URL: http://www.elsevier.nl/locate/entcs/volume84.html 13 pages

k-Valued Non-Associative Lambek Grammars are Learnable from Function-Argument Structures

Denis Béchet [1], Annie Foret [2]

IRISA, INRIA & Université de Rennes 1
Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 Rennes Cedex
France

Abstract

This paper is concerned with learning categorial grammars in the model of Gold. We show that rigid and k-valued non-associative Lambek grammars are learnable from function-argument structured sentences. In fact, function-argument structures are natural syntactical decompositions of sentences in sub-components with the indication of the head of each sub-component.

This result is interesting and surprising because for every k, the class of k-valued *NL* grammars has infinite elasticity and one could think that it is not learnable, which is not true. Moreover, these classes are very close to unlearnable classes like k-valued associative Lambek grammars learned from function-argument sentences or k-valued non-associative Lambek calculus grammars learned from well-bracketed list of words or from strings. Thus, the k-valued non-associative Lambek grammars learned from function-argument sentences is at the frontier between learnable and unlearnable classes of languages.

Keywords: grammatical inference, categorial grammars, non-associative Lambek calculus, learning from positive examples, model of Gold, computational linguistic.

1 Introduction

Lexicalized grammars of natural languages are well adapted to learning perspectives. The model of Gold [11] used here consists in defining an algorithm on a finite set of structured sentences that converge to obtain a grammar in the class that generates the examples.

[1] Email: Denis.Bechet@irisa.fr
[2] Email: Annie.Foret@irisa.fr

Let $\mathcal{CG}$ be a class of grammars that we wish to learn from positive examples. Let $\mathcal{L}(G)$ denote the language associated with a grammar G. A learning algorithm is a function ϕ from finite sets of (structured) strings to $\mathcal{CG}$, such that for any $G \in \mathcal{CG}$ and $< e_i >_{i\in\mathbb{N}}$ any enumeration of $\mathcal{L}(G)$, there exist a grammar $G' \in \mathcal{CG}$ such that $\mathcal{L}(G') = \mathcal{L}(G)$ and $n_0 \in \mathbb{N}$ such that $\forall n > n_0\ \phi(\{e_0, \ldots, e_n\}) = G'$.

After pessimistic unlearnability results in [11], learnability of non trivial classes has been proved in [2,19]. Recent works[12,18] following [6] have answered the problem for different sub-classes of classical categorial grammars (the whole class of classical categorial grammars and the whole class of (non)-associative Lambek grammars are equivalent to context free grammars and thus is not learnable in Gold's model).

In fact, the learnable or unlearnable problem for a class of grammars depends both of the informations that the input structures carry and the model that defines the language associated to a given grammar. The input informations can be just a string, the list of words of the input sentence. It can be a tree that describes the sub-components with or without the indication of the head of each sub-component. More complex input informations give natural deduction structure or semantics informations. For k-valued categorial grammars [3], classical categorial grammars [3], noted *AB* grammars, are learnable from strings, the simplest form of informations[12]. Associative Lambek categorial grammars [14], noted L grammars, are learnable from natural deduction structures [4] but not from strings and sub-component trees [9,10].

Non-associative Lambek categorial grammars [15], noted *NL* grammars, lie between classical categorial grammars and associative Lambek grammars since for the same categorial grammar G, the associated language $\mathcal{L}_{NL}(G)$ includes the corresponding classical categorial language $\mathcal{L}_{AB}(G)$ but is a subset of the associative Lambek language from the same lexicon, $\mathcal{L}_L(G)$. Thus, the learnability problem for this class is interesting.

Usually, to prove that a class of language is learnable in Gold's model, we prove that the class has finite elasticity [22,17]. However, we show here that this does not hold for k-valued non-associative Lambek categorial grammars. However, we can bypass this difficulty. In fact, this class is learnable as it is shown in the paper.

The paper is organized as follows. Section 2 gives some background knowledge on three main aspects: non-associative Lambek categorial grammars; learning in Gold's model; learning non-associative Lambek categorial grammars from function-argument structures. Section 3 presents the proof that the class of rigid (and thus k-valued) non-associative Lambek categorial grammars have infinite elasticity and thus is not easily learnable in Gold's model. Section 4 shows our main result by building a learning algorithm and by proving

[3] A k-valued lexicalized grammar is a lexicalized grammar where each word has at most k entries and for a categorial grammar, at most k types.

that it learns in k-valued non-associative Lambek categorial grammars from function-argument structures in Gold's model. 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 articles written by Lambek [14,15] or more recently in [13,1,5,16,7,8]. We use in the paper non-associative Lambek calculus without empty sequence and without product.

Types. The *types* Tp, or formulas, are generated from a set of *primitive types* Pr, or atomic formulas, by two binary connectives [4] "/" (over) and "\" (under):

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

Rigid and k-valued categorial grammars.

- A *categorial grammar* is a structure $G = (\Sigma, I, S)$ where:
  - Σ is a finite alphabet (the words in the sentences);
  - $I : \Sigma \mapsto \mathcal{P}^f(T)$ is a function that maps a set of types to each element of Σ (the possible categories of each word);
  - $S \in Pr$ is the *main type* associated to correct sentences.
- if $X \in I(a)$, we say that G associates X to a and we write $G : a \mapsto X$.
- a *k-valued categorial grammar* is a categorial grammar where, for every word $a \in \Sigma$, $I(a)$ has at most k elements.
- a *rigid categorial grammar* is a 1-valued categorial grammars

2.2 Non-associative Lambek Calculus NL

2.2.1 NL derivation $\vdash_{NL}$

As a logical system, we use Gentzen-style sequent presentation. A sequent $\Gamma \vdash A$ is composed of a binary tree of formulas Γ (the set of tree is noted $\mathcal{T}_{Tp}$) which is the antecedent configuration and a succedent formula A. A context $\Gamma[i\cdot]$ is a binary tree of formulas with a hole. For X, a formula or a binary tree of formulas, $\Gamma[X]$ is the binary tree obtained from $\Gamma[\cdot]$ by filling the hole with X. A sequent is valid in NL and is noted $\Gamma \vdash_{NL} A$ iff $\Gamma \vdash A$ can be deduced from the following rules:

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

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

[4] product connective is not used in the paper

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

2.2.2 NL languages

Yield. If T is a tree where the leaves are elements of a set $\mathcal{E}$, $yield_{\mathcal{E}}(T) \in \mathcal{E}^+$ is the list of leaves of T. This notation will be used for well-bracketed list of words $yield_{\Sigma}$, for binary trees of formulas $yield_{Tp}$ and also for *FA* structures (see below).

Language. Let $G = (\Sigma, I, S)$ be a categorial grammar over Σ.

- *G generates a well-bracketed list of words $T \in \mathcal{T}_\Sigma$* (in *NL* model) iff it exists Γ a binary tree of types, $c_1, \ldots, c_n \in \Sigma$ and $A_1, \ldots, A_n \in Tp$ such that:
$$\begin{cases} G : c_i \mapsto A_i \ (1 \leq i \leq n) \\ \Gamma = T[c_1 \rightarrow A_1, \ldots, c_n \rightarrow A_n] \\ \Gamma \vdash_{NL} S \end{cases}$$
where $T[c_1 \rightarrow A_1, \ldots, c_n \rightarrow A_n]$ means the binary tree obtained from T by substituting the left to right occurrences of $c_1, \ldots, c_n$ by $A_1, \ldots, A_n$.
- *G generates a string $c_1 \cdots c_n \in \Sigma^+$* iff it exists $T \in \mathcal{T}_\Sigma$ such that $yield_\Sigma(T) = c_1 \cdots c_n$ and G generates T.
- The *language of well-bracketed lists of words* corresponding to G, written $\mathcal{L}_{NL}^{\mathcal{T}_\Sigma}(G)$, is the set of well-bracketed lists of words generated by G.
- The *language of strings* corresponding to G, written $\mathcal{L}_{NL}^{\Sigma^+}(G)$, is the set of strings generated by G.

Example 2.1 Let $\Sigma_1 = \{John, Mary, likes\}$ and let $Pr_1 = \{S, N\}$. We define:
$$G_1 = \begin{cases} John \ \mapsto N \\ Mary \mapsto N \\ likes \ \mapsto N \backslash (S/N) \end{cases}$$
G_1 is a rigid (or 1-valued) grammar. We can prove that $((N,\ N\backslash(S/N)),\ N) \vdash_{NL} S$. Thus, we get:
$$John\ likes\ Mary \quad \in \mathcal{L}_{NL}^{\Sigma^+}(G_1)$$
$$((John\ likes)\ Mary) \in \mathcal{L}_{NL}^{\mathcal{T}_\Sigma}(G_1)$$

2.3 Learning and Elasticity

2.3.1 Learning algorithm

For a class $\mathcal{CG}$ of grammars, we write $\mathcal{L}(G)$ the language that is generated by $G \in \mathcal{CG}$ and $\mathcal{CL} = \{\mathcal{L}(G) \mid G \in \mathcal{CG}\}$ the class of generated languages. A learning algorithm ϕ on $\mathcal{CG}$ is an algorithm that takes as input a finite set of (structured) sentences and returned a grammar of $\mathcal{CG}$. ϕ learns $\mathcal{CG}$ in Gold's model iff for any enumeration $< e_i >_{i \in \mathbb{N}}$ of a language $\mathcal{L}(G)$ where $G \in \mathcal{CG}$, there exists $n_0 \in \mathbb{N}$ and a grammar $G' \in \mathcal{CG}$ such that $\mathcal{L}(G') = \mathcal{L}(G)$ and $\forall n \geq n_0, \phi(\{e_0, \ldots, e_i\}) = G'$.

2.3.2 Infinite elasticity

- A class $\mathcal{CL}$ of languages has *infinite elasticity* iff it exists an infinite sequence $< e_i >_{i\in\mathbb{N}}$ of sentences and an infinite sequence $< L_i >_{i\in\mathbb{N}}$ of languages in $\mathcal{CL}$ such that $\forall n \in \mathbb{N} : e_n \notin L_n$ and $\{e_0, \ldots, e_{n-1}\} \subseteq L_n$.
- A class $\mathcal{CL}$ of languages has *finite elasticity* iff it has not infinite elasticity.

Finite elasticity implies learnability. If the languages corresponding to a class of grammars $\mathcal{CG}$ have finite elasticity then $\mathcal{CG}$ is *learnable* in Gold's model [22].

2.4 Learning from FA structures

2.4.1 FA structures

Let Σ be an alphabet, a *FA structure* over Σ is a binary tree where each leaf is labelled by an element of Σ and each internal node is label by *FApp* (forward application) or *BApp* (backward application):

$$\mathcal{FA}_\Sigma ::= \Sigma \mid \mathit{FApp}(\mathcal{FA}_\Sigma, \mathcal{FA}_\Sigma) \mid \mathit{BApp}(\mathcal{FA}_\Sigma, \mathcal{FA}_\Sigma)$$

Yield and tree yield. $yield_\Sigma$ can be naturally extended to *FA* structures. Moreover, the well-bracketed list of words obtained from a *FA* structure F over Σ by forgetting *FApp* and *BApp* labels is called the *tree yield* of F (notation $tree_\Sigma(F)$). More generally, if $\mathcal{E}$ is a set, $\mathcal{T}_\mathcal{E}$ denotes the set of well-bracketed list of elements of $\mathcal{E}$ and if T is a binary tree where the leaves are elements of $\mathcal{E}$, $tree_\mathcal{E}(T) \in \mathcal{T}_\mathcal{E}$ is the well-bracketed list of elements of $\mathcal{E}$ corresponding to T ($tree_\mathcal{E}$ forget the information on the internal nodes of T).

2.4.2 GAB Deduction

A *generalized AB deduction*, or GAB deduction, over *Tp* is a binary tree using the following conditional rules ($C \vdash_{NL} B$ must be valid in *NL*):

$$\frac{A/B \quad C}{A}\ \mathit{FApp} \qquad \frac{C \quad B\backslash A}{A}\ \mathit{BApp}$$

$$C \vdash_{NL} B \text{ valid in } NL$$

In fact, a deduction must be justified, for each node, by a proof of the corresponding sequent in *NL*. Thus, a rule has three premises: the two sub-deductions and a *NL* derivation. Moreover, a *GAB* deduction can be seen as a *FA* structure where the leaves are types and the nodes need a logical justification. We write $FA_{Tp}(D)$ for the *FA* structure that corresponds to D (internal types and NL derivations are forgotten). We also write $tree_{Tp}(D)$ for the corresponding well-bracketed list of types.

- For $F \in FA_{Tp}$ and $A \in Tp$, we say that D is a *GAB* deduction of $F \vdash_{GAB} A$ when A is the type of the conclusion of D and when $FA_{Tp}(D) = F$.
- For $\Gamma \in \mathcal{T}_{Tp}$ and $A \in Tp$, we say that D is a *GAB* deduction of $\Gamma \vdash_{GAB} A$ when A is the type of the conclusion of D and when $tree_{Tp}(D) = F$.

2.4.3 GAB Languages

Like *NL*, we can associate to each categorial grammar a language of *FA* structures. Let $G = (\Sigma, I, S)$ be a categorial grammar over Tp:

- $G = (\Sigma, I, S)$ *generates a FA structure* $F \in FA_\Sigma$ (in the *GAB* derivation model) iff it exists a *GAB* derivation of a *FA* structure D, $c_1, \ldots, c_n \in \Sigma$ and $A_1, \ldots, A_n \in Tp$ such that:

$$\begin{cases} G : c_i \mapsto A_i \ (1 \leq i \leq n) \\ D = T[c_1 \rightarrow A_1, \ldots, c_n \rightarrow A_n] \\ D \vdash_{GAB} S \end{cases}$$

  where $T[c_1 \rightarrow A_1, \ldots, c_n \rightarrow A_n]$ means the *FA* structure obtained from F by substituting respectively the left to right occurrences of $c_1, \ldots, c_n$ by $A_1, \ldots, A_n$.
- G *generates a well-bracketed list of words* $T \in \mathcal{T}_\Sigma$ iff it exists $F \in FA_\Sigma$ such that $tree_\Sigma(T) = T$ and G generates F.
- G *generates a string* $c_1 \cdots c_n \in \Sigma^+$ iff it exists $F \in FA_\Sigma$ such that $yield_\Sigma(F) = c_1 \cdots c_n$ and G generates F.
- The *language of FA structures* corresponding to G, written $\mathcal{L}_{GAB}^{FA_\Sigma}(G)$, is the set of *FA* structures generated by G.
- The *language of well-bracketed lists of words* corresponding to G, written $\mathcal{L}_{GAB}^{\mathcal{T}_\Sigma}(G)$, is the set of well-bracketed lists of words generated by G.
- The *language of strings* corresponding to G, written $\mathcal{L}_{GAB}^{\Sigma^+}(G)$, is the set of strings generated by G.

The language of *FA* structures $\mathcal{L}_{GAB}^{FA_\Sigma}(G)$, the language of well-bracketed list of words $\mathcal{L}_{GAB}^{\mathcal{T}_\Sigma}(G)$ and the language of strings $\mathcal{L}_{GAB}^{\Sigma^+}$ that are associated to a categorial grammar G are the set of *FA* structures, the set of well-bracketed lists of words and the set of strings that are generated by this grammar.

Example 2.2 If we take the categorial grammar that is defined in Example 1, we get:

$$\begin{aligned} John\ likes\ Mary &\in \mathcal{L}_{GAB}^{\Sigma^+}(G_1) \\ ((John, likes), Mary) &\in \mathcal{L}_{GAB}^{\mathcal{T}_\Sigma}(G_1) \\ FApp(BApp(John, likes), Mary) &\in \mathcal{L}_{GAB}^{FA_\Sigma}(G_1) \end{aligned}$$

because we can build the following deduction:

$$\dfrac{\dfrac{\overbrace{N}^{John} \quad \overbrace{N\backslash(S/N)}^{likes}}{S/N}\,BApp \quad \overbrace{N}^{Mary}}{S}\,FApp$$

2.5 NL and GAB languages

In fact, there is a strong correspondence between *GAB* deductions and *NL* derivations. Thus, it is not necessary to distinguish the two different concepts.

Theorem 2.3 *If A is an atomic formula, $\Gamma \vdash_{GAB} A$ iff $\Gamma \vdash_{NL} A$*

Corollary 2.4 *$\mathcal{L}_{NL}^{\mathcal{T}_\Sigma}(G) = \mathcal{L}_{GAB}^{\mathcal{T}_\Sigma}(G)$ and $\mathcal{L}_{NL}^{\Sigma^+} = \mathcal{L}_{GAB}^{\Sigma^+}$*

We write, for the rest of the paper, $\mathcal{L}^{FA_\Sigma}(G)$, $\mathcal{L}^{\mathcal{T}_\Sigma}(G)$ and $\mathcal{L}^{\Sigma^+}$ in place of $\mathcal{L}_{GAB}^{FA_\Sigma}(G)$, $\mathcal{L}_{GAB}^{\mathcal{T}_\Sigma}(G) = \mathcal{L}_{NL}^{\mathcal{T}_\Sigma}(G)$ and $\mathcal{L}_{GAB}^{\Sigma^+} = \mathcal{L}_{NL}^{\Sigma^+}$. We usually write $\mathcal{L}(G)$ for $\mathcal{L}^{FA_\Sigma}(G)$.

Proof of $\Gamma \vdash_{GAB} A \Rightarrow \Gamma \vdash_{NL} A$: This is relatively easy because a *GAB* deduction is just a mixed presentation of an *NL* proof using a natural deduction part and a *NL* derivation part (hypotheses on nodes). We can transform recursively a *GAB* deduction. The last rule of a *GAB* deduction corresponding to a *FA* structure $FApp(F_1, F_2)$ is:

$$\dfrac{\begin{array}{cc} D_1 & D_2 \\ \vdots & \vdots \\ A/B & C \end{array}}{A}\ FApp$$

We know that $C \vdash_{NL} B$ and we have two sub-deductions D_1 and D_2 that correspond to F_1 and F_2. The first one, D_1, concludes with A/B and the second, D_2, with C. By induction hypothesis, the two deductions correspond to two *NL* derivations of $tree_{Tp}(F_1) \vdash_{NL} A/B$ and $tree_{Tp}(F_2) \vdash_{NL} C$. Now, using two cuts and $(/E)$, we find that $tree_{Tp}(FApp(F_1, F_2)) = (tree_{Tp}(F_1), tree_{Tp}(F_2)) \vdash_{NL} A$. The other possibility ($BApp$) as first rule) is very similar and the base case is obvious. ■

Proof of $\Gamma \vdash_{NL} A \Rightarrow \Gamma \vdash_{GAB} A$: This property results from an alternative presentation of *NL* where contexts are in a limited form [1]:

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

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

Aarts and Trautwein in [1] have proved the equivalence of *NL* and this system called NLD_0^{**}. Now, if we have a *NL* derivation of $\Gamma \vdash_{NL} A$ with A atomic, the first rule must be a left rule. For instance, for $(/L)$, Γ can be written $\Delta[(B/C, D)]$ and we get a NLD_0^{**} derivation of $D \vdash C$ and another one of $\Delta[B] \vdash A$. We can apply our hypothesis to the second derivation. At this point, we have a *GAB* deduction $P[B]$ of $\Delta[B] \vdash_{GAB} A$. In this deduction, we

replace the leaf node corresponding to B by a new node corresponding to the conclusion of (*FApp*) rule:

$$\begin{array}{ccc} & & \dfrac{B/C \quad D}{B} \textit{FApp} \\ B & & \\ \vdots & \rightarrow & \vdots \\ P & & P \end{array}$$

This transformation gives a *GAB* deduction corresponding to $\Delta[(B/C, D)]$ since $D \vdash C$. The other possibility for $(\backslash L)$ is symmetrical and the base case where the derivation is an axiom is obvious. ■

3 Infinite Elasticity Theorem

We prove, in this section that, for each $k \in \mathbb{N}$, the class of k-valued *NL* languages of *FA* structures has infinite elasticity. Thus, the learning problem which is solved in section 4 is difficult for this class.

The problem here is to find an infinite sequence $< G_i >_{i\in\mathbb{N}}$ of categorial grammars and an infinite sequence $< F_i >_{i\in\mathbb{N}}$ of *FA* structures such that, for all $n \in \mathbb{N}$:

$$\begin{cases} F_n \notin \mathcal{L}^{FA_\Sigma}(G_n) \\ \{F_0, \ldots, F_{n-1}\} \subseteq \mathcal{L}^{FA_\Sigma}(G_n) \end{cases}$$

3.1 Definition of the Infinite Sequences

The primitive types are $Pr = \{A, S\}$. We define by induction formulas $D_0 = A$ and $D_{n+1} = D_n/(D_n \backslash D_n)$. The alphabet is $\Sigma = \{a, m, b\}$. We define:

$$G_n : \begin{cases} a \mapsto A\backslash A \\ b \mapsto D_n \\ c \mapsto S/D_n \end{cases}$$

We define by induction *FA* structures $E_0 = b$ and $E_{n+1} = \textit{FApp}(E_n, a)$. Finally the sequence of *FA* structures is defined by $< F_n = \textit{FApp}(c, E_{n+1}) >$.

Proof of $\forall n \in \mathbb{N} : \{F_1, \ldots, F_n\} \subseteq \mathcal{L}^{FA_\Sigma}(G_{n+1})$**:** In fact we can first prove that $\forall n \in \mathbb{N}, D_n \vdash_{NL} D_{n+1}$. This is obvious because $D_{n+1} = D_n/(D_n \backslash D_n)$ is a type-raising of D_n. Thus, if $0 \leq i \leq n$, we have $D_i \vdash_{NL} D_n$. Secondly, we can prove by induction that $A\backslash A \vdash_{NL} D_n \backslash D_n$. For $n = 0$, it is obvious and for $n > 0$, by hypothesis, we have $A\backslash A \vdash_{NL} D_{n-1} \backslash D_{n-1}$ and because $D_{n-1} \vdash_{NL} D_n$, we have $(D_{n-1}/(D_{n-1} \backslash D_{n-1}), A\backslash A) \vdash_{NL} D_n$. Then $A\backslash A \vdash_{NL} D_n/(D_{n-1}/(D_{n-1} \backslash D_{n-1})) = D_n/D_n$. For the rest, we have to check that we

can put these derivation on unique the *FA* structure on *Tp* that correspond to F_n (G_n is rigid and there is no choice for the type of each element of Σ). ■

Proof of $F_n \notin \mathcal{L}^{FA_\Sigma}(G_n)$: In fact, with *FA* structures, we know the structure of a corresponding derivation and we just have to find a justification for internal rules. For a derivation corresponding to F_n in $\mathcal{L}^{FA_\Sigma}(G_n)$, since $G : b \mapsto D_n$ and $G : a \mapsto A\backslash A$, the deepest internal node for $n > 0$ is:

$$\dfrac{\overbrace{D_n = D_{n-1}/(D_{n-1}\backslash D_{n-1})}^{b} \quad \overbrace{A\backslash A}^{a}}{D_{n-1}}\ FApp$$
$$\vdots$$

If we go from the deepest node to the root, we find successively formulas $D_{n-1}, \ldots$. But, because the *FA* structure have $n+1$ "a", the derivation looks like:

$$\vdots$$
$$\dfrac{\overbrace{S/D_n}^{c} \qquad \dfrac{D_0 = A \quad \overbrace{A\backslash A}^{a}}{?}\ FApp}{S}\ FApp$$

which is impossible because A is atomic and can not be the function in a function-argument rule (this is the reason why a "?" appears on the deduction). ■

4 Learnability Theorem

Previous section shows that the class of *NL* languages has infinite elasticity. Thus, it is not possible to use a general property given by learning theory. To solve this problem, we define sub-classes of *NL* grammars and prove that they have finite elasticity. Then, we use learning algorithms that learn these classes and define a learning algorithm for the whole class.

4.1 Order of NL languages

Order of *FA* structures. The order of a *FA* structure on *Tp* corresponds to the maximum number of arguments of each function in the structure. It does not correspond to the "arity" of a functional expression but is bound by the maximum "arity" [5] of the types on the leaves of the structure. It is defined

[5] Arity can be defined by induction: $arity(A) = 0$ if $A \in Pr$ and $arity(A/B) = arity(B\backslash A) = arity(A) + 1$.

by:

$$order_{fa}(A) = 0 \quad \text{if} A \in Tp$$
$$order_{fa}(FApp(F_1, F_2)) = max(order_{fa}(F_1) + 1, order_{fa}(F_2))$$
$$order_{fa}(BApp(F_1, F_2)) = max(order_{fa}(F_1), order_{fa}(F_2) + 1)$$

Order of $\mathcal{L}^{FA_\Sigma}(G)$. For a categorial grammar G, we define the order of the *NL* language associated to this grammar by the maximum order of its *FA* structures: $order_{fa}(\mathcal{L}^{FA_\Sigma}(G) = max\{order_{fa}(F) \mid F \in \mathcal{L}^{FA_\Sigma}(G)\}$. This maximum exists for k-valued categorial grammars because the order of a *FA* structure is bound be the maximum arity of the types on the leaves of the structure which is bound by the maximum arity of the types that appear in the grammar.

Order-bounded *NL* languages. The class of *NL* languages of *FA* structures whom order is bound by n is noted $\mathcal{CL}^{(order_{fa} \leq n)}$. The corresponding grammars are noted $\mathcal{CG}^{(order_{fa} \leq n)}$. For k-valued categorial grammars, we write $\mathcal{CL}_k^{(order_{fa} \leq n)}$ and $\mathcal{CG}_k^{(order_{fa} \leq n)}$ and for rigid categorial grammars, $\mathcal{CL}_1^{(order_{fa} \leq n)}$ and $\mathcal{CG}_1^{(order_{fa} \leq n)}$

4.2 $\mathcal{CL}_1^{(order_{fa} \leq n)}$ has finite elasticity.

This lemma is essential to our proof because as a corollary of general results[12], the corresponding classes of languages on well-bracketed lists of words and on strings have also finite elasticity. Moreover, this result can also be extended to k-valued grammars. As a consequence, all these classes are learnable in the Gold's model and we can find a learning algorithm for each of them.

Proof: To prove that $\mathcal{CL}_1^{(order_{fa} \leq n)}$ has finite elasticity, we use a result by Shinohara [19,20] showing that *formal systems* having *finite thickness* must have finite elasticity. In [20] this is applied to *length-bounded elementary formal systems with at most k rules* and also to *context sensitive languages* that are definable by at most k rules. Formal systems in [20] do not describe only languages of strings but also languages of terms like our *FA* structures. Thus, here is just a new application of this theorem to $\mathcal{CL}_1^{(order_{fa} \leq n)}$. For this class, the sketch of proof is as follows:

(i) **Definition.** A categorial grammar $G_1 = (\Sigma_1, I_1, S)$ is *included* in a categorial grammar $G_2 = (\Sigma_2, I_2, S)$ (notation $G_1 \subseteq G_2$) iff $\Sigma_1 \subseteq \Sigma_2$ and $\forall x \in \Sigma_1$, $I_1(x) \subseteq I_2(x)$.

(ii) **Definition and lemma.** The mapping $\mathcal{L}^{FA_\Sigma}$ from $G \in \mathcal{CG}_1^{(order_{fa} \leq n)}$ to $G \in \mathcal{CL}_1^{(order_{fa} \leq n)}$ is monotonic: if $G_1 \subseteq G_2$ then $\mathcal{L}^{FA_\Sigma}(G_1) \subseteq \mathcal{L}^{FA_\Sigma}(G_2)$.

(iii) **Definition.** A grammar G is *reduced with respect to a set* $X \subseteq FA_\Sigma$ iff $X \subseteq \mathcal{L}^{FA_\Sigma}(G)$ and for each grammar $G' \subseteq G$, $X \not\subseteq \mathcal{L}^{FA_\Sigma}(G')$. Intuitively, a grammar that is reduced with respect to X, does not have redundant expressions to cover all the structures of X.

(iv) **Lemma.** For each finite set $X \subseteq \mathcal{L}^{FA_\Sigma}(G)$, there is a finite set of languages in $\mathcal{CL}_1^{(order_{fa} \leq n)}$ that correspond to grammars reduced from X. This is the main part of the proof and is a consequence of the fact that a k-valued *GAB* grammars of order not greater than n can be completely described by the function-argument possible applications of the types that appear in the lexicon and their main subtypes limited to a depth of n [6]. This boolean system has at most $(n+1)^2 \times k^2 \times \#(\Sigma)^2$ values because there are at most $k \times \#(\Sigma)$ types in G and we need two types or subtypes (thus at most $n+1$ values for each type) one as function and one as argument.

(v) **Definition.** Monotonicity and the previous property define a system that has *bounded finite thickness*.

(vi) **Theorem.** Shinohara proves in [20] that a formal system that has bounded finite thickness has finite elasticity.

(vii) **Corollary.** $\mathcal{CL}_k^{(order_{fa} \leq n)}$ has finite elasticity as a consequence of a general theorem on classes that are related by a finite-valued relation: finite elasticity of one class is equivalent to finite elasticity of the other. This construction is, for instance, proved and used in [12] to go from rigid to k-valued classical categorial grammars which have also finite elasticity. ■

4.3 k-valued NL language is learnable from FA structures

Because, for each n and k, the class $\mathcal{CL}_k^{(order_{fa} \leq n)}$ has finite elasticity, there exists an algorithm ϕ_k^n that learns the languages of this class from *FA* structures in Gold's model. We define the following algorithm ϕ_k that takes a finite list of *FA* structures $F_1, \ldots, F_l$ and returns a categorial grammar (or fails):

(i) Compute the maximum order r of the l input *FA* structures.

(ii) Apply algorithm ϕ_k^r on $F_1, \ldots, F_l$.

This algorithm defines a learning mechanism for k-valued *NL* grammars from *FA* structures because if for a language L that corresponds to a k-valued *NL* grammar, there exists at least one *FA* structure F such that $order_{fa}(F) = order_{fa}(L)$. Thus, for every enumeration on the *FA* structure of L, their exists an integer s such that for every $l \geq s$, the number r computed by ϕ_k is $order_{fa}(L)$. From this integer, ϕ_k applies the proper algorithm $\phi_k^{order_{fa}(L)}$ that converge to L.

[6] The subtypes of a type limited to a depth of n can be defined by $subtypes_0(A) = \emptyset$, $subtypes_n(A) = \{A\}$ if A is atomic and $subtypes_n(A/B) = subtypes_n(B \backslash A) = \{B\} \cup subtypes_{n-1}(A)$.

5 Conclusion

Learnability from function-argument structures. We have shown first in the paper how we can define languages of function-argument structures of sentences based on non-associative Lambek calculus. Secondly, we have proved that, for each $k \geq 0$, the class of k-valued non-associative Lambek languages of function-argument structures has infinite elasticity and thus is difficult to learn in Gold's model. Finally, we have shown how we can bypass this problem and define a learning algorithm for this class of languages.

Learnability from strings and well-bracketed lists of words. Unfortunately, the learning algorithm on function-argument structures can not be adapted to the problems of learning non-associative Lambek languages from strings or from well-bracketed lists of words because we need to bound the effective arity of each element of the lexicon. This information is given by *FA* structures but not by strings or well-bracketed lists of words. Thus the paper does not solve this problem.

References

[1] Aarts, E. and K. Trautwein, *Non-associative Lambek categorial grammar in polynomial time*, Mathematical Logic Quaterly **41** (1995), pp. 476–484.

[2] Angluin, D., *Inductive inference of formal languages from positive data*, Information and Control **45** (1980), pp. 117–135.

[3] Bar-Hillel, Y., *A quasi arithmetical notation for syntactic description*, Language **29** (1953), pp. 47–58.

[4] Bonato, R. and C. Retoré, *Learning rigid lambek grammars and minimalist grammars from struc tured sentences*, Third workshop on Learning Language in Logic, Strasbourg (september 2001).

[5] Buszkowski, W., *Mathematical linguistics and proof theory*, in: van Benthem and ter Meulen [21] pp. 683–736.

[6] Buszkowski, W. and G. Penn, *Categorial grammars determined from linguistic data by unification*, Studia Logica **49** (1990), pp. 431–454.

[7] de Groote, P., *Non-associative Lambek calculus in polynomial time*, in: *8^th Workshop on theorem proving with analytic tableaux and related methods*, number 1617 in Lecture Notes in Artificial Intelligence (1999), pp. 128–139.

[8] de Groote, P. and F. Lamarche, *Classical non-associative lambek calculus*, Studia Logica **71.1 (2)** (2002).

[9] Foret, A. and Y. Le Nir, *Lambek rigid grammars are not learnable from strings*, in: *COLING'2002, 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.

[10] Foret, A. and Y. Le Nir, *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 (2002), pp. 106–119.

[11] Gold, E., *Language identification in the limit*, Information and control **10** (1967), pp. 447–474.

[12] Kanazawa, M., "Learnable classes of categorial grammars," Studies in Logic, Language and Information, FoLLI & CSLI, 1998, distributed by Cambridge University Press.

[13] Kandulski, M., *The non-associative lambek calculus*, in: W. M. W. Buszkowski and J. V. Bentem, editors, *Categorial Grammar*, Benjamins, Amsterdam, 1988 pp. 141–152.

[14] Lambek, J., *The mathematics of sentence structure*, American mathematical monthly **65** (1958), pp. 154–169.

[15] Lambek, J., *On the calculus of syntactic types*, in: R. Jakobson, editor, *Structure of language and its mathematical aspects*, American Mathematical Society, 1961 pp. 166–178.

[16] Moortgat, M., *Categorial type logic*, in: van Benthem and ter Meulen [21] pp. 93–177.

[17] Motoki, T., T. Shinohara and K. Wright, *The correct definition of finite elasticity: Corrigendum to identification of unions*, in: *The fourth Annual Workshop on Computational Learning Theory* (1991), p. 375.

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

[19] Shinohara, T., *Inductive inference from positive data is powerful*, in: *The 1990 Workshop on Computational Learning Theory* (1990), pp. 97–110.

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

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

[22] Wright, K., *Identifications of unions of languages drawn from an identifiable class*, in: *The 1989 Workshop on Computational Learning Theory* (1989), pp. 328–333.