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

Denis Béchet and Annie Foret [a,b]

[a] *LINA – CNRS FRE 2729, Université de Nantes, France*

[b] *IRISA – Université de Rennes 1, France*

Abstract

This paper is concerned with learning categorial grammars from positive examples in the model of Gold. Functor-argument structures (written FA) are usual syntactical decompositions of sentences in sub-components distinguishing the functional parts from the argument parts defined in the case of classical categorial grammars also known as AB-grammars. In the case of non-associative type-logical grammars, we propose a similar notion that we call generalized functor-argument structures and we show that these structures capture the essence of non-associative Lambek calculus (NL) without product.

We show that (i) rigid and k-valued non-associative Lambek (NL without product) grammars are learnable from generalized functor-argument structured sentences.

We also define subclasses of k-valued grammars in terms of arity. We first show that (ii) for each k and each bound on arity the class of *FA*-arity bounded k-valued *NL* languages of *FA* structures is finite and (iii) that *FA*-arity bounded k-valued NL grammars are learnable both from strings and from *FA* structures as a corollary.

Result (i) is obtained from (ii); this learnability result (i) is interesting and surprising when compared to other results: in fact we also show that (iv) this class has infinite elasticity. Moreover, these classes are very close to classes like rigid associative Lambek grammars learned from natural deduction structured sentences (that are different and much richer than FA or generalized FA) or to k-valued non-associative Lambek grammars unlearnable from strings or even from bracketed strings. Thus, the class of k-valued non-associative Lambek grammars learned from generalized functor-argument sentences is at the frontier between learnable and unlearnable classes of languages.

Key words: grammatical inference, categorial grammars, non-associative Lambek calculus, learning from positive examples, model of Gold

1 Introduction

Lexicalized grammars of natural languages are well adapted to learning perspectives. The model of Gold [1] used here consists in defining, given a class $\mathcal{G}$ of grammars, an algorithm on a finite set of structured sentences, computing a grammar in $\mathcal{G}$; given any infinite sequence enumerating a language of a grammar in $\mathcal{G}$, this algorithm must converge to obtain a grammar in $\mathcal{G}$ generating the same language.

After pessimistic unlearnability results in [1], learnability of non trivial classes has been proved in [2,3]. Recent works[4,5] 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 are not learnable in Gold's model).

In fact, the learnable-or-unlearnable problem for a class of grammars depends both on the information that the input structures carry and on the model that defines the language associated to a given grammar. The input information 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 [1] , classical categorial grammars [7], noted AB grammars, are learnable from strings, the simplest form of informations[4]. Rigid (1-valued) associative Lambek categorial grammars [8], denoted L grammars, are learnable from natural deduction structures [9] (that are different from functor-argument structures) but not from strings [10,11] ; in their commutative-associative version, 1-valued Lambek grammars are neither not learnable from strings [12,13].

Non-associative Lambek categorial grammars [14], denoted NL grammars, lie between classical categorial grammars and associative Lambek grammars since for the same assignments of types to the lexicon of a 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 [15,16]. However, we show here that

Email addresses: Denis.Bechet@univ-nantes.fr, Annie.Foret@irisa.fr (Denis Béchet and Annie Foret).

[1] A k-valued lexicalized grammar is a lexicalized grammar where each word has at most k entries ; in the case of categorial grammars, this means at most k types.

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. This is not the first example of a learnable class with infinite elasticity : the famous class of languages recognized by k-reversible automata does not have finite elasticity, but is nevertheless learnable [17] .

The paper is organized as follows. Section 2 gives some background knowledge on non-associative Lambek categorial grammars and on learning in Gold's model. In section 3 we define alternative deduction rules for NL-grammars (without product) and we define generalized FA-structures; in fact these rules are extensions of the cancelation rules of classical categorial grammars that lead to the generalization of FA-structures proposed here. Section 4 presents the proof that the class of 1-valued (and thus k-valued) non-associative Lambek categorial grammars have infinite elasticity and thus is not easily learnable in Gold's model. Section 5 shows that k-valued non-associative Lambek categorial grammars are learnable from generalized FA-structures in Gold's model. Section 6 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 [8,14] or more recently in [18–23]. We use in the paper non-associative Lambek calculus without empty sequence and without product.

Definition 1 (Types) *The* types *Tp, or formulas, are generated from a set of* primitive types *Pr, or atomic formulas, by two binary connectives*[2] *"/" (over) and "\" (under):*

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

Definition 2 (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);*
- $I : \Sigma \mapsto \mathcal{P}^f(Tp)$ *is a function (called a lexicon) that assigns a finite 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

[2] no product connective is used in the paper

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 grammar.*

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 such trees is noted $\mathcal{T}_{Tp}$) which is the antecedent configuration and a succedent formula A. A context $\Gamma[\cdot]$ 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.

Definition 3 (NL) *A sequent is valid in NL and is noted* $\Gamma \vdash_{NL} A$ *iff* $\Gamma \vdash A$ *can be deduced from the following rules:*

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

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

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

2.2.2 NL languages

$\mathcal{E}^+$ denotes the set of non-empty strings over $\mathcal{E}$. Let $\mathcal{T}_\mathcal{E}$ denote the set of (non-empty) well-bracketed lists (binary trees) of elements of $\mathcal{E}$.

Definition 4 (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 lists of words $yield_\Sigma$, for binary trees of formulas $yield_{Tp}$ and will be extended to *FA* structures (see further Definition 11).

Definition 5 (Language) *Let* $G = (\Sigma, I, S)$ *be a categorial grammar.*

- G generates a well-bracketed list of words $T \in \mathcal{T}_\Sigma$ *(in NL model) iff there exists* Γ *a binary tree of types,* $c_1, \ldots, c_n \in \Sigma$ *and* $A_1, \ldots, A_n \in Tp$ *such*

that:

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

where $T[c_1 \to A_1, \ldots, c_n \to A_n]$ 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 there 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* $\mathcal{BL}_{NL}(G)$*, is the set of well-bracketed lists of words generated by* G.
- *The* language of strings *corresponding to* G*, written* $\mathcal{L}_{NL}(G)$*, is the set of strings generated by* G.

Example 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:

$$John\ likes\ Mary \quad \in \mathcal{L}_{NL}(G_1)$$

$$((John\ likes)\ Mary) \in \mathcal{BL}_{NL}(G_1)$$

One interest of *NL* when compared to classical categorial grammars lies in its possibility to easily encode a restriction on the use of a basic category. For instance when we want to distinguish between a noun phrase and pronouns in subject position or object position, we can proceed as follows.

Example 2 *Let* $\Sigma_2 = \{John, Mary, likes, he, she, him, her\}$ *and let* $Pr_2 = \{S, N, X_1, X_2\}$. *We define the following rigid grammar:*

$$G_2 = \begin{cases} John, Mary \mapsto N \\ he, she \quad \mapsto N_1 \\ him, her \quad \mapsto N_2 \\ likes \quad \mapsto N_1 \backslash (S/N_2) \end{cases}$$

where $N_1 = X_1/(N\backslash X_1)$ and $N_2 = X_2/(N\backslash X_2)$.

We get: $((He\ likes)\ Mary) \in \mathcal{BL}_{NL}(G_2)$ *but:* $John\ likes\ she \notin \mathcal{L}_{NL}(G_2)$

Definition 6 (Grammar System) *A grammar system is a triple* $\langle \mathbb{G}, \mathbb{S}, \mathbb{L} \rangle$ *where:*

- $\mathbb{G}$ *is a "hypothesis space" (hereafter a set of grammars, for example categorial grammars on* Σ *)*
- $\mathbb{S}$ *is a "sample space" (for example* Σ^* *or structured sentences like* $\mathcal{T}_\Sigma$*)*
- $\mathbb{L}$ *is a function from* $\mathbb{G}$ *to subsets of* $\mathbb{S}$ *(for instance* $\mathcal{L}_{NL}$ *or* $\mathcal{BL}_{NL}$*)*

Let $\langle \mathbb{G}, \mathbb{S}, \mathbb{L} \rangle$ denote a grammar system. A learning algorithm ϕ on $\mathbb{G}$ is an algorithm that takes as input a finite list of $\mathbb{S}$ (a list of (structured) sentences) and returns an element of $\mathbb{G}$ (a grammar) as follows.

Definition 7 (Learning function) *In a grammar system* $\langle \mathbb{G}, \mathbb{S}, \mathbb{L} \rangle$ *a function* ϕ *is said to learn* $\mathbb{G}$ *in Gold's model iff for any* $G \in \mathbb{G}$ *and for any enumeration* $\langle e_i \rangle_{i \in \mathbb{N}}$ *of* $\mathbb{L}(G)$ *there exists* $n_0 \in \mathbb{N}$ *and a grammar* $G' \in \mathbb{G}$ *such that* $\mathbb{L}(G') = \mathbb{L}(G)$ *and* $\forall n \geq n_0, \phi(\langle e_0, \ldots, e_n \rangle) = G'$*. A class of grammars of* $\langle \mathbb{G}, \mathbb{S}, \mathbb{L} \rangle$ *is said learnable when there exists a computable function that learns* $\mathbb{G}$*. It is said unlearnable otherwise.*

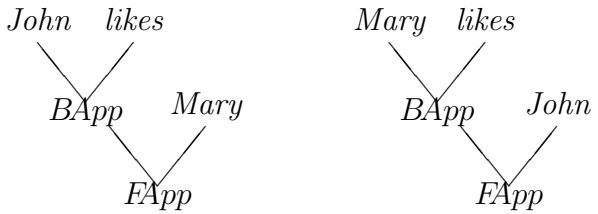
Definition 8 (Finite and infinite elasticity) *A class* $\mathcal{C}$ *of languages has* infinite elasticity *iff there exists an infinite sequence* $\langle e_i \rangle_{i \in \mathbb{N}}$ *of sentences and an infinite sequence* $\langle L_i \rangle_{i \in \mathbb{N}}$ *of languages in* $\mathcal{C}$ *such that* $\forall n \in \mathbb{N} \ : \ e_n \notin L_n$ *and* $\{e_0, \ldots, e_{n-1}\} \subseteq L_n$*. A class* $\mathcal{C}$ *of languages has* finite elasticity *iff it does not have infinite elasticity.*

Theorem 9 (Finite elasticity implies learnability [Wright [15]]) *If the languages corresponding to a class of grammars* $\mathbb{G}$ *of a grammar system* $\langle \mathbb{G}, \mathbb{S}, \mathbb{L} \rangle$ *have finite elasticity then* $\mathbb{G}$ *is* learnable *in Gold's model (provided that the class of grammars is recursively enumerable and the universal membership problem for this class is decidable).*

We now examplify categorial grammar inference in the simpler variant of AB-rigid grammars, with positive structured examples (called FA-structures in the AB framework); this structure represents the decompositions of sentences in sub-components distinguishing the functional parts from the argument parts; the internal nodes indicate the direction of application (forward by *FApp* or backward by *BApp*) [3]. W. Buszkowski and G. Penn have provided a unification algorithm on types to construct the most general lexicon generating the positive examples. This method has been used and extended in [4]. We give below an example that illustrates the algorithm.

[3] with a left application rule: $A/B, B \rightarrow A$ and a right application rule: $B, B\backslash A \rightarrow A$

Example 3 *We consider the following two structured examples :*



Argument types are first generated: the root is labelled S, distinct variables label the argument nodes ; here X_1 and X_2 for the first sentence, X_3 and X_4 for the second one. Next step is the computation of the types for functor nodes. The final step is the unification of the types associated to the same lexical entry. This is summarized in the following table :

John	X_2, X_3	$X_2 = X_3$	X_1
likes	$X_2 \setminus (S/X_1)$,	$X_2 = X_4$	
	$X_4 \setminus (S/X_3)$	$X_3 = X_1$	$X_1 \setminus (S/X_1)$
Mary	X_1, X_4	$X_1 = X_4$	X_1

3 GAB deductions and generalized FA-structures

3.1 FA structures over a set $\mathcal{E}$

We give a general definition of *FA* structures over a set $\mathcal{E}$, whereas in practice $\mathcal{E}$ is either an alphabet Σ or a set of types such as Tp.

Definition 10 (*FA* structures) *Let $\mathcal{E}$ be a set, a FA* structure *over $\mathcal{E}$ is a binary tree where each leaf is labelled by an element of $\mathcal{E}$ and each internal node is labelled by FApp (forward application) or BApp (backward application):*

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

Definition 11 (Tree yield) *The well-bracketed list of words obtained from a FA structure F over $\mathcal{E}$ by forgetting FApp and BApp labels is called the* tree yield *of F over $\mathcal{E}$ (notation $tree_{\mathcal{E}}(F)$).*

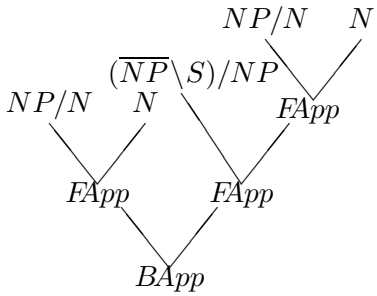
Definition 12 (*GAB* Deduction) Generalized *AB* deductions *(GAB deductions) over Tp are the deductions built from formulas on Tp (the base case) using the following conditional rules ($C \vdash_{NL} B$ must be valid in NL):*

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

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

GAB deductions can be seen as a generalization of *AB* deductions in the following sense: for *AB* application rules C and B must be the same formula.

Definition 13 (*FA* structure of a *GAB* deduction) *To each GAB deduction $\mathcal{P}$, we associate a FA structure, written $FA_{Tp}(\mathcal{P})$, such that each internal node corresponds to the application of a rule in $\mathcal{P}$ and is labelled by the name of this rule and where the leaves are the same as in $\mathcal{P}$.*



$$\dfrac{\dfrac{NP/N \quad N}{NP}\ FApp \quad \dfrac{(\overline{NP}\backslash S)/NP \quad \dfrac{NP/N \quad N}{NP}\ FApp}{\overline{NP}\backslash S}\ FApp}{S}\ BApp$$

Here, $\overline{NP} = X/(NP\backslash X)$ and thus $NP \vdash_{NL} \overline{NP}$

Definition 14 (*GAB* Deductions of $F \vdash_{GAB} A$ or $\Gamma \vdash_{GAB} A$)

- *For a FA structure over types $F \in FA_{Tp}$ and $A \in Tp$, we say that $\mathcal{P}$ is a*[4] *GAB deduction of $F \vdash_{GAB} A$ when A is the type of the conclusion of $\mathcal{P}$ and when $FA_{Tp}(\mathcal{P}) = F$.*
- *For a tree over types $\Gamma \in \mathcal{T}_{Tp}$ and $A \in Tp$, we say that $\mathcal{P}$ is a GAB deduction of $\Gamma \vdash_{GAB} A$ when A is the type of the conclusion of $\mathcal{P}$ and when $tree_{Tp}(FA_{Tp}(\mathcal{P})) = \Gamma$.*

[4] in fact, given a *FA* structure F, there is at most one *GAB* deduction $\mathcal{P}$ s.t. $FA_{Tp}(\mathcal{P}) = F$

Similarly to classical categorial grammars, we can associate to each categorial grammar a language of *FA* structures.

Definition 15 (*GAB* Languages) *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 there exists a GAB derivation of a FA structure $D \in FA_{Tp}$, $c_1, \ldots, c_n \in \Sigma$ and $A_1, \ldots, A_n \in Tp$ such that:*

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

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

Example 4 *If we take the categorial grammar that is defined in Example 2, we get:*

$$He\ likes\ Mary \qquad \in \mathcal{L}_{GAB}(G_2)$$

$$((He\ likes)\ Mary) \qquad \in \mathcal{BL}_{GAB}(G_2)$$

$$FApp(BApp(He, likes), Mary) \in \mathcal{FL}_{GAB}(G_2)$$

because we can build the following deduction (where $N_2 = X_2/(N \setminus X_2)$ that entails $N \vdash N_2$):

$$\dfrac{\dfrac{\overbrace{N_1}^{He} \quad \overbrace{N_1 \setminus (S/N_2)}^{likes}}{S/N_2}\, BApp \quad \overbrace{N}^{Mary}}{S}\, FApp$$

however:

$$Mary\ likes\ he \notin \mathcal{L}_{GAB}(G_2)$$

3.3 NL and GAB languages

In fact, there is a strong correspondence between *GAB* deductions and *NL* derivations. In particular with Theorem 16, we show that the respective string languages and binary tree languages are the same.

Theorem 16 *If A is an atomic formula, $\Gamma \vdash_{GAB} A$ iff $\Gamma \vdash_{NL} A$*

Corollary 17 *$\mathcal{BL}_{NL}(G) = \mathcal{BL}_{GAB}(G)$ and $\mathcal{L}_{NL}(G) = \mathcal{L}_{GAB}(G)$*

We write, for the rest of the paper, $\mathcal{FL}(G)$, $\mathcal{BL}(G)$ and $\mathcal{L}(G)$ in place of $\mathcal{FL}_{GAB}(G)$, $\mathcal{BL}_{GAB}(G) = \mathcal{BL}_{NL}(G)$ and $\mathcal{L}_{GAB}(G) = \mathcal{L}_{NL}(G)$.

Proof of $\Gamma \vdash_{GAB} A \Rightarrow \Gamma \vdash_{NL} A$ (A does not need to be atomic): 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. Suppose that the last rule of a *GAB* deduction corresponding to a *FA* structure $FApp(F_1, F_2)$ is:

$$\dfrac{\begin{array}{cc}\mathcal{P}_1 & \mathcal{P}_2 \\ \vdots & \vdots \\ A/B & C\end{array}}{A}\ FApp$$

We know that $C \vdash_{NL} B$ and we have two sub-deductions $\mathcal{P}_1$ and $\mathcal{P}_2$ that correspond to F_1 and F_2. The first one, $\mathcal{P}_1$, concludes with A/B and the second, $\mathcal{P}_2$, 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 $(/L)$ for $(A/B, B) \vdash A$ and two cuts, 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 $\Gamma \vdash_{NL} A \Rightarrow \Gamma \vdash_{GAB} A$ (A atomic): This property results from an alternative presentation of *NL* where contexts are in a limited form [19]:

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

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

Aarts and Trautwein in [19] 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 on the main branch of the derivation 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 $\mathcal{P}[B]$ 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:

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

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. ■

4 Infinite Elasticity Theorem

We prove, in this section that, the class of rigid (also k-valued for each k) *NL* languages of *FA* structures has infinite elasticity. Thus, the learning problem which is solved in section 5 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 $n \in \mathbb{N}$:

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

Construction 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 $\Sigma = \{a, b, c\}$. 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 $E_{n+1} = \mathit{FApp}(E_n, a)$. Finally the sequence of *FA* structures is defined by $< F_n = \mathit{FApp}(c, E_{n+1}) >$.

Theorem 18 *The class of rigid (also k-valued for each k) NL languages of FA structures over Σ has infinite elasticity.*

Proof of $\forall n \in \mathbb{N} : \{F_1, \ldots, F_n\} \subseteq \mathcal{FL}(G_{n+1})$: In fact we can first prove that $\forall n \in \mathbb{N}, D_n \vdash_{NL} D_{n+1}$. This is easy 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 $A \backslash A \vdash_{NL} (D_{n-1}/(D_{n-1} \backslash D_{n-1})) \backslash D_n = D_n \backslash D_n$. For the rest, we have to check that we can put these derivation on the unique *FA* structure on *Tp* that corresponds to F_n (G_n is rigid and there is no choice for the type of each element of Σ).

Proof of $F_n \notin \mathcal{FL}(G_n)$: 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 $\mathcal{FL}(G_n)$, since $G : b \mapsto D_n$ and $G : a \mapsto A \backslash A$, the deepest internal node for $n > 0$ is:

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

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

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

which is impossible because A is atomic and can not be the functor in a

functor-argument rule (this is the reason why a "?" appears on the deduction). ■

5 Learnability Theorems

Previous section shows that the class of rigid (also k-valued for each k) *NL* languages of *FA* structures over Σ has infinite elasticity. Thus, usual general properties given by learning theory do not apply here. To solve this problem, we define sub-classes of *NL* grammars (in terms of arity) and prove that the corresponding classes of languages are finite (and thus have finite elasticity). Then, we prove the learnability of the class of rigid (also k-valued for each k) *NL* languages of *FA* structures over Σ.

5.1 Type-arity, FA-arity and related subclasses

The arity of types is defined as usual on formulas, but in curryfied and non-commutative forms where $\backslash$ and / stand for right anf left implications. We recall the definition below in Definition 19. We then introduce another kind of arity, that is based directly on *FA* structures rather than on types. When we consider the lexicon of a grammar and the languages that are generated, these two measures are in fact connected as shown after. Whereas the *t-arity* on types can be computed directly on the lexicon, the *fa-arity* on *FA* structures can be computed directly on the data given in a learning process.

Definition 19 (Type-arity) *The arity of a type, written t-arity is :*

$$\begin{aligned} t\text{-}arity(A) &= 0 \ \ if \ \ A \in Pr \\ t\text{-}arity(A/B) &= t\text{-}arity(B\backslash A) = t\text{-}arity(A) + 1 \end{aligned}$$

The type-arity of a grammar G, written $t\text{-}arity(G)$ is the maximum arity of the types assigned by G.

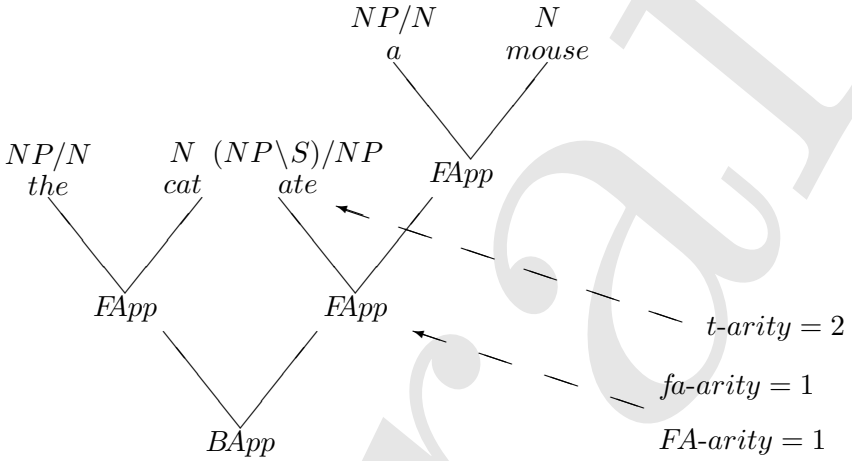
Definition 20 (*FA*-arity of *FA* structures over $\mathcal{E}$) *The FA-arity of a FA structure on $\mathcal{E}$ corresponds to the maximum number of arguments of each*

function in the structure. It is defined by:

$$
\begin{aligned}
&\textit{fa-arity}(A) && = 0 \quad \textit{if } A \in \mathcal{E} \\
&\textit{fa-arity}(\textit{FApp}(F_1, F_2)) && = \textit{fa-arity}(\textit{BApp}(F_2, F_1)) = \textit{fa-arity}(F_1) + 1 \\
&\textit{FA-arity}(A) && = 0 \quad \textit{if } A \in \mathcal{E} \\
&\textit{FA-arity}(\textit{FApp}(F_1, F_2)) && = \textit{FA-arity}(\textit{BApp}(F_2, F_1)) \\
& && = \textit{max}(\textit{fa-arity}(F_1) + 1, \textit{FA-arity}(F_1), \textit{FA-arity}(F_2))
\end{aligned}
$$

These definitions are extended to finite sets as the maximum computed for the given set (and possibly on infinite sets when such a maximum exists).

Note that the *FA*-arity on $\mathcal{FA}_{Tp}$ does not correspond to the usual arity of a functional expression, but is bounded by the maximum *t-arity* of the types on the leaves of the structure as shown below.



Property 21 *Let F denote a FA structure on Tp, we have*

$$\textit{for } F \in \mathcal{FA}_{Tp} \quad : \quad \textit{FA-arity}(F) \leq \textit{max}\{\textit{t-arity}(A_i) \mid A_i \in \textit{Tp leaf of } F\}$$

We can formulate a similar property for *FA* structures on Σ, provided we can relate the type and the word by a lexicon I :

$$\text{for } F \in \mathcal{FA}_{\Sigma} \quad : \quad \textit{FA-arity}(F) \leq \textit{max}\{\textit{t-arity}(A_i) \mid A_i \in I(c) \textit{ and } c \text{ leaf of } F\}$$

Proof : in fact we show a more detailed property : (1) for any subtree F of a *FA* structure on *Tp* :

$$(1)\ \textit{fa-arity}(F) + \textit{t-arity}(\textit{root}(F)) = \textit{t-arity}(\widehat{F})$$

where $\widehat{F}$ is the leaf type that labels the ultimate functor [5] of F and $\textit{root}(F)$

[5] More formally : $\widehat{A} = A$ if $A \in Tp; \widehat{\textit{FApp}(F_1, F_2)} = \widehat{F_1} = \widehat{\textit{BApp}(F_2, F_1)}$

is the type that labels the root in the GAB-derivation corresponding to F.

Property (1) entails the desired property since the *FA*-arity of a given *FA* structure is also the maximum *fa-arity* on the *FA* structures that occur in it.

We now prove (1) by induction on the structure F. In the base case where F is a leaf node (1) clearly holds because $\mathit{fa\text{-}arity}(F) = 0$ and $\mathit{root}(F) = \widehat{F}$. Let $F = \mathit{FApp}(F_1, F_2)$, we have $\widehat{F} = \widehat{F_1}$, $\mathit{fa\text{-}arity}(F) = 1 + \mathit{fa\text{-}arity}(F_1)$ and $\mathit{root}(F_1)$ must be $\mathit{root}(F)/B$ for some type B (such that rule *FApp* can apply in the derivation) therefore $\mathit{t\text{-}arity}(\mathit{root}(F_1)) = 1 + \mathit{t\text{-}arity}(\mathit{root}(F))$, hence by induction (1) holds for F. The case $F = \mathit{BApp}(F_1, F_2)$ is similar ■

Property 21 allows to define the *FA*-arity of a grammar as follows.

Definition 22 (*FA*-arity of a grammar) *For a categorial grammar G or the corresponding language $\mathcal{FL}(G)$, we define their FA-arity as the maximum FA-arity of the FA structures of $\mathcal{FL}(G)$:*

$$\mathit{FA\text{-}arity}(G) = \mathit{FA\text{-}arity}(\mathcal{FL}(G)) = \mathit{max}\{\mathit{FA\text{-}arity}(F) \mid F \in \mathcal{FL}(G)\}$$

This maximum exists for a k-valued categorial grammar because: (i) the *FA*-arity of a set of *FA* structures on Σ is the same as the *FA*-arity of the *FA* structures on *Tp* that generate these structures; (ii) from Property 21, each *FA* structure on *Tp* has a *FA*-arity that is bounded by the maximum *t-arity* of the types on the leaves of the structure; (iii) the *t-arity* of the types on the leaves of the structure are bounded by the maximum arity of the types that appear in the grammar (in finite number).

Definition 23 (*FA*-arity bounded subclasses) *We consider the following subclasses of NL languages and grammars (over Σ):*

- *the class of NL grammars whose FA-arity is bounded by n is noted $\mathcal{CG}^{(\mathit{FA\text{-}arity} \leq n)}$; the corresponding classes of languages of FA structures and of strings are written $\mathcal{CFL}^{(\mathit{FA\text{-}arity} \leq n)}$ and $\mathcal{CL}^{(\mathit{FA\text{-}arity} \leq n)}$;*
- *the class of NL k-valued categorial grammars, whose FA-arity is at most n, is written $\mathcal{CG}_k^{(\mathit{FA\text{-}arity} \leq n)}$; we write $\mathcal{CFL}_k^{(\mathit{FA\text{-}arity} \leq n)}$ and $\mathcal{CL}_k^{(\mathit{FA\text{-}arity} \leq n)}$ for the classes of NL languages of FA structures and of strings generated by these grammars.*

5.2 Each class of rigid arity-bounded languages $\mathcal{CFL}_1^{(\mathit{FA\text{-}arity} \leq n)}$ is finite.

We first give some technical definitions and properties related to GAB-deductions.

Definition 24 *The main subtype of depth n for a given type is defined by*

$$main_0(A) = A$$
$$main_n(A) = \text{ is undefined} \qquad \text{if } A \text{ is atomic and } n > 0$$
$$main_n(A/B) = main_n(B\backslash A) = main_{n-1}(A) \qquad \text{if } n > 0$$

Remark. *if A/B (or $B\backslash A$) is the main subtype of depth k for a formula C then A is the main subtype of depth $k+1$ for this formula C.*

Theorem 25 (Types in *GAB*-deductions) *The types that appear in a GAB-deduction $\mathcal{P}$ are main subtypes of the leaves of $\mathcal{P}$ with a depth less or equal to the FA-arity of the FA structure associated to $\mathcal{P}$, that is they belong to :*

$$\{main_n(A_i) \mid A_i \text{ leaf of } \mathcal{P} \wedge n \leq \textit{FA-arity}(FA_{Tp}(\mathcal{P})) \wedge main_n(A_i) \text{ is defined } \}$$

Proof : by induction on the *GAB*-deduction $\mathcal{P}$, we also show that (o) the conclusion of $\mathcal{P}$ is a main subtype of a leaf in $\mathcal{P}$ of depth $\textit{fa-arity}(FA_{Tp}(\mathcal{P}))$.
- The base case when $\mathcal{P}$ is a formula on Tp is obvious since a formula is also its main subtype of depth 0.
- Suppose $\mathcal{P}$ ends with the application of *FApp* on two sub-deductions $\mathcal{P}_1$ of A/B and $\mathcal{P}_2$ of C, such that $C \vdash_{NL} B$; let $F_1 = FA_{Tp}(\mathcal{P}_1)$ and $F_2 = FA_{Tp}(\mathcal{P}_2)$, we have $FA_{Tp}(\mathcal{P}) = \textit{FApp}(F_1, F_2)$.

$$\dfrac{\begin{array}{cc}\mathcal{P}_1 & \mathcal{P}_2 \\ \vdots & \vdots \\ A/B & C\end{array}}{A}\ \textit{FApp}$$

* We first get (o) since A/B is a main subtype of a leaf in $\mathcal{P}_1$ with a depth equal to $\textit{fa-arity}(F_1)$, which implies that A must be a main subtype of this leaf with a depth equal to $1 + \textit{fa-arity}(F_1) = \textit{fa-arity}(\textit{FApp}(F_1, F_2)) = \textit{fa-arity}(FA_{Tp}(\mathcal{P}))$.

* Then by induction hypothesis: the types that appear in $\mathcal{P}_1$ are main subtypes of the leaves of $\mathcal{P}_1$ with a depth $\leq \textit{FA-arity}(F_1)$ and similarly for $\mathcal{P}_2$ with a depth $\leq \textit{FA-arity}(F_2)$. We conclude the *FApp* case using:
$\textit{FA-arity}(\textit{FApp}(F_1, F_2)) = max(\textit{fa-arity}(F_1) + 1, \textit{FA-arity}(F_1), \textit{FA-arity}(F_2))$
- The case when $\mathcal{P}$ ends with the application of *BApp* is similar. ■

Given a grammar G, the main subtypes of the types assigned by G are used to define tables that are intended to capture its *NL*-derivation possibilities.

Definition 26 (Deduction table) *Let $G = (\Sigma, I, S)$ be a rigid categorial grammar, its deduction table of depth n is:*

$$
\begin{aligned}
&\mathit{Tab(G;n)}[\langle a, i\rangle, \langle b, j\rangle] \in \{\bot, \mathit{FApp}, \mathit{BApp}\}\\
&= \mathit{FApp} \quad \mathit{iff}\ \exists A, B, C, D : \begin{cases} I(a) = \{A\}, I(b) = \{B\}, \\ main_j(B) = C/D, main_i(A) \vdash_{NL} D \end{cases}\\
&= \mathit{BApp} \quad \mathit{iff}\ \exists A, B, C, D : \begin{cases} I(a) = \{A\}, I(b) = \{B\}, \\ main_j(B) = D\backslash C, main_i(A) \vdash_{NL} D \end{cases}\\
&= \bot \qquad \mathit{elsewhere}
\end{aligned}
$$

where $a, b \in \Sigma$ and $0 \leq i, j \leq n$.

Theorem 27 *Two grammars with the same deduction tables for each depth, (or equivalently for their maximum FA-arity) generate the same FA-structure languages : let $n \geq 0$, $G = (\Sigma, I, S)$, $G' = (\Sigma, I', S)$ with $G, G' \in \mathcal{CG}_1^{(FA\text{-}arity \leq n)}$: if Tab(G;n)=Tab(G';n) then $\mathcal{FL}(G) = \mathcal{FL}(G')$*

Proof : by induction on a *GAB*-deduction $\mathcal{P}$ for $\Gamma \vdash C$, where Γ is a tree of only main subtypes of G of depth $\leq$ *FA-arity*(G), we show that a similar *GAB*-deduction $\mathcal{P}'$ is obtained for the types of G' by replacing each main subtype for G in $\mathcal{P}$ by the corresponding main subtype for G' (that is each occurrence $main_i(A)$ where $I(a) = \{A\}$ for $a \in \Sigma$ is replaced by $main_i(A')$ where $I'(a) = \{A'\}$) ■

Theorem 28 *For each n, $\mathcal{CFL}_1^{(FA\text{-}arity \leq n)}$ is finite.* [6]

Proof : let us fix $n \geq 0$; the number of deduction tables of depth n is finite (since Σ is finite and fixed for the class). From Theorem 27, all grammars in $\mathcal{CG}_1^{(FA\text{-}arity \leq n)}$ with the same deduction tables have the same langage, therefore $\mathcal{CFL}_1^{(FA\text{-}arity \leq n)}$ is finite ■

This property is essential in this work. 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.

5.3 FA-arity bounded k-valued NL languages are learnable from strings

We get as a corollary that *FA*-arity bounded k-valued *NL* languages are learnable from strings as explained below.

[6] each class corresponds to a given finite alphabet often left implicit

We have seen that the class of *FA*-arity bounded k-valued *NL* languages of *FA* structures is finite. We can deduce that the class of *FA*-arity bounded k-valued *NL* languages of strings is finite and thus learnable from positive examples.

Corollary 29 *$\mathcal{CL}_k^{(FA\text{-}arity\leq n)}$ is finite for each k and n and thus the corresponding classes of grammars $\mathcal{CG}_k^{(FA\text{-}arity\leq n)}$ are learnable from strings.*

A similar corollary holds for well-bracketed strings using similar arguments

5.4 k-valued NL languages are learnable from FA structures

We now adress languages of *FA* structures that are not necessarily arity bounded and we show in this section a more general result.

Property 30 *$\mathcal{CFL}_k$ has infinite elasticity for each k whereas the corresponding classes of grammars $\mathcal{CG}_k$ are learnable from FA-structures.*

Proof: Because, for each n and k, the class $\mathcal{CFL}_k^{(FA\text{-}arity\leq n)}$ 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):

(1) Compute the maximum *FA*-arity r of the l input *FA* structures.
(2) 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 for a language L that corresponds to a k-valued *NL* grammar, there exists at least one *FA* structure F such that $FA\text{-}arity(F) = FA\text{-}arity(L)$. Thus, for every enumeration on the *FA* structure of L, there exists an integer r_1 such that for every $l \geq r_1$, the number r computed by ϕ_k is $FA\text{-}arity(L)$. From this integer, ϕ_k applies the proper algorithm $\phi_k^{FA\text{-}arity(L)}$ that converges to L.

6 Conclusion

Learnability from functor-argument structures. We have shown first in the paper how we can define languages of functor-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 functor-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 functor-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 strings. In fact, as shown in [24,25] by limit points, each class of k-valued non-associative Lambek grammar is unlearnable from strings and even from well-bracketed strings. This result expresses the need for further restrictions or for an adequate structure on strings like the notion of FA-arity bounded language.

Learnable subclasses. Another recent work in [26] applies in particular to unidirectional k-valued NL-grammars and yields their learnability from strings. [27] deals with k-valued NL-grammars with types of t-arity at most n and their learnability from strings only, these latter classes being based on a bound on types rather than on the FA-structures generated. Here, we have shown that FA-arity bounded k-valued NL languages are learnable from strings. In fact, these results of [27] can be recovered from results in this paper using Property 21.

We now give a summary of some results on the learnability of NL classes of grammars from (structured) examples (* is proved in the article) :

Restriction\Structure	strings	Well bracketed strings	Generalized *FA*
all	no	no	no
k-valued	no [24]	no [25]	yes*
k-valued and *t-arity* bounded	yes [27]	yes corollary of [27]	yes corollary of [27]
k-valued and *FA-arity* bounded	yes*	yes*	yes*

References

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

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

[3] T. Shinohara, Inductive inference from positive data is powerful, in: The 1990 Workshop on Computational Learning Theory, Morgan Kaufmann, San Mateo, California, 1990, pp. 97–110.

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

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

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

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

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

[9] R. Bonato, C. Retoré, Learning rigid Lambek grammars and minimalist grammars from structured sentences, Third workshop on Learning Language in Logic, Strasbourg (september 2001) 23–34.

[10] A. Foret, Y. Le Nir, Lambek rigid grammars are not learnable from strings, in: COLING'2002, 19th International Conference on Computational Linguistics, Taipei, Taiwan, 2002, pp. 274–279.

[11] A. Foret, Y. Le Nir, On limit points for some variants of rigid Lambek grammars, in: ICGI'2002, the 6th International Colloquium on Grammatical Inference, no. 2484 in Lecture Notes in Artificial Intelligence, Springer-Verlag, 2002, pp. 106–119.

[12] C. C. Florêncio, A limit point for rigid associative-commutative Lambek grammars, in: A. Copestake, J. H. c (Eds.), Proceedings of EACL2003, Tenth Conference of the European Chapter of the Association for Computational Linguistics, 2003, pp. 72–82.

[13] C. C. Florêncio, Learning categorial grammars, Ph.D. thesis, Utrecht Institute of Linguistics OTS, Utrecht University (Nov. 2003).

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

[15] K. Wright, Identifications of unions of languages drawn from an identifiable class, in: The 1989 Workshop on Computational Learning Theory, Morgan Kaufmann, San Mateo, Calif., 1989, pp. 328–333.

[16] T. Motoki, T. Shinohara, K. Wright, The correct definition of finite elasticity: Corrigendum to identification of unions, in: The fourth Annual Workshop on Computational Learning Theory, Morgan Kaufmann, San Mateo, Calif., 1991, p. 375.

[17] D. Angluin, Inference of reversible languages, Journal of the ACM (JACM) 29 (3) (1982) 741–765.

[18] M. Kandulski, The non-associative Lambek calculus, in: W. M. W. Buszkowski, J. V. Bentem (Eds.), Categorial Grammar, Benjamins, Amsterdam, 1988, pp. 141–152.

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

[20] W. Buszkowski, Mathematical linguistics and proof theory, in: van Benthem and ter Meulen [28], Ch. 12, pp. 683–736.

[21] M. Moortgat, Categorial type logic, in: van Benthem and ter Meulen [28], Ch. 2, pp. 93–177.

[22] P. de Groote, Non-associative Lambek calculus in polynomial time, in: 8^th Workshop on theorem proving with analytic tableaux and related methods, no. 1617 in Lecture Notes in Artificial Intelligence, Springer-Verlag, 1999, pp. 128–139.

[23] P. de Groote, F. Lamarche, Classical non-associative Lambek calculus, Studia Logica 71(3) (2002) 355–388.

[24] D. Béchet, A. Foret, k-valued non-associative Lambek categorial grammars are not learnable from strings, in: ACL (Ed.), Proccedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003), 2003, pp. 351–358.

[25] D. Béchet, A. Foret, On intermediate structures for non-associative Lambek grammars and learnability, in: Proccedings of the 7th conference on Categorial Grammars (CG2004), Montpellier, France, 2004, pp. 180–194.

[26] Y. Le Nir, Structures des analyses syntaxiques catégorielles. application à l'inférence grammaticale, Ph.D. thesis, Rennes 1 University (Dec. 2003).

[27] D. Béchet, A. Foret, Apprentissage des grammaires de Lambek rigides et d'arité bornée pour le traitement automatique des langues, in: Actes de la Conférence d'APprentissage 2003 (CAP'2003), 2003, pp. 155–167.

[28] J. van Benthem, A. ter Meulen (Eds.), Handbook of Logic and Language, North-Holland Elsevier, Amsterdam, 1997.