Parsing Pregroup Grammars with Letter Promotions in Polynomial Time

Katarzyna Moroz

Faculty of Mathematics and Computer Science,
Adam Mickiewicz University, Poznań
moroz@amu.edu.pl

Abstract. We consider pregroup grammars with letter promotions of the form $p^{(m)} \Rightarrow q^{(n)}$, $p \Rightarrow 1, 1 \Rightarrow q$. We prove a variant of Lambek's normalization theorem [5] for the calculus of pregroups enriched with such promotions and present a polynomial parsing algorithm for the corresponding pregroup grammars. The algorithm extends that from [8], elaborated for pregroup grammars without letter promotions. The normalization theorem, restricted to letter promotions without 1, was proved in [3,4] while the present version was stated in [4] without proof and used to show that the word problem for letter promotions with unit is polynomial. Our results are contained in the unpublished PhD thesis [9].

1 Introduction and Preliminaries

The paper continues and extends some results of [4] and [8]. [4] considered pregroups and pregroup grammars with letter promotions and with letter promotions with unit. A Lambek-style normalization theorem for pregroups with letter promotions is proved. A similar theorem for pregroups with letter promotions with unit is stated and here we give the proof of the theorem. The proof of the normalization theorem for letter promotions with 1 essentially refines that from [3,4], but does not follow directly from them, since one must handle new contraction and expansion steps. In [4] it is also proved that the word problem and the membership problem for pregroups with letter promotions can be solved in polynomial time. Similar results are given for pregroups with letter promotions with unit. In [8] we propose a polynomial dynamic parsing algorithm for pregroup grammars and give the proof of its correctness. In this paper we show that the algorithm can be modified to work for pregroup grammars with letter promotions with unit. The results were stated in an unpublished PhD thesis [9].

Pregroups were introduced by Lambek [5] as an algebraic tool for the syntactical analysis of sentences. Pregroup grammars belong to lexical grammars since most of linguistic information is encoded in the lexicon. Similarly as in Lambek categorial grammars, syntactical properties of words are described by a finite set of pregroup types. However, the structure of types is different and so is the logic. Pregroup types are elements of a free monoid, generated by iterated adjoints of some atoms, and they are processed using a calculus of free pregroups,

G. Morrill and M.-J. Nederhof (Eds.): FG 2012/2013, LNCS 8036, pp. 52–68, 2013.

also called Compact Bilinear Logic **CBL**. The computational complexity of pregroup grammars is polynomial in contrast to the computational complexity of categorial grammars. Pregroups and pregroup grammars have been successfully applied to parsing diverse natural languages like English, Italian, Polish and Japanese.

However, not all natural language phenomena can be easily described by the formalism of pregroups. Therefore some extensions to pregroup grammars have been proposed. Mater and Fix [7] consider pregroup grammars enriched with some more general assumptions. An interesting problem they propose is called a *letter promotion problem for pregroups*. Their idea is further developed in [3], [4]. Buszkowski and Lin [3] show that the word problem for pregroups with letter promotions, that is assumptions of the form $p^{(m)} \Rightarrow q^{(n)}$, is polynomial when the size of $p^{(n)}$ is counted as $|n|+1$. [4] extends this result for letter promotions with unit $p^{(m)} \Rightarrow 1$, $1 \Rightarrow q^{(n)}$.

CBL enriched with some assumptions is interesting for a few reasons. First of all, we look for extensions to pregroups allowing easier description of some natural language phenomena, see [6]. Moreover, we know that the calculus of pregroups is solvable in polynomial time (see [1]), but the associative Lambek calculus is NP-complete. The calculi admitting letter promotions and letter promotions with unit are still polynomial. It is interesting how far the calculus of pregroups can be generalized while remaining polynomial. Finally, pregroup grammars with letter promotions can directly simulate any cancelation grammar. Cancelation grammars are defined by Buszkowski in [4].

Definition 1. *A* cancelation grammar *is a tuple $G = (\Sigma, V, X, R, I)$, where Σ and V are finite disjoint alphabets (terminal and auxiliary respectively), $X \in V^*$, R is a finite set of cancelation rules and I assigns a finite set of strings over V to any element of Σ. The cancelation rules are of the form $A, B \Rightarrow \varepsilon$ and $A \Rightarrow B$, for $A, B \in V$. G assigns a string $Y \in V^*$ to a string $a_1...a_n$ of elements from Σ if there exist strings $Y_1...Y_n$ such that $Y_i = I(a_i)$, $i = 1, ..., n$ and $Y_1...Y_n$ reduces to Y by a finite number of applications of rules from R.*

The *language of a cancelation grammar* consists of all strings on Σ^+ which are assigned the designated type X by G. Pregroup grammars (also with letter promotions) are a special kind of cancelation grammars. Conversely, every cancelation rule $A, B \Rightarrow \varepsilon$ can directly be simulated by the letter promotion $A \Rightarrow B^l$, i.e. $A \Rightarrow B^{(-1)}$.

Definition 2. *A* pregroup *is an algebra $(M, \leqslant, \cdot, l, r, 1)$, such that $(M, \leqslant, \cdot, 1)$ is a partially ordered monoid and l, r are unary operations on M satisfying* adjoint laws*:*

$$(Al)\ a^l a \leqslant 1 \leqslant a a^l$$

$$(Ar)\ a a^r \leqslant 1 \leqslant a^r a,$$

for all $a \in M$.

The elements a^l and a^r are called *left* and *right adjoint*, respectively. Let us notice that left and right adjoints are unique for each element of M.

One defines *iterated adjoints* $a^{(n)} = a^{rr\ldots r}$ and $a^{(-n)} = a^{ll\ldots l}$, where n is a non-negative integer, ^{r} and ^{l} are iterated n times and a is any element of a pregroup M. By definition $a^{(0)} = a$.

To apply pregroups to parsing natural languages Lambek uses the notion of a *free pregroup* generated by a poset. Let us assume $(P, \leqslant)$ is a nonempty, finite poset. Then elements of P are called *atoms* and they are denoted by letters p, q, r. Expressions of the form $p^{(n)}$, where $p \in P$ and n is any integer, are called *terms* and denoted by t, u. Finite strings of terms are called *types* and denoted by X, Y, Z. Types are assigned to words in the lexicon and they refer to the role the given word takes in a sentence. Usually, one word can be assigned many different types.

The following rules define a binary relation $\Rightarrow$ on the set of types:

(CON) $X, p^{(n)}, p^{(n+1)}, Y \Rightarrow X, Y$,
(EXP) $X, Y \Rightarrow X, p^{(n+1)}, p^{(n)}, Y$,
(POS) $X, p^{(n)}, Y \Rightarrow X, q^{(n)}, Y$, if $p \leqslant q$ and n is even or $q \leqslant p$ and n is odd.

(CON), (EXP), (POS) are called Contraction, Expansion and Poset rules, respectively. Poset rules were originally called Induced Step (IND) by Lambek [5]. Actually, $\Rightarrow$ is a reflexive and transitive closure of the relation defined by these clauses. Let us notice that $X \Rightarrow Y$ is true iff X can be rewritten into Y by a finite number of applications of these rules. This rewriting system is Lambek's original form of the logic of pregroups, which is also called ***CBL***.

Additionaly, one defines a useful rule called *Generalized Contraction*, which combines (CON) and (POS) and similarly Generalized Expansion (GEXP):

(GCON) $X, p^{(n)}, q^{(n+1)}, Y \Rightarrow X, Y$,
(GEXP) $X, Y \Rightarrow X, p^{(n+1)}, q^{(n)} Y$,

where in both cases $p \leqslant q$ and n is even, or $q \leqslant p$ and n is odd.

Lambek proves a normalization theorem for **CBL** [5]:

> if $X \Rightarrow Y$ in **CBL**, then there exist Z and U such that $X \Rightarrow Z$ by applying (GCON) only, $Z \Rightarrow U$ by applying (POS) only, and $U \Rightarrow Y$ by applying (GEXP) only.

Therefore, if Y is a term or $Y \Rightarrow \varepsilon$, then $X \Rightarrow Y$ in **CBL** if and only if X can be reduced to Y without (GEXP), that is using (CON) and (POS) only.

It is useful to define X^r and X^l for any type X:

$$\varepsilon^l = \varepsilon = \varepsilon^r;$$
$$X^r = (p_1^{(n_1)} \ldots p_k^{(n_k)})^r = p_k^{(n_k+1)} \ldots p_1^{(n_1+1)},$$
$$X^l = (p_1^{(n_1)} \ldots p_k^{(n_k)})^l = p_k^{(n_k-1)} \ldots p_1^{(n_1-1)},$$

where $n_1, \ldots, n_m$ are arbitrary integers.

Definition 3. *A* pregroup grammar *is a structure $G = (\Sigma, P, I, s, R)$ where Σ is a finite alphabet (that is a lexicon), P is a finite set of atoms, I is a finite relation assigning types on P to symbols from Σ, s is the denoted type s, and R is a partial ordering on P.*

If R is fixed, one writes $p \leqslant r$ for pRq. For $a \in \Sigma,\ I(a)$ denotes the set of all types X such that $(a, X) \in I$. Let us assume $x \in \Sigma^+,\ x = a_1...a_n\ (a_i \in \Sigma)$. One says that the grammar G *assigns* type Y to x if there exist types $X_i \in I(a_i)$, $i = 1, ..., n$, such that $X_1, ..., X_n \Rightarrow Y$ in **CBL**; we write $x \rightarrow_G Y$. Then *the language of a pregroup grammar* G, denoted by $L(G)$, consists of all strings $x \in \Sigma^+$ to which the grammar G assigns the denoted type s. Due to the normalization theorem while parsing pregroup grammars, one may restrict the rules to only (CON) and (POS).

To describe a pregroup grammar for a given language one has to define a set of terms, fix a lexicon consisting of words of the language and types assigned to them and define a partial order on types, that is, fix the assumptions. One can say that assumptions express different forms of subtyping. Let us consider a few examples. If *he* is assigned type π_3 (subject in third person), *likes* - $\pi_3^r\ s_1\ o^l$ and *books* - n_2 (plural noun), then the sentence *He likes books.* can be parsed as follows: $(\pi_3)(\pi_3^r\ s_1\ o^l)(n_2) \leqslant s_1 o^l n_2 \leqslant s_1 o^l o \leqslant s_1$ (using the assumption $n_2 \leqslant o$, as a plural noun can take part of an object in a sentence). The string is assigned type s_1 i.e. the type of statement in present tense and by the assumption $s_1 \leqslant s$, it is a statement (type s). Clearly, the types assigned above are not unique. For example *books* is also of type $\pi_3^r s_1 o^l$ (a transitive verb in third person, present tense). The above reduction can be depicted by the following links:

He *likes* *books.*
$\underbrace{(\pi_3)(\pi_3^r}\ s_1\ \underbrace{o^l)(n_2)}$

Let us consider a complete system of **CBL** with letter promotions obtained by modifying (POS) to Promotion Rules (PRO):

(PRO) $X, p^{(m+k)}, Y \Rightarrow X, q^{(n+k)}, Y$ if either k is even and $p^{(m)} \Rightarrow q^{(n)}$ is an assumption, or k is odd and $q^{(n)} \Rightarrow p^{(m)}$ is an assumption.

A pregroup grammar with letter promotions is defined similarly to a pregroup grammar.

Definition 4. *A* pregroup grammar with letter promotions *is a pregroup grammar* $G = (\Sigma, P, I, s, R)$ *in which* R *is the set of assumptions extended by a set of letter promotions. We require that* $P(R) \subseteq P$, *where* $P(R)$ *denotes the set of atoms appearing in assumptions from* R.

By $R \vdash_{\mathbf{CBL}} X \Rightarrow Y$ we mean that X can be transformed into Y by a finite number of applications of (CON), (EXP) and (PRO), restricted to the assumption from a set of letter promotions R. (POS) is treated as an instance of (PRO).

Assuming that $t \Rightarrow u$ is an instance of (PRO) restricted to the assumptions from R, that is X, Y are empty, we write $t \Rightarrow_R u$. We write $t \Rightarrow_R^* u$ if there exist terms $t_0, \ldots, t_k$ such that $k \geqslant 0,\ t_0 = t,\ t_k = u$, and $t_{i-1} \Rightarrow_R t_i$, for all $i = 1, \ldots, k$. Hence, $\Rightarrow_R^*$ is the reflexive and transitive closure of $\Rightarrow_R$.

For **CBL** with letter promotions one defines a generalization of the rules (CON) and (EXP), which are derivable in **CBL** with assumptions from R.

(GCON-R) $X, p^{(m)}, q^{(n+1)}, Y \Rightarrow X, Y$ if $p^{(m)} \Rightarrow_R^* q^{(n)}$,
(GEXP-R) $X, Y \Rightarrow X, p^{(m+1)}, q^{(n)}, Y$ if $p^{(m)} \Rightarrow_R^* q^{(n)}$.

Clearly, (CON) is a special instance of (GCON-R) while (EXP) is a special instance of (GEXP-R). One can treat any iteration of (PRO) as a single step.

(PRO-R) $X, t, Y \Rightarrow X, u, Y$ if $t \Rightarrow_R^* u$.

As a consequence of the normalization theorem, we get $R \vdash_{\mathbf{CBL}} t \Rightarrow u$ iff $t \Rightarrow_R^* u$; see [3,4].

Definition 5. *The* letter promotion problem for pregroups **(LPPP)** *is stated as follows: for the given finite set R of letter promotions, and terms t, u, verify whether $t \Rightarrow u$ in **CBL** enriched with all promotions from R as assumptions.*

Shortly, **(LPPP)** consists of verifying whether $t \Rightarrow_R^* u$ for given R, t, u.

Buszkowski and Lin [3] prove that **LPPP** is polynomial, provided that the size of $p^{(n)}$ is counted as $|n| + 1$ (it is natural, since $p^{(n)}$ abbreviates the n-th iteration of l's or r's). This count is used in the present paper. [4] extends this result for letter promotions with unit and shows that the membership problem for the corresponding pregroup grammars is polynomial, and these grammars are equivalent to CFGs (but the latter does not directly imply the polynomiality of the membership problem; see [2] for discussion).

The main result of the present paper is a polynomial parsing algorithm for pregroup grammars admitting letter promotions of the form $p^{(m)} \Rightarrow q^{(n)}$, $p \Rightarrow 1$, $1 \Rightarrow q$. This algorithm refines an earlier one for pregroup grammars [8]; the latter adapts a method of Savateev [11], elaborated for Unidirectional Lambek Calculus. Section 3 presents our algorithm, the proof of its correctness and a related algorithm returning the reduction. Since a Lambek-style normalization theorem is essentially used, we give its full proof in section 2 (the theorem was proved in [9] and stated without proof in [4]).

2 The Normalization Theorem

Letter promotions with unit are promotions allowing 1, that is letter promotions of the form: $p^{(m)} \Rightarrow 1$ or $1 \Rightarrow q^{(n)}$. We add 1 to the set of terms. Notice that in pregroups the assumption $p^{(m)} \Rightarrow 1$ is equivalent to $p \Rightarrow 1$ if m is even and to $1 \Rightarrow p$ if m is odd. Similarly, the assumption $1 \Rightarrow p^{(m)}$ is equivalent to $1 \Rightarrow p$ if m is even and to $p \Rightarrow 1$ if m is odd. Therefore, in what follows, we consider letter promotions with unit only of the form $p \Rightarrow 1$ and $1 \Rightarrow q$.

A complete system of **CBL** with letter promotions with unit is obtained by adding two new rules to **CBL** with letter promotions.

Definition 6. *A* complete system of **CBL** with letter promotions with unit *consists of the following rules:*

(CON) $X, p^{(n)}, p^{(n+1)}, Y \Rightarrow X, Y$,
(EXP) $X, Y \Rightarrow X, p^{(n+1)}, p^{(n)}, Y$,
(PRO) $X, p^{(m+k)}, Y \Rightarrow X, q^{(n+k)}, Y$, *if either k is even and $p^{(m)} \Rightarrow q^{(n)}$ is an assumption, or k is odd and $q^{(n)} \Rightarrow p^{(m)}$ is an assumption.*

(PRO-C) $X, p^{(m)}, Y \Rightarrow X, Y$, *if either* m *is even and* $p \Rightarrow 1$ *is an assumption, or* m *is odd and* $1 \Rightarrow p$ *is an assumption,*
(PRO-E) $X, Y \Rightarrow X, q^{(n)}, Y$, *if either* n *is even and* $1 \Rightarrow q$ *is an assumption, or* n *is odd and* $q \Rightarrow 1$ *is an assumption.*

Consequently, (PRO-C) is a contracting promotion step, (PRO-E) is an expanding promotion step and (PRO) is a neutral promotion step.

We fix a finite set $R1$ of letter promotions, possibly with unit. We write $t \Rightarrow_{R1} u$ if $t \Rightarrow u$ is an instance of (PRO), (PRO-C) or (PRO-E) restricted to the set of assumptions $R1$ (X, Y are empty). We write $t \Rightarrow^*_{R1} u$ if there exist terms $t_0, ..., t_k$ such that $k \geqslant 0,\ t_0 = t,\ t_k = u,\ t_{i-1} \Rightarrow_{R1} t_i$, for all $i = 1, ..., k$.

We introduce some new rules derivable in **CBL** with assumptions from $R1$.

(GCON-R1) $X, p^{(m)}, q^{(n+1)}, Y \Rightarrow X, Y$, if $p^{(m)} \Rightarrow^*_{R1} q^{(n)}$
(GEXP-R1) $X, Y \Rightarrow X, p^{(n+1)}, q^{(m)}, Y$, if $p^{(n)} \Rightarrow^*_{R1} q^{(m)}$
(PRO-R1) $X, t, Y \Rightarrow X, u, Y$ if $t \Rightarrow^*_{R1} u$ and $t \neq 1$ and $u \neq 1$
(PRO-C-R1) $X, t, Y \Rightarrow X, Y$ if $t \Rightarrow^*_{R1} 1$ and $t \neq 1$
(PRO-E-R1) $X, Y \Rightarrow X, u, Y$ if $1 \Rightarrow^*_{R1} u$ and $u \neq 1$

There holds a normalization theorem for **CBL** with letter promotions with unit:

Theorem 1. *If* $R1 \vdash_{\mathbf{CBL}} X \Rightarrow Y$, *then there exist* Z, U *such that* $X \Rightarrow Z$ *by a finite number of instances of (GCON-R1) and (PRO-C-R1),* $Z \Rightarrow U$ *by a finite number of instances of (PRO-R1) and* $U \Rightarrow Y$ *by a finite number of instances of (GEXP-R1) and (PRO-E-R1).*

Proof. Let us start with some notions. A sequence $X_0, \ldots, X_k$ such that $X = X_0,\ Y = X_k$ and, for any $i = 1, \ldots, k,\ X_{i-1} \Rightarrow X_i$ is an instance of (GCON-R1), (GEXP-R1), (PRO-E-R1), (PRO-C-R1) or (PRO-R1) is called a *derivation of* $X \Rightarrow Y$ *from the set of assumptions R1.* Clearly, $R1 \vdash X \Rightarrow Y$ iff there exists a derivation of $X \Rightarrow Y$ of this form. k is the length of the derivation. If a derivation has a form required by the Theorem, then it is called a *normal derivation.*

We prove that every derivation $X_0, \ldots, X_k$ of $X \Rightarrow Y$ can be transformed into a normal derivation of length not greater than k. We proceed by induction on k.

We should notice that for $k = 0$ and $k = 1$ the initial derivation is normal. For $k = 0$, it suffices to take $X = Z = U = Y$. For $k = 1$, if $X \Rightarrow Y$ is an instance of (GCON-R1) or (PRO-C-R1), one takes $Z = U = Y$, if $X \Rightarrow Y$ is an instance of (GEXP-R1) or (PRO-E-R1), one takes $X = Z = U$, and if $X \Rightarrow Y$ is an instance of (PRO-R1), one takes $X = Z$ and $U = Y$.

Assume now $k > 1$. The derivation $X_1, \ldots, X_k$ is shorter, whence it can be transformed into a normal derivation $Y_1, \ldots, Y_l$ such that $X_1 = Y_1,\ X_k = Y_l$ and $l \leqslant k$. If $l < k$, then $X_0, Y_1, \ldots, Y_l$ is a derivation of $X \Rightarrow Y$ of length less than k, whence it can be transformed into a normal derivation, by the induction hypothesis. So assume $l = k$.

Case 1. $X_0 \Rightarrow X_1$ is an instance of (GCON-R1). Then $X_0, Y_1, \ldots., Y_l$ is a normal derivation of $X \Rightarrow Y$ from $R1$.

Case 2. $X_0 \Rightarrow X_1$ is an instance of (PRO-C-R1). Then $X_0, Y_1, \ldots, Y_l$ is a normal derivation of $X \Rightarrow Y$ from $R1$.

Case 3. $X_0 \Rightarrow X_1$ is an instance of (GEXP-R1), assume $X_0 = UV$; $X_1 = Up^{(n+1)}q^{(m)}V$, and $p^{(n)} \Rightarrow^*_{R1} q^{(m)}$. We consider two subcases.

Case 3.1. Neither any (GCON-R1)-step nor any (PRO-C-R1)-step of $Y_1, \ldots, Y_l$ acts on the designated occurrences of $p^{(n+1)}, q^{(m)}$. If also no (PRO-R1)-step of $Y_1, \ldots, Y_l$ acts on these designated terms, then we drop $p^{(n+1)}q^{(m)}$ from all types appearing in (GCON-R1)-steps, (PRO-C-R1)-steps and (PRO-R1)-steps of $Y_1, \ldots, Y_l$, then we introduce $p^{(n+1)}q^{(m)}$ by a single instance of (GEXP-R1), and continue the (GEXP-R1)-steps and (PRO-E-R1)-steps of $Y_1, \ldots, Y_l$; this yields a normal derivation of $X \Rightarrow Y$ of length k. Otherwise, let $Y_{i-1} \Rightarrow Y_i$ be the first (PRO-R1)-step of $Y_1, \ldots, Y_l$ which acts on $p^{(n+1)}$ or $q^{(m)}$.

(I) If $Y_{i-1} \Rightarrow Y_i$ acts on $p^{(n+1)}$, then there exists a term $r^{(m')}$ and types T, W such that $Y_{i-1} = Tp^{(n+1)}W$, $Y_i = Tr^{(m')}W$ and $p^{(n+1)} \Rightarrow^*_{R1} r^{(m')}$. Consequently, $r^{(m'-1)} \Rightarrow^*_{R1} p^{(n)}$, whence $r^{(m'-1)} \Rightarrow^*_{R1} q^{(m)}$. Then we can replace the derivation $X_0, Y_1, \ldots, Y_l$ by a shorter derivation: first apply (GEXP-R1) of the form $U, V \Rightarrow U, r^{(m')}, q^{(m)}, V$, then derive $Y_1, \ldots, Y_{i-1}$ in which $p^{(n+1)}$ is replaced by $r^{(m')}$, drop Y_i, and continue $Y_{i+1}, \ldots, Y_l$. By the induction hypothesis, this derivation can be transformed into a normal derivation of length less than k.

(II) If $Y_{i-1} \Rightarrow Y_i$ acts on $q^{(m)}$, then there exist a term $r^{(m')}$ and types T, W such that $Y_{i-1} = Tq^{(m)}W$, $Y_i = Tr^{(m')}W$ and $q^{(m)} \Rightarrow^*_{R1} r^{(m')}$. Consequently, $p^{(n)} \Rightarrow^*_{R1} r^{(m')}$, and we can replace the derivation $X_0, Y_1, ldots, Y_l$ by a shorter derivation: first apply (GEXP-R1) of the form $U, V \Rightarrow U, p^{(n+1)}, r^{(m')}, V$, then derive $Y_1, \ldots, Y_{i-1}$ in which $q^{(m)}$ is replaced by $r^{(m')}$, drop Y_i, and continue $Y_{i+1}, \ldots, Y_l$. Again we apply the induction hypothesis.

Case 3.2. Some (GCON-R1)-step of $Y_1, \ldots, Y_l$ acts on (some of) the designated occurrences of $p^{(n+1)}, q^{(m)}$. Let $Y_{i-1} \Rightarrow Y_i$ be the first step of that kind. There are three possibilities.

(I) This step acts on both $p^{(n+1)}$ and $q^{(m)}$. Then, the derivation $X_0, Y_1, \ldots, Y_l$ can be replaced by a shorter derivation: drop the first application of (GEXP-R1), then derive $Y_1, \ldots, Y_{i-1}$ in which $p^{(n+1)}q^{(m)}$ is omitted, drop Y_i, and continue $Y_{i+1}, \ldots, Y_l$. We apply the induction hypothesis.

(II) This step acts on $p^{(n+1)}$ only. Then, $Y_{i-1} = Tr^{(m')}p^{(n+1)}q^{(m)}W$, $Y_i = Tq^{(m)}W$ and $r^{(m')} \Rightarrow^*_{R1} p^{(n)}$. The derivation $X_0, Y_1, \ldots, Y_l$ can be replaced by a shorter, normal derivation: drop the first application of (GEXP-R1), then derive $Y_1, \ldots, Y_{i-1}$ in which $p^{(n+1)}q^{(m)}$ is omitted, derive Y_i by a (PRO-R1)-step (notice $r^{(m')} \Rightarrow^*_{R1} q^{(m)}$), and continue $Y_{i+1}, \ldots, Y_l$.

(III) This step acts on $q^{(m)}$ only. Then, $Y_{i-1} = Tp^{(n+1)}q^{(m)}r^{(m'+1)}W$, $Y_i = Tp^{(n+1)}W$ and $q^{(m)} \Rightarrow^*_{R1} r^{(m')}$. The derivation $X_0, Y_1, \ldots, Y_l$ can be replaced by a shorter derivation: drop the first application of (GEXP-R1), then derive $Y_1, \ldots, Y_{i-1}$ in which $p^{(n+1)}q^{(m)}$ is omitted, derive Y_i by

a (PRO-R1)-step (notice $r^{(m'+1)} \Rightarrow^*_{R1} p^{(n+1)}$), and continue $Y_{i+1}, \ldots, Y_l$. We apply the induction hypothesis.

Case 3.3. Some (PRO-C-R1)-step of $Y_1, \ldots, Y_l$ acts on (some of) the designated occurrences of $p^{(n+1)}, q^{(m)}$. Let $Y_{i-1} \Rightarrow Y_i$ be the first step of that kind. There are two possibilities.

(I) This step acts on $p^{(n+1)}$. Then $Y_{i-1} = Tp^{(n+1)}W$, $Y_i = TW$ and $p^{(n+1)} \Rightarrow^*_{R1} 1$. Thus, $1 \Rightarrow^*_{R1} p^{(n)}$, and hence $1 \Rightarrow^*_{R1} q^{(m)}$. We can replace the derivation $X_0, Y1, \ldots, Y_l$ by a shorter derivation: start with an application of (PRO-E-R1) of the form $UV \Rightarrow Uq^{(m)}V$ derive $Y_1, \ldots, Y_{i-1}$ in which $p^{(n+1)}$ is replaced by 1, drop Y_i, and continue $Y_{i+1}, \ldots, Y_l$. Again we apply the induction hypothesis.

(II) This step acts on $q^{(m)}$. Then, $Y_{i-1} = Tq^{(m)}W$, $Y_i = TW$ and $q^{(m)} \Rightarrow^*_{R1} 1$. Thus, $p^{(n)} \Rightarrow^*_{R1} 1$, and we can replace the derivation $X_0, Y_1, \ldots, Y_l$ by a shorter derivation: start with an application of (PRO-E-R1) of the form $U, V \Rightarrow U, p^{(n+1)}, V$, derive $Y_1, \ldots, Y_{i-1}$ in which $q^{(m)}$ is replaced by 1, drop Y_i, and continue $Y_{i+1}, \ldots, Y_l$. Again we apply the induction hypothesis.

Case 4. $X_0 \Rightarrow X_1$ is an instance of (PRO-E-R1), assume $X_0 = UV$, $X_1 = Uq^{(m)}V$, and $1 \Rightarrow^*_{R1} q^{(m)}$. There are three subcases.

Case 4.1. Neither any (GCON-R1)-step nor any (PRO-C-R1) of $Y_1, \ldots, Y_l$ acts on the designated occurrence of $q^{(m)}$. If also no (PRO-R1)-step of $Y_1, \ldots, Y_l$ acts on this designated term, then we drop the first application of the (PRO-E-R1)-step, omit $q^{(m)}$ in all types appearing in (GCON-R1)-steps, (PRO-C-R1)-steps and (PRO-R1)-steps of $Y_1, \ldots, Y_l$, then introduce $q^{(m)}$ by a single instance of (PRO-E-R1), and continue with the (GEXP-R1)-steps of $Y_1, \ldots, Y_l$; this yields a normal derivation of $X \Rightarrow Y$ of length k.

Otherwise, let $Y_{i-1} \Rightarrow Y_i$ be the first (PRO-R1)-step of $Y_1, \ldots, Y_l$ which acts on $q^{(m)}$. Then, there exist a term $r^{(m')}$ and types T, W such that $Y_{i-1} = Tq^{(m)}W$, $Y_i = Tr^{(m')}W$ and $q^{(m)} \Rightarrow^*_{R1} r^{(m')}$. Thus $1 \Rightarrow^*_{R1} r^{(m')}$, and we can replace the derivation $X_0, Y_1, \ldots, Y_l$ by a shorter derivation: first apply (PRO-E-R1) of the form $UV \Rightarrow Ur^{(m')}V$, then derive $Y_1, \ldots, Y_{i-1}$ in which $q^{(m)}$ is replaced by $r^{(m')}$, drop Y_i, and continue $Y_{i+1}, \ldots, Y_l$. Again we apply the induction hypothesis.

Case 4.2. Some (GCON-R1)-step of $Y_1, \ldots, Y_l$ acts on the designated occurrence of $q^{(m)}$. Let $Y_{i-1} \Rightarrow Y_i$ be the first step of that kind. Then, $Y_{i-1} = Tq^{(m)}r^{(m'+1)}W$, $Y_i = TW$ and $q^{(m)} \Rightarrow^*_{R1} r^{(m')}$. The derivation $X_0, Y_1, \ldots, Y_l$ can be replaced by a shorter derivation: drop the first application of (PRO-E-R1), then derive $Y_1, \ldots, Y_{i-1}$ in which $q^{(m)}$ is omitted, derive Y_i by a (PRO-C-R1)-step (notice $r^{(m'+1)} \Rightarrow^*_{R1} 1$), and continue $Y_{i+1}, \ldots, Y_l$. We apply the induction hypothesis.

Case 4.3. Some (PRO-C-R1)-step of $Y_1, \ldots, Y_l$ acts on the designated occurrence of $q^{(m)}$. Let $Y_{i-1} \Rightarrow Y_i$ be the first step of that kind. Then, $Y_{i-1} = Tq^{(m)}W$, $Y_i = TW$ and $q^{(m)} \Rightarrow^*_{R1} 1$. Then we can replace the derivation $X_0, Y_1, \ldots, Y_l$ by a shorter, normal derivation: drop the first

application of the (PRO-E-R1)-step, derive $Y_1, \ldots, Y_{i-1}$ in which $q^{(m)}$ is omitted, drop Y_i, and continue $Y_{i+1}, \ldots, Y_l$.

Case 5. $X_0 \Rightarrow X_1$ is an instance of (PRO-R1), say $X_0 = UtV$, $X_1 = UuV$, $t \Rightarrow^*_{R1} u$, $u \neq 1$ and $t \neq 1$.

Case 5.1. Neither any (GCON-R1)-step nor any (PRO-C-R1)-step of $Y_1, \ldots, Y_l$ acts on the designated occurrence of u. Then $X_0, Y_1, \ldots, Y_l$ can be transformed into a normal derivation of the length k: drop the first application of (PRO-R1), apply all (GCON-R1)-steps of $Y_1, \ldots, Y_l$ in which the designated occurrence of u is replaced by t and apply all (PRO-C-R1)-steps, then apply a (PRO-R1)-step which changes t into u, and continue the remaining steps of $Y_1, \ldots, Y_l$.

Case 5.2. Some (GCON-R1)-step of $Y_1, \ldots, Y_l$ acts on the designated occurrence of u. Let $Y_{i-1} \Rightarrow Y_i$ be the first step of that kind. There are two possibilities.
(I) $Y_{i-1} = Tuq^{(n+1)}W$, $Y_i = TW$ and $u \Rightarrow^*_{R1} q^{(n)}$. Since $t \Rightarrow^*_{R1} q^{(n)}$, then $X, Y_1, \ldots, Y_l$ can be transformed into a shorter derivation: drop the first application of (PRO-R1), derive $Y_1, \ldots, Y_{i-1}$ in which the designated occurrence of u is replaced by t, derive Y_i by a (GCON-R1)-step of the form $T, t, q^{(n+1)}, W \Rightarrow T, W$, and continue $Y_{i+1}, \ldots, Y_l$. We apply the induction hypothesis.
(II) $u = q^{(n+1)}, Y_{i-1} = Tp^{(m)}uW$, $Y_i = TW$ and $p^{(m)} \Rightarrow^*_{R1} q^{(n)}$. Let $t = r(n')$. We have $q^{(n)} \Rightarrow^*_{R1} r^{(n'-1)}$, whence $p^{(m)} \Rightarrow^*_{R1} r^{(n'-1)}$. The derivation $X_0, Y_1, \ldots, Y_l$ can be transformed into a shorter derivation: drop the first application of (PRO-R1), derive $Y_1, \ldots, Y_{i-1}$ in which the designated occurrence of u is replaced by t, derive Y_i by a (GCON-R1)-step of the form $T, p^{(m)}, r^{(n')}, W \Rightarrow T, W$, and continue $Y_{i+1}, \ldots, Y_l$. We apply the induction hypothesis.

Case 5.3. Some (PRO-C)-step of $Y_1, \ldots, Y_l$ acts on the designated occurrence of u. Let $Y_{i-1} \Rightarrow Y_i$ be the first step of that kind. Then there exists types T, W, such that $Y_{i-1} = TuW$, $Y_i = TW$ and $u \Rightarrow^*_{R1} 1$. Thus $t \Rightarrow^*_{R1} 1$. The derivation $X_0, Y_1, \ldots, Y_l$ can be transformed into a normal derivation of length k: drop the first application of (PRO-R1), apply a (PRO-C-R1)-step of the form: $TtW \Rightarrow^*_{R1} TW$ derive $Y_1, \ldots, Y_{i-1}$ in which the designated occurrence of u is omitted, drop Y_i and continue $Y_{i+1}, \ldots, Y_l$.

Consequently, there holds: if $R1 \vdash_{\mathbf{CBL}} X \Rightarrow t$, where t is a term, then X can be reduced to t by (GCON-R1), (PRO-C-R1) and (PRO-R1) only. Moreover one can prove that: $R1 \vdash_{\mathbf{CBL}} t \Rightarrow u$ if, and only if, $t \Rightarrow^*_{R1} u$ ([4,9]).

Definition 7. The letter promotion problem for pregroups with unit (**LPPP1**) *can be stated as follows: verify, whether* $t \Rightarrow^*_{R1} u$ *for given* t, u *and* $R1$.

The problem can be solved in polynomial time. The algorithm for solving the problem is based on the idea given for pregroups with letter promotions. For details see [4,9].

3 The Parsing Algorithm

Definition 8. *A* pregroup grammar with letter promotions with unit *is a tuple* $G = (\Sigma, P, R1, s, I)$*, such that* $R1$ *is the set of assumptions (letter promotions) and* $P(R1) \subseteq P$*, where* $P(R1)$ *is the set of atoms appearing in assumptions from* $R1$*.* Σ, P, s, I *are defined in the same way as for pregroup grammars.*

Assume $T^+(G)$ is a set of all types appearing in I and $T(G)$ is a set of all terms forming types from $T^+(G)$. Then all generalized contractions $tu \Rightarrow \varepsilon$ $(t, u \in T(G))$ derivable from $R1$ in **CBL** can be computed in polynomial time.

We define a polynomial, dynamic, parsing algorithm for pregroup grammars with letter promotions with unit on the basis of the algorithm for pregroup grammars described in [8]. The algorithm can also be used for pregroup grammars with letter promotions without unit by omitting the cases when promotions with 1 are considered, see [9]. Our goal is to obtain an appropriate derivation of the given string $x = a_1, \ldots, a_n$, $a_i \in \Sigma$, $i = 1, \ldots, n$ if x is a member of the language generated by a pregroup grammar with letter promotions with unit G, that is if $x \in L(G)$. The algorithm is constructed in a style proposed by Savateev (see [11]) for Unidirectional Lambek Calculus. It is a dynamic algorithm working on a special form of a string, containing all possible type assignments for words of the sentence to parse.

We fix a grammar $G = (\Sigma, P, R1, s, I)$. We take a string of words $x \in \Sigma^+$ such that $x = a_1 ... a_n$. We use special symbols $*, \langle, \rangle$. Let us denote:

- $\mathcal{Z}$ - the set of integers,
- $T = \{p^{(n)} : p \in P,\ n \in \mathcal{Z}\}$ - the set of terms,
- $X, Y, Z, ..$ - elements of T^*,
- $k^a = |I(a)|$,
- X_j^a - the j-th possible assignment of type to a, $1 \leqslant j \leqslant k^a$ (hence $I(a) = \{X_1^a, ...X_{k^a}^a\}$),
- $Q^a = \langle *X_1^a * X_2^a * ... * X_{k^a}^a *\rangle$,
- $W^x = Q^{a_1} ... Q^{a_n} \langle *s^{(1)}$, $W^x \in (T \cup \{*, \langle, \rangle\})^*$
- W_i^x - the i-th symbol of the string W^x, $1 \leqslant i \leqslant |W^x|$,
- $W_{[i,j]}^x = W_i^x W_{i+1}^x ... W_j^x$ - the substring of W^x, $1 \leqslant i \leqslant j \leqslant |W^x|$ ($W_{[i,i]}^x$ stands for W_i^x).

In the following, by a reduction to 1 we mean a reduction to ε in **CBL** with letter promotions with 1. We define an auxiliary function M as follows. Let $M'(i,j), 1 \leqslant i \leqslant j \leqslant |W^x|$ be a function such that $M'(i,j) = 1$ iff one of the following conditions holds:

- **M1.** $W_{[i,j]}^x \in T^+$ and it reduces to 1.
- **M2a.** $W_{[i,j]}^x$ is of the form $\langle ... \rangle ... \langle \mathbf{V} * Z$, where:
  - $Z \in T^+$
  - $\mathbf{V}$ contains no angle brackets
  - in $W_{[i,j]}^x$ there are precisely g $(g \geqslant 0)$ pairs of matching angle brackets; for the h-th pair of them there is a substring of the form $*X_h*$ in between them such that $X_h \in T^+$ and the string $X_1 ... X_g Z$ reduces to 1

- **M2b.** $W^x_{[i,j]}$ is of the form $Y * \mathbf{U}\rangle ... \langle ...\rangle$, where:
  - $Y \in T^+$
  - $\mathbf{U}$ contains no angle brackets
  - in $W^x_{[i,j]}$ there are precisely g ($g \geqslant 0$) pairs of matching angle brackets; for the h-th pair of them there is a substring of the form $*X_h*$ in between them such that $X_h \in T^+$ and the string $YX_1...X_g$ reduces to 1
- **M3.** $W^x_{[i,j]}$ is of the form $Y * \mathbf{U}\rangle ... \langle \mathbf{V} * Z$, where:
  - $Y, Z \in T^+$
  - $\mathbf{U}, \mathbf{V}$ contain no angle brackets
  - in $W^x_{[i,j]}$ there are precisely g ($g \geqslant 0$) pairs of matching angle brackets; for the h-th pair of them there is a substring of the form $*X_h*$ in between them such that $X_h \in T^+$ and the string $YX_1...X_gZ$ reduces to 1
- **M4.** $W^x_{[i,j]}$ is of the form $\langle ...\rangle ... \langle ...\rangle$, where:
  - in $W^x_{[i,j]}$ there are precisely g ($g \geqslant 1$) pairs of matching angle brackets; for the h-th pair of them there is a substring of the form $*X_h*$ in between them such that $X_h \in T^+$ and the string $X_1...X_g$ reduces to 1

In all other cases $M'(i,j) = 0$.

Clearly, the whole string W^x is of the form **M2a**. Therefore, $M'(1, |W^x|) = 1$ entails the existence of a string $X_1 \dots X_n s^r$ reducing to 1. Each X_i is the type that needs to be found for a_i. Thus, a solution to the recognition problem is found, i.e. $x \in L(G)$. On the other hand, if $M'(1, |W^x|) = 0$, then there is no string reducing to 1, in which exactly one element comes from each pair of angle brackets and which reduces to 1. It means $x \notin L(G)$.

We start the algorithm by determining the set $Pairs$ of all pairs $(p^{(m)}, q^{(n+1)})$ such that $p, q \in P$ and $p^{(m)} \Rightarrow^*_{R1} q^{(n)}$, and a set $Reducible$ of terms t such that $t \Rightarrow^*_{R1} 1$, which can be done in polynomial time, see [3].

We compute $M'(i,j)$ dynamically. There are two initial cases. The first one computes $M'(i,i) = 1$ in the case when $W^x_i = t$ and $t \in Reducible$. Secondly, one looks for two adjacent terms W^x_i and W^x_{i+j} belonging to the set Pairs. If $(W^x_i, W^x_{i+j}) \in Pairs$ then we put $M'(i, i+1) = 1$.

When we already know $M'(g,h)$, for all $1 \leqslant g < h \leqslant |W^x|$ such that $h - g < j - i$, we can compute $M'(i,j)$. There are several cases:

- **A0.** $W^x_{[i,j]}$ is of the form $p^{(m)} * \mathbf{U}\rangle\langle \mathbf{V} * q^{(n+1)}$, $(p^m, q^{(n+1)}) \in Pairs$ and strings $\mathbf{U}$, $\mathbf{V}$ contain no angle brackets. Then, we put $M'(i,j) = 1$.
- **A1a.** $W^x_i, W^x_j \in T$. If there exists k such that $i \leqslant k < j$, $W^x_k \in T$, $W^x_{(k+1)} \in T$ and both $M'(i,k)$ and $M'(k+1,j)$ are equal to 1, then we put $M'(i,j) = 1$.
- **A1a'.** $W^x_i, W^x_j \in T$. If there exists k such that $i < k \leqslant j$, $W^x_k \in T$, $W^x_{(k+1)} \in T$ and both $M'(i,k-1)$ and $M'(k,j)$ are equal to 1, then we put $M'(i,j) = 1$.
- **A1b.** $W^x_i, W^x_j \in T$. If there exists k such that $i < k < j-1$, $W^x_k = \rangle$, $W^x_{(k+1)} = \langle$ and both $M'(i,k)$ and $M'(k+1,j)$ are equal to 1, then we put $M'(i,j) = 1$.
- **A2.** $W^x_i = p^{(m)}$, $W^x_j = q^{(n+1)}$ and $(p^{(m)}, q^{(n+1)}) \in Pairs$.
  If $M'(i+1, j-1) = 1$, then $M'(i,j) = 1$.

- **A3a.** $W^x_{[i,j]}$ is of the form $\langle ...\rangle ...\langle ...p^{(m)}$, $p \in P, m \in \mathcal{Z}$. If there exists k such that $i < k < j$, $W^x_k = *, W^x_{[i+1,k]}$ contains no angle brackets and $M'(k+1,j) = 1$, then $M'(i,j) = 1$.
- **A3b.** $W^x_{[i,j]}$ is of the form $p^{(m)} ...\rangle ...\langle ...\rangle$, $p \in P, m \in \mathcal{Z}$. If there exists k such that $i < k < j$, $W^x_k = *, W^x_{[k,j-1]}$ contains no angle brackets and $M'(i,k-1) = 1$, then we put $M'(i,j) = 1$.
- **A4a.** $W^x_{[i,j]}$ is of the form $p^{(m)} * ...\rangle ...\langle ...q^{(n+1)}$ and $(p^{(m)}, q^{(n+1)}) \in Pairs$. If $M'(k,j-1) = 1$, where k is the position of the first left angle bracket in the string $W^x_{[i,j]}$, then we put $M'(i,j) = 1$.
- **A4b.** $W^x_{[i,j]}$ is of the form $p^{(m)} ...\rangle ...\langle ... * q^{(n+1)}$ and $(p^{(m)}, q^{(n+1)}) \in Pairs$. If $M'(i+1,k) = 1$, where k is the position of the last right angle bracket in the string $W^x_{[i,j]}$, then $M'(i,j) = 1$.
- **A4c.** $W^x_{[i,j]}$ is of the form $p^{(m)} * ...\rangle \langle ... * q^{(n+1)}$, where the string "..." in between the angle brackets is not empty and $(p^{(m)}, q^{(n+1)}) \in Pairs$. If $M'(k,k') = 1$, where k is the position of the first left angle bracket in the string $W^x_{[i,j]}$ and k' is the position of the last right angle bracket in the string $W^x_{[i,j]}$, then $M'(i,j) = 1$.
- **A4d.** $W^x_{[i,j]}$ is of the form $p^{(m)} * ...\rangle ...\langle ...$, and $p^{(m)} \in Reducible$. If $M'(k,j) = 1$, where k is the position of the first left angle bracket in the string $W^x_{[i,j]}$, then we put $M'(i,j) = 1$.
- **A4e.** $W^x_{[i,j]}$ is of the form $...\rangle ...\langle ...*q^{(n)}$, and $q^{(n)} \in Reducible$. If $M'(i,k) = 1$, where k is the position of the last right angle bracket in the string $W^x_{[i,j]}$, then $M'(i,j) = 1$.
- **A5.** $W^x_{[i,j]}$ is of the form $\langle ...\rangle ...\langle ...\rangle$. If $M'(k,k') = 1$, where W^x_k is a term in between the first pair of angle brackets, $W^x_{k'}$ is a term in between last pair of angle brackets in the string $W^x_{[i,j]}$ and $W^x_{k-1} = *$ and $W^x_{k'+1} = *$, then $M'(i,j) = 1$.
- **A6a.** $W^x_{[i,j]}$ is of the form $p^{(m)} q^{(n)} ...$, and $p^{(m)} \in Reducible$. If $M'(i+1,j) = 1$, then we put $M'(i,j) = 1$.
- **A6b.** $W^x_{[i,j]}$ is of the form $...p^{(m)} q^{(n)}$, and $q^{(n)} \in Reducible$. If $M'(i,j-1) = 1$, then we put $M'(i,j) = 1$.

In all other cases $M'(i,j) = 0$.

Note that the high number of cases is due to the variety of the form of the string W^x. Moreover, observe that from each pair of matching angle brackets exactly one type must be chosen. Let us call any substring of W^x that satisfies all conditions of any of the forms of **M** *accepted*. We start with finding all terms and pairs of adjacent terms that can be reduced to 1, that is the shortest accepted substrings. Then we try to extend them to obtain a longer accepted substring. Obviously the procedure is continued until there are no more possibilities of extending obtained substrings in an acceptable way or the whole string W^x can be reached. For example, cases **(A1a)**, **(A1a')** and **(A1b)** show which conditions two substrings have to satisfy to be concatenated. Case **(A2.)** explains when the substring can be surrounded by a link (i.e. two terms that reduce to 1) while all

of the cases **A3**, **A4** and **A5** ensure all possibilities of lengthening the substring by terms from the adjacent pair of angle bracket (and all symbols between).

We claim:

Theorem 2. *The algorithm computes $M'(i,j)$ correctly.*

Proof. We will show at first that, if the algorithm computes $M'(i,j)=1$, then $M'(i,j)=1$ according to the definition of M. We will prove it by induction on the length of the string $W^x_{(i,j)}$.

For strings of length one, $M'(i,j)=1$ only in case when $W^x_i = p^{(m)}$ and $p^{(m)} \in Reducible$. $W^x_{[i,i]}$ is then of the form **(M1)**, since $W^x_{[i,i]} \in T^+$ and the string $W^x_{[i,i]}$ reduces to 1. Hence, $M'(i,i)=1$ according to the definition of M.

Consider now the strings of length two. The algorithm computes $M'(i,i+1)=1$ only in case when $W^x_i = p^{(m)}$ and $W^x_{i+1} = q^{(n+1)}$ and $(p^{(m)}, q^{(n+1)}) \in Pairs$. $W^x_{[i,j]}$ is then of the form **(M1)**, since $W^x_{[i,i+1]} \in T^+$ and the string $W^x_{[i,i+1]}$ reduces to 1. Hence, $M'(i,i+1)=1$ according to the definition of M.

Now let us consider the recursive cases when the algorithm computes $M'(i,j)=1$. We present only some cases different from those in the proof for pregroup grammars, given in [8] (see [9] for the full proof).

Case A4d. $W^x_{[i,j]}$ is of the form **(A4d)**. $W^x_{[i,j]} = \underbrace{p^{(m)} * \ldots *}_{i,\ no\ \langle,\rangle} \underbrace{\rangle\langle \ldots \rangle}_{k,\ j,\ M'(k,j)=1}$,

where $p^{(m)} \in Reducible$ and k is the position of the first left angle bracket in the string $W^x_{[i,j]}$. $W^x_{[k,j]}$ is shorter than $W^x_{[i,j]}$. Hence, by the induction hypothesis, $M'(k,j)=1$ according to the definition of M. The string $W^x_{[k,j]}$ must then be of the form:

- **(M2a)**. Then $W^x_{[k,j]} = \langle \ldots \rangle \ldots \langle \mathbf{V} * Z$, where Z is the string of terms, $\mathbf{V}$ contains no angle brackets and there are precisely g ($g \geqslant 0$) pairs of matching angle brackets, for the h-th of them there is the substring $*X_h*$ in between them, such that $X_h \in T^+$ and $X_1 \ldots X_g Z$ reduces to 1. Let $Y = p^{(m)}$. Then $YX_1 \ldots X_g Z$ also reduces to 1, and the string $W^x_{[i,j]}$ is therefore of the form **(M3)**. Then $M'(i,j)=1$ in accordance with the definition of M.
- **(M4)**. Then $W^x_{[k,j]} = \langle \ldots \rangle \ldots \langle \ldots \rangle$ and there are precisely g ($g > 0$) pairs of matching angle brackets, for the h-th of them there is the substring $*X_h*$ in between them, such that $X_h \in T^+$ and $X_1 \ldots X_g$ reduces to 1. Let $Y = p^{(m)}$. Then $YX_1 \ldots X_g$ also reduces to 1, and the string $W^x_{[i,j]}$ is therefore of the form **(M2b)**. Then $M'(i,j)=1$ in accordance with the definition of M.

Case A6b. $W^x_{[i,j]}$ is of the form $W^x_{[i,j]} = \underbrace{\ldots p^{(m)}}_{i,\ j-1,\ M'(i,j-1)=1} q^{(n)}$, j

where $q^{(n)} \in Reducible$. $W^x_{[i,j-1]}$ is shorter than $W^x_{[i,j]}$. Hence, by the induction hypothesis, $M'(i,j-1)=1$ according to the definition of M. The string $W^x_{[i,j-1]}$ can therefore be of the form:

- **(M1)**. Then $W^x_{[i,j-1]} \in T^+$ and $W^x_{[i,j-1]}$ reduces to 1. Then $W^x_{[i,j]}$ is also of the form **(M4)** and it reduces to 1. Thus, $M'(i,j) = 1$ in accordance with the definition of M..
- **(M2a)**. Then $W^x_{[i,j-1]} = \langle ... \rangle ... \langle \mathbf{V} * Z$, where Z is the string of terms, $\mathbf{V}$ contains no angle brackets and there are precisely g ($g \geqslant 0$) pairs of matching angle brackets, for the h-th of them there is the substring $*X_h*$ in between them, such that $X_h \in T^+$ and $X_1 ... X_g Z$ reduces to 1. Let $Z' = Zq^{(n)}$. Then $X_1 ... X_g Z'$ also reduces to 1, and the string $W^x_{[i,j]}$ is therefore of the form **(M2a)**. Then $M'(i,j) = 1$ in accordance with the definition of M.
- **(M3)**. Then $W^x_{[i,j-1]} = Y * \mathbf{U} \rangle ... \langle ... \rangle \mathbf{V} * Z$, where Y, Z are the strings of terms, $\mathbf{U}, \mathbf{V}$ contain no angle brackets and there are precisely g ($g \geqslant 0$) pairs of matching angle brackets, for the h-th of them there is the substring $*X_h*$ in between them, such that $X_h \in T^+$ and $YX_1 ... X_g Z$ reduces to 1. Let $Z' = Zq^{(n)}$. Then $YX_1 ... X_g Z'$ reduces to 1, and the string $W^x_{[i,j]}$ is therefore also of the form **(M3)**. Then $M'(i,j) = 1$ in accordance with the definition of M.

We will prove that the algorithm correctly finds all substrings for which the function $M'(i,j) = 1$ by induction on the length of the substring of W^x. Note that there are no such substrings that contain asterisks but no angle brackets.

The only strings of length one for which $M'(i,j) = 1$ is of the form $p^{(m)}$, where $p^{(m)} \in T$ and $p^{(m)} \in \mathit{Reducible}$ and the algorithm finds them correctly (the string is of the form **M1**). The only strings of length two, for which $M'(i,j) = 1$, are of the form $p^{(m)}q^{(n+1)}$, where $(p^{(m)}, q^{(n+1)}) \in \mathit{Pairs}$ (that is of the form **(M1)**), and the algorithm finds them correctly.

Let us now consider the substrings of the length $l > 2$ such that for all $l' < l$ the algorithm finds the substrings of the length l' correctly (we present some chosen cases).

Case **3.** $W^x_{[i,j]}$ is of the form **(M3)**. $W^x_i W^x_j$ takes part in the reduction to 1 in $W^x_{[i,j]}$.

First, let us assume $W^x_i W^x_{i'}$ reduce to 1 where $i' \neq j$. Then $W^x_{i'+1}$ can be:

- term. Then $W^x_{[i,i']}$ and $W^x_{[i'+1,j]}$ are of the form **(M1)** or **(M3)**, they are shorter and both $M'(i,i') = 1$ and $M'(i'+1,j) = 1$. Hence, by the induction hypothesis, these strings are found by the algorithm correctly, so $M'(i,j) = 1$ (case (**A1a**)).
- asterisk. Then $W^x_{[i,i']}$ is of the form **(M1)** or **(M3)**, it is shorter and $M'(i,i') = 1$. Hence, by the induction hypothesis, the string is found by the algorithm correctly. Let k be the position of the first right angle bracket following i'. $W^x_{[i,k]}$ is shorter than $W^x_{[i,j]}$ and it is of the form **(M2b)**. So, by the induction hypothesis $M'(i,k) = 1$ (case (**A3b**)). Similarly, the substring $W^x_{[k+1,j]}$ is of the form **(M2a)**, it is shorter than $W^x_{[i,j]}$. So, by the induction hypothesis $M'(k+1,j) = 1$ (case (**A3a**)). Hence, $M'(i,j) = 1$, by the induction hypothesis (case (**A1b**)).

Otherwise, let us assume $W_i^x \in \mathit{Reducible}$. Then W_{i+1}^x can be:

- term. Then $W_{[i+1,j]}^x$ is of the form **(M3)**, it is shorter, so by the induction hypothesis $M'(i+1,j)=1$. Moreover, $W_{[i,i]}^x$ is of the form **M1**, it is shorter and $M'(i,i)=1$. Hence, by the induction hypothesis, these strings are found by the algorithm correctly. Then $M'(i,j)=1$ (case (**A1a**)).
- asterisk. Let k be the position of the first right angle bracket following i. $W_{[i,k]}^x$ is shorter than $W_{[i,j]}^x$ and it is of the form **(M2b)**. So, by the induction hypothesis $M'(i,k)=1$. Similarly, the substring $W_{[k+1,j]}^x$ is of the form **(M2a)**, it is shorter than $W_{[i,j]}^x$. So, by the induction hypothesis $M'(k+1,j)=1$. Hence, $M'(i,j)=1$, by the induction hypothesis (case (**A1b**)).

Let us assume $i'=j$ so $W_i^x W_j^x$ reduces to 1. There are the following cases.

- W_{i+1}^x, W_{j-1}^x are terms. Then the substring $W_{[i+1,j-1]}^x$ is of the form **(M3)**. It is shorter than $W_{[i,j]}^x$, therefore $M'(i+1,j-1)=1$, as by the induction hypothesis, that string is found by the algorithm correctly. Then $M'(i,j)=1$ (case (**A2**)).
- $W_{i+1}^x = *, W_{j-1}^x \in T$. So $W_{[i,j]}^x$ is of the form $p^{(m)} * ... \rangle ... \langle ... q^{(n+1)}$. Let i' be the position of the first left angle bracket in $W_{[i,j]}^x$. There exists $i' < k \leqslant j-1$ such that $W_k^x W_{j-1}^x$ reduces to, or if $k=j-1$, then $W_{j-1}^x \in \mathit{Reducible}$. Hence, $W_{[i',j-1]}^x$ is of the form **(M2a)**. It is shorter than $W_{[i,j]}^x$, therefore $M'(i',j-1)=1$, as by the induction hypothesis, that string is found by the algorithm correctly. Then $M'(i,j)=1$ (case (**A4a**)).
- $W_{i+1}^x \in T, W_{j-1}^x = *$. It is proved symmetrically to the case $W_{i+1}^x = *, W_{j-1}^x \in T$.
- $W_{i+1}^x = *, W_{j-1}^x = *$. Then the string $W_{[i,j]}^x$ can be of the form $p^{(m)} * \mathbf{U} \rangle \langle \mathbf{V} * q^{(n+1)}$ and then $M'(i,j)=1$ by the initial case of the description of the algorithm. If $W_{[i,j]}^x$ contains more brackets, then there is a string $W_{[k,k']}^x$, where k is the index of the first left angle bracket in the string $W_{[i,j]}^x$ and k' is the index of the last right angle bracket in the string $W_{[i,j]}^x$. $W_{[k,k']}^x$ is of the form **(M4)**. It is shorter than $W_{[i,j]}^x$, therefore, by the induction hypothesis, $M'(k,k')=1$. So, $M'(i,j)=1$ by (**A4c**).

Obviously, the algorithm is not yet a real parsing algorithm since it answers only the question whether there exists any reduction to the designated type. However it can easily be modified to find such reductions, still in polynomial time. It can be done exactly as in the algorithm for pregroup grammars.

Each obtained reduction is described by the set of links involved in the reduction. If we want to obtain only one reduction, the complexity of the algorithm does not increase. The set of links $L(i,j)$ represents a reduction of some term to 1. Links are denoted by pairs of integers (k,l) such that $i \leqslant k < l \leqslant j$. We find the set of links by backtracking the indices of the function $M'(i,j)=1$, obviously starting with $M'(1,|W^x|)$. We also define an auxiliary function $\mathit{Prev}(i,j)$ to help us follow the backtracking (as the value of the function $M'(i,j)$ does

not say how it was obtained). The value of the function $Prev(i,j)$ is a sequence of three pairs $((l_1,l_2),(m1_1,m1_2),(m2_1,m2_2))$, where l_1,l_2 are indices of the link, $m1_1,m1_2,m2_1,m2_2$ are indices of function M on which the computation $M'(i,j)=1$ is based. If any of the values is not used, it is set to 0. Every time when the algorithm computes the value of the function $M'(i,j)=1$ we set the value of the function $Prev(i,j)$ in the following way. If the computation was executed by one of the cases:

- any initial case or (**A0**), then $Prev(i,j)=((i,j),(0,0),(0,0))$,
- (**A2**), (**A4a**), (**A4b**), (**A4c**), then $Prev(i,j)=((i,j),(k,l),(0,0))$, where (k,l) is the pair of indices for which the value of the function M was 1 in the current computation (that is e.g. in (**A2**) a pair $(k,l)=(i+1,j-1)$),
- (**A3a**), (**A3b**), (**A5**), then $Prev(i,j)=((0,0),(k,l),(0,0))$, where (k,l) is the pair of indices for which the value of the function M was 1 in the current computation,
- (**A1a**), (**A1a'**), (**A1b**), then $Prev(i,j)=((0,0),(i,k),(k+1,j))$,
- (**A4d**), then $Prev(i,j)=((i,i),(k,j),(0,0))$, where (k,j) is the pair of indices for which the value of the function M was 1 in the current computation,
- (**A4e**), then $Prev(i,j)=((j,j),(i,k),(0,0))$, where (i,k) is the pair of indices for which the value of the function M was 1 in the current computation,
- (**A6a**), then $Prev(i,j)=((i,i),(i+1,j),(0,0))$,
- (**A6b**), then $Prev(i,j)=((j,j),(i,j-1),(0,0))$.

Obviously, one can choose whether the algorithm should remember the first computed reduction or the last computed reduction. In the first case if $Prev(i,j)\neq((0,0),(0,0),(0,0))$, then it cannot be modified. In the latter, $Prev(i,j)$ is updated every time when the algorithms computes $M'(i,j)=1$. When the computation of the functions M and $Prev$ is finished, we easily compute the set $L(1,|W^x|)$. The definition of the function $L(i,j)$ is as follows:

- if $Prev(i,j)=((i,j),(0,0),(0,0))$, where $0<i<j$, then $L(i,j)=\{(i,j)\}$,
- if $Prev(i,j)=((i,j),(k,l),(0,0))$, where $0<i\leqslant k<l\leqslant j$, then $L(i,j)=L(k,l)\cup\{(i,j)\}$,
- if $Prev(i,j)=((0,0),(k,l),(0,0))$, where $0<i\leqslant k<l\leqslant j$, then $L(i,j)=L(k,l)$,
- if $Prev(i,j)=((0,0),(i,k),(k+1,j))$, where $0<i<k<j$, then $L(i,j)=L(i,k)\cup L(k+1,j)$.

The algorithm is polynomial and works in time proportional to n^3, where n is the length of the string W_x, assuming the set *Pairs* and *Reducible* are determined. The procedures for computing the set *Pairs* and the set *Reducible* are polynomial.

References

1. Buszkowski, W.: Lambek Grammars Based on Pregroups. In: de Groote, P., Morrill, G., Retoré, C. (eds.) LACL 2001. LNCS (LNAI), vol. 2099, pp. 95–109. Springer, Heidelberg (2001)
2. Buszkowski, W., Moroz, K.: Pregroup grammars and context-free grammars. In: Casadio, C., Lambek, J. (eds.) Computational Algebraic Approaches to Natural Language, Polimetrica, pp. 1–21 (2008)
3. Buszkowski, W., Lin, Z.: Pregroup Grammars with Letter Promotions. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.) LATA 2010. LNCS, vol. 6031, pp. 130–141. Springer, Heidelberg (2010)
4. Buszkowski, W., Lin, Z., Moroz, K.: Pregroup Grammars with Letter Promotions: Complexity and Context-Freeness. Journal of Computer and System Sciences 78(6), 1899–1909 (2012)
5. Lambek, J.: Type Grammars Revisited. In: Lecomte, A., Perrier, G., Lamarche, F. (eds.) LACL 1997. LNCS (LNAI), vol. 1582, pp. 1–27. Springer, Heidelberg (1999)
6. Lambek, J.: From Word to Sentence: a computational algebraic approach to grammar. Polimetrica (2008)
7. Mater, A.H., Fix, J.D.: Finite Presentations of Pregroups and the Identity Problem. In: Proceedings of FG-MoL 2005, pp. 63–72. CSLI (electronic) (2005)
8. Moroz, K.: A Savateev-style parsing algorithm for pregroup grammars. In: de Groote, P., Egg, M., Kallmeyer, L. (eds.) FG 2009. LNCS, vol. 5591, pp. 133–149. Springer, Heidelberg (2011)
9. Moroz, K.: Algorithmic questions for pregroup grammars, PhD thesis, Adam Mickiewicz University, Poznań (2010)
10. Pentus, M.: Lambek calculus is NP-complete. Theoretical Computer Science 357, 186–201 (2006)
11. Savateev, Y.: Unidirectional Lambek Grammars in Polynomial Time. Theory of Computing Systems 46, 662–672 (2010)