Noname manuscript No.
(will be inserted by the editor)

A Faithful Representation of Non-Associative Lambek Grammars in Abstract Categorial Grammars

Christian Retoré, Sylvain Salvati

Abstract This paper solves a natural but still open question: can Abstract Categorial Grammars (ACGs) respresent usual categorial grammars? Despite their name and their claim to be a unifying framework, up to now there was no faithful representation of usual categorial grammars in ACGs. This paper shows that Non-Associative Lambek grammars as well as their derivations can be defined using ACGs of order two. To conclude, the outcomes of such a representation are discussed.

1 Introduction

Abstract Categorial Grammars (ACGs) have been defined by de Groote [1] as a uniform way to define formal grammars in simply typed, even linear, λ-calculus, that is the smallest functional system.

These grammars share with categorial grammars the definition of the grammar as a lexicon, and the use of λ-terms as objects and derivations. In particular, they use a systematic view of strings and trees as user defined types with constructors *à la* ML.

Nevertheless, they also differ from standard categorial grammars, taking apart the hierarchical structure and the mappings to word sequences or to semantic representations. This two level approach can be connected to a broader contemporary trend, including minimalist grammars [2], lambda grammars [3], and the two step approach [4].

The expressivity of ACGs has already been explored in [5], but one may wonder whether one can *represent faithfully* any categorial formalism. By *represent faithfully* we mean that not only the string language is recognized, but also the proof structures, which are trees that can be represented within simply typed lambda calculus as ACGs usually do.

To do so, we encode the natural deduction trees corresponding to grammatical analyses of a Non-Associative Lambek Grammar (NLCG) as terms built on a second

Christian Retoré, Sylvain Salvati INRIA Bordeaux Sud-Ouest, LaBRI (Université de Bordeaux et C.N.R.S.),
351, cours de la Libération, F-33405 Talence cedex France
Work supported by ANR research grant PRELUDE E-mail: christian.retore@labri.fr, sylvain.salvati@labri.fr

order signature (the type of any argument always is a base type and never a functional one) and the main result of this paper can be stated as follows:

Theorem 1 *Given a non associative Lambek categorial grammar G defined by a lexicon there exists a second order ACG whose object language precisely is the set of the derivations of G viewed as lambda terms.*

From this one easily obtains that languages generated by NLCGs are context free (see section 6.1) and a parsing complexity in $O(m^2n^3)$ where m is the number of symbols in the NLCG lexicon and n the number of words, thus competing with the most recent results in this area [6] — recall that when parsing in NLCG the correct binary tree structure on the words is not given but has to be found by the parsing strategy.

Another interest of such a representation is that it allows one to import into ACGs the analyses of grammatical phenomena implemented in categorial grammars, which are the core of Multimodal Categorial Grammars and which encompass a number of linguistic descriptions [7] as well as practical large scale implementations [8].

An expected step is to look within the ACG setting at the corner stone of categorial grammar, namely their easy syntax/semantics interface. It consists in mapping the λ-terms provided by our translation to some meaning representation, hence it is at least as simple as in the plain categorial setting, and perfectly suits in with the ACG framework.

The paper is organized as follows. We first define the simply typed λ-calculus and Abstract Categorial Grammars in section 2. In section 3, we present the non-associative Lambek calculus and NL-grammars. Section 4 describes the embedding we propose. Section 5 sketches the conversion of a tiny NL grammar. Section 6 explains the outcomes of our result. And finally section 7 offers an outline of future work.

2 λ-calculus and Abstract Categorial Grammars

2.1 Reminder on simply typed λ-calculus

We consider the usual simply typed λ-calculus, with a presentation *à la* Church, that is variables are provided with their type, and, consequently, typable terms have a unique type. Given a set of base types $\mathcal{A}$, the types over $\mathcal{A}$ is defined as $\mathcal{T}_\mathcal{A} ::= \mathcal{A} \mid \mathcal{T}_\mathcal{A} \rightarrow \mathcal{T}_\mathcal{A}$ We assume that $\rightarrow$ associates to the right and therefore that $\alpha_1 \rightarrow \cdots \alpha_n \rightarrow \beta$ stands for type $(\alpha_1 \rightarrow \cdots (\alpha_n \rightarrow \beta) \cdots)$.

Regarding λ-terms we write $\lambda x_1^{\alpha_1} \ldots x_n^{\alpha_n}.M$ for $\lambda x_1^{\alpha_1}. \ldots \lambda x_n^{\alpha_n}.M$ and $M_0 M_1 \ldots M_n$ for $(\ldots(M_0 M_1)\ldots M_n)$. Moreover we implicitly assume that when $n = 0$ $\lambda x_1^{\alpha_1} \ldots x_n^{\alpha_n}.M$ denotes M and that $M_0 M_1 \ldots M_n$ denotes M_0. We take for granted that the notion of free variables ($FV(M)$ denotes the set of free variables of M), α-conversion, β-conversion, normal form are known (see [9] and [10]). The *head of the term* $\lambda x_1 \ldots x_n.h M_1 \ldots M_p$ is said to be h whenever h is either a variable or a constant. In general, when we refer to the head of a term, we refer to the head of its normal form.

Contexts are λ-terms with a hole. Given a set of lambda terms Λ, contexts are defined according to the following grammar:

$$\mathbf{C} ::= [] \mid (\mathbf{C}\Lambda) \mid (\Lambda\mathbf{C}) \mid \lambda x^{\alpha}.\mathbf{C}$$

We write $C[N]$ (*resp.* $C[C'[]]$) the term obtained by inserting the term N (*resp.* the context $C'[]$) in place of the hole in $C[]$. Note that inserting a term or a context in another context can bind variables. For example when $C[] = \lambda x.[]$ and $N = x$, we have $C[N] = \lambda x.x$.

A term M is said to be in *long form* whenever, for any context $C[]$ and term N of type $\alpha \rightarrow \beta$ such that $C[N] = M$, it verifies one of the following property:

1. N is of the form $\lambda x^{\alpha}.N'$,
2. $C[]$ is of the form $C'[([]N')]$.

The set of terms in long form is closed under β-reduction [10].

A term M is said to be *linear* if $M = x^{\alpha}$; if $M = c$; if $M = (MN)$, M and N are linear, and $FV(M) \cap FV(N) = \emptyset$; or if $M = \lambda x^{\beta}.M'$, M' is linear and $x^{\beta} \in FV(M')$.

2.2 Higher order signatures and abstract categorial grammars

A *higher order signature* Σ is a triple $(\mathcal{A}, \mathcal{C}, \tau)$ where

- $\mathcal{A}$ is a finite set of *atomic types*,
- $\mathcal{C}$ is a finite set of *constants*
- τ is a *typing function*, *i.e.* a function from $\mathcal{C}$ to $\mathcal{T}_{\mathcal{A}}$

Unless otherwise stated , we assume that the triple defining the signature Σ_i is $(\mathcal{A}_i, \mathcal{C}_i, \tau_i)$.

Let $\Lambda_{\Sigma}^{\alpha}$ with $\alpha \in \mathcal{T}_{\mathcal{A}}$ stand for the set of terms of type α. The usual definition of typed terms using a set of constants as above may be stated as follows:

1. $x^{\alpha} \in \Lambda_{\Sigma}^{\alpha}$ (x^{α} is a λ-*variable*),
2. $c \in \Lambda_{\Sigma}^{\tau(c)}$,
3. $M_1 \in \Lambda_{\Sigma}^{\beta\rightarrow\alpha}$ and $M_2 \in \Lambda_{\Sigma}^{\beta}$ imply that $(M_1 M_2) \in \Lambda_{\Sigma}^{\alpha}$, and
4. $\lambda x^{\beta}.M \in \Lambda_{\Sigma}^{\beta\rightarrow\alpha}$ whenever $M \in \Lambda_{\Sigma}^{\beta}$.

Given a set of terminals W, the *string signature* over W is the higher order signature $\Sigma_W : (\{*\}, W, \tau)$ where $\tau(c) = * \rightarrow *$. The finite sequences of terminals are represented as closed terms (*i.e.* terms with no free variables) of $\Lambda_{\Sigma}^{*\rightarrow*}$: a string $c_1 \cdots c_n$ is represented by the term $\lambda x^*.c_1(\ldots(c_n x^*)\ldots)$ denoted by $/c_1 \ldots c_n/$. Regarding strings in λ-calculus, we recall that:

- The concatenation operation (associative) can be represented by $+ = \lambda u^{*\rightarrow*}\lambda v^{*\rightarrow*}\lambda x^*.(u(vx))$ for $u, v \in \Lambda_{\Sigma}^{*\rightarrow*}$. Clearly, $((+u)v)$ reduces to $/uv/$.
- We also use operator like notation: $u +_{\backslash} v$ and $u +_{/} v$ are short hands, respectively, for $u +_{\backslash} v = ((+u)v)$ and $u +_{/} v = ((+v)u)$.
- The empty string, which is the unit of $+$, is $\lambda x^*.x^*$.

A *homomorphism* between the signatures Σ_1 and Σ_2 is a pair (g, h) such that g maps $\mathcal{T}_{\mathcal{A}_1}$ to $\mathcal{T}_{\mathcal{A}_2}$, h maps Λ_{Σ_1} to Λ_{Σ_2} and verify the following properties:

1. $g(\alpha \rightarrow \beta) = g(\alpha) \rightarrow g(\beta)$,
2. $h(x^{\alpha}) = x^{g(\alpha)}$,
3. $h(c)$ is a closed term (*i.e.* $FV(h(c)) = \emptyset$) of $\Lambda_{\Sigma_2}^{g(\tau(c))}$,
4. $h(M_1 M_2) = h(M_1)h(M_2)$ and
5. $h(\lambda x^{\beta}.M) = \lambda x^{g(\beta)}.h(M)$.

A homomorphism is said to be *linear* whenever closed linear terms are mapped onto constants. We write $\mathcal{H}(\alpha)$ and $\mathcal{H}(M)$ respectively instead of $g(\alpha)$ and of $h(M)$ for a given homomorphism $\mathcal{H} = (g, h)$. Note that if $\mathcal{H}$ is a homomorphism from Σ_1 to Σ_2 and $M \in \Lambda^{\alpha}_{\Sigma_1}$ then $\mathcal{H}(M) \in \Lambda^{\mathcal{H}(\alpha)}_{\Sigma_2}$, note furthermore that if $\mathcal{H}$ and M are both linear then so is $\mathcal{H}(M)$.

An *Abstract Categorial Grammar* [1] (ACG) is a 4-tuple $(\Sigma_1, \Sigma_2, \mathcal{L}, S)$ where Σ_1 is the *abstract vocabulary*, Σ_2 is the *object vocabulary*, $\mathcal{L}$ is linear homomorphism, the *lexicon*, and S is an element of $\mathcal{A}_1$, the *distinguished type*. A *non-linear Abstract Categorial Grammar* is an ACG whose lexicon may be an arbitrary homomorphism. An ACG $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{L}, S)$ (*resp.* a non-linear ACG) defines two languages: the *abstract language*: $\mathcal{A}(\mathcal{G}) = \{M \in \Lambda^{S}_{\Sigma_1} | M \text{ is closed and linear}\}$, the *object language*: $\mathcal{O}(\mathcal{G}) = \{M | \exists N \in \mathcal{A}(\mathcal{G}) \wedge M =_{\beta\eta} \mathcal{L}(N)\}$.

Note that given a homomorphism $\mathcal{L}'$ from Σ_2 to Σ_3, and an ACG $(\Sigma_1, \Sigma_2, \mathcal{L}, S)$, then $(\Sigma_1, \Sigma_3, \mathcal{L}' \circ \mathcal{L}, S)$ is an ACG when $\mathcal{L}'$ is linear and otherwise it is a non-linear ACG.

3 The non-associative Lambek calculus

In this paper we deal with the non-associative Lambek calculus without product known as NL [11]. Given a finite set Cat, called the *basic set of categories*, the set of categories built on Cat, NL_{Cat}, is the smallest set containing Cat and having the property that if $A, B \in NL_{Cat}$ then $\backslash(B, A)$ and $/(B, A)$ are in NL_{Cat}.[1] Categories will be represented by the roman uppercase letters A, B, C, D, E and F (possibly with some indices). A *hypothesis base*, or simply a *base*, is a binary tree whose leaves are elements of NL_{Cat}; any element of NL_{Cat} can be considered as a base, and given two bases Γ and Δ, we write (Γ, Δ) the new base obtained from them. Given a base Γ, we write $\overline{\Gamma}$ the list of the leaves of Γ taken from left to right.

The non-associative Lambek calculus derives judgements of the form $\Gamma \vdash A$ where Γ is a base and A belongs to NL_{Cat}. These judgements are obtained with the following rules:

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

$$\frac{\Gamma \vdash \backslash(B, A) \quad \Delta \vdash B}{(\Delta, \Gamma) \vdash A}\,/\ E \qquad \frac{\Gamma \vdash /(B, A) \quad \Delta \vdash B}{(\Gamma, \Delta) \vdash A}\,\backslash\ E$$

If we define the functions $f_{\backslash}$ and $f_{/}$ as binary operators over bases such that $f_{\backslash}(\Gamma, \Delta) = (\Delta, \Gamma)$ and $f_{/}(\Gamma, \Delta) = (\Gamma, \Delta)$, taking advantage of the symmetry of the calculus, we may write the introduction and the elimination rules as rules parametrized by the operator $c \in \{\backslash; /\}$:

$$\frac{f_c(\Gamma, B) \vdash A}{\Gamma \vdash c(B, A)}\,c\ I \qquad \frac{\Gamma \vdash c(B, A) \quad \Delta \vdash B}{f_c(\Gamma, \Delta) \vdash A}\,c\ E$$

This remark will simplify the notation of derivations in the following of the paper. We will also write $\widetilde{c}$ for $\backslash$ if $c = /$ and for $/$ when $c = \backslash$.

[1] In the literature $\backslash(B, A)$ is rather written as $B\backslash A$ and $/(B, A)$, as A/B. We adopt these non-conventional notations so as to facilitate the encoding of NL-grammars in ACGs.

A non-associative Lambek grammar (NL-grammar) is a 4-tuple $G = (W, Cat, \chi, S)$ where W is a set of *words*, Cat is a set of *atomic categories*, χ is a function from W to finite subsets of NL_{Cat}, and $S \in Cat$. In order to account for grammaticality, we now allow a new kind of hypothesis in hypothesis bases. These new hypotheses are pairs $\langle w, A\rangle$ such that $A \in \chi(w)$. They represent the use of a *lexical entry* of the NL-grammar in a derivation. A base is said to be *lexical* if all its leaves are such pairs. We also add the following rule:

$$\frac{A \in \chi(w)}{\langle w, A\rangle \vdash A}\, Lex$$

The only distinction between this new kind of hypothesis and the usual one is that they cannot be discharged by using the rules $\backslash\ I$ and $/\ I$.

A sentence $w_1 \dots w_n$ of W^* is said to be *accepted* by G if there is a lexical base Γ such that $\overline{\Gamma} = [\langle w_1, A_1\rangle; \dots; \langle w_n, A_n\rangle]$ and $\Gamma \vdash S$ is derivable. Such a derivation is called a *grammatical analysis* of $w_1 \dots w_n$. The language defined by G is the set of sentences in W^* that it accepts.

The Curry-Howard correspondence is a mapping from derivations in intuitionistic logic to typed λ-terms. Since non associative Lambek calculus is a subcalculus of intuitionistic logic (linear, non commutative, non associative), one can easily obtain a λ-term, even a linear one, out of an NL proof. In this later case, however, our mapping is not an isomorphism, since the simply typed λ-calculus is not enough to recover the NL proof: for instance the directionality is lost. Nevertheless, the essential features of the proofs are preserved in such a representation. Indeed, these are the terms that are used to obtain the semantics of a derivation, and we will see that the sentence that this proof analyses can also be recovered from that λ-term.

An NL type A is mapped to a simple type $|A|$ as usual with $|c(B, A)| = |B| \to |A|$.

The following system shows how to transform a derivation in NL into a linear λ-term built on the signature Σ_G with the same atomic types as G (*i.e.* $\mathcal{A}_G = Cat$), with as constants all the lexical items $\langle w, A\rangle$ with type $|A|$ *i.e.* $\mathcal{C}_G = \{\langle w, A\rangle | w \in W \wedge A \in \chi(w)\}$ and $\tau_G(\langle w, A\rangle) = |A|$.

In the following rules, we show how one can compute from the NL proof the associated λ-term; an NL formula A on the left handside is labelled with a λ-variable of type $|A|$, while the NL formula on the right handside is labelled with a λ-term of type $|A|$.

$$\frac{A \in \chi(w)}{\langle w, A\rangle \vdash \langle w, A\rangle : A} \qquad \frac{}{x^{|A|} : A \vdash x^{|A|} : A}\, Ax.$$

$$\frac{f_c(\Gamma, x^{|B|} : B) \vdash M : A}{\Gamma \vdash \lambda x^{|A|}.M : c(B, A)}\, c\, I \quad \frac{\Gamma \vdash M_1 : c(B, A) \ \ \Delta \vdash M_2 : B}{f_c(\Gamma, \Delta) \vdash (M_1 M_2) : A}\, c\, E$$

This correspondence allows us to talk about derivations in normal form, derivations in long forms, and also about the head of a derivation as induced from the λ-calculus. In particular the derivations that are in normal form enjoy the so-called subformula property; for grammatical derivation, this property implies that the formulae used in the grammatical analyses in long normal form of G can only be S or some subformula of $A \in \bigcup_{w \in W} \chi(w)$. We adopt the notation $\mathcal{F}_G$ for this set of formulae for an NL-grammar G.

3.1 Turning an NL grammatical derivation into the analysed string

One can devise a linear homomorphism $\mathcal{Y}$ which maps every atomic category to the type $* \rightarrow *$ so that every term M representing a grammatical analysis of $w_1 \ldots w_n$ is mapped by $\mathcal{Y}$ to the string $/w_1 \ldots w_n/$ in Σ_W as defined in the first section.

It should be observed that the analysed string is reconstructed from the λ-term corresponding to the proof (which forgot the directionality) and from the constants associated with lexical entries.

Let us first define $\varphi : NL_{Cat} \mapsto \Lambda_{\Sigma_W}$ and $\psi : \Lambda_{\Sigma_W} \times NL_{Cat} \mapsto \Lambda_{\Sigma_W}$:

1. $\varphi(A) = \lambda x.x$ if A is an atomic category,
2. $\varphi(c_1(B_1, \ldots c_n(B_n, A))) = \lambda x_1 \ldots x_n.\psi(x_1, B_1) + \ldots + \psi(x_n, B_n)$, with $c \in \{\backslash; /\}$,
3. $\psi(M, B) = M$ if B is an atomic category,
4. $\psi(M, c(B, A)) = \psi(M\varphi(B), A)$ with $c \in \{\backslash; /\}$

The important property which arises from this definition is that $\psi(\varphi(B), B) = \lambda x^*.x^*$.

From ϕ and ψ one can define $\rho : \mathbf{C} \times \Lambda_{\Sigma_W} \times NL_{Cat} \rightarrow \Lambda_{\Sigma_W}$:

1. $\rho(C[], M, A) = C[M]$ when A is an atomic category,
2. $\rho(C[], M, c(B, A)) = \rho(C[\lambda x.[]], \psi(x, B) +_c M, B)$ with $c \in \{\backslash; /\}$

Now one defines $\mathcal{Y}$ by $\mathcal{Y}(\langle w, A\rangle) = \rho([], w, A)$ and a mere computation shows that:

Proposition 1 *The Λ_{Σ_W} term $\mathcal{Y}(t)$ reduces to $/w_1 \cdots w_n/$ whenever t is the Λ_{Σ_G} term representing an NL analysis of $w_1 \cdots w_n$.*

Furthermore, the meaning representation of the sentence, for Montague-like semantics, can be obtained from those λ-terms, by means of a non-linear lexicon $\mathcal{L}_{Sem}$.

4 Coding NL-grammars into ACGs

In this section we define an ACG $\mathcal{G}_G = (\Sigma_{\delta G}, \Sigma_G, \mathcal{L}_G, [\bullet \vdash S])$ (Σ_G is as defined in the previous section) so that $\mathcal{O}(\mathcal{G}_G)$ contains exactly the λ-terms in that are associated to the grammatical analyses of some given NL-grammar, $G = (W, Cat, \chi, S)$. $\Sigma_{\delta G}$ is a second order signature (*i.e.* every constants have types of the form $\alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \alpha_0$ where for all $i \in [0, n]$, α_i is an atomic type) whose terms encode derivations in NL. The role of the lexicon is to decode these terms into λ-terms representing the derivation. With such an ACG, it is easy to construct other ACGs that implement the usual interface between syntax and semantics of the NL-grammar. Indeed, we have seen in the previous section how to interpret with the homomorphism $\mathcal{Y}$ the λ-terms representing the derivations of an NL-grammar in order to obtain the corresponding surface structures. Obtaining the semantics can be done as usual in the categorial approach. The overall architecture of our approach is summarized by the following picture:

$$\begin{tikzcd}
 & \Sigma_{\delta G} \arrow[d, "\mathcal{L}_G"'] & \\
 & \Sigma_G \arrow[dl, "\mathcal{Y}"'] \arrow[dr, "\mathcal{L}_{Sem}"] & \\
\Sigma_W & & \Sigma_{Sem}
\end{tikzcd}$$

Our construction is based on a careful study of the shape of the grammatical analyses of NL-grammars. Thus we divide this section into two subsections. First we study the structure of NL-derivations and second, based on this study we construct our ACG.

4.1 The structure of NL-derivations

We here try to understand the general structure of NL-grammars. This means that we do not try to give an account of any proof in NL, but rather that we try to account for the specific proofs that are grammatical analyses of NL-grammars. In proof-theory, the object of study is the nature of proofs and in particular, proofs are identified modulo some congruence relation, usually up to some reduction process. Here, NL proofs are considered as equal whenever they have the same long normal form as it is common in proof-theoretical studies (see [12] and [10]). Such derivations are represented by λ-terms of the form

$$\lambda x_1 \dots x_n . h M_1 \dots M_p$$

where h is either a constant or a variable (remind that h is called the head of the term and, by extension, the head of the derivation that the term represents) and M_1, ..., M_p represent proofs in long normal forms. We will see now some more properties of the grammatical analyses of NL-grammars.

Grammatical analyses of an NL-grammars are derivations of sequents of the form $\Gamma \vdash A$ where Γ is a lexical base. Because we are interested in long normal derivations, we know that if A is a category of the form

$$c_1(A_1, \dots c_n(A_n, B) \dots) \text{ where } B \text{ is atomic}$$

then the derivation must finish with a sequence of n introduction rules and, therefore, be of the form:

$$\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{f_{c_n}(A_n, \dots f_{c_1}(A_1, \Gamma) \dots) \vdash B}\, c_n\ I}{\vdots}}{f_{c_1}(A_1, \Gamma) \vdash c_2(A_2, \dots c_n(A_n, B)}\, c_2\ I}{\Gamma \vdash c_1(A_1, \dots c_n(A_n, B) \dots)}\, c_1\ I$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{f_{c_1}(\Gamma,A_1)\vdash D_2=c_2(C_2,D_3)} \quad \dfrac{\vdots}{A_2\vdash C_2}}{f_{c_2}(f_{c_1}(\Gamma,A_1),A_2)\vdash D_3}\,c_2\ E}{\vdots}}{\Gamma_{n-1}\vdash c_n(C_n,B)} \quad \dfrac{\vdots}{A_n\vdash C_n}}{f_{c_n}(\Gamma_{n-1},A_n)\vdash B}\,c_n\ E}{\vdots}\,c_n\ I}{f_{c_1}(\Gamma,A_1)\vdash B_2}\,c_2\ I}{\Gamma\vdash c_1(A_1,B_2)=B_1=A}\,c_1\ I
$$

The notations are as follows: $A = c_1(A_1, \dots c_n(A_n, B)\dots)$, $B_n = D_n = c_n(A_n, B)$, $B_k = c_k(A_k, B_{k+1})$, $D_k = c_k(C_k, D_{k+1})$, $\Gamma_1 = \Gamma$, and $\Gamma_{k+1} = f_{c_k}(\Gamma_k, A_k)$

Fig. 1 Shape of the long normal derivation of $\Gamma \vdash A$

In the representation of the proof as λ-terms, $\lambda x_1 \dots x_n.hM_1 \dots M_p$, this sequence of introduction rules is responsible for the n λ-abstractions in front of the term. Furthermore $hM_1 \dots M_p$ is the representation of the proof of $f_{c_n}(A_n, \dots f_{c_1}(A_1, \Gamma)\dots) \vdash B$ which is finished by a sequence of elimination rules which are represented by the p applications in $hM_1 \dots M_p$.

First of all we remark that because of the particular shape of the context $f_{c_n}(A_n, \dots f_{c_1}(A_1, \Gamma)\dots)$, the structure of the proof $f_{c_n}(A_n, \dots f_{c_1}(A_1, \Gamma)\dots) \vdash B$ is fully determined for its $n-1$ last steps and must be of the form:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{f_{c_1}(A_1,\Gamma)\vdash c_2(C_2,\dots c_n(C_n,B))} \quad \dfrac{\vdots}{A_2\vdash C_2}}{\vdots}\,c_2\ E}{f_{c_{n-1}}(A_{n-1},\dots f_{c_1}(A_1,\Gamma)\vdash c_n(C_n,B)}\,c_{n-1}\ E \quad \dfrac{\vdots}{A_n\vdash C_n}}{f_{c_n}(A_n,\dots f_{c_1}(A_1,\Gamma)\dots)\vdash B}\,c_n\ E
$$

This implies that $hM_1 \dots M_p$ is actually of the form $hN_1 \dots N_k R_2 \dots R_n$ where R_i represents a proof of $A_i \vdash C_i$ (this entails that if the hypothesis A_i is represented by the λ-variable x_i then R_i only contains a unique free variable, x_i).

The overall structure of the proof of $\Gamma \vdash A$ that we have analysed so far is summarized in figure 1. Then the shape of the proof of $f_{c_1}(A_1, \Gamma) \vdash c_2(C_2, \dots c_n(C_n, B))$ is determined by the nature of its head, *i.e.* by whether h is a variable or a constant. In the first case the proof is of the shape

$$
\dfrac{\dfrac{}{A_1\vdash \tilde{c_1}(C_1,c_2(C_2,\dots c_n(C_n,B)))}\,Ax. \quad \dfrac{\vdots}{\Gamma\vdash C_1}}{f_{c_1}(A_1,\Gamma)\vdash c_2(C_2,\dots c_n(C_n,B))}\,\tilde{c_1}\ E
$$

and therefore $A_1 = \tilde{c_1}(C_1, c_2(C_2, \ldots c_n(C_n, B))$. Furthermore, the term that represents the proof of $\Gamma \vdash A$ is of the form

$$\lambda x_1 \ldots x_n . x_1 N R_2 \ldots R_n$$

where N is a closed term and R_i is a term that does not contain any constant and whose only free variable is x_i.

In the second case, the case where h is a constant, the first rule that is used to prove the sequent $f_{c_1}(A_1, \Gamma) \vdash c_2(C_2, \ldots c_n(C_n, B))$ must be an elimination of the operator c_1 thus the proof must be of the form:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{B_1 \in \chi(w)}{\langle w, F_1\rangle \vdash F_1 = c_1'(E_1, F_2)}\,Lex \qquad \dfrac{\vdots}{\Theta_1 \vdash E_1}}{f_{c_1'}(\langle w, F_1\rangle, \Theta_1) = \Delta_2 \vdash F_2}\,c_1'\ E}{\vdots}}{\Gamma \vdash D_1 = c_1(C_1, D_2)}\,c_p'\ E \qquad \dfrac{\vdots}{A_1 \vdash C_1}}{f_{c_1}(\Gamma, A_1) \vdash D_2}\,c_1\ E$$

Note that the bases Θ_i must be lexical since Γ is lexical. This implies that the whole proof of $\Gamma \vdash A$ (in the case where h is a constant) is represented by a λ-term of the form $\lambda x_1 \ldots x_n . \langle w, F_1\rangle N_1 \ldots N_l R_1 \ldots R_n$ where the N_i are closed λ-terms and the R_i are λ-terms that do not contain any constant and contain a unique free variable, namely x_i.

If we summarize what we have seen from our analysis, we note the long normal derivations of sequents of the forms $\Gamma \vdash A$ where Γ is a lexical base are composed only by derivations of sequents whose base is lexical and of sequents of the form $C \vdash A$. Furthermore the way these derivations are composed so as to obtain a derivation of $\Gamma \vdash A$ is fully determined by the shape of A and its head (whether it is a variable or a constant).

In order to complete our study of the structure of the grammatical analyses of NL-grammars, we have to analyse the structure of the long normal proofs of sequents of the form $C \vdash A$.

An analysis similar to the one that we conducted for the previous case leads to the fact that the general shape of the derivation of $C \vdash A$ obeys the general scheme presented in figure 1 where $\Gamma = C$. Now, there are two possibilities for the derivation of $f_{c_1}(C, A_1) \vdash D_2$: either the head of the derivation is the hypothesis C or it is the hypothesis A_1. If the head is C then we get the following derivation for $f_{c_1}(C, A_1) \vdash D_2$:

$$\dfrac{\dfrac{}{C \vdash C = c_1(C_1, D_2)}\,Ax. \qquad \dfrac{\vdots}{A_1 \vdash C_1}}{f_{c_1}(C, A_1) \vdash D_2}\,c_1\ E$$

In that case, the term that represents the derivation is $\lambda x_1 \ldots x_n . y R_1 \ldots R_n$ where y is the λ-variable that represents the hypothesis C and R_i is a λ-term that does not contain any constant and whose only free variable is x_i.

When A_1 is the head, the derivation of $f_{c_1}(C, A_1) \vdash D_2$ is:

$$\dfrac{\dfrac{}{A_1 \vdash A_1 = \tilde{c_1}(C_1, D_2)}\,Ax. \qquad \dfrac{\vdots}{C \vdash C_1}}{f_{\tilde{c_1}}(A_1, C) = f_{c_1}(C, A_1) \vdash D_2}\,\tilde{c_1}\ E$$

and the term that represents the derivation is $\lambda x_1 \ldots x_n.x_1 N R_2 \ldots R_n$ where N is a term that does not contain any constant and such that $FV(N) = \{y\}$ if y is the λ-variable that represents the hypothesis C and R_i are terms with the same properties as in the previous case.

We can summarise what this section has achieved so far:

Lemma 1 *Given an NL-grammar G, any long normal sub-derivations of a long normal grammatical analysis of G is either:*

L_l *A derivation of a sequent $\Gamma \vdash A$ with Γ being a lexical base and the head of the derivation being lexical or,*

L_v *A derivation of a sequent $\Gamma \vdash A$ with Γ being a lexical base and the head of the derivation being a non lexical hypothesis (a variable) or,*

C_h *A derivation of $C \vdash A$ with the head of the derivation being the hypothesis of type C or,*

C_o *A derivation of $C \vdash A$ with the head of the derivation being another hypothesis.*

Furthermore A and C are elements of $\mathcal{F}_G$

Note that we also detailed the way those derivations are composed together in a precise way. This will be of particular importance in the next subsection.

4.2 Representing NL-grammars as ACGs

We now define the second order signature $\Sigma_{\delta G}$ which encodes the grammatical analyses of an NL-grammar $G = (W, Cat, \chi, S)$. The set of types of $\Sigma_{\delta G}$ is given by $\mathcal{A}_G = \{[\bullet \vdash A] | A \in \mathcal{F}_G\} \cup \{[C \vdash A] | A, C \in \mathcal{F}_G\}$. The types of the form $[\bullet \vdash A]$ are the type of the terms encoding the derivations whose conclusion is of the form $\Gamma \vdash A$ with Γ being a lexical base; a type of the form $[C \vdash A]$ is the type of the encodings of the derivations whose conclusion is $C \vdash A$. The constants of $\Sigma_{\delta G}$ are divided in four sets which correspond to the four cases of lemma 1. These sets will be named accordingly to the cases they refer to. Thus, we let $\mathcal{C}_{\delta G} = L_l \cup L_v \cup C_h \cup C_o$. Along with the definition of those sets we define $\tau_{\delta G}$ but also $\mathcal{L}_{\delta G}$ (we let $\mathcal{L}_{\delta G}([\bullet \vdash A]) = |A|$ and $\mathcal{L}_{\delta G}([C \vdash A]) = |C| \rightarrow |A|$) the lexicon that *decodes* the terms built with those constants into linear λ-terms that represent the encoded derivations. Even though they are technical, the following definitions follow exactly the cases we detailed in the previous subsection while describing the general structure of NL-derivations.

$$\begin{aligned} L_l = \{\langle c, F, A\rangle | \quad & F \in \chi(c) \wedge A \in \mathcal{F}_G \\ & \wedge F = c'_1(E_1, \ldots c'_p(E_p, c_1(C_1, \ldots c_n(C_n, B) \ldots)) \ldots) \\ & \wedge A = c_1(A_1, \ldots c_n(A_n, B) \ldots) \\ & \wedge B \in Cat\} \end{aligned}$$

In this case we have $\tau_{\delta G}(\langle c, F, A\rangle) = \beta_1 \rightarrow \cdots \beta_p \rightarrow \alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow [\bullet \vdash A]$ with $\beta_i = [\bullet \vdash E_i]$ and $\alpha_j = [A_j \vdash C_j]$. Furthermore:

$$\mathcal{L}_G(\langle c, F, A\rangle) = \lambda x_1 \ldots x_p y_1 \ldots y_n z_1 \ldots z_n.\langle c, F\rangle x_1 \ldots x_p (y_1\, z_1) \ldots (y_n\, z_n)$$

$$\begin{aligned} L_v = \{\langle \bullet, A\rangle | \quad & A \in \mathcal{F}_G \\ & \wedge A = c_1(A_1, \dots c_n(A_n, C) \dots) \\ & \wedge A_1 = c'_1(C_1, \dots c'_n(C_n, C) \dots) \\ & \wedge c'_1 = \tilde{c_1} \wedge 1 < i \leq n \Rightarrow c'_i = c_i \\ & \wedge C \in Cat\} \end{aligned}$$

Here $\tau_{\delta G}(\langle \bullet, A\rangle)$ is $\alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow [\bullet \vdash A]$ where $\alpha_1 = [\bullet \vdash C_1]$ and $\alpha_i = [A_i \vdash C_i]$ when $1 < i \leq n$; and:

$$\mathcal{L}_G(\langle \bullet, A\rangle) = \lambda x_1 \dots x_n z_1 \dots z_n . z_1\, x_1(x_2\, z_2) \dots (x_n\, z_n)$$

$$\begin{aligned} C_h = \{\langle C, A\rangle_1 | \quad & A \in \mathcal{F}_G \wedge C \in \mathcal{F}_G \\ & \wedge A = c_1(A_1, \dots c_n(A_n, B) \dots) \\ & \wedge C = c_1(C_1, \dots c_n(C_n, B) \dots) \\ & \wedge B \in Cat\} \end{aligned}$$

For that set we let $\tau_{\delta G}(\langle C, A\rangle_1) = \alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow [C \vdash A]$ with $\alpha_i = [A_i \vdash C_i]$, and also:

$$\mathcal{L}_G(\langle C, A\rangle_1) = \lambda x_1 \dots x_n z_0 \dots z_n . z_0(x_1\, z_1) \dots (x_n\, z_n)$$

$$\begin{aligned} C_o = \{\langle C, A\rangle_2 | \quad & A \in \mathcal{F}_G \wedge C \in \mathcal{F}_G \\ & \wedge A = c_1(A_1, \dots c_n(A_n, B) \dots) \\ & \wedge A_1 = c'_1(C_1, \dots c'_n(C_n, B) \dots) \\ & \wedge c'_1 = \tilde{c_1} \wedge 1 < i \leq n \Rightarrow c'_i = c_i \\ & \wedge B \in Cat\} \end{aligned}$$

Finally we let $\tau_{\delta G}(\langle C, A\rangle_2) = \alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow [C \vdash A]$ such that $\alpha_1 = [C \vdash C_1]$ and for $1 < i \leq n$, $\alpha_i = [A_i \vdash C_i]$; and

$$\mathcal{L}_G(\langle C, A\rangle_2) = \lambda x_1 \dots x_n z_0 \dots z_n . z_1(x_1\, z_0)(x_1\, z_2) \dots (x_n\, z_n)$$

Lemma 1 assures that he constants that are declared in $\Sigma_{\delta G}$ define all the necessary basic building blocks of the grammatical analyses of G. Furthermore, the way those constants are typed assures that those blocks are assembled in a way that is compliant with NL rules. Finally, the lexicon $\mathcal{L}_G$ mimics the structures that we have showed in the previous subsection. This allows us to obtain the following property:

Theorem 2 *Given an NL grammar G, the object language of the ACG*

$$\mathcal{G}_G = (\Sigma_{\delta G}, \Sigma_G, \mathcal{L}_G, [\bullet \vdash S])$$

is exactly the set of λ-terms that represent grammatical analyses of G.

Note that the size of $\Sigma_{\delta G}$ and the size of $\mathcal{L}_G$ are quadratic with respect to the size of G since each constant in $\Sigma_{\delta G}$ is depends on two elements taken from $\mathcal{F}_G$.

5 An example

We now give an example of our construction. In this example we use an NL-grammar with the lexical entries:

- aime : $(np\backslash S)/np$,
- Philippe, Rachel : np,
- qui : $(np\backslash np)\backslash(np/S)$,
- dort : $np\backslash S$

We show how to represent the following grammatical derivation as an abstract term of the second order ACG that encodes this grammar:

$$\dfrac{\dfrac{\text{Philippe} \vdash np \quad \dfrac{\text{qui} \vdash (np\backslash np)/(np\backslash S) \quad \dfrac{\dfrac{x \vdash x : np \quad \dfrac{\text{aime} \vdash (np\backslash S)/np \quad \text{Rachel} \vdash np}{(\text{aime},\text{Rachel}) \vdash \text{aime}\,\text{Rachel} : S}}{(x,(\text{aime},\text{Rachel})) \vdash \text{aime}\,\text{Rachel}\,x : np\backslash S}}{(\text{aime},\text{Rachel}) \vdash \lambda x.\text{aime}\,\text{Rachel}\,x : np\backslash S}}{(\text{qui},(\text{aime},\text{Rachel})) \vdash \text{qui}(\lambda x.\text{aime}\,\text{Rachel}\,x) : np\backslash np}}{(\text{Philippe},(\text{qui},(\text{aime},\text{Rachel}))) \vdash \text{qui}(\lambda x.\text{aime}\,\text{Rachel}\,x)\text{Philippe} : np} \quad \text{dort} \vdash np\backslash S}{((\text{Philippe},(\text{qui},(\text{aime},\text{Rachel}))),\text{dort}) \vdash \text{dort}(\text{qui}(\lambda x.\text{aime}\,\text{Rachel}\,x)\text{Philippe}) : np}$$

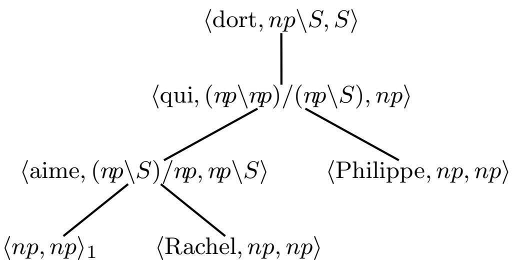
In order to represent the term that encodes this grammatical analysis, we will use the following constants of the abstract vocabulary of the ACG we would obtain from our construction:

- $\langle \text{dort}, np\backslash S, S\rangle : [\bullet \vdash np] \rightarrow [\bullet \vdash S]$,
- $\langle \text{aime}, (np\backslash S)/np, np\backslash S\rangle : [np \vdash np] \rightarrow [\bullet \vdash np] \rightarrow [\bullet \vdash np\backslash S]$,
- $\langle \text{qui}, (np\backslash np)/(np\backslash S), np\rangle : [\bullet \vdash np] \rightarrow [\bullet \vdash np\backslash S] \rightarrow [\bullet \vdash np]$,
- $\langle \text{Philippe}, np, np\rangle : [\bullet \vdash np]$,
- $\langle \text{Rachel}, np, np\rangle : [\bullet \vdash np]$,
- $\langle np, np\rangle_1 : [np \vdash np]$

Those constants are mapped as follows λ-terms by the lexicon $\mathcal{L}$:

- $\mathcal{L}(\langle \text{dort}, np\backslash S, S\rangle) = \lambda x.\text{dort}x$,
- $\mathcal{L}(\langle \text{aime}, (np\backslash S)/np, np\backslash S\rangle) = \lambda xyz.\text{aime}\,x\,(y\,z)$,
- $\mathcal{L}(\langle \text{qui}, (np\backslash np)/(np\backslash S), np\rangle) = \lambda x_1 x_2.\text{qui}\,x_1\,x_2$,
- $\mathcal{L}(\langle \text{Philippe}, np, np\rangle) = \text{Philippe}$,
- $\mathcal{L}(\langle \text{Rachel}, np, np\rangle) = \text{Rachel}$,
- $\mathcal{L}(\langle np, np\rangle_1) = \lambda z.z$

Thus the term representing the proof is (in a tree-like notation):



One can easily check that $\mathcal{L}$ maps this tree to the λ-term $\text{dort}(\text{qui}(\lambda x.\text{aime}\,\text{Rachel}\,x)\text{Philippe})$.

6 Outcomes of the construction

Interestingly we can use some known results about ACGs on the construction we propose and then obtain alternative proofs of some known results about NL-grammars.

6.1 Context-freeness of NL

The lexicon that is obtained by composing $\mathcal{L}_G$ with $\mathcal{Y}$ is somewhat complicated in the sense that it does not obviously show that the language recognized by G is context free. This can be fixed by remarking that we can define a lexicon $\mathcal{L}_{sym}$ from $\Sigma_{\delta G}$ to Σ_W that maps exactly the terms of the abstract language to the string of which they represent a grammatical analysis. The lexicon $\mathcal{L}_{sym}$ may be defined by:

1. $\mathcal{L}_{syn}([\bullet \vdash A]) = * \rightarrow *$,
2. $\mathcal{L}_{syn}([C \vdash A]) = * \rightarrow *$,
3. $\mathcal{L}_{syn}(\langle c, F, A\rangle) = \lambda x_1 \dots x_p y_1 \dots y_n.((\dots(((\dots(c{+_{c_1}}x_1)\dots){+_{c_p}}x_p){+_{c'_1}}y_1)\dots){+_{c'_n}} y_n)$ if $F = c'_1(E_1, \dots c'_p(E_p, c_1(C_1, \dots c_n(C_n, B)\dots))\dots)$ and $A = c_1(A_1, \dots c_n(A_n, B)\dots)$,
4. $\mathcal{L}_{syn}(\langle \bullet, A\rangle) = \lambda x_1 \dots x_n.x_1 + \cdots + x_n$ if $A = c_1(A_1, \dots c_n(A_n, B)\dots)$,
5. $\mathcal{L}_{syn}(\langle C, A\rangle_1) = \lambda x_1 \dots x_n.x_1 + \dots + x_n$ when $A = c_1(A_1, \dots c_n(A_n, B)\dots)$,
6. $\mathcal{L}_{syn}(\langle C, A\rangle_2) = \lambda x_1 \dots x_n.x_1 + \dots + x_n$ when $A = c_1(A_1, \dots c_n(A_n, B)\dots)$

The definition of $\mathcal{L}_{syn}$ is based on the fact that terms of type $[\bullet \vdash A]$ represent proofs of the form $\Gamma \vdash A$ where Γ is lexical. Such terms give the analysis of the fact that the phrase $\overline{\Gamma}$ is of category A. Meanwhile terms of type $[C \vdash A]$ show that if the empty string is considered of type C then it can be considered as being of type A. Hence, $\mathcal{L}_{syn}$ operates on the yields of the lexical contexts as NL-calculus does. And, as opposed to the composition of $\mathcal{L}_G$ with $\mathcal{Y}$, $\mathcal{L}_{syn}$ ignores the functional complexity induced by the order of the categories typing strings. As a consequence, this gives this simpler formulation. It was showed in [5] that ACG like $(\Sigma_{Der\,G}, \Sigma_W, \mathcal{L}_{syn}, [\bullet \vdash S])$ exactly define a context free language recognized by a context free grammar.

6.2 Parsing complexity

In [5], we can remark that the context free grammar whose language is the same as an ACG like $(\Sigma_{Der\,G}, \Sigma_W, \mathcal{L}_{syn}, [\bullet \vdash S])$ has the same size as this ACG. As context free grammars can be parsed in time $\mathcal{O}(mn^3)$ where m is the size of the grammar (the number of occurrences of symbols in the rules of the grammar) and n is the number of word in the parsed sentence, our construction shows that NL-grammars can be parsed in $\mathcal{O}(m^2n^3)$ with m being the size of the NL-grammar (the number of symbols in the lexicon) and n^3 being the size of the considered sentence. The quadratic dependence of the complexity comes from the fact that the ACG that we obtain has a size that is quadratic in the size of the original NL-grammar.

7 Conclusion

We defined a faithful embedding NL-grammars into second order ACGs. Our proposal accounts easily for the interface between syntax and semantics for those grammars in the ACG framework. Besides the resolution of this open question, the construction we propose gives alternative proofs to already known properties of NL-grammars.

Indeed, we obtain a one-to-one representation of the grammatical analyses of an NL-grammar as a set of local trees. This fact could be derived from the results of [13] which shows that the set of grammatical analyses represented as normal natural deduction trees of a given NL-grammar could be seen as a regular set of trees. Nevertheless, as far as we know our construction is different and easily shows that the size of the automaton that recognizes this set is quadratic with respect to the size of the original NL-grammar.

We can also see that this local set of trees as the parse trees of a context free grammar that is equivalent to the original NL-grammar. From these facts, we also get a new proof, different from the one in [6], that NL-grammars can be parsed in time $\mathcal{O}(m^2n^3)$ where m is the size of the grammar and n is the size of the sentence, which shows that the universal membership problem is polynomial for NL-grammars.

The investigation we have undertaken tries to relate ACGs and elder categorial formalisms. This line of research tends to develop the connections between formal language theory and proof-theory in the spirit of [13]. In the near future we shall explore with similar methods related categorial systems like: NL-grammars with product, NL-grammars with non-logical axioms and the associative Lambek Calculus. Furthermore we have seen that the homomorphism $\mathcal{L}_{syn}$ yields to a simpler description of the language of the NL-grammar than the composition of $\mathcal{L}_G$ with $\mathcal{Y}$. This raises the question of possible automation of simplification of lexicons and also the problem of deciding whether they are equivalent. This last question has already received an answer in [14], since such homomorphisms may be seen as deterministic MSO-transductions. It nevertheless remains to see in which cases such a problem is tractable.

References

1. de Groote, P.: Towards abstract categorial grammars. In for Computational Linguistic, A., ed.: Proceedings 39th Annual Meeting and 10th Conference of the European Chapter, Morgan Kaufmann Publishers (2001) 148–155
2. Stabler, E.: Derivational minimalism. In Retoré, C., ed.: Logical Aspects of Computational Linguistics, LACL'96. Volume 1328 of LNCS/LNAI., Springer-Verlag (1997) 68–95
3. Muskens, R.: Lambda Grammars and the Syntax-Semantics Interface. In van Rooy, R., Stokhof, M., eds.: Proceedings of the Thirteenth Amsterdam Colloquium, Amsterdam (2001) 150–155
4. Morawietz, F.: Two-Step Approaches ot Natural Language Formalisms. Studies in Generative Grammar. Mouton de Gruyter, Berlin · New York (2003)
5. de Groote, P., Pogodalla, S.: On the expressive power of abstract categorial grammars: Representing context-free formalisms. Journal of Logic, Language and Information **13**(4) (2004) 421–438
6. Capelletti, M.: Parsing with Structure-Preserving Categorial Grammars. PhD thesis, UIL-OTS, Universiteit Utrecht (2007)
7. Moortgat, M.: Categorial Investigations: Logical & Linguistic Aspects of the Lambek Calculus. Foris Pubns USA (1988)
8. Moot, R.: Grail: an interactive parser for categorial grammars. In Delmonte, R., ed.: Proceedings of VEXTAL 1999. (1999) 255–261
9. Barendregt, H.P.: The Lambda Calculus: Its Syntax and Semantics. Volume 103. Studies in Logic and the Foundations of Mathematics, North-Holland Amsterdam (1984) revised edition.
10. Huet, G.: Résolution d'équations dans des langages d'ordre 1,2,...,ω. Thèse de doctorat es sciences mathématiques, Université Paris VII (1976)

11. Lambek, J.: On the calculus of syntactic types. In Jakobson, R., ed.: Studies of Language and its Mathematical Aspects, Proceedings of the 12th Symposium of Applied Mathematics. (1961) 166–178
12. Girard, J.Y., Taylor, P., Lafont, Y.: Proofs and types. Cambridge University Press (1989)
13. Tiede, H.J.: Deductive Systems and Grammars: Proofs as Grammatical Structures. PhD thesis, Indiana University (1999)
14. Engelfriet, J., Maneth, S.: The equivalence problem for deterministic mso tree transducers is decidable. Inf. Process. Lett. **100**(5) (2006) 206–212
15. Kandulski, M.: Derived tree languages of nonassociative lambek categorial grammars with product. Fundam. Inform. **55**(3-4) (2003) 349–362
16. Buszkowski, W.: Mathematical linguistics and proof theory. In van Benthem, J., ter Meulen, A., eds.: Handbook of Logic and Language. Elsevier science BV and MIT press (1997) 683–736