Denis Béchet

Parsing pregroup grammars and Lambek calculus using partial composition

Abstract. The paper presents a way to transform pregroup grammars into context-free grammars using *functional composition*. The same technique can also be used for the proof-nets of multiplicative cyclic linear logic and for Lambek calculus allowing empty premises.

Keywords: pregroup, Lambek calculus, linear logic, parsing, context-free grammars.

1. Introduction

Pregroup grammars [10] and Lambek calculus [9] are based on the idea that sentences are produced from words using lexical rules. The syntactical properties of each word are characterized by a finite set of types. Buszkowski in [4] proved that languages generated by pregroup grammars are context-free. However, the proof does not give a direct algorithm for the construction of a context-free grammar starting from a pregroup grammar even if one is defined in [5]. Our transformation follows the structure of a pregroup derivation but the steps are switched and grouped together. Each group is an instance of a context-free rule with one or two right hand sides. This technique can also be applied to related formalisms such as multiplicative intuitionistic cyclic linear logic presented as proof-nets, or its equivalent: Lambek calculus allowing empty premises. Pentus proposed earlier an algorithm which transforms a grammar based on Lambek calculus into a context-free grammar [15, 16]. Although this idea produces a resulting grammar with rules with binary right hand sides, it uses an interpolation theorem which allows to compute all the possible compositions of two types. Here, the idea consists in showing that the parsing of a string involves a restricted form of partial compositions of two types with a result less or equal to the maximal sizes of its arguments. The restricted partial composition is called *functional*. It can be computed directly on types for pregroups, or on flat interfaces for cyclic linear logic. Either similar results on Lambek calculus,

Presented by **Name of Editor**; *Received* December 1, 2002

Studia Logica **68**: 1–26, 2001.

[14, 7] also use modules and proof-nets but the transformation is global and do not define a notion of partial composition of types.

This paper presents a transformation of pregroup grammars into context-free grammars. Then, the same technique is used to show the transformation of Lambek calculus using flat interface calculus.

2. Pregroup grammar

2.1. Notations

We denote $\mathbb{N}$ the set of natural numbers including 0 and $\mathbb{Z}$ the set of all integers. Let Σ be a non-empty set called an *alphabet*. We define a *string* over the alphabet Σ as a finite sequence $a_1, a_2, \ldots, a_n$ of elements of Σ. Σ^* is the set of all finite sequences over Σ and Σ^+ is the set of all finite sequences over Σ except the empty sequence.

2.2. Pregroup and free pregroup

Definition 2.1 (Pregroup). A *pregroup* is a structure $(P, \leq, \cdot, l, r, 1)$ such that $(P, \leq, \cdot, 1)$ is a partially ordered monoid[1] and l, r are two unary operations on P that satisfy: $\forall a \in P : a^l a \leq 1 \leq a a^l$ and $a a^r \leq 1 \leq a^r a$.
Iterated adjoints are defined for $i \in \mathbb{Z}$: $a^{(0)} = a$, for $i \leq 0 : a^{(i-1)} = \left(a^{(i)}\right)^l$, for $i \geq 0 : a^{(i+1)} = \left(a^{(i)}\right)^r$

Definition 2.2 (Free pregroup, width). Let $(P, \leq_P)$ be a partially ordered set of primitive types, $P^{(\mathbb{Z})} = \{p^{(i)} \mid p \in P, i \in \mathbb{Z}\}$ is the set of atomic types and $\mathcal{C}at_{(P,\leq_P)} = \left(P^{(\mathbb{Z})}\right)^* = \{p_1^{(i_1)} \cdots p_n^{(i_n)} \mid 1 \leq k \leq n, p_k \in P, i_k \in \mathbb{Z}\}$ is the set of types[2]. We define the *width* of a type $C = p_1^{u_1} \ldots p_n^{u_n}$ as $wd(C) = n$ (the number of atomic types). For $X, Y \in \mathcal{C}at_{(P,\leq_P)}$, $X \leq Y$ iff this relation is deducible in the following system proposed by Buszkowski where $p, q \in P$, $n, k \in \mathbb{Z}$ and $X, Y, Z \in \mathcal{C}at_{(P,\leq_P)}$[3]:

---

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

[2] In strings, we use commas to distinguish the components, but atomic types occur without punctuation marks even if in both cases they define sequences.

[3] In fact, the resulting structure is a quasi-pregroup because $\leq$ is only a preorder on $\mathcal{C}at_{(P,\leq_P)}$. However, we can define a pregroup from this quasi-pregroup by constructing the quotient-structure from the congruence $\sim$ on $\mathcal{C}at_{(P,\leq_P)}$ defined by $x \sim y \Leftrightarrow x \leq y \wedge y \leq x$.

$$X \leq X \ (Id) \qquad \frac{XY \leq Z}{Xp^{(n)}p^{(n+1)}Y \leq Z}\,(A_L) \qquad \frac{Xp^{(k)}Y \leq Z}{Xq^{(k)}Y \leq Z}\,(IND_L)$$

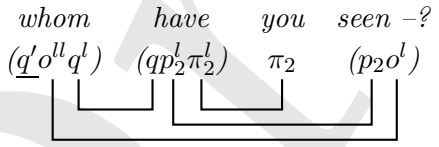
$$\frac{X \leq Y \quad Y \leq Z}{X \leq Z}\,(Cut) \qquad \frac{X \leq YZ}{X \leq Yp^{(n+1)}p^{(n)}Z}\,(A_R) \qquad \frac{X \leq Yq^{(k)}Z}{X \leq Yp^{(k)}Z}\,(IND_R)$$

$q \leq_P p$ if k is even
or $p \leq_P q$ if k is odd

2.3. Pregroup grammars

Definition 2.3 (Pregroup grammar). Let $(P, \leq_P)$ be a finite partially ordered set. A *pregroup grammar* based on $(P, \leq_P)$ is a lexicalized[4] grammar $G = (\Sigma, I, s)$ such that $s \in P$; G assigns a type X to a string $v_1, \ldots, v_n$ of Σ^* iff for $1 \leq i \leq n$, $\exists X_i \in I(v_i)$ such that $X_1 \cdots X_n \leq X$ in the free pregroup based on $(P, \leq_P)$. *The language* $\mathcal{L}(G)$ is the set of strings in Σ^* that are assigned s by G.

Example 2.4. *Our example is taken from [11] with the primitive types:* π_2 *= second person,* p_2 *= past participle,* o *= object,* q *= yes-or-no question,* q' *= question. This sentence receives the type* q' *(*$q' \leq_P s$*), where the dash indicates a Chomskian trace:*



2.4. Context-free grammars

Definition 2.5 (Context-free grammar). A *context-free grammar* is a quadruple $\langle \Sigma, \mathcal{W}, S, \mathcal{R} \rangle$, where the set of terminal symbols Σ and the set of non-terminal symbols $\mathcal{W}$ are two disjoint finite sets, $S \in \mathcal{W}$ is the start symbol and $\mathcal{R}$ is a finite set of *context-free productions* of the form[5] $\alpha \to u$, where $\alpha \in \mathcal{W}$ and $u \in (\mathcal{W} \cup \Sigma)^+$.

A string $w' \in (\mathcal{W} \cup \Sigma)^+$ is *directly derivable* from a string $w \in (\mathcal{W} \cup \Sigma)^+$ in a grammar $G = \langle \Sigma, \mathcal{W}, S, \mathcal{R} \rangle$ (denoted by $w \xrightarrow{\mathcal{R}} w'$) iff $w = v_1, \alpha, v_2$, $w' = v_1, u, v_2$ and $\alpha \to u$ is a context-free production of G. $\xrightarrow{\mathcal{R}^*}$ is the reflexive and transitive closure of $\xrightarrow{\mathcal{R}}$. We indicate that w' is *derivable* from

[4] A lexicalized grammar is a triple (Σ, I, s): Σ is a finite alphabet, I assigns a finite set of types to each $c \in \Sigma$, s is a type associated to grammatical sentences.

[5] We use context-free languages without the empty string.

w (denoted by $w \xrightarrow{\mathcal{R}^*} w'$). The *language generated by* G, denoted by $\mathcal{L}(G)$ is the set of strings of Σ^+, derivable in this grammar from S.

3. Parsing pregroup grammars

Pregroup languages are context-free and the parsing is polytime. In this section we present a construction that defines a context-free grammar from a pregroup grammar.

3.1. Rewriting systems

Rewriting systems, denoted by $\xrightarrow{\mathcal{R}}$ where $\mathcal{R}$ consists of one or several rules, are defined on sequences of types $(\mathcal{C}at_{(P,\leq_P)})^*$. $\xrightarrow{\mathcal{R}}{}^*$ is the reflexive-transitive closure of $\xrightarrow{\mathcal{R}}$. In the following rules, p, q are atomic types, X and Y range over types and Γ, Δ over sequences of types:

- **GCON (generalized contraction)** : if $q \leq_P p$ and n is even or if $p \leq_P q$ and n is odd: $Xq^{(n)}p^{(n+1)}Y \xrightarrow{\mathbf{GCON}} XY$

LEMMA 3.1 (Lambek switching lemma). *Let* $X \in \mathcal{C}at_{(P,\leq_P)}$ *and* $s \in P$*:*

$$X \leq s \text{ iff } \exists q \in P \text{ such that } X \xrightarrow{\mathbf{GCON}^*} q \text{ and } q \leq_P s$$

Using this property, the parsing of strings can be produced using a planar graph where the atomic types are linked two by two except for one primitive type $q \leq_P s$ that is underlined. Example 2.4 uses this notation: the underlined atomic type is q' and the links are shown below the types.

In order to introduce partial composition, we extend the notion of inference to the lists of types, so that it reflects the separations between the words of the initial string:

- **M (merge)** : $\Gamma, X, Y, \Delta \xrightarrow{\mathbf{M}} \Gamma, XY, \Delta$.
- **I (internal)** : if $q \leq_P p$ and n is even or if $p \leq_P q$ and n is odd: $\Gamma, Xq^{(n)}p^{(n+1)}Y, \Delta \xrightarrow{\mathbf{I}} \Gamma, XY, \Delta$,
- **E (external)** : if $q \leq_P p$ and n is even or if $p \leq_P q$ and n is odd: $\Gamma, Xq^{(n)}, p^{(n+1)}Y, \Delta \xrightarrow{\mathbf{E}} \Gamma, X, Y, \Delta$,

This system is equivalent with the deduction system when the final right element is a primitive type (or 1). As a consequence, the parsing can be done using $\xrightarrow{(\mathbf{MIE})^*}$.

LEMMA 3.2. *Let* $X_1, \ldots, X_n \in \mathcal{C}at_{(P,\leq_P)}$ *and* $s \in P$*:*

$$X_1 \cdots X_n \leq s \text{ iff } \exists q \in P \text{ such that } X_1, \ldots, X_n \xrightarrow{(\mathbf{MIE})^*} q \text{ and } q \leq_P s$$

PROOF. $\xrightarrow{\mathbf{I}}$ and $\xrightarrow{\mathbf{E}}$ are the two possible versions of $\xrightarrow{\mathbf{GCON}}$ in the context of lists of types. In fact $\xrightarrow{\mathbf{E}}$ is not necessary because we can perform $n-1$ $\xrightarrow{\mathbf{M}}$ steps first then several $\xrightarrow{\mathbf{I}}$ steps. ■

COROLLARY 3.3. *$G = (\Sigma, I, s)$ generates a string $v_1 \cdots v_n$ iff for $1 \leq i \leq n$, $\exists X_i \in I(v_i)$ and $\exists p \in P$ such that $X_1, \cdots, X_n \xrightarrow{(\mathbf{MIE})^*} p$ and $p \leq_P s$.*

EXAMPLE 3.4. *Parsing of "whom have you seen ?" ($q' \leq_P s$)*

whom	*have*	*you*	*seen*
$q'o^{ll}q^l$	$qp_2^lp_2^l$	π_2	p_2o^l

$$q'o^{ll}\underbrace{q^l}, qp_2^l\pi_2^l, \pi_2, p_2o^l \xrightarrow{\mathbf{E}} q'o^{ll}, \boxed{p_2^l\pi_2^l, \pi_2}, p_2o^l \xrightarrow{\mathbf{M}} q'o^{ll}, p_2^l\underbrace{\pi_2^l}\pi_2, p_2o^l$$
$$\xrightarrow{\mathbf{I}} q'o^{ll}, \underbrace{p_2^l}, \underbrace{p_2o^l} \xrightarrow{\mathbf{E}} q'o^{ll}, \boxed{1, o^l} \xrightarrow{\mathbf{M}} q'\underbrace{o^{ll}}, \underbrace{o^l} \xrightarrow{\mathbf{E}} q' \text{ and } q' \leq_P s$$

In fact, every $\xrightarrow{\mathbf{I}}$ step can be performed before the $\xrightarrow{\mathbf{M}}$ and $\xrightarrow{\mathbf{E}}$ steps.

LEMMA 3.5. *$\Gamma_1 \xrightarrow{(\mathbf{MIE})^*} \Gamma_2$ iff $\exists \Delta$ such that $\Gamma_1 \xrightarrow{\mathbf{I}^*} \Delta$ and $\Delta \xrightarrow{(\mathbf{ME})^*} \Gamma_2$*

PROOF. We can always switch a $\xrightarrow{\mathbf{I}}$ step and a $\xrightarrow{\mathbf{E}}$ step. For a $\xrightarrow{\mathbf{M}}$ step followed by a $\xrightarrow{\mathbf{I}}$ step: if the merged types do not overlap the contracted type or if the contracted atomic types do not come from two types before the $\xrightarrow{\mathbf{M}}$ step, then the two steps can be switched. In the other case, the $\xrightarrow{\mathbf{I}}$ step is transformed into a $\xrightarrow{\mathbf{E}}$ step followed by a $\xrightarrow{\mathbf{M}}$ step. ■

EXAMPLE 3.6. *In the previous example:*

$$q'o^{ll}\underbrace{q^l}, qp_2^l\pi_2^l, \pi_2, p_2o^l \xrightarrow{\mathbf{E}} q'o^{ll}, \boxed{p_2^l\pi_2^l, \pi_2}, p_2o^l \xrightarrow{\mathbf{M}} q'o^{ll}, p_2^l\underbrace{\pi_2^l}\pi_2, p_2o^l$$
$$\xrightarrow{\mathbf{I}} q'o^{ll}, \underbrace{p_2^l}, \underbrace{p_2o^l} \xrightarrow{\mathbf{E}} q'o^{ll}, \boxed{1, o^l} \xrightarrow{\mathbf{M}} q'\underbrace{o^{ll}}, \underbrace{o^l} \xrightarrow{\mathbf{E}} q'$$

becomes:

$$q'o^{ll}\underbrace{q^l}, qp_2^l\pi_2^l, \pi_2, p_2o^l \xrightarrow{\mathbf{E}} q'o^{ll}, p_2^l\underbrace{\pi_2^l}, \pi_2, p_2o^l \xrightarrow{\mathbf{E}} q'o^{ll}, \boxed{p_2^l, 1}, p_2o^l$$
$$\xrightarrow{\mathbf{M}} q'o^{ll}, \underbrace{p_2^l}, \underbrace{p_2o^l} \xrightarrow{\mathbf{E}} q'o^{ll}, \boxed{1, o^l} \xrightarrow{\mathbf{M}} q'\underbrace{o^{ll}}, \underbrace{o^l} \xrightarrow{\mathbf{E}} q'$$

The external reductions corresponding to the same pair of types and a merge reduction can be joined together such that, at each step, the number of types decreases.

- **C (k-partial composition)** : For $k \in \mathbb{N}$, if $q_i \leq_P p_i$ and n_i is even or if $p_i \leq_P q_i$ and n_i is odd, for $1 \leq i \leq k$:
  $\Gamma, Xq_1^{(n_1)} \cdots q_k^{(n_k)}, p_k^{(n_k+1)} \cdots p_1^{(n_1+1)}Y, \Delta \xrightarrow{\mathbf{C}} \Gamma, XY, \Delta$

Remark. A 0-partial composition corresponds to a $\xrightarrow{\mathbf{M}}$ step, a 1-partial composition to a $\xrightarrow{\mathbf{E}}$ step followed by a $\xrightarrow{\mathbf{M}}$ step, and a k-partial composition to k $\xrightarrow{\mathbf{E}}$ steps followed by a $\xrightarrow{\mathbf{M}}$ step. For A and B, $\exists k_0, k_0 \leq min(wd(A), wd(B))$ such that $\forall k, 0 \leq k \leq k_0$, A and B are k-partially composable but are not (k_0+1)-partially composable.

LEMMA 3.7. *For* $\Gamma \in (\mathcal{C}at_{(P,\leq_P)})^+$, $X \in \mathcal{C}at_{(P,\leq_P)}$: $\Gamma \xrightarrow{(\mathbf{ME})^*} X$ *iff* $\Gamma \xrightarrow{\mathbf{C}^*} X$

PROOF. The $\xrightarrow{\mathbf{M}}$ steps form a tree with the types of Γ as leaves and X as root. Except for the atomic types of X, each atomic type of Γ is involved in a $\xrightarrow{\mathbf{E}}$ step with another atomic type. Using the tree, each couple of such atomic types corresponds to a unique $\xrightarrow{\mathbf{M}}$ step. Thus, every $\xrightarrow{\mathbf{E}}$ step corresponds to a unique $\xrightarrow{\mathbf{M}}$ step. Now, we can switch the rules such that all the $\xrightarrow{\mathbf{E}}$ steps that correspond to the same $\xrightarrow{\mathbf{M}}$ step are grouped just before the $\xrightarrow{\mathbf{M}}$ step (in the same order as they appear in the initial derivation) and we can join them to form a $\xrightarrow{\mathbf{C}}$ step. The converse is obvious because Rule $\xrightarrow{\mathbf{C}}$ is the composition of Rule $\xrightarrow{\mathbf{E}}$ and Rule $\xrightarrow{\mathbf{M}}$. ■

EXAMPLE 3.8. *In the previous example:*

$$q'o^{ll}\underbrace{q^l}, \underbrace{qp_2^l}\pi_2^l, \pi_2, p_2o^l \xrightarrow{\mathbf{E}} q'o^{ll}, \underbrace{p_2^l\pi_2^l}, \pi_2, p_2o^l \xrightarrow{\mathbf{E}} q'o^{ll}, \boxed{p_2^l, 1}, p_2o^l$$
$$\xrightarrow{\mathbf{M}} q'o^{ll}, \underbrace{p_2^l, p_2o^l} \xrightarrow{\mathbf{E}} q'o^{ll}, \boxed{1, o^l} \xrightarrow{\mathbf{M}} q'\underbrace{o^{ll}, o^l} \xrightarrow{\mathbf{E}} q'$$

becomes

$$\boxed{q'o^{ll}\underbrace{q^l}, \underbrace{qp_2^l}\pi_2^l}, \pi_2, p_2o^l \xrightarrow{\mathbf{C}} \boxed{q'o^{ll}\underbrace{p_2^l\pi_2^l}, \pi_2}, p_2o^l \xrightarrow{\mathbf{C}} \boxed{q'\underbrace{o^{ll}\underbrace{p_2^l}, \underbrace{p_2o^l}}} \xrightarrow{\mathbf{C}} q'$$

The first and second steps are 1-partial compositions. The last step is a 2-partial composition.

To define a polytime parsing algorithm, we have to constraint Rule $\xrightarrow{\mathbf{C}}$ such that the size of the resulting type never exceeds the maximal size of the two original types. In this case the k-partial composition is called a k-functional composition. The k-functional composition of two types A and B is a k-partial composition such that $k \geq min(\frac{wd(A)}{2}, \frac{wd(B)}{2})$.

- **F (k-functional composition)** : For $k \in \mathbb{N}$,
  $\Gamma, Xq_1^{(n_1)} \cdots q_k^{(n_k)}, p_k^{(n_k+1)} \cdots p_1^{(n_1+1)}Y, \Delta \xrightarrow{\mathbf{F}} \Gamma, XY, \Delta$, if $q_i \leq_P p_i$ and n_i is even or if $p_i \leq_P q_i$ and n_i is odd, for $1 \leq i \leq k$ and if $wd(XY) \leq max(wd(Xq_1^{(n_1)} \cdots q_k^{(n_k)}), wd(p_k^{(n_k+1)} \cdots p_1^{(n_1+1)}Y)) = k + max(wd(X), wd(Y))$.

Lemma 3.10 will prove that this rule can replace $\xrightarrow{\mathbf{C}}$ if the final type is 1 because for any outerplanar graph (planar graph in which the vertices are placed on a single line) there always exists at least one vertex that is either isolated or only connected to its left or right neighbours (or to both of them) exclusively. This vertex is then composed with any of its neighbours if it is isolated (the type must be 1 and we use a 0-functional composition). If it is connected to only one neighbour, it is composed with the connected neighbour (if the width of the type is k, we use a k-functional composition). If it is connected to its two neighbours, we choose the one that interacts the most with the vertex. In the three cases the composition is functional.

Lemma 3.9 (property of outerplannar graphs). *Let (V, A) be a graph where the vertices are linearly ordered $V = v_1, \ldots, v_n$. The edges A are non-oriented and we write $A[i, j]$ if there exists an edge between v_i and v_j.*

If the graph is an outerplanar graph w.r.t. the linear order of V then $\exists$ a vertice only connected to its neighbours:
If $\forall i, j, k, l, 1 \leq i < j < k < l \leq n \ \neg(A[i, k] \wedge A[j, l])$ then
$\exists i, 1 \leq i \leq n$ such that $\forall j, (1 \leq j \leq i-2) \vee (i+2 \leq j \leq n), \neg A[i, j]$

Proof. By induction on n. We also prove that if $n \geq 3$, the vertex that is only connected to its neighbours is neither v_1 nor v_n. For 1 or 2 vertices, the property is obvious. For 3 vertices, the middle vertex always verifies the property (and is neither v_1 nor v_3). For $n > 3$ vertices, if v_1 and v_n are connected, this edge is remove. We look at edges that connect two vertices that are not neighbours. If none are found, every vertex is only connected to its neighbours and any vertex v_i, $2 \leq i \leq n-1$ is only connected to its neighbours (and is neither v_1 nor v_n). Otherwise, if it exists an edge between v_i and v_j with $i \leq j$ and $j - i \geq 2$ ($j - i < n$ because v_1 and v_n are not connected) and we consider recursively the subgraph corresponding to $v_i, \ldots, v_j$. It has a vertex v_k, $i < k < j$ only connected to its neighbours. v_k cannot be connected to any other vertex before v_i or after v_j. ∎

Lemma 3.10. *For $\Gamma \in (\mathcal{C}at_{(P,\leq_P)})^+$, $\Gamma \xrightarrow{(\mathbf{ME})^*} 1$ iff $\Gamma \xrightarrow{\mathbf{F}^*} 1$.*

Proof. If the length of Γ is 1, $\xrightarrow{\mathbf{E}}$, $\xrightarrow{\mathbf{M}}$ and $\xrightarrow{\mathbf{F}}$ cannot be applied and Γ must be 1. For $n \geq 2$, starting with a derivation $X_1, \ldots, X_n \xrightarrow{(\mathbf{ME})^*} 1$, we build a graph with n vertices corresponding to $X_1, \ldots, X_n$. We connect vertex i' to vertex j' if there exists at least one $\xrightarrow{\mathbf{E}}$ step $\Gamma', X'q^{(n')}, p^{(n'+1)}Y', \Delta' \xrightarrow{\mathbf{E}} \Gamma', X', Y', \Delta'$ in the derivation where $q^{(n')}$ is initially an atomic type of $X_{i'}$

and $p^{(n'+1)}$ is initially an atomic type of $X_{j'}$.
This graph is outerplanar. Lemma 3.9 says that it exists a vertex that is isolated or only connected to its neighbours. This vertex corresponds to a type X_i and each $\xrightarrow{\mathbf{E}}$ step on the atomic types of X_i concerns an atomic type of X_{i-1} or X_{i+1}. Let j be the number of $\xrightarrow{\mathbf{E}}$ concerning X_i and X_{i-1} ($j = 0$ if $i = 1$) and k be the number of $\xrightarrow{\mathbf{E}}$ concerning X_i and X_{i+1} ($k = 0$ if $i = n$). i and j can be both 0. We have $wd(X_i) = j + k$ because every atomic type is involved in one $\xrightarrow{\mathbf{E}}$ step (the derivation ends with type 1). If $j \geq k$, we can choose all the $\xrightarrow{\mathbf{E}}$ steps concerning X_i and X_{i-1}. If $j \leq k$, we can choose all the $\xrightarrow{\mathbf{E}}$ steps concerning X_i and X_{i+1}. If for instance we choose all the $\xrightarrow{\mathbf{E}}$ steps concerning X_i and X_{i-1} (the other case is similar) then we have $j \geq \frac{wd(X_i)}{2}$ and $\exists X'', Y''$ such that $X_{i-1} = X'' q_1^{(n_1)} \cdots q_j^{(n_j)}$ and $X_i = p_j^{(n_j+1)} \cdots p_1^{(n_1+1)} Y''$ with $q_l \leq_P p_l$ and n_l is even or $p_l \leq_P q_l$ and n_l is odd, for $1 \leq l \leq j$. Now, we remove the $\xrightarrow{\mathbf{E}}$ steps from the initial derivation together with the first $\xrightarrow{\mathbf{M}}$ step that concerns X_i. Now, we have a derivation of $X_1, \ldots, X_{i-2}, X''Y'', X_{i+2}, \ldots, X_n \xrightarrow{(\mathbf{ME})^*} 1$ with $n-1$ types that corresponds by induction hypothesis to a derivation using only $\xrightarrow{\mathbf{F}}$ steps. We prepend the step: $X_1, \ldots, X'' q_1^{(n_1)} \cdots q_j^{(n_j)}, p_j^{(n_j+1)} \cdots p_1^{(n_1+1)} Y'', \ldots, X_n \xrightarrow{\mathbf{F}} X_1, \ldots, X''Y'', \ldots, X_n$ and find a complete $\xrightarrow{\mathbf{F}^*}$ derivation. This composition is a j-functional composition because $j \geq wd(Y'')$ and $wd(X''Y'') \leq wd(X'') + j \leq j + max(wd(X''), wd(Y''))$ ■

Lemma 3.11. *For $\Gamma \in (\mathcal{C}at_{(P,\leq_P)})^+$, $p \in P^{(\mathbb{Z})}$: $\Gamma \xrightarrow{(\mathbf{ME})^*} p$ iff $\Gamma \xrightarrow{\mathbf{F}^*} p$.*

Proof. Because $X \leq_P p$ iff $Xp^r \leq_P 1$, the proof is similar to the proof of Lemma 3.10 but used a derivation of $\Gamma, s^r \xrightarrow{(\mathbf{ME})^*} 1$. ■

Example 3.12. *The first $\xrightarrow{\mathbf{C}}$ step of the previous example is not a $\xrightarrow{\mathbf{F}}$ step:*

$$\boxed{q'o^{ll}\underset{\sqcup}{q^l}, qp_2^l\pi_2^l}, \pi_2, p_2o^l \xrightarrow{\mathbf{C}} \boxed{q'o^{ll}p_2^l\underset{\sqcup}{\pi_2^l}, \pi_2}, p_2o^l \xrightarrow{\mathbf{C}} \boxed{q'o^{ll}\underset{\sqcup}{p_2^l}, \underset{\sqcup}{p_2o^l}} \xrightarrow{\mathbf{C}} q'$$

becomes

$$q'o^{ll}q^l, \boxed{qp_2^l\underset{\sqcup}{\pi_2^l}, \pi_2}, p_2o^l \xrightarrow{\mathbf{F}} q'o^{ll}q^l, \boxed{qp_2^l, p_2o^l} \xrightarrow{\mathbf{F}} \boxed{q'o^{ll}\underset{\sqcup}{q^l}, \underset{\sqcup}{qo^l}} \xrightarrow{\mathbf{F}} q'$$

COROLLARY 3.13. *$G = (\Sigma, I, s)$ generates a string $v_1 \cdots v_n$ iff for $1 \leq i \leq n$, $\exists X_i \in I(v_i)$, $\exists Y_i \in \mathcal{C}at_{(P,\leq_P)}$ and $\exists p \in P$ such that $\forall i, 1 \leq n, X_i \xrightarrow{\mathbf{I}^*} Y_i$, $Y_1, \cdots, Y_n \xrightarrow{\mathbf{F}^*} p$ and $p \leq_P s$.*

3.2. A parsing algorithm

A Cocke-Kasami-Younger parsing algorithm [6] consists in the following steps:

1. Search for the types associated to the n words by the lexicon.
2. For each word, add the types deduced with $\xrightarrow{\mathbf{I}^*}$.
3. Compute recursively the possible types associated to a contiguous segment of words using $\xrightarrow{\mathbf{F}}$.

The third step uses a function that takes the positions of the first and last words in the segment as parameters. The result is a set of types with a bounded width (i.e. by the maximum width of the types in the lexicon).

PROPERTY 3.14. *For a given grammar, this algorithm is n^3 (wrt the number n of words of input strings).*

EXAMPLE 3.15. *Parsing of "whom have you seen ?" The cell of line i (from the bottom) and column j contains the types of the fragment starting at the i^{th} word and ending at the j^{th} word.*

	...whom	*...have*	*...you*	*...seen*
seen...				$\{p_2 o^l\}$
you...			$\{\pi_2\}$	$\emptyset$
have...		$\{q p_2^l \pi_2^l\}$	$\{q p_2^l\}$	$\{q o^l\}$
whom...	$\{q' o^{ll} q^l\}$	$\emptyset$	$\{q' o^{ll} p_2^l\}$	$\{q', q' o^{ll} o^l\}$

The cell of line i (from the bottom) and column j contains the types of the fragment starting at the i^{th} word and ending at the j^{th} word. The sentence has two types (but only one is atomic) because $q' o^{ll} p_2^l, p_2 o^l \xrightarrow{\mathbf{F}} q'$ using 2-functional composition but also $q' o^{ll} p_2^l, p_2 o^l \xrightarrow{\mathbf{F}} q' o^{ll} o^l$ using 1-functional composition.

3.3. From pregroup grammars to context-free grammars

Using Corollary 3.13, we can deduce a context-free grammar from a pregroup grammar that generates the same language.

[6] In fact, this algorithm tests whether a sentence is grammatical. A real parsing algorithm must also reverse all its operations in order to give one or all possible parse trees.

DEFINITION 3.16. Let $G = (\Sigma, I, s)$ be a pregroup grammar on $Cat_{(P,\leq_P)}$. Let $N = \max(\{wd(X) \mid a \in \Sigma, X \in I(a)\} \cup \{1\})$ be the maximum width of the types used by G and $M = \max(\{|i| \mid a \in \Sigma, Xp^{(i)}Y \in I(a)\} \cup \{0\})$ the maximum of the absolute value of the exponents of the atomic types used by G. We denote by $\mathcal{C}at_{(P,\leq_P)}\langle N, M\rangle$ the set of types of $\mathcal{C}at_{(P,\leq_P)}$ with a width less or equal to N and where exponents are in $-M, \ldots, M$. $\mathcal{C}at_{(P,\leq_P)}\langle N, M\rangle$ is finite. We define the context-free grammar $\mathcal{G}(G) = \langle \Sigma, \mathcal{C}at_{(P,\leq_P)}\langle N, M\rangle, s, \mathcal{R}\rangle$ where:

$$\begin{array}{ll} \forall X \in \mathcal{C}at_{(P,\leq_P)}\langle N, M\rangle, a \in \Sigma & X \xrightarrow{\mathcal{R}} a \text{ if } \exists Y \in I(a), Y \xrightarrow{\mathbf{I}^*} X \\ \forall X, Y, Z \in \mathcal{C}at_{(P,\leq_P)}\langle N, M\rangle & X \xrightarrow{\mathcal{R}} Y, Z \text{ if } Y, Z \xrightarrow{\mathbf{F}} X \\ \forall p \in P & s \xrightarrow{\mathcal{R}} p \text{ if } p \leq_P s \end{array}$$

COROLLARY 3.17. *Let G be a pregroup grammar, $\mathcal{L}(G) = \mathcal{L}(\mathcal{G}(G))$*

EXAMPLE 3.18. *With the pregroup grammar such that $I(whom) = \{q'o^{ll}q^l\}$, $I(have) = \{qp_2^l\pi_2^l\}$, $I(you) = \{\pi_2\}$ and $I(seen) = \{p_2o^l\}$ and where $\Sigma = \{whom, have, you, seen\}$ and $P = \{\pi_2, p_2, o, q', s\}$ with $q' \leq_P s$. The maximum width N is 3 and the maximum absolute exponent M is 2. The set of terminals of the context-free grammar is Σ. The non-terminals are $\mathcal{C}at_{(P,\leq_P)}\langle 3, 2\rangle = \{1\} \cup \{p^{(i)} \mid p \in P, -2 \leq i \leq 2\} \cup \{p^{(i)}q^{(j)} \mid p, q \in P, -2 \leq i, j \leq 2\} \cup \{p^{(i)}q^{(j)}r^{(k)} \mid p, q, r \in P, -2 \leq i, j, k \leq 2\}$. The start symbol is s. The rules are divided in 3 parts. For the terminals: $q'o^{ll}q^l \xrightarrow{\mathcal{R}} whom$, $qp_2^l\pi_2^l \xrightarrow{\mathcal{R}} have$, $\pi_2 \xrightarrow{\mathcal{R}} you$ and $p_2o^l \xrightarrow{\mathcal{R}} seen$ and no other because it is not possible to apply Rule $\xrightarrow{\mathbf{I}}$ on $q'o^{ll}q^l$, $qp_2^l\pi_2^l$, π_2 or p_2o^l. The binary rules correspond to all the possible applications of Rule $\xrightarrow{\mathbf{F}}$ to two types in $\mathcal{C}at_{(P,\leq_P)}\langle 3, 2\rangle$. For instance, we find $o \xrightarrow{\mathcal{R}} o\pi_2^l p_2, p_2^r\pi_2$ but also $o\pi_2^l\pi_2 \xrightarrow{\mathcal{R}} o\pi_2^l p_2, p_2^r\pi_2$ and $o\pi_2^l p_2 \xrightarrow{\mathcal{R}} o\pi_2^l p_2, 1$ and $1 \xrightarrow{\mathcal{R}} o^r\pi_2^l p_2, p_2^r\pi_2 o^{rr}$ and $1 \xrightarrow{\mathcal{R}} 1, 1$. The last rules for primitive types less or equal to s: $s \xrightarrow{\mathcal{R}} s$ (not very useful) and $s \xrightarrow{\mathcal{R}} q'$.*

4. Lambek calculus

4.1. Lambek calculus allowing empty premises

DEFINITION 4.1. The types *Tp* of Lambek calculus are generated from a set of atomic types P by three binary connectives "/" (over), "\" (under) and "•" (product). A sequent $\Gamma \vdash A$ is composed of a sequence of types Γ, possibly empty, which is the antecedent configuration and a succedent

type A. We write $\Gamma \vdash_{L_\emptyset} A$ when $\Gamma \vdash A$ can be deduced in the following system:

$$\frac{}{A \vdash A}\,\mathbf{Ax} \qquad \frac{\Gamma \vdash A \quad \Delta_1, A, \Delta_2 \vdash B}{\Delta_1, \Gamma, \Delta_2 \vdash B}\,\mathbf{Cut}$$

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \bullet B, \Delta \vdash C} \bullet L \qquad \frac{\Gamma \vdash A \quad \Delta_1, B, \Delta_2 \vdash C}{\Delta_1, B/A, \Gamma, \Delta_2 \vdash C} /L \qquad \frac{\Gamma \vdash A \quad \Delta_1, B, \Delta_2 \vdash C}{\Delta_1, \Gamma, A \setminus B, \Delta_2 \vdash C} \setminus L$$

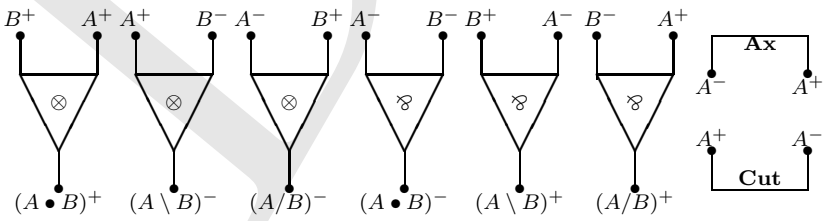
$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \bullet B} \bullet R \qquad \frac{\Gamma, B \vdash A}{\Gamma \vdash A/B} /R \qquad \frac{A, \Gamma \vdash B}{\Gamma \vdash A \setminus B} \setminus R$$

4.2. Proof-nets for Lambek calculus

Lambek calculus allowing empty premises is equivalent to the intuitionistic multiplicative fragment of cyclic linear logic [8, 18, 1] and we use proof-nets and modules rather than sequent calculus.

Definition 4.2 (Link, proof-structure). A *polarized type* is a couple constituted of a type in *Tp* and a polarity in $\{+, -\}$. Polarities are either *input* denoted by $-$ or *output* denoted by $+$. For $X \in Tp$, we define $(A^-)^\perp = A^+$ and $(A^+)^\perp = A^-$. *Sequents* are circular lists of polarized type. *Intuitionistic sequents* are circular lists of polarized types where one and only one type is output. They are written $A_1, \cdots, A_n \vdash A$ where A^+ is the output type and $A_1^-, \cdots, A_n^-$ are the input types: The circular list is cut between A^+ and A_1^-.

- A *link* is a drawing that connects (the occurrences of) polarized types. There are 8 drawings: three $\otimes$-links, three $\parr$-links, a **Ax**-link and a **Cut**-link:



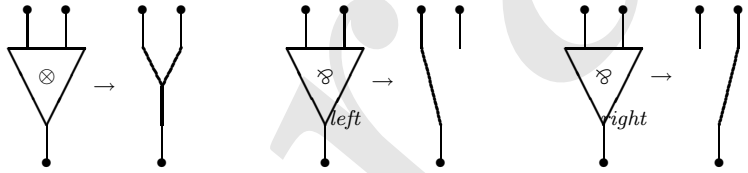
  The polarized types situated above the link are the *premises*, the ones under the link are the *conclusions*.

- A *proof-structure* is a planar drawing where vertices are polarized types and drawings are links such that a polarized type is the premise of at most one link and the conclusion of exactly one link. The polarized types that are not the premise of a link are the conclusions of the proof-structure. An *intuitionistic proof-structure* is a proof-structure such that the conclusions form an intuitionistic sequent (there is exactly one output type). The conclusions must be on the unbounded face (not inside the proof-structure but on the border).

Definition 4.3 (Proof-net). A *proof-net* is an intuitionistic proof-structure that corresponds to a sequential proof in Lambek calculus.

Remark. Not every intuitionistic proof-structure is a proof-net. But, a criteria exists that tells us whether a proof-structure is a proof-net. In this article, we use the Danos-Regnier criterion [6].

Definition 4.4 (Switching). A switching of a proof-structure is a selection for every $\parr$-link between *left* or *right* position. This switching induces a planar graph by replacing each link by the edges shown below:

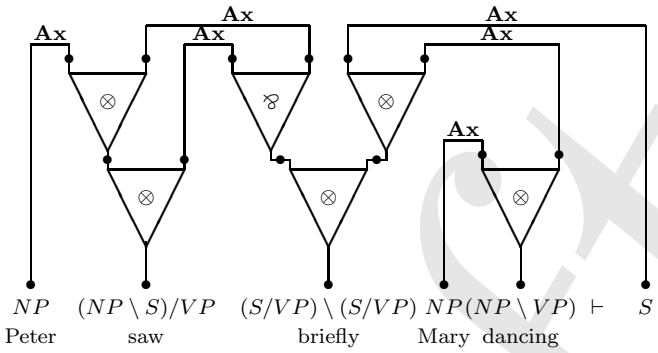


Definition 4.5 (Correct proof-structure). A proof-structure is correct (it verifies the correctness criterion) if and only if for every switching, the graph is acyclic and connected.

Theorem 4.6. *The correct intuitionistic proof-structures are the proof-nets.*

This is the usual correctness and sequentialization theorem of proof-nets. In [8, 6], a theorem is given for classical linear logic. Here, it must be taken into account that the proofs are also intuitionistic and cyclic [17]. This is the role of polarities and planarity.

Example 4.7. *Below is a proof-net corresponding to the sentence "Peter saw briefly Mary dancing". The types of each word are supposed to be stored in the lexicon. The type of internal nodes is not mentioned. This proof-structure verifies the correctness criterion: there is only one $\parr$-link and the two possible switchings lead to two acyclic and connected graphs. It is also a valid sequent in Lambek calculus.*



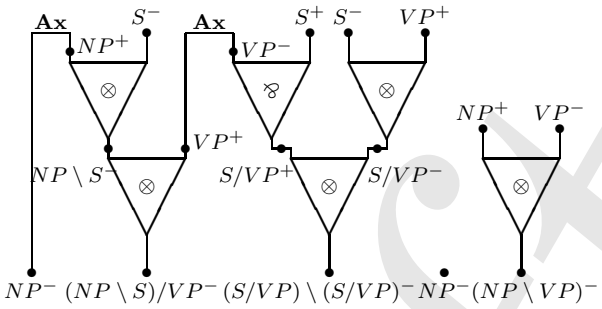
5. Parsing using flat interface calculus

5.1. Module calculus

A proof-net is a correct proof-structure. To build a proof-net from a sequent, attention must be given to the different possible proof-structures corresponding to this sequent, and their correction. Because we want to introduce a notion of partial composition similar to the partial composition of pregroup types, we have to study the parts of proof-structures. These objects are called modules [6, 13, 12, 2, 3].

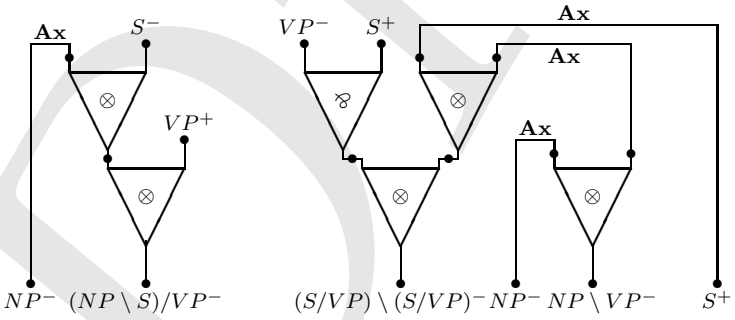
Definition 5.1 (Module, border). A *module* is a "proof-structure with premises": a planar drawing where each polarized type is the premise of at most one link, and the conclusion of at most one link. At least one polarized type must not be the conclusion of a link. The polarized types that are not the premises of links are the *conclusions* of the module. The polarized types that are not the conclusion of a link are its *premises*. The *border* of the module is the list (from left to right on a drawing of a module) of its premises. We write $M[\overset{A_1}{\bullet}\cdots\overset{A_k}{\bullet}]$ a module with a border $A_1, \ldots, A_k$. We write $\overset{A}{\bullet}$ the module with only one vertex A and no link. The border is A. The list of conclusions is also A.

Example 5.2. *Below is a module composed of five $\otimes$-links, one $\parr$-link and two axiom links. Six polarized types are the premise and the conclusion of some links, four are only conclusion, six are only premise and one NP^- is neither premise nor conclusion. Thus, the conclusions of the module are, from left to right, NP^-, $(NP \setminus S)/VP^-$, $(S/VP) \setminus (S/VP)^-$, NP^- and $(NP\backslash VP)^-$. The premises are, from left to right, S^-, S^+, S^-, VP^+, NP^-, NP^+ and VP^-. The type NP^- that is not connected to a link appeared both as a conclusion and as a premise of the module:*



We can split a proof-structure in several modules by splitting the circular list of conclusions in several contiguous parts. Axiom links that connect the parts are cut. Thus the modules have premises (one for each cut axiom link) that are not in the initial proof-structure.

Example 5.3. *For instance, we can split the proof-net of Example 4.7 in two modules corresponding to "Peter saw" and "briefly Mary dancing* $\vdash S$*". In this example, we split the conclusions between succedent* S *and the first type* NP *(remember that the list is circular) and between the second type* $(NP \setminus S)/VP$ *and the third type* $(S/VP) \setminus (S/VP)$*. The left module has two conclusions* NP^- *and* $(NP \setminus S)/VP^-$ *and two premises* VP^+ *and* S^-*. The right module has four conclusions* $(S/VP) \setminus (S/VP)^-$*,* NP^-*,* $NP \setminus VP^-$ *and* S^+ *and two premises* S^+ *and* VP^-*:*



Definition 5.4 (Switching, correct module). As for proof-structures, a *switching* of a module is a selection for every ⅋-link between *left* or *right* position. This switching induces a planar graph by replacing each link by the edges shown in Definition 4.4.

A module is *correct* (it verifies the correctness criterion for modules) if and only if for every switching, the graph is acyclic and each vertex is connected to a polarized type of the border.

DEFINITION 5.5 (Orthogonal modules). A module M with conclusions $A_1, \ldots, A_k$ and a border $B_1, \ldots, B_n$ is *orthogonal* to a module N with conclusions $C_1, \ldots, C_m$ and border $B_n^\perp, \ldots, B_1^\perp$, (notation $M \perp N$) if and only if the proof-structure obtained by linking the n premises of M to the n premises of N by n axiom links is a proof-net.

THEOREM 5.6. *If $M \perp N$ then M and N are correct modules.*

PROOF. The proof is straightforward. If $M \perp N$ then the proof-structure obtained by linking the n premises of M to the n premises of N by n axiom links is a proof-net: for every switching of the $\parr$-links, the graph is acyclic and connected. Thus, for every switching of the $\parr$-links of M (or N), the graph is acyclic and every vertex is connected to (at least) one type of the border: M (or N) is a correct module. ■

In fact, Lambek calculus proof search can be performed using the following rules on list of correct modules[7]:

- $\otimes$ **($\otimes$ introduction)** : $..., M[\cdots \overset{A^p}{\bullet} \cdots], ... \longrightarrow ..., M[\cdots \overset{B^q}{\bullet}\ \overset{C^r}{\bullet} \cdots], ...$ if $A^p = (C \bullet B)^+$, $q = +$ and $r = +$ or $A^p = (B \setminus C)^-$, $q = +$ and $r = -$ or $A^p = (B/C)^-$, $q = -$ and $r = +$. On the right hand side, A^p is replaced by B^q and C^r in the border.
- $\parr$ **($\parr$ introduction)** : $..., M[\cdots \overset{A^p}{\bullet} \cdots], ... \longrightarrow ..., M[\cdots \overset{B^q}{\bullet}\ \overset{C^r}{\bullet} \cdots], ...$ if $A^p = (B \bullet C)^-$, $q = -$ and $r = -$ or $A^p = (C \setminus B)^+$, $q = +$ and $r = -$ or $A^p = (C/B)^+$, $q = -$ and $r = +$. On the right hand side, A^p is replaced by B^q and C^r in the border.
- **A (axiom)** : $..., M[\cdots \overset{A^p}{\bullet}\ \overset{A^q}{\bullet} \cdots], ... \xrightarrow{\mathbf{A}} ..., M[\cdots \overset{\mathbf{Ax}}{A^p\ \ A^q} \cdots], ...$ if A is atomic, if $p = +$ and $q = -$ or $p = -$ and $q = +$ and if the result is a correct module[8]. On the right hand side, A^p and A^q are removed from the border.
- **M (merge)** : $..., M[\overset{A_1}{\bullet} \cdots \overset{A_k}{\bullet}], N[\overset{B_1}{\bullet} \cdots \overset{B_l}{\bullet}], ... \xrightarrow{\mathbf{M}} ..., M[\overset{A_1}{\bullet} \cdots \overset{A_k}{\bullet}] N[\overset{B_1}{\bullet} \cdots \overset{B_l}{\bullet}], ...$ The modules are juxtaposed. On the right hand side, the border is the concatenation of the two borders of the left hand side.

[7] One can easily check that the rules preserve correctness on modules.

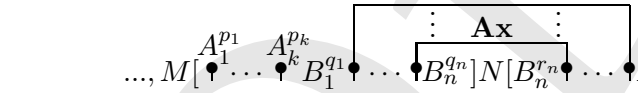
[8] In particular, the resulting border must not be empty.

THEOREM 5.7. *If $A_1, \ldots, A_n$ are types and S is an atomic type,*
$A_1, \ldots, A_n \vdash_{L_\emptyset} S$ *iff* $\exists M[\overset{S^-}{\bullet}]$ *such that* $\overset{A_1^-}{\bullet}, \ldots, \overset{A_n^-}{\bullet} \xrightarrow{(\otimes\parr\mathbf{AM})^*} M[\overset{S^-}{\bullet}]$

PROOF. In fact, if $A_1, \ldots, A_n \vdash_{L_\emptyset} S$, there exists a cut-free proof where the axioms are limited to atomic types that corresponds to a proof-net without **Cut**-link and where **Ax**-links are only on polarized atomic types. We consider the module corresponding to the n input types. This is a correct module with a border equal to the single polarized atomic type S^- that can be built using the above rules. Conversely, we add S^+ to the list of the conclusions of $M[\overset{S^-}{\bullet}]$ and an axiom and we have a proof-structure. This proof-structure is correct because $M[\overset{S^-}{\bullet}]$ is a correct module: for any switching of the proof-structure, the graph is acyclic because the corresponding switching of the module gives an acyclic graph. It is connected because for the corresponding switching of the module every vertex of the graph is connected to the vertex of the atomic type S^-. ■

Using the same technique as for pregroups, we can merge together n $\xrightarrow{\mathbf{A}}$ and one $\xrightarrow{\mathbf{M}}$ into a n-functional composition:

- **F (n-functional composition)** :
$..., M[\overset{A_1^{p_1}}{\bullet}\ldots\overset{A_k^{p_k}}{\bullet}\overset{B_1^{q_1}}{\bullet}\ldots\overset{B_n^{q_n}}{\bullet}], N[\overset{B_n^{r_n}}{\bullet}\ldots\overset{B_1^{r_1}}{\bullet}\overset{C_1^{s_1}}{\bullet}\ldots\overset{C_l^{s_l}}{\bullet}], ... \xrightarrow{\mathbf{F}}$



$..., M[\overset{A_1^{p_1}}{\bullet}\ldots\overset{A_k^{p_k}}{\bullet}B_1^{q_1}\bullet\ldots\bullet B_n^{q_n}]N[B_n^{r_n}\bullet\ldots\bullet B_1^{r_1}\overset{C_1^{s_1}}{\bullet}\ldots\overset{C_l^{s_l}}{\bullet}], ...$
if $k + l \leq n + max(k, l)$, if $\forall i, A_i, B_i, C_i$ are atomic, if $\forall i$, $q_i = +$ and $r_i = -$ or $q_i = -$ and $r_i = +$ and if the result is a correct module[9]. On the right hand side, $\forall i$, $B_i^{q_i}$ and $B_i^{r_i}$ are removed from the border.

THEOREM 5.8. *If $A_1, \ldots, A_n$ are types and S is an atomic type,*
$A_1, \ldots, A_n \vdash_{L_\emptyset} S$ *iff* $\exists N_1, \ldots, N_n, M_1, \ldots, M_n, M[\overset{S^-}{\bullet}]$ *such that*

$$\begin{cases} \forall i, 1 \leq i \leq n : \overset{A_i^-}{\bullet} \xrightarrow{(\otimes\parr)^*} N_i \\ \forall i, 1 \leq i \leq n : N_i \xrightarrow{\mathbf{A}^*} M_i \\ M_1, \ldots, M_n \xrightarrow{\mathbf{F}^*} M[\overset{S^-}{\bullet}] \end{cases}$$

[9] In particular, the resulting border must not be empty ($k + l \geq 1$).

PROOF. The proof is the same as for pregroup grammars because the axiom links of a proof-net define an outerplanar graph where the atomic types that appear in $A_1, \ldots, A_n$ are the vertices. ■

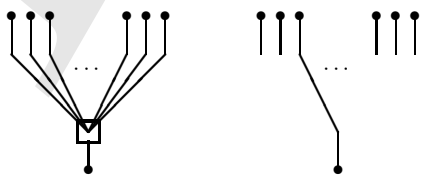
Remark. This module calculus using functional composition does not give a polytime algorithm for parsing Lambek calculus because the size of the modules that are composed using $\xrightarrow{\mathbf{F}}$ is not bound for a particular grammar. This problem leads to the search of a more synthetic representation for modules bound for a particular border. This issue is studied in the next section on flat interfaces that characterize correct modules.

5.2. Interface calculus

We want to characterize the connectiveness of a module from its border point of view. A first approch was presented in [6]. In the context of commutative linear logic, the algorithm computes the set of partitions of the border of the module induced by every switching of the $\parr$-links. If a module has k $\parr$-links, this is done in 2^k steps even if the number of premises is bounded. This technique can be used here but we prefer to introduce flat interfaces which is a more compact representation of the connectiveness properties of the border of a correct module.

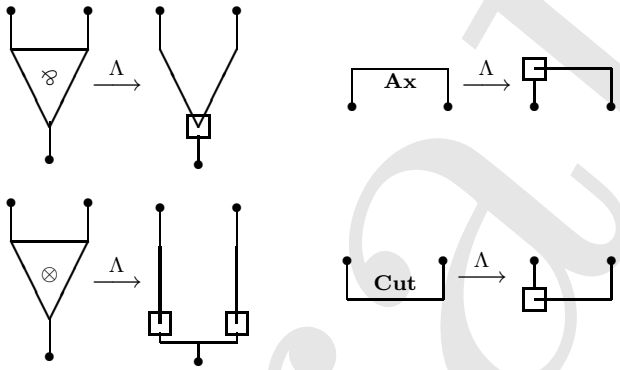
DEFINITION 5.9 (Interface). An interface is a quadruple $\langle B, V, L, m\rangle$ where:

- $B = A_1, \ldots, A_k$, the *border* of the interface, is a non-empty list of polarized types.
- $\langle V, L\rangle$, the *net* of the interface, is a (non-planar) drawing made from vertices V and k-ary $\parr$-links L shown below. This figure shows also one of its k switching positions. Each link is connected to a particular vertex called the *conclusion* and to a multiset of vertices that are its *premises*. The arity is the number of premises and must be greater or equal to 1. Arity must not be necessarily equal for all the $\parr$-links of an interface.



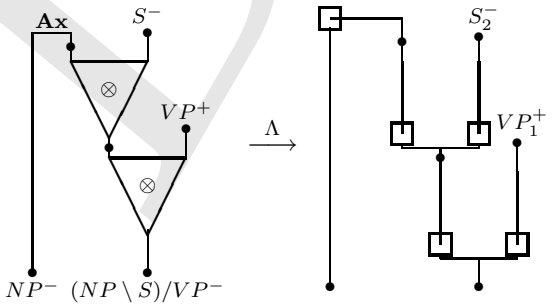
- m, the *mapping* of the interface, is a function $m : \{1, \ldots, k\} \to V$ that maps each type of the border to a (not necessarily different) vertex of the net. To simplify the notation, we write $m(A_k)$ in place of $m(k)$.

Definition 5.10 (Natural interface). The net of the *natural interface* of a module M is defined by transforming each polarized type into a vertex and each link by the corresponding local net defined below. A $\parr$-link is transformed into a binary $\parr$-link, a $\otimes$-link into 2 unary $\parr$-links and a **Cut** or **Ax** link into a single unary $\parr$-link:



The border of the interface is the border of M. The mapping of the interface associates to each premise of M the corresponding vertex of the net. The other vertices are not labelled by any polarized type. This natural interface of M is denoted by $\Lambda(M)$.

Example 5.11. *Below is the natural interface of the left module of Example 5.3. There are six vertices, three unary $\parr$-links, one binary $\parr$-link. The border has two types denoted by VP_1^+ and S_2^- (because the net does not need to be planar, a number gives the position of a polarized type in the border) that are mapped to two different vertices:*



An interface looks like a module except that it does not have a list of conclusions, the net is not necessarily planar, vertices can be connected to more than two links, only the polarized types of the border are associated to a vertex and one vertex can be associated to zero, one of more polarized types. However, a correctness criterion can be defined on it.

Definition 5.12 (Switching, correct interface). A switching of an interface is a selection for every k-ary $\parr$-link from one of its k positions. This switching induces a graph by replacing each link by an edge following its switching position (see Definition 5.9).

An interface is *correct* (it verifies the correctness criterion for interfaces) if and only if for every switching, the graph is acyclic and each component has at least one vertex that is mapped to a polarized type of the border.

Remark. The correctness of interfaces and of modules are very similar. Moreover, the natural interface $\Lambda(M)$ of a correct module M is a correct interface.

Definition 5.13 (Orthogonal interfaces). $I_1 = \langle (A_1, \ldots, A_n), V_1, L_1, m_1 \rangle$ and $I_2 = \langle (A_n^\perp, \ldots, A_1^\perp), V_2, L_2, m_2 \rangle$, two correct interfaces are *orthogonal* (notation $I_1 \perp I_2$) if and only if for every switching of the nets of I_1 and I_2, the graph obtained from the union of the two graphs by connecting the vertices $m_1(A_1)$ to $m_2(A_1)$, $\ldots$, $m_1(A_n)$ to $m_2(A_n)$ is acyclic and connected.

Example 5.14. *For instance, because the modules of Example 5.3 are defined by splitting the proof-net of Example 4.7, the natural interfaces of the modules are orthogonal.*

Theorem 5.15. *Let M_1 and M_2 be two correct modules. $M_1 \perp M_2$ iff $\Lambda(M_1) \perp \Lambda(M_2)$*

Proof. The proof is straightforward. ∎

Definition 5.16 (Equivalent interfaces). Two correct interfaces I_1 and I_2 with the same border are equivalent (notation $I_1 \equiv I_2$) if and only if for every correct interface J: $I_1 \perp J \iff I_2 \perp J$

The next theorem says that we can use any interface that is equivalent to the natural interface of a correct module for proving that we have a proof-net.

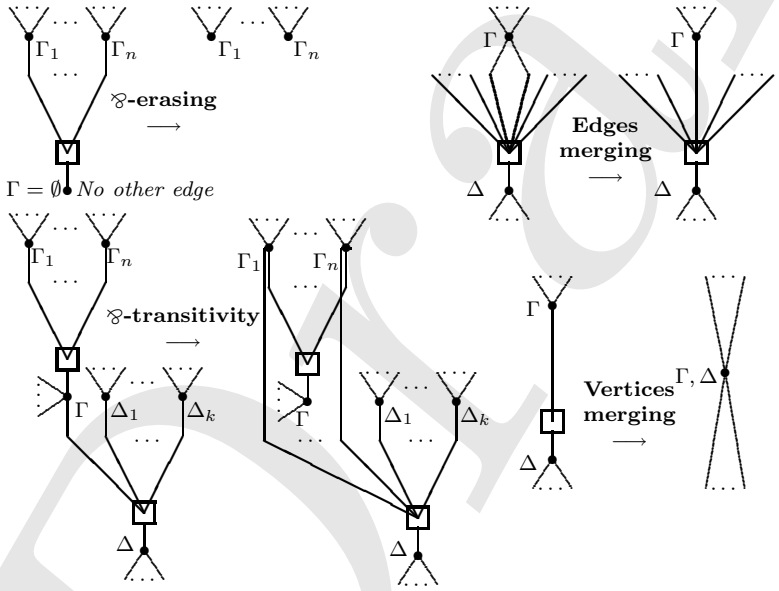
Theorem 5.17. *Let M_1 and M_2 be two correct modules and I_1 be a correct interface such that $I_1 \equiv \Lambda(M_1)$ then $M_1 \perp M_2$ iff $I_1 \perp \Lambda(M_2)$*

Proof. This is a direct consequence from Definition 5.16 and Theorem 5.15. ∎

5.3. Flat Interface

For a particular border, there exists an infinite set of modules thus an infinte set of natural interfaces. But only a finite set of equivalent classes exists. Here, we prove this property by showing that each interface is equivalent to a flat interface and that the set of flat interfaces with a given border is finite.

Definition 5.18 (Flat interface transformations). Flat interface transformations are transformations of correct interfaces. The four rules are $\parr$-erasing, $\parr$-transitivity, Edge merging and Vertices merging. We denote by $I_1 \xrightarrow{\mathcal{F}} I_2$ the transformation of I_1 into I_2 using one of these rules. We say that a correct interface is a *flat interface* when it is not possible to apply one of the flat interface transformation to it[10]:

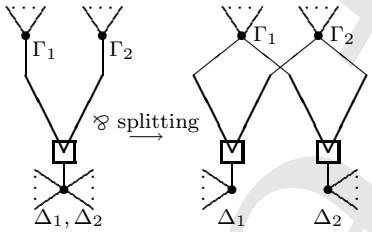


The transformations need some explanations. Capital Greek letters like Γ or Δ represent types from the border that are mapped to a vertex. For all rules except $\parr$-erasing, this mapping is free. In $\parr$-erasing, the conclusion of the n-ary $\parr$-link must not be mapped to any type of the border. Moreover, this vertex must be neither the premise nor the conclusion of another $\parr$-link. In Vertices merging, a unary $\parr$-link disappears and two vertices are merged.

[10]The vertices of the net of a flat interface cannot be both the premise and the conclusion of one or two $\parr$-links of the interface.

THEOREM 5.19. *On the net of a correct interface, flat interface transformations preserve correctness and are compatible with interface equivalence (they do not change the class of the resulting interface).*

PROOF. The proof of the correctness of the right hand side when the left hand side is correct is straightforward. For each rule, we can prove also that the nets on both sides have the same orthogonal. In fact, we use the following local key transformation which preserves correctness and equivalence in the interfaces (compatibility):



A local transformation that preserves correctness and compatibility does not change the equivalence class of a correct interface (congruence). Thus this key transformation is a small brick that is used to prove the correctness and the compatibility of $\parr$-transitivity, Edges and Vertices merging. ■

THEOREM 5.20. *The rewriting system on correct interfaces that performs flat interface transformations always terminates. We write $\mathcal{NF}(I)$ an arbitrary flat interface of I: if $J = \mathcal{NF}(I)$ then J is a flat interface and $I \xrightarrow{\mathcal{F}^*} J$. Of course, $I \equiv J$*

PROOF. For each transformation, the number of vertices and $\parr$-links of the net cannot increase. For $\parr$-erasing and Vertices merging, it decreases: an infinite reduction can not involve an infinte number of such steps. Edges merging merges two edges of the same P-link. Thus, if an infinite reduction exists for an interface with an infinite number of Edges merging and an infinite number of $\parr$-transitivity, it exists an infinite reduction with only a finite number of Edges merging (and an infinite number of $\parr$-transitivity). A problem might occur with the use of $\parr$-transitivity in an infinte reduction. This is not possible because the interface is correct: it does not have a cycle following from the conclusion to the premise of a circular list of $\parr$-links. ■

Remark. $\xrightarrow{\mathbf{F}^*}$ is not confluent ($\mathcal{NF}(I)$ is not uniquely defined). It is possible to add some other rules so that the resulting rewriting system is normalizing and confluent [3] but this complex system is not necessary here.

THEOREM 5.21. *For a given border, the set of flat interfaces is finite.*

PROOF. Let $\langle B, V, N, m\rangle$ be a normalized flat interface. In N, each node is either isolated, the premise of one or more $\parr$-links or the conclusion of one or more $\parr$-links.
Each vertex that is the premise of one or more $\parr$-links must be connected to the border through m. Otherwise it is possible to find a switching such that the vertex is isolated (this is not possible because the interface is correct). This is also the case for isolated vertices and for vertices that are the conclusion of only one $\parr$-link (because of $\parr$-erasing). Thus the number of vertices that are isolated, the premise of one or more $\parr$-links or the conclusion of only one $\parr$-link is less than or equal to the length of B.
The number of vertices that are the conclusion of two or more $\parr$-links cannot be greater than or equal to the number of vertices that are premises of one or more $\parr$-links because otherwise, there exists a switching of the $\parr$-links that creates a cycle (this is not possible because the interface is correct). Thus, their number is less than the length of B.
Finally, the premises of all the $\parr$-links with the same conclusion are the premises of only one of these $\parr$-links otherwise there exists a cycle for a particular switching of the $\parr$-links . Thus the number of $\parr$-links with the same conclusion is also bounded by the length of B. ■

The following theorem says that there is only one flat interface corresponding to a correct module with a border of length one.

COROLLARY 5.22. *Let $M[\overset{A}{\bullet}]$ be a correct module with a border reduced to a single polarized type then $\exists v_A$ such that*

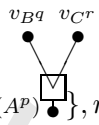
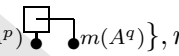
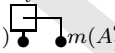
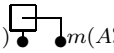
$$\mathcal{NF}(M[\overset{A}{\bullet}]) = \langle (A), \{v_A\}, \emptyset, \{A \mapsto v_A\}\rangle$$

We write $\nabla(A)$ this flat interface.

5.4. Parsing using flat interfaces

Flat interface calculus is module calculus where flat interfaces are used in place of modules. The rules of Section 5.1 are adapted to flat interfaces.

- $\otimes$ **($\otimes$ introduction)** :
  $..., \langle(\ldots, A^p, \ldots), V, E, m\rangle, ... \xrightarrow{\otimes} ..., \langle(\ldots, B^q, C^r, \ldots), V, E, m'\rangle, ...$
  If $A^p = (C \bullet B)^+$, $q = +$ and $r = +$ or $A^p = (B \setminus C)^-$, $q = +$ and $r = -$ or $A^p = (B/C)^-$, $q = -$ and $r = +$. $m'(B^q) = m'(C^r) = m(A)$ and $\forall X \neq A^p, m'(X) = m(X)$

- $\parr$ (**$\parr$ introduction**) :
  $..., \langle (\ldots, A^p, \ldots), V, E, m\rangle, ... \xrightarrow{\parr}$
  $..., \mathcal{NF}(\langle (\ldots, B^q, C^r, \ldots), V \cup \{v_{B^q}, v_{C^r}\}, E \cup \{m(A^p)$  $\}, m'\rangle), ...$
  If $A^p = (B \bullet C)^-$, $q = -$ and $r = -$ or $A^p = (C \setminus B)^+$, $q = +$ and $r = -$ or $A^p = (C/B)^+$, $q = -$ and $r = +$. Here, v_{B^q} and v_{C^r} are two new vertices. $m'(B^q) = v_{B^q}$, $m'(C^r) = v_{C^r}$ and $\forall X \neq A^p, m'(X) = m(X)$
- **A (axiom)** : $..., \langle (\ldots, A^p, A^q, \ldots), V, E, m\rangle, ... \xrightarrow{\mathbf{A}}$
  $..., \mathcal{NF}(\langle (\ldots, \ldots), V, E \uplus \{m(A^p)$  $m(A^q)\}, m'\rangle), ...$
  If A is atomic, if $p = +$ and $q = -$ or $p = -$ and $q = +$ and if the resulting interface (before normalization) is correct.
- **M (merge)** : $..., \langle B, V, E, m\rangle, \langle B', V', E', m'\rangle, ... \xrightarrow{\mathbf{M}}$
  $..., \langle (B \circ B'), V \uplus V', E \uplus E', m \uplus m'\rangle, ...$
  $B \circ B'$ is the concatenation of B and B', $V \uplus V'$, $E \uplus E'$ are the disjoint unions of V and V' and of E and E'. $m \uplus m'$ is the function that maps X to $m(X)$ if X is in B or to $m'(X)$ if X is in B'.
- **F (n-functional composition)** :
  $..., \langle (A_1^{p_1}, \ldots, A_k^{p_k}, B_1^{q_1}, \ldots, B_n^{q_n}), V, E, m\rangle,$
  $\langle (B_n^{r_n}, \ldots, B_1^{r_1}, C_1^{s_1}, \ldots, C_l^{s_l}), V'E', m'\rangle, ...$
  $\xrightarrow{\mathbf{F}} ..., \mathcal{NF}(\langle (A_1^{p_1}, \ldots, A_k^{p_k}, C_1^{s_1}, \ldots, C_l^{s_l}), V \uplus V', E \uplus E' \uplus E'', m''\rangle), ...$
  If $k + l \leq n + max(k, l)$, if $\forall i, A_i, B_i, C_i$ are atomic, if $\forall i$, $q_i = +$ and $r_i = -$ or $q_i = -$ and $r_i = +$ and if the result is a correct interface. Here, $E'' = \{m(A_1^{p_1})$  $m(A_1^{q_1}), \ldots, m(A_n^{p_n})$  $m(A_n^{q_n})\}$ and m'' is $m \uplus m'$ restricted to $A_1^{p_1}, \ldots, A_k^{p_k}, C_1^{s_1}, \ldots, C_l^{s_l}$.

THEOREM 5.23. *If $A_1, \ldots, A_n$ are types and S is an atomic type,*
$A_1, \ldots, A_n \vdash_{L_\emptyset} S$ *iff* $\nabla(A_1^-), \ldots, \nabla(A_n^-) \xrightarrow{(\otimes \parr \mathbf{AM})^*} \nabla(S^-)$

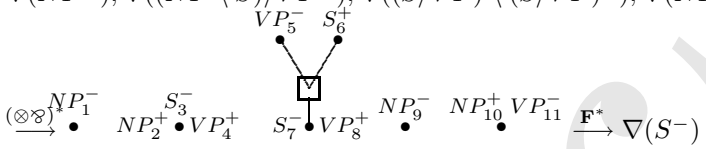
PROOF. We use flat interface in place of module in Theorem 5.7. ∎

THEOREM 5.24. *If $A_1, \ldots, A_n$ are types and S is an atomic type,*
$A_1, \ldots, A_n \vdash_{L_\emptyset} S$ *iff* $\exists I_1, \ldots, I_n, J_1, \ldots, J_n$ *flat interfaces such that*

$$\begin{cases} \forall i, 1 \leq i \leq n : \nabla(A_i^-) \xrightarrow{(\otimes \parr)^*} I_i \\ \forall i, 1 \leq i \leq n : I_i \xrightarrow{\mathbf{A}^*} J_i \\ J_1, \ldots, J_n \xrightarrow{\mathbf{F}^*} \nabla(S^-) \end{cases}$$

PROOF. Similar to the proof of Corollary 3.13. ∎

EXAMPLE 5.25. *For the proof-net of Example 4.7:*
$\nabla(NP^-), \nabla((NP \setminus S)/VP^-), \nabla((S/VP) \setminus (S/VP)^-), \nabla(NP^-), \nabla(NP \setminus VP^-)$



The reduction does not use $\xrightarrow{\mathbf{A}}$ *steps.*

5.5. From Lambek calculus to context-free grammars

Using flat interfaces, it is possible to transform a Lambek grammar into a context-free grammar. The construction is very similar to the construction for pregroup grammars presented in the first part of the article.

DEFINITION 5.26. Let $G = (\Sigma, I, S)$ be a Lambek grammar on Tp (atomic types P). Let $N = \max(\{size(X) \mid a \in \Sigma, X \in I(a)\} \cup \{1\})$ be the maximum number of atomic types of the types used by G. We denote by $\mathcal{I}\langle N, P\rangle$ the set of flat interfaces with a border in $P^{\{1,\ldots,N\}}$[11]. This set is finite by Theorem 5.21. We define the context-free grammar $\mathcal{G}(G) = \langle \Sigma, \mathcal{I}\langle N, P\rangle, \nabla(S^-), \mathcal{R}\rangle$ where:

$$\forall X \in \mathcal{I}\langle N, P\rangle, a \in \Sigma \quad X \xrightarrow{\mathcal{R}} a \begin{cases} \text{if } \exists A \in I(a), J \in \mathcal{I}\langle N, P\rangle \text{ such that} \\ \nabla(A^-) \xrightarrow{(\otimes\parr)^*} J \text{ and } J \xrightarrow{\mathbf{A}^*} X \end{cases}$$

$$\forall X, Y, Z \in \mathcal{I}\langle N, P\rangle \quad X \xrightarrow{\mathcal{R}} Y, Z \text{ if } Y, Z \xrightarrow{\mathbf{F}} X$$

THEOREM 5.27. *Let G be a Lambek grammar, $\mathcal{L}(G) = \mathcal{L}(\mathcal{G}(G))$*

PROOF. This is a direct consequence of Theorem 5.24. ∎

6. Conclusion

This paper introduces the notion of partial composition for pregroups and for modules of multiplicative cyclic linear logic or their flat interfaces. A restricted form of partial composition, named functional composition, valid for both formalisms enables us to transform into a context-free grammar a pregroup grammar or a grammar based on Lambek calculus allowing empty premises.

[11] The length of the border is limited to N.

References

[1] Michele Abrusci. Phase semantics and sequent calculus for pure non-commutative classical linear logic. *Journal of Symbolic Logic*, 56(4):1403–1451, December 1991.

[2] Denis Béchet. Minimality of the correctness criterion for multiplicative proof nets. *Mathematical Structures in Computer Science*, 8:543–558, 1998.

[3] Denis Béchet. Incremental parsing of Lambek calculus using proof-net interfaces. In ACL/SIGPARSE, editor, *Proceedings of the Eigth International Workshop on Parsing Technologies, Nancy, France, April 2003*, pages 31–42. INRIA, April 2003.

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

[5] Wojciech Buszkowski and Katarzyna Moroz. PTIME transformation of pregroup grammar into cfg and pda. In *37th Pozna Linguistic Meeting (PLM), Poznań*, 2006.

[6] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[7] Philippe de Groote. A dynamic programming approach to categorial deduction. In *Conference on Automated Deduction, CADE'99*, Lecture Notes in Artificial Intelligence. Springer-Verlag, July 1999.

[8] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[9] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

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

[11] Joachim Lambek. Mathematics and the mind. In V.M. Abrusci and C. Casadio, editors, *New Perspectives in Logic and Formal Linguisitics, Proceedings Vth ROMA Workshop*. Bulzoni Editore, 2001.

[12] Alain Lecomte and Christian Retoré. Words as modules and modules as partial proof nets in a lexicalised grammar. In Michele Abrusci and Claudia Casadio, editors, *Third Roma Workshop: Proofs and Linguistics Categories – Applications of Logic to the analysis and implementation of Natural Language*, pages 187–198. Bologna:CLUEB, 1996.

[13] Alain Lecomte and Christian Retoré. Words as modules: a lexicalised grammar in the framework of linear logic proof nets. In Carlos Martin-Vide, editor, *Mathematical and Computational Analysis of Natural Language — selected papers from ICML'96*, volume 45 of *Studies in Functional and Structural Linguistics*, pages 129–144. John Benjamins publishing company, 1998.

[14] Glyn V. Morrill. Memoisation of categorial proof nets: parallelism in categorial processing. In *Third Roma Workshop: Proofs and Linguistics Categories – Applications of Logic to the analysis and implementation of Natural Language*. Bologna:CLUEB, 1996.

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

[16] Mati Pentus. Product-free Lambek calculus and context-free grammars. *Journal of Symbolic Logic*, 62(2):648–660, 1997.

[17] Dirk Roorda. Resource logics: Proof-theoretical investigations. Phd thesis, University of Amsterdam, 1991.

[18] David N. Yetter. Quantales and (non-commutative) linear logic. *Journal of Symbolic Logic*, 55:41–64, 1990.

Denis Béchet
LINA & CNRS FRE 2729
2, rue de la Houssinière
BP 92208
44322 Nantes Cedex 03
France
Denis.Bechet@univ-nantes.fr