Pregroup Grammars and Context-free Grammars

Wojciech Buszkowski[1,2] and Katarzyna Moroz[1]

Adam Mickiewicz University in Poznań[1]
Rovira i Virgili University in Tarragona[2]

1 Introduction

Pregroup grammars were introduced by Lambek [20] as a new formalism of type-logical grammars. They are weakly equivalent to context-free grammars [8]. The proof in one direction uses the fact that context-free languages are closed under homomorphism and inverse homomorphism. Here we present a direct construction of a context-free grammar and a push-down automaton, equivalent to a given pregroup grammar. The size of the resulting context-free grammar (push-down automaton) is polynomial in the size of the pregroup grammar, while the construction proposed in Béchet [4] is exponential. We also obtain similar results for pregroup grammars based on a full system of Compact Bilinear Logic in the form of [11].

First, we recall some basic notions related to type logics, pregroups, Compact Bilinear Logic and pregroup grammars. Context-free grammars and push-down automata will not be defined; the reader is referred to any standard textbook on mathematical linguistics.

1.1 Type logics

Pregroup grammars are closely related to type-logical grammars, based on the Lambek calculus. The Lambek calculus **L**, due to Lambek [18], is a standard logic for type-logical grammars. *Types* (formulas) are formed out of atoms (variables or constants) by means of operation symbols $\otimes, \backslash, /$ (product, right residuation, left residuation). A Gentzen-style system for **L** admits the following axioms and rules:

$$\text{(Id)}\ A \to A,$$

$$(\backslash\text{L})\ \frac{\Gamma, B, \Delta \to C;\ \Phi \to A}{\Gamma, \Phi, A\backslash B, \Delta \to C}\quad (\backslash\text{R})\ \frac{A, \Gamma \to B}{\Gamma \to A\backslash B}$$

$$(/\text{L})\ \frac{\Gamma, B, \Delta \to C;\ \Phi \to A}{\Gamma, B/A, \Phi, \Delta \to C}\quad (/\text{R})\ \frac{\Gamma, A \to B}{\Gamma \to B/A}$$

$$(\otimes\text{L})\ \frac{\Gamma, A, B, \Delta \to C}{\Gamma, A \otimes B, \Delta \to C}\quad (\otimes\text{R})\ \frac{\Gamma \to A;\ \Delta \to B}{\Gamma, \Delta \to A \otimes B}$$

for any types A, B, C and type sequences Γ, Δ, Φ provided that $\Gamma \neq \epsilon$ in rules (\R) and (/R). The cut-elimination theorem for **L** states that the following rule is admissible [18]:

$$\text{(CUT)}\ \frac{\Phi \to A;\ \Gamma, A, \Delta \to B}{\Gamma, \Phi, \Delta \to B}.$$

Different variants of the Lambek calculus appear in literature. Admitting $\Gamma = \epsilon$ in (\R), (/R) yields the Lambek calculus with (possibly) empty antecedents (**L***). It is a conservative fragment of **L1**; the latter admits one new constant 1, one new rule and one new axiom:

$$\text{(1L)}\ \frac{\Gamma, \Delta \to A}{\Gamma, 1, \Delta \to A}\ \text{(1R)}\ \ \to 1.$$

By affixing additives $\vee$ (join) and $\wedge$ (meet) one obtains Full Lambek Calculus (**FL**)[15]. Other variants admit some structural rules, e.g. Exchange, Weakening, or drop associativity of product $\otimes$. The cut-elimination theorem is true for all these variants, and the provability problem is decidable. This problem is NP-complete for **L**, **L*** [26] and solvable in polynomial time for non-associative systems without additives [16]. For the latter, even the consequence relations are polynomial time decidable [10].

Abstract algebraic models for **L** are *residuated semigroups*: structures $\mathcal{M} = (M, \leq, \cdot, \backslash, /)$ such that $\leq$ is a partial ordering on M, $\cdot$ is an associative, binary operation on M (product), and $\backslash, /$ are binary operations on M (residual operations), which satisfy *the residuation law*:

$$\text{(RES)}\ a \cdot b \leq c \text{ iff } b \leq a\backslash c \text{ iff } a \leq c/b$$

for all $a, b, c \in M$. In these models, $\otimes$ is interpreted as $\cdot$, and $\to$ as $\leq$. From (RES) it follows that $(M, \leq, \cdot)$ is a partially ordered semigroup ($\cdot$ is monotone in both arguments). For **L***, **L1**, one employs *residuated monoids* (residuated semigroups with an element 1, satisfying: $1a = a = a1$, for all elements a).

Standard models are language frames $P(\Sigma^*)$ (for **L***) and $P(\Sigma^+)$ (for **L**), in which $L_1 \cdot L_2 = \{xy : x \in L_1, y \in L_2\}$ is the concatenation of languages, $L_1\backslash L_2 = \{x \in \Sigma^* : L_1 \cdot \{x\} \subseteq L_2\}$, $L_2/L_1 = \{x \in \Sigma^* : \{x\} \cdot L_1 \subseteq L_2\}$, $1 = \{\epsilon\}$, and $\leq$ is inclusion (for $P(\Sigma^+)$, put $x \in \Sigma^+$ in the definitions of residuals). For instance, if S is the set of sentences,and N_0 is the set of proper nouns, then $N_0\backslash S$ contains all verb phrases, $S/(N_0\backslash S)$ contains all noun phrases, and so on (we ignore agreement). Since $A \to B/(A\backslash B)$ is provable in **L**, then it is valid in residuated semigroups. Consequently, every proper noun can also be assigned the type of noun phrase.

L* is a conservative subsystem of Bilinear Logic **BL** [19], which amounts to the multiplicative fragment of Noncommutative Linear Logic of Abrusci [1]. Models of **BL** are residuated monoids with *a dualizing element* 0, satisfying: $a = 0/(a\backslash 0) = (0/a)\backslash 0$, for all elements a. One defines two *negations*: $a^r = a\backslash 0$, $a^l = 0/a$, and proves:

$$a^{rl} = a = a^{lr},\ aa^r \leq 0,\ a^l a \leq 0,\ (a^r b^r)^l = (a^l b^l)^r.$$

One also defines the operation *par*: $a \oplus b = (b^r a^r)^l$, which is associative and, together with $\cdot$, satisfies De Morgan laws: $(ab)^l = b^l \oplus a^l$, $(a \oplus b)^l = b^l a^l$, and similarly for right negation.

1.2 Pregroups and pregroup grammars

Pregroups are models of **BL** in which $ab = a \oplus b$, for all elements a, b, and consequently $1 = 0$. They can also be defined directly, as follows. *A pregroup* is a structure $\mathcal{M} = (M, \leq, \cdot, ^l, ^r, 1)$ such that $(M, \leq, \cdot, 1)$ is a partially ordered monoid, and ^{l}, ^{r} are unary operations on M, satisfying the adjoint laws:

$$(\mathrm{A}l)\ a^l a \leq 1 \leq a a^l,\ (\mathrm{A}r)\ a a^r \leq 1 \leq a^r a,$$

for all $a \in M$. Pregroups were introduced by Lambek [20] as a generalization of partially ordered groups. a^l (resp. a^r) is called *the left* (resp. *right*) *adjoint* of a.

If $\cdot$ is commutative, then $a^l a = 1 = a a^l$, $a a^r = 1 = a^r a$, whence $a^l = a^{-1} = a^r$. Commutative pregroups are simply partially ordered abelian groups. They are not adequate for language description, so Lambek focuses on noncommutative pregroups and even free pregroups, defined below.

First, we note that the following laws are easily derivable from $(\mathrm{A}l)$, $(\mathrm{A}r)$:

$$1^l = 1 = 1^r,\ a^{lr} = a = a^{rl},\ (ab)^l = b^l a^l,\ (ab)^r = b^r a^r,$$

$$a \leq b \text{ iff } b^l \leq a^l \text{ iff } b^r \leq a^r.$$

For any pregroup, one defines $a \backslash b = a^r b$, $a / b = a b^l$ and easily proves (RES). Accordingly, every pregroup is a residuated monoid with these residual operations. It follows that all sequents provable in **L1** are valid in pregroups under this translation of residuals. The converse is not true; $(ab)/c \leq a(b/c)$ is valid in pregroups, but not in residuated monoids.

For an integer $n \geq 0$, one defines $a^{(n)} = a^{rr\ldots r}$ and $a^{(-n)} = a^{ll\ldots l}$, where adjoints are iterated n times. The following laws are provable:

$$a^{(n)} a^{(n+1)} \leq 1 \leq a^{(n+1)} a^{(n)}, \text{ for any integer } n. \tag{1}$$

Let $(P, \leq)$ be a (finite) poset. Elements of P are called *atoms* and denoted p, q, r, s. *Simple terms* are of the form $p^{(n)}$, for any $p \in P$ and $n \in \mathcal{Z}$, where $\mathcal{Z}$ denotes the set of integers. *A term* is a finite sequence (string) of simple terms. Terms are also called *types*. Greek capitals Γ, Δ, Φ range on terms, and t denotes a simple term. If $p \leq q$ in P, then we write $p^{(n)} \leq q^{(n)}$, if n is even, and $q^{(n)} \leq p^{(n)}$, if n is odd. One defines a binary relation $\Rightarrow$ on the set of terms as the least reflexive and transitive relation, satisfying the clauses:

(CON) $\Gamma, p^{(n)}, p^{(n+1)}, \Delta \Rightarrow \Gamma, \Delta$,

(EXP) $\Gamma, \Delta \Rightarrow \Gamma, p^{(n+1)}, p^{(n)}, \Delta$,

(IND) $\Gamma, p^{(n)}, \Delta \Rightarrow \Gamma, q^{(n)}, \Delta$, if $p^{(n)} \leq q^{(n)}$.

(CON), (EXP), (IND) are called Contraction, Expansion and Induced Step, respectively. They can be treated as rules of a term rewriting system. $\Gamma \Rightarrow \Delta$ is true iff Γ can be rewritten into Δ by a finite number of applications of these rules. This rewriting system is Lambek's original form of the logic of pregroups. This logic is also called Compact Bilinear Logic (**CBL**).

One defines Γ^l and Γ^r as follows:

$$\epsilon^l = \epsilon^r = \epsilon,\ (p^{(n)})^l = p^{(n-1)},\ (p^{(n)})^r = p^{(n+1)},$$

$$(t_1, \ldots, t_n)^l = (t_n)^l \ldots (t_1)^l,\ (t_1, \ldots, t_n)^r = (t_n)^r \ldots (t_1)^r).$$

The set of all terms with the relation $\Rightarrow$, operations $\cdot$ (concatenation), ^{l}, ^{r}, and the unit element ϵ constitutes a preordered pregroup. Preordered pregroups are defined like pregroups except that $\leq$ can be a preorder (i.e. a reflexive and transitive relation), and in monoid equations $=$ is replaced by $\sim$, where: $a \sim b$ iff $a \leq b$ and $b \leq a$. If $\mathcal{M}$ is a preordered pregroup, then $\mathcal{M}/\sim$ is a pregroup.

So, $\Gamma \sim \Delta$ iff $\Gamma \Rightarrow \Delta$ and $\Delta \Rightarrow \Gamma$. The relation $\sim$ is nontrivial also for a trivial poset $(P, =)$; for instance $p, p^{(1)}, p \sim p$; $p, p^{(-1)}, p \sim p$. (In pregroups, $aa^r a = a$ and $aa^l a = a$.) It is a congruence on the preordered pregroup, defined above. The quotient-structure with the ordering defined by:

$$[\Gamma] \leq [\Delta] \text{ iff } \Gamma \Rightarrow \Delta$$

is a pregroup, called the free pregroup generated by $(P, \leq)$, and denoted $F(P, \leq)$ or $F(P)$. One easily proves that it is free in the sense that every function f from P to a pregroup $\mathcal{M}$ which preserves the order ($p \leq q$ entails $f(p) \leq f(q)$) has a unique extension to an (order-preserving) homomorphism from $F(P)$ to $\mathcal{M}$. This yields the completeness theorem: $\Gamma \Rightarrow \Delta$ iff $f([\Gamma]) \leq f([\Delta])$, for any order-preserving function f from P to a pregroup $\mathcal{M}$.

Lambek [20] proves the following normalization theorem for **CBL** (also called the Lambek switching lemma).

Lambek's theorem. If $\Gamma \Rightarrow \Delta$ then there exists Φ such that $\Gamma \Rightarrow \Phi$ without (EXP) and $\Phi \Rightarrow \Delta$ without (CON).

Consequently, if $\Gamma \Rightarrow t$, where t is a simple term or ϵ, then Γ can be reduced to t by (CON) and (IND) only. Such reductions are easily computable and can be simulated by a context-free grammar. This yields the polynomial time decidability of **CBL**. (Notice that $\Gamma \Rightarrow \Delta$ iff $\Delta^l, \Gamma \Rightarrow \epsilon$ iff $\Gamma, \Delta^r \Rightarrow \epsilon$.) Here it is essential that, for any finite set T of simple types, one can construct in polynomial time (depending on the size of T) a context-free grammar G such that $L(G) = \{\Gamma \in T^* : \Gamma \Rightarrow \epsilon\}$; this construction is similar to that of G_1 in section 2. For **L**, **L*** the situation is different: given a finite set T of types and $A \in T$, the set of all $\Gamma \in T^*$ such that $\Gamma \Rightarrow A$ is provable is a context-free language [24, 25], but one cannot construct the context-free grammar in polynomial time (if P$\neq$NP).

The Lambek normalization theorem is equivalent to the cut-elimination theorem for a sequent system of **CBL** [9]. *Sequents* are of the form $\Gamma \rightarrow \Delta$. The

axioms are all sequents $\Gamma \to \Gamma$. The inference rules are divided in left rules, which act on antecedents of sequents, and right rules, which act on consequents of sequents.

$$\text{(AL)}\ \frac{\Gamma, \Gamma' \to \Delta}{\Gamma, p^{(n)}, p^{(n+1)}, \Gamma' \to \Delta},\ \text{(AR)}\ \frac{\Gamma \to \Delta, \Delta'}{\Gamma \to \Delta, p^{(n+1)}, p^{(n)}, \Delta'},$$

$$\text{(IL)}\ \frac{\Gamma, q^{(n)}, \Gamma' \to \Delta}{\Gamma, p^{(n)}, \Gamma' \to \Delta},\ \text{(IR)}\ \frac{\Gamma \to \Delta, p^{(n)}, \Delta'}{\Gamma \to \Delta, q^{(n)}, \Delta'};$$

in (IL), (IR), one assumes $p^{(n)} \leq q^{(n)}$. We admit the following form of the cut rule:

$$\text{(TRAN)}\ \frac{\Gamma \to \Phi;\ \Phi \to \Delta}{\Gamma \to \Delta}.$$

Clearly (AL) corresponds to (CON), (AR) to (EXP), and (IL), (IR) to (IND). The rewriting system admits (TRAN), by definition. It is easy to see that $\Gamma \Rightarrow \Delta$ in the rewriting system if and only if $\Gamma \to \Delta$ is provable in the sequent system with (TRAN). From the Lambek normalization theorem it follows that $\Gamma \Rightarrow \Delta$ if and only if $\Gamma \to \Delta$ is provable in the sequent system without (TRAN). This yields the cut-elimination theorem for the sequent system. Conversely, the latter implies the Lambek normalization theorem.

A pregroup grammar is a quintuple $G = (\Sigma, P, \leq, s, I)$ such that Σ is a nonempty, finite alphabet, $(P, \leq)$ is a finite poset, $s \in P$, and I is a finite relation between symbols from Σ and nonempty types (on P). For $a \in \Sigma$, $I(a)$ denotes the set of all types Γ such that $(a, \Gamma) \in I$. Let $x \in \Sigma^+$, $x = a_1 \dots a_n$ ($a_i \in \Sigma$). One says that G assigns type Δ to x, if there exist types $\Gamma_i \in I(a_i)$, $i = 1, \dots, n$, such that $\Gamma_1, \dots, \Gamma_n \Rightarrow \Delta$ in **CBL**; we write $x \to_G \Delta$. The set $L(G) = \{x \in \Sigma^+ : x \to_G s\}$ is called *the language* of G.

For more information about pregroup grammars the reader is referred to e.g. [21, 12] and the special issue 'Categorial grammars and pregroups', edited by W. Buszkowski and A. Preller, *Studia Logica* 87.2/3, 2007.

To illustrate pregroups parsing of natural language we take the following set of basic types (after Lambek [20]):

π_k - k person singular (plural forms are of type π_2)

$\mathbf{s}_i$ - statement, in the present tense if $i = 1$, and in the past tense if $i = 2$

$\mathbf{p}_i$ - participle (present or past)

o - object

Examples are presented in Table 1.

she	*goes*			
π_2	$\pi_2^r \mathrm{s}_1$	$= (\pi_2 \pi_2^r)\mathrm{s}_1$	$\leq\ \mathrm{s}_1$	
we	*have*	*been*	*seen*	
π_2	$\pi_2^r \mathrm{s}_1 \mathrm{p}_2^l$	$\mathrm{p}_2 \mathrm{o}^{ll}\ \mathrm{p}_2^l$	$\mathrm{p}_2 \mathrm{o}^l$	$= \pi_2 \pi_2^r\ \mathrm{s}_1 \mathrm{p}_2^l \mathrm{p}_2 \mathrm{o}^{ll} \mathrm{p}_2^l \mathrm{p}_2 \mathrm{o}^l \leq\ \mathrm{s}_1$

Table 1: Linguistic examples

2 The equivalence theorem

Pregroup grammars are weakly equivalent to $\epsilon-$free context-free grammars [8]. The proof consists of two parts. Below we survey its main lines. We also discuss a different approach of Béchet [4], based on 'partial composition', and state a stronger form of his lemma on binary reductions (with an outline of proof). This section ends with some remarks on fully lexicalized pregroups.

We abbreviate 'context-free grammar' as CFG, 'classical categorial grammar' as CCG, and 'pregroup grammar' as PG. Grammars G_1, G_2 are *weakly equivalent* (shortly: equivalent) if $L(G_1) = L(G_2)$.

Recall that a CCG is a triple $G = (\Sigma, I, s)$ such that Σ is a finite alphabet, s is a designated atomic type, and I is a finite relation between elements of Σ and product-free types (in the sense of **L**). G assigns a type A to a string $x \in \Sigma^+$, $x = a_1 \dots a_n$ (write: $x \rightarrow_G A$), if there exist types $A_1, \dots, A_n$ assigned to $a_1, \dots, a_n$ by I such that $A_1, \dots, A_n \rightarrow A$ is provable in the logic **AB** (after Ajdukiewicz [2] and Bar-Hillel [3]). **AB** can be axiomatized by (Id), (\L) and (/L). Equivalently, it can be presented as a rewriting system, based on the reduction rules: $A, A\backslash B \Rightarrow B$ and $B/A, A \Rightarrow B$. The language of G is defined as $L(G) = \{x \in \Sigma^+ : x \rightarrow_G s\}$.

First, one proves that every $\epsilon-$free CFG is equivalent to some PG. In [3], it was proved that every $\epsilon-$free CFG G is equivalent to some CCG G' in which all types are of the form p, p/q, $(p/q)/r$, where p, q, r are atoms. Let P consist of all atoms appearing in G'. Define a PG $G_0 = (\Sigma, P, =, s, I)$, by setting: Σ is the alphabet of G', s is the designated type of G', and $(a, \Gamma) \in I$ iff there exists a type A, assigned to a by G', such that Γ is the translation of A in the language of **CBL**. For instance, if G' assigns $(p/q)/r$ to a, then $(a, pq^l r^l) \in I$. Let $t(A)$ denote the translation of A. One proves: $A_1, \dots, A_n \Rightarrow s$ in **AB** iff $t(A_1), \dots, t(A_n) \Rightarrow s$ in **CBL** (all A_i are restricted as above). This yields $L(G_0) = L(G') = L(G)$. Consequently, every $\epsilon-$free CFG is equivalent to some PG which is based on a trivial poset $(P, =)$ and employs neither right adjoints, nor iterated left adjoints $p^{ll\cdots l}$ [8].

Second, one proves that every PG is equivalent to an $\epsilon-$free CFG. The Lambek normalization theorem is essential. Let $G = (\Sigma, P, \leq, s, I)$ be a PG. For $a_i \in \Sigma$, $i = 1, \dots, n$, we have: $a_1 \dots a_n \in L(G)$ iff $\Gamma_1, \dots, \Gamma_n \Rightarrow s$, for some types $\Gamma_i \in I(a_i)$, $i = 1, \dots, n$. Only rules (CON) and (IND) can be regarded in reductions to s.

Let $T(G)$ and $S(G)$ denote the set of all types involved in I and all simple terms appearing in types from $T(G)$, respectively. We define a CFG G_1 such that $L(G_1) = \{\Gamma \in (S(G))^+ : \Gamma \Rightarrow s\}$. $S(G)$ is the terminal alphabet of G_1. The nonterminal alphabet N contains an additional symbol E and all types $q^{(n)}$ such that $p^{(n)} \leq q^{(n)}$, for some $p^{(n)} \in S(G)$. The start symbol is s. The production rules of G_1 are all possible rules of the form:

$$q^{(n)} \mapsto p^{(n)};\ X \mapsto EX;\ X \mapsto XE;\ E \mapsto p^{(n)} p^{(n+1)}, \tag{2}$$

where all symbols are in N, and $p^{(n)} \leq q^{(n)}$ in the first rule. One proves: $\Gamma \Rightarrow t$

in **CBL** iff $t \to^* \Gamma$ in G_1, for all strings Γ on $N - \{E\}$ and simple terms $t \in N$. $\to^*$ denotes the relation of derivability in the given CFG, i.e. the reflexive transitive closure of the relation of one-step derivability $\to$; $x \to y$ holds if x can be rewritten into y by applying one production rule.

For any $a \in \Sigma$ such that $I(a) = \{\Gamma_1, \ldots, \Gamma_k\}$, $(k > 0)$, we choose k new symbols $a(1), \ldots, a(k)$. The alphabet Σ' consists of all new symbols, corresponding to symbols from Σ. We define a homomorphism h from $(\Sigma')^*$ to $(S(G))^*$, by setting $h(a(i)) = \Gamma_i$, and a homomorphism g from $(\Sigma')^*$ to Σ^*, by setting $g(a(i)) = a$. Clearly, $L(G) = g(h^{-1}(L(G_1)))$, and consequently, $L(G)$ is a context-free language [8].

Béchet [4] notices that the above proof does not yield any direct construction of a CFG equivalent to the given pregroup grammar. Indeed, this proof relies upon the fact that the class of context-free languages is closed under homomorphisms and inverse homomorphisms. On the other hand, these closure properties can be proved in an effective way, which yields a direct construction, actually. We present this construction in section 3. Below we discuss an alternative construction, proposed by Béchet [4].

The main idea is *partial composition*. It is a multiple (generalized) contraction of the form:

$$\text{(PC) } \Gamma p_1^{(n_1)} \cdots p_k^{(n_k)}, q_k^{(n_k+1)} \cdots q_1^{(n_1+1)} \Delta \Rightarrow \Gamma\Delta$$

such that $p_i^{(n_i)} \leq q_i^{(n_i)}$, for all $i = 1, \ldots, k$ and $l(\Gamma\Delta) \leq m$. (Notation: in contexts like $\Gamma, \Delta \Rightarrow \Phi$, comma stands for the concatenation of sequences, but we prefer to write $\Gamma \cdot \Delta$ or $\Gamma\Delta$, when we consider the very sequence; as usual, a string of symbols $a_1, \ldots, a_n$ is written $a_1 \ldots a_n$.)

Define *the length* of Γ ($l(\Gamma)$) as the total number of simple terms occurring in Γ. The following lemma on binary reductions is crucial:

Béchet's lemma. If $\Gamma_1, \ldots, \Gamma_n \Rightarrow p$, $n \geq 2$, in the free pregroup, then there exist $1 \leq i < n$ and a type Δ such that $\Gamma_i, \Gamma_{i+1} \Rightarrow \Delta$ without (EXP), $\Gamma_1, \ldots, \Gamma_{i-1}, \Delta, \Gamma_{i+2}, \ldots, \Gamma_n \Rightarrow p$, and $l(\Delta) \leq \max\{l(\Gamma_j) : j = 1, \ldots, n\}$.

Now, a CFG G' equivalent to the pregroup grammar G can be defined as follows. The terminal alphabet is Σ. Let m be the maximal length of types in $T(G)$. The nonterminal alphabet N' consists of all symbols of the form $[\Gamma]$ such that Γ is a nonempty string of types from N (in the sense of G_1), $l(\Gamma) \leq m$. The start symbol is $[s]$. Production rules are of the form $[\Delta] \mapsto [\Gamma][\Gamma']$ such that $[\Delta], [\Gamma], [\Gamma'] \in N'$ and $\Gamma, \Gamma' \Rightarrow \Delta$ without (EXP); in the context of $\Rightarrow$, E is to be replaced by ϵ. One also adds lexical rules $[\Gamma] \mapsto a$, for all $a \in \Sigma$, $\Gamma \in N'$ such that $\Gamma \in I(a)$. Applying Béchet's lemma, one proves $L(G') = L(G)$.

The size of G' is exponential in the size of G. More precisely, if k is the cardinality of N, then N' contains $k^1 + k^2 + \cdots + k^m$ nonterminal symbols.

Our formulation of Béchet's lemma slightly differs from the one in [4] but is equivalent to it; in particular, [4] uses 1 instead of E. The proof of this lemma in [4] is based on some properties of outer-planar graphs, i.e. planar graphs whose nodes are located on a straight line and all edges are drawn in one semiplane,

determined by this line. Clearly graphs of links corresponding to (generalized) contractions in free pregroups are of that kind. First, a variant of the lemma with ϵ instead of p is proved. Second, the author notices that the lemma follows from the latter by the fact that $\Gamma \Rightarrow p$ iff $\Gamma, p^r \Rightarrow \epsilon$ and a reconstruction (not presented in detail) of the (PC)-tree for $\Gamma, p^r \Rightarrow \epsilon$; here $\Gamma = \Gamma_1 \cdots \Gamma_n$.

We note that another proof of Bechet's lemma can be obtained by methods of Pentus [24, 25], using some counts of atoms. Even a stronger form with Γ_{n+1} instead of p and $l(\Delta) \leq \max\{l(\Gamma_j) : j = 1, \ldots, n, n+1\}$ can be proved. We outline the main lines of the proof.

It is easy to see that any reduction $\Gamma \Rightarrow \Delta$ can be rearranged to a reduction in which all (IND)-steps are performed at the very beginning and at the very end. First, apply the Lambek normalization theorem. Then, observe that, if (IND) immediately follows (CON), then one can change the order of rules, and similarly, if (IND) immediately precedes (EXP).

Accordingly, a reduction $\Gamma_1, \ldots, \Gamma_n \Rightarrow \Gamma_{n+1}$ can be divided in four parts:

(P1) $\Gamma_1, \ldots, \Gamma_n \Rightarrow \Gamma'_1, \ldots, \Gamma'_n$, by multiple (IND) only,

(P2) $\Gamma'_1, \ldots, \Gamma'_n \Rightarrow \Gamma'_{n+1}$, by multiple (CON) only,

(P3,4) $\Gamma'_{n+1} \Rightarrow \Gamma_{n+1}$, first by multiple (EXP), then by multiple (IND).

Clearly $l(\Gamma'_{n+1}) \leq l(\Gamma_{n+1})$. It suffices to prove the following form of 'binary reduction'.

(BR) If $\Gamma'_1, \ldots, \Gamma'_n \Rightarrow \Gamma'_{n+1}$, by multiple (CON), $n \geq 2$, then there exist $1 \leq i < n$ and a type Δ such that $\Gamma'_i, \Gamma'_{i+1} \Rightarrow \Delta$, by multiple (CON), $\Gamma'_1, \ldots, \Gamma'_{i-1}, \Delta, \Gamma'_{i+2}, \ldots, \Gamma'_n \Rightarrow \Gamma'_{n+1}$, by multiple (CON), and $l(\Delta)$ is not greater than $\max\{l(\Gamma'_j) : j = 1, \ldots, n+1\}$.

Béchet's lemma follows from (BR), since $\Gamma_j \Rightarrow \Gamma'_j$ (by (IND)), for all $j = 1, \ldots, n$. All reductions in (BR) are performed in **CBL** without poset rules, i.e. pure **CBL**. This logic can be axiomatized as a cut-free sequent system with axioms (Id) and rules (AL), (AR). ((AR) is not needed, since we do not regard (CON) in (BR), but let us admit it.) The proof of (BR) relies upon three essential facts.

First, one can prove a Roorda-style interpolation lemma for this system [29]: if $\Gamma_1, \Phi, \Gamma_2 \Rightarrow \Gamma$, where $\Phi \neq \epsilon$, then there exists a type Δ such that $\Gamma_1, \Delta, \Gamma_2 \Rightarrow \Gamma$, $\Phi \Rightarrow \Delta$ and, for any variable p, the number of occurrences of p in Δ is not greater than the minimum of the numbers of occurrences of p in Φ and in $\Gamma_1 \cdot \Gamma_2 \cdot \Gamma$ (the context of Φ). (Use induction on cut-free proofs or, equivalently, on the length of reductions based on (CON), (EXP), in which (EXP) never precedes (COM).)

Second, every sequent provable in pure **CBL** arises from a provable sequent, in which each variable has two occurrences or none, by a substitution of variables for variables. This is obvious, by the form of the sequent system.

Third, every provable sequent becomes an equality valid in a free group, after one has replaced $\rightarrow$ by $=$ and $p^{(n)}$ by p, if n is even, and by p^{-1}, if n is odd.

These three facts are keystones of Pentus' proof of an analogue of (BR) for **L**, **L***. One can repeat the proof without changes.

Pregroup grammars based on a trivial poset $(P, =)$ are said to be *fully lexicalized* [5]. Every pregroup grammar is weakly equivalent to some fully lexicalized pregroup grammar. This easily follows from the proof of the weak equivalence of pregroup grammars and context-free grammars [8], recalled at the beginning of this section: each PG G is equivalent to some $\epsilon-$free CFG, and the latter to some fully lexicalized PG. A direct transformation of a PG into an equivalent fully lexicalized PG can also be based on the normalization (P1)-(P3,4). If $G = (\Sigma, P, \leq, s, I)$ is a PG, then $G' = (\Sigma, P, =, s, I')$ is equivalent to G, where I' results from I by performing all possible (IND)-steps on the types assigned by I. The size of G' is exponential in the size of G. Béchet and Foret [5] provide a polynomial transformation; the main idea is to replace poset arrows by some arrows provable in pure **CBL**.

3 Polynomial constructions

We present another construction of CFG equivalent to PG, which roughly follows the original proof from [8], applying inverse homomorphism. Our construction is polynomial in the size of the pregroup grammar. (On the other hand, Bechet's construction resembles analogous constructions of CFGs equivalent to grammars based on **L** [6, 24, 25].) We also provide a (simpler) construction of a push-down automaton equivalent to the given PG. Similar results are obtained for pregroup grammars based on a full system of **CBL**. At the end of this section we briefly discuss tree structures determined by pregroup grammars.

3.1 A construction of a CFG

The alphabet N has been defined in section 2 (it depends on the given PG $G = (\Sigma, P, \leq, s, I)$). We construct a CFG G_2. For any $a \in \Sigma$, we create a new nonterminal symbol $X(a)$. The terminal alphabet of G_2 is $S(G) \cup \{X(a) : a \in \Sigma\}$. The nonterminal alphabet of G_2 is N. The start symbol is s, and the production rules are all rules of G_1 and all rules $Y \mapsto X(a)Y$, for $Y \in S(G)$. G_2 can insert symbols $X(a)$ in front of terms from $S(G)$ within the strings generated by G_1.

The next step is similar to the construction of a CFG which generates the intersection of a context-free language and a regular language. Let $Q(G)$ denote the set of all types Δ such that $\Gamma\Delta \in T(G)$, for some Γ. A new CFG G_3 is defined as follows. The terminal alphabet is Σ. The nonterminal alphabet consists of all symbols $([\Gamma], X, [\Delta])$ such that $\Gamma, \Delta \in Q(G)$ and X is a terminal or nonterminal symbol of G_2. The start symbol is $([\epsilon], s, [\epsilon])$. The production rules of G_3 are divided in four groups.
(I) $([\Gamma_1], X, [\Gamma_3]) \mapsto ([\Gamma_1], Y, [\Gamma_2])([\Gamma_2], Z, [\Gamma_3])$, for any rule $X \mapsto YZ$ of G_2 and all $\Gamma_1, \Gamma_2, \Gamma_3 \in Q(G)$.
(II) $([\Gamma_1], X, [\Gamma_2]) \mapsto ([\Gamma_1], Y, [\Gamma_2])$, for all rules $X \mapsto Y$ of G_2 and $\Gamma_1, \Gamma_2 \in Q(G)$.

(III) $([t\Gamma], t, [\Gamma]) \mapsto \epsilon$, for all $\Gamma \in Q(G)$ and $t \in S(G)$.
(IV) $([\epsilon], X(a), [\Gamma]) \mapsto a$, for all $a \in \Sigma$, $\Gamma \in I(a)$.

Proposition 1. *$L(G) = L(G_3)$.*

Proof. In a derivation of a string $x \in L(G_3)$ rules (III), (IV) can be applied at the very end. Below we denote $\Gamma_i = t^i_1 \cdots t^i_{k(i)}$, $\Gamma^i_j = t^i_j t^i_{j+1} \cdots t^i_{k(i)}$, for $i = 1, \ldots, n$, $j = 1, \ldots, k(i)$. We also denote:

$$\Delta_i = ([\epsilon], X(a_i), [\Gamma_i])([\Gamma^i_1], t^i_1, [\Gamma^i_2]) \ldots ([t^i_{k(i)}], t^i_{k(i)}, [\epsilon]);$$

clearly, $\Gamma^i_1 = \Gamma_i$ and $\Gamma^i_{k(i)} = t^i_{k(i)}$.

For $a_1, \ldots, a_n \in \Sigma$, $a_1 \ldots a_n \in L(G_3)$ iff there exist $\Gamma_1 \in I(a_1), \ldots, \Gamma_n \in I(a_n)$ such that the string $\Delta_1 \cdots \Delta_n$ (according to the above notation) is derivable from $([\epsilon], s, [\epsilon])$ by rules (I), (II) only. The latter is equivalent to the condition that $\Gamma_1 \cdots \Gamma_n$ is derivable from s in G_1, which yields the thesis. □

G_3 contains nullary rules. They can be eliminated in a standard way. The following lemma is useful.

Lemma 1. *$([\Gamma], Y, [\Delta]) \to^* \epsilon$ in G_3 if and only if there exist $t_1, \ldots, t_n \in S(G)$, $n > 0$, such that $\Gamma = t_1 \cdots t_n \Delta$ and $Y \to^* t_1 \cdots t_n$ in G_1.*

Proof. We prove the 'if' part. Assume that the condition holds. Then, the following string:

$$([\Gamma], t_1, [t_2 \cdots t_n \Delta])([t_2 \cdots t_n \Delta], t_2, [t_3 \cdots t_n \Delta]) \ldots ([t_n], t_n, [\Delta])$$

is derivable from $([\Gamma], Y, [\Delta])$ in G_3. Now one applies (III) n times.

The 'only if' part is proved by induction on the length of the derivation of ϵ from $([\Gamma], Y, [\Delta])$ in G_3. If ϵ is derivable in one step, then the rule must be (III) with $Y = t$, $\Gamma = t\Delta$, whence our claim is true. Assume that the derivation has more than one step. We consider two cases.

Case 1. The first step employs rule (I). The rule must be of the form $([\Gamma], Y, [\Delta]) \mapsto ([\Gamma], X, [\Phi])([\Phi], Z, [\Delta])$, where $Y \mapsto XZ$ is a rule of G_2 and $([\Gamma], X, [\Phi]) \to^* \epsilon$, $([\Phi], Z, [\Delta]) \to^* \epsilon$, both in less steps. By the induction hypothesis, there exist $t_1, \ldots, t_m \in S(G)$ and $u_1, \ldots, u_n \in S(G)$, $m, n > 0$, such that $\Gamma = t_1 \cdots t_m \Phi$, $\Phi = u_1 \cdots u_n \Delta$ and $X \to^* t_1 \cdots t_m$, $Z \to^* u_1 \cdots u_n$ in G_1. Consequently, $X, Z \in N$, whence $Y \mapsto XZ$ is a rule of G_1. Clearly, $\Gamma = t_1 \cdots t_m u_1 \cdots u_n \Delta$ and $Y \to^* t_1 \cdots t_m u_1 \cdots u_n$ in G_1, which yields our claim.

Case 2. The first step employs rule (II). The rule must be of the form $([\Gamma], Y, [\Delta]) \mapsto ([\Gamma], Z, [\Delta])$, where $Y \mapsto Z$ is a rule of G_2 and $([\Gamma], Z, [\Delta]) \to^* \epsilon$ in less steps. By the induction hypothesis, there exist $t_1, \ldots, t_n \in S(G)$, $n > 0$, such that $\Gamma = t_1 \cdots t_n \Delta$ and $Z \to^* t_1 \cdots t_n$ in G_1. Consequently, $Z \in N$, whence $Y \mapsto Z$ is a rule of G_1. Then, $Y \to^* t_1 \cdots t_n$ in G_1, which yields our claim. □

We transform G_3 into G_4 by dropping rules (III) and affixing all rules of the form $([\Gamma_1], X, [\Gamma_3]) \mapsto ([\Gamma_2], Z, [\Gamma_3])$, for any rule (I) such that $([\Gamma_1], Y, [\Gamma_2]) \rightarrow^* \epsilon$ in G_3, and all rules $([\Gamma_1], X, [\Gamma_3]) \mapsto ([\Gamma_1], Y, [\Gamma_2])$, for any rule (I) such that $([\Gamma_2], Z, [\Gamma_3]) \rightarrow^* \epsilon$ in G_3. A standard argument of formal language theory yields:

Proposition 2. $L(G_4) = L(G_3)$.

Hence, by Proposition 1, we obtain:

Theorem 1. $L(G) = L(G_4)$.

G_4 contains unary rules. As usual, one can eliminate them and obtain a grammar in Chomsky Normal Form. The size of G_3 and G_4 is polynomial in the size of G. The latter, denoted by $|G|$, is defined as the sum of lengths of types in $T(G)$ plus the cardinality of P. The size of a CFG is defined as the sum of the number of nonterminal symbols, the number of terminal symbols and the number of production rules. Assuming $I(a) \neq \emptyset$, for any $a \in \Sigma$, the cardinality of Σ is not greater than $|G|$. The cardinality of N is $O(|G|^2)$. The size of G_1 and G_2 is $O(|G|^2)$. The cardinality of $Q(G)$ is $O(|G|)$. The size of G_3 is $O(|G|^5)$, and G_3 has $O(|G|^4)$ nonterminals. The size of G_4 is also $O(|G|^5)$. Clearly G_1, G_2, G_3 and G_4 can be constructed from G in polynomial time. In particular, the construction of G_4 from G_3 applies Lemma 1; the condition of the lemma can be checked in time $O(|G|^3)$, with applying a cubic algorithm for provability in **CBL**. The elimination of unary rules can also be done in polynomial time.

3.2 A construction of a push-down automaton

In a similar way, we construct a push-down automaton (PDA) equivalent to G. This construction is polynomial in the size of G. The input alphabet is Σ. The stack alphabet consists of all symbols from $N_1 = N - \{E\}$. States are of the form $[\Gamma]$, for $\Gamma \in Q(G)$; we also admit a new state f, which is the only accepting state. The PDA accepts a string $x \in \Sigma^*$, if it reads the input and ends in state f with the empty stack. The initial state is $[\epsilon]$. The transition rules are as follows.
(i) In state $[\epsilon]$ read a and go to state $[\Gamma]$, for $\Gamma \in I(a)$.
(ii) In any state except f replace $p^{(n)}$ on the stack by $q^{(n)}$ such that $p^{(n)} \leq q^{(n)}$.
(iii) In state $[p^{(n)}\Gamma]$ push $p^{(n)}$ on the stack and go to state $[\Gamma]$.
(iv) In state $[p^{(n+1)}\Gamma]$ pull $p^{(n)}$ of the stack and go to state $[\Gamma]$.
(v) In state $[\epsilon]$ pull s of the stack and go to state f.

A different construction is proposed by Francez and Kaminski [14], where push-down automata are applied in their new proof of the equivalence of pregroup grammars and context-free grammars. Let us notice that our construction has already been announced in a lecture on Poznań Linguistic Meeting 2006 and later on 1st International Workshop on Non-Classical Formal Languages in Linguistics 2007 in Budapest (both lectures published in pre-proceedings).

Deterministic push-down automata, working in linear time, can be constructed for special pregroup grammars, studied by Preller [27], if the entry uniquely assigns types to elements of Σ.

3.3 Pregroup grammars on a full system of CBL

In [11], **CBL** is presented in a different language. Types are formed out of atoms from P and the constant 1 by operation symbols $\otimes$ (binary) and ^{r},^{l} (unary). A, B range over types, and Γ, Δ, Φ over finite strings of types. In metalanguage, the notation $A^{(n)}$, where $n \in \mathcal{Z}$, is defined as $a^{(n)}$ in section 1. The rewriting system for **CBL** in the extended language is defined by the following rules.

(R1) $\Gamma, 1^{(n)}, \Gamma' \Rightarrow \Gamma, \Gamma'$, (R2) $\Gamma, A^{(n)}, A^{(n+1)}, \Gamma' \Rightarrow \Gamma, \Gamma'$,

(R3) $\Gamma, (A \otimes B)^{(2n)}, \Gamma' \Rightarrow \Gamma, A^{(2n)}, B^{(2n)}, \Gamma'$,

(R4) $\Gamma, (A \otimes B)^{(2n+1)}, \Gamma' \Rightarrow \Gamma, B^{(2n+1)}, A^{(2n+1)}, \Gamma'$,

(R5) $\Gamma, (A^{(m)})^{(n)}, \Gamma' \Rightarrow \Gamma, A^{(m+n)}, \Gamma'$,

(E1) $\Gamma, \Gamma' \Rightarrow \Gamma, 1^{(n)}, \Gamma'$, (E2) $\Gamma, \Gamma' \Rightarrow \Gamma, A^{(n+1)}, A^{(n)}, \Gamma'$,

(E3) $\Gamma, A^{(2n)}, B^{(2n)}, \Gamma' \Rightarrow \Gamma, (A \otimes B)^{(2n)}, \Gamma'$,

(E4) $\Gamma, B^{(2n+1)}, A^{(2n+1)}, \Gamma' \Rightarrow \Gamma, (A \otimes B)^{(2n+1)}, \Gamma'$,

(E5) $\Gamma, A^{(m+n)}, \Gamma' \Rightarrow \Gamma, (A^{(m)})^{(n)}, \Gamma'$,

(IND) (the same as for the Lambek rewriting system).

It is shown in [11] that (R2), (E2) can be restricted to variables A.

This system allows a direct derivation of e.g. $(p \otimes q^r)^l \otimes p \Rightarrow q$. It is naturally related to the framework of free compact 2-categories [28] and can be expanded by other operations and modalities. The Lambek-style normalization theorem, proved in [11], is the following: if $\Gamma \Rightarrow \Delta$, then there exists Φ such that $\Gamma \Rightarrow \Phi$ without E-rules and $\Phi \Rightarrow \Delta$ without R-rules (here 'R-rule' means 'reducing rule', and 'E-rule' means 'expanding rule'). The proof employs an equivalent sequent system, which admits cut-elimination. Every rule has only one premise. Like for the system discussed in section 1, R-rules of the rewriting system correspond to left rules of the sequent system, and E-rules of the rewriting system correspond to right rules of the sequent system. (IND) is exceptional; it is represented in both left and right rules of the sequent system.

Pregroup grammars based on this system are defined as above except that I is a finite relation between symbols from Σ and types. A grammar G assigns type A to a string $a_1 \dots a_n$, $(a_i \in \Sigma)$, if there exist types $A_i \in I(a_i)$, $i = 1, \dots, n$, such that $A_1, \dots, A_n \Rightarrow A$. $L(G)$ is defined as above.

We prove that pregroup grammars in the new sense are equivalent to $\epsilon-$free CFGs. Furthermore, a pregroup grammar can be transformed into an equivalent CFG in polynomial time. It suffices to observe that a pregroup grammar G in

the new sense is equivalent to a pregroup grammar G' in the old sense, and the size of G' is polynomial in the size of G. For any type A, one defines a string of terms $s(A)$, by the following recursion: $s(p^{(n)}) = p^{(n)}$, $s(1^{(n)}) = \epsilon$, $s((A^{(m)})^{(n)}) = s(A^{(m+n)})$, $s((A \otimes B)^{(n)}) = s(A^{(n)})s(B^{(n)})$, if n is even, $s((A \otimes B)^{(n)}) = s(B^{(n)})s(A^{(n)})$, if n is odd. It suffices to define $s((A^{(m)})^{(n)})$ for $m \cdot n < 0$. If $m \cdot n \geq 0$ then $(A^{(m)})^{(n)} = A^{(m+n)}$; if $m \cdot n < 0$ then the complexity of $(A^{(m)})^{(n)}$ is greater than the complexity of $A^{(m+n)}$.

Lemma 2. *$A_1, \ldots, A_n \Rightarrow p$ in the new system if and only if $s(A_1), \ldots, s(A_n) \Rightarrow p$ in the old system.*

Proof. We prove the 'only if' part. Assume that $A_1, \ldots, A_n \Rightarrow p$ holds in the new system. We have $s(A_i) \Rightarrow A_i$, by E-rules, whence $s(A_1), \ldots, s(A_n) \Rightarrow p$ holds in the new system. By the normalization theorem, this reduction can be performed without E-rules, which yields a reduction in the old system.

We prove the 'if' part. Assume that $s(A_1), \ldots, s(A_n) \Rightarrow p$ holds in the old system. Then, it also holds in the new system which is an extension of the old one. We have $A_i \Rightarrow s(A_i)$, by R-rules, whence $A_1, \ldots, A_n \Rightarrow p$ in the new system. □

The grammar G' is defined as G except that I' of G' assigns $s(A)$ to a iff I of G assigns A to a. By lemma 2, $L(G) = L(G')$. So, we can apply our former results. They must be adapted to pregroup grammars admitting empty types in I, but it causes no problem: if $\epsilon \in I(a)$, then G_2 contains a new rule $E \mapsto X(a)$, and G_3 contains $([\epsilon], X(a), [\epsilon]) \mapsto a$. Our construction yields a CFG $(G_4)'$, satisfying:

Theorem 2. *$L(G) = L((G_4)')$.*

3.4 Trees

An interesting research area is to study tree languages generated by pregroup grammars. We defer this problem for future investigation. Here we briefly discuss some basic points.

We confine ourselves to pregroup grammars, defined in section 1. *Trees* on Σ can be formed out of symbols from Σ by the rule: if X, Y are trees, then (X, Y) is a tree. A pregroup grammar $G = (\Sigma, P, \leq, s, I)$ generates trees on Σ according to the following rules: (i) $a \rightarrow^t_G \Gamma$, for any $a \in \Sigma$, $\Gamma \in I(a)$, (ii) if $X \rightarrow^t_G \Gamma$ and $\Gamma \Rightarrow \Delta$, then $X \rightarrow^t_G \Delta$, (iii) if $X \rightarrow^t_G \Gamma p^{(n)}$ and $Y \rightarrow^t_G p^{(n+1)}\Delta$, then $(X, Y) \rightarrow^t_G \Gamma\Delta$. *The tree language* of G, denoted by $L^t(G)$, consists of all trees X such that $X \rightarrow_G s$. Clearly $L(G)$ is the yield of $L^t(G)$.

If $\Rightarrow$ in rule (ii) is the complete derivability relation of **CBL** (admitting (CON), (EXP) and (IND)), then $L^t(G)$ is total with respect to $L(G)$, which is precisely stated by the following proposition (an analogous theorem for **L** was proved in [7]). In the proof, we write $\Gamma\Delta \Rightarrow \Phi$ instead of $\Gamma, \Delta \Rightarrow \Phi$ to avoid collision with tree structure.

Proposition 3. *For any pregroup grammar G, $L^t(G)$ contains all possible trees whose yields belong to $L(G)$.*

Proof. As usual $X[Y]$ denotes a tree with a designated subtree Y (precisely, $X[Y]$ is the substitution of Y for '-' in the tree context $X[\text{-}]$). It suffices to prove the equivalence:

$$V[((X,Y),Z)] \rightarrow^t_G s \text{ iff } V[(X,(Y,Z))] \rightarrow^t_G s\,,$$

for any tree context $V[\text{-}]$ and all trees X, Y, Z on Σ. We prove the 'only if' part; the proof of the 'if' part is similar. Assume $V[((X,Y),Z)] \rightarrow^t_G s$. The subtree $((X,Y),Z)$ must be obtained by a double application of rule (iii), possibly mixed with (ii). First, $X \rightarrow^t_G \Gamma p^{(n)}$ and $Y \rightarrow^t_G p^{(n+1)}\Delta$ yield $(X,Y) \rightarrow^t_G \Gamma\Delta$, by (iii). Then, $\Gamma\Delta \Rightarrow \Phi q^{(m)}$ and $Z \rightarrow^t_G q^{(m+1)}\Psi$ yield $((X,Y),Z) \rightarrow^t_G \Phi\Psi$, by (ii) and (iii). One can proceed in a different way. From $Y \rightarrow^t_G p^{(n+1)}\Delta$ derive $Y \rightarrow^t_G p^{(n+1)}\Delta q^{(m+1)}q^{(m)}$, by (ii) (using (EXP)), then $(Y,Z) \rightarrow^t_G p^{(n+1)}\Delta q^{(m+1)}\Psi$, by (iii). Applying (iii) again, one gets $(X,(Y,Z)) \rightarrow^t_G \Gamma\Delta q^{(m+1)}\Psi$. Since $\Gamma\Delta \Rightarrow \Phi q^{(m)}$, then $(X,(Y,Z)) \rightarrow^t_G \Phi\Psi$, by (ii). In the derivation of $V[((X,Y),Z)] \rightarrow^t_G s$ one replaces the subderivation of $((X,Y),Z) \rightarrow^t_G \Phi\Psi$ by the derivation of $(X,(Y,Z)) \rightarrow^t_G \Phi\Psi$, given above. This yields $V[(X,(Y,Z))] \rightarrow^t_G s$. □

Consequently, if $L(G)$ is a non-regular context-free language, then $L^t(G)$ is not a regular tree language, since the relation:

$$X \sim Y \text{ iff, for all } Z[\text{-}],\ Z[X] \in L^t(G) \text{ iff } Z[Y] \in L^t(G)$$

has infinitely many equivalence classes.

Usually, parsing with pregroup grammars employs the restricted derivability relation $\Rightarrow_c$, based on (CON) and (IND) only. If we replace $\Rightarrow$ by $\Rightarrow_c$ in rule (ii), then Proposition 3 cannot be proved. We leave it as an open problem whether the resulting tree languages are regular.

Tree languages generated by context-free grammars are regular. Then, the context-free grammar G_4, equivalent to the given pregroup grammar G, generates a proper subset of $L^t(G)$ (based on $\Rightarrow$), and similarly for the construction of Béchet (see section 2). Actually, the latter seems to provide more trees than G_4. Consider the second sentence of Table 1. Béchet's grammar generates trees:

1. (they, (have,(been, seen))),
2. ((they, have),(been, seen)),
3. (((they, have), been), seen),

but not (they,((have, been), seen)). The latter tree can be derived without (EXP). Our grammar G_4 only generates 2. We could modify the construction of G_4 in order to get more trees. It remains, however, an open question whether a polynomial construction can transform any pregroup grammar into a context-free grammar whose tree language equals the tree language of Béchet's grammar.

References

[1] Abrusci, V. M., Phase semantics and sequent system for pure noncommutative classical propositional logic, *Journal of Symbolic Logic* 56:1403–1454, 1991.

[2] Ajdukiewicz, K., Die syntaktische Konnexität, *Studia Philosophica* 1:1–27, 1935.

[3] Bar-Hillel,Y., C. Gaifman, and E. Shamir, On categorial and phrase structure grammars, *Bull. Res Council Israel* F9:155–166, 1960.

[4] Béchet, D., Parsing Pregroup Grammars and Lambek Calculus using Partial Composition, *Studia Logica* 87.2/3, 2007. To appear.

[5] Béchet, D. and A. Foret, Fully Lexicalized Pregroup Grammars, in: D. Leivant and P. de Queiroz (eds.), *Logic, Language, Information and Computation*, LNCS 4576:12–25, Springer, 2007.

[6] Buszkowski, W., The Equivalence of Unidirectional Lambek Categorial Grammars and Context-free Grammars, *Zeitschrift f. math. Logik und Grundlagen d. Math.* 31:369–384, 1985.

[7] Buszkowski, W., Generative Power of Categorial Grammars, [23]:69–94.

[8] Buszkowski, W., Lambek Grammars Based on Pregroups, [17]:95–109.

[9] Buszkowski, W., Sequent Systems for Compact Bilinear Logic, *Mathematical Logic Quarterly*, 49:467–474, 2003.

[10] Buszkowski, W., Lambek Calculus with Non-Logical Axioms, [13]:77–93.

[11] Buszkowski, W., Type Logics and Pregroups, *Studia Logica* 87.2/3, 2007. To appear.

[12] Casadio, C. and J. Lambek, An Algebraic Analysis of Clitic Pronouns in Italian, [17]:110–124.

[13] Casadio, C., P. J. Scott, and R. Seely, eds., *Language and Grammar. Studies in Mathematical Linguistics and Natural Language*, CSLI Lecture Notes 168, Stanford, 2005.

[14] Francez, N. and M. Kaminski, Pushdown automata with cancellation and commutation-augmented pregroup grammars, Pre-proceedings of LATA 2007:7–25, Universitat Rovira i Virgili , Tarragona, 2007.

[15] Galatos, N., P. Jipsen, T. Kowalski and H. Ono, *Residuated Lattices: an Algebraic Glimpse at Substructural Logics*, Elsevier, Amsterdam, 2007.

[16] de Groote, P. and F. Lamarche, Classical Nonassociative Lambek Calculus, *Studia Logica* 71.2:355–388, 2002.

[17] de Groote, P., G. Morrill, and C. Retoré, eds., *Logical Aspects of Computational Linguistics*, LNAI 2099, Springer, 2001.

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

[19] Lambek, J., From categorial grammar to bilinear logic, [30]:207–237.

[20] Lambek, J., Type Grammars Revisited, [22]:1–27.

[21] Lambek, J., Type Grammars as Pregroups, *Grammars* 4:21–39, 2001.

[22] Lecomte, A., F. Lamarche, and G. Perrier, eds., *Logical Aspects of Computational Linguistics*, LNAI 1582, Springer, 1999.

[23] Oehrle, R. T., E. Bach and D. Wheeler, eds., *Categorial Grammars and Natural Language Structures*, D. Reidel, Dordrecht, 1988.

[24] Pentus, M., Lambek Grammars are Context-Free, *Proc. 8th IEEE Symp. Logic in Computer Scie.*, 429–433, 1993.

[25] Pentus, M., Lambek Calculus and Formal Grammars, in: *Provability, Complexity, Grammars*, AMS Translations, series 2, Providence, 1999.

[26] Pentus, M., Lambek calculus is NP-complete, *Theoretical Computer Science* 357:186–201, 2006.

[27] Preller, A., Linear Processing with Pregroups, *Studia Logica* 87.2/3, 2007. To appear.

[28] Preller, A. and J. Lambek, Free compact 2-categories, *Mathematical Structures in Computer Science* 17:309–40, 2007.

[29] Roorda, D. *Resource logics. Proof-theoretic investigations*, PhD Thesis, Amsterdam, 1991.

[30] Schroeder-Heister, P. and K. Dosen, eds., *Substructural Logics*, Clarendon Press, Oxford, 1993.