A Savateev-style parsing algorithm for pregroup grammars

Katarzyna Moroz

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

Abstract. We present a new cubic parsing algorithm for ambiguous pregroup grammars. It modifies the recognition algorithm of Savateev [10] for categorial grammars based on $\mathbf{L}^{\backslash}$. We show the correctness of the algorithm and give some examples. We compare our algorithm with the algorithm of Oehrle [8] for pregroup grammars and the algorithm of Savateev [10].

1 Introduction and preliminaries

Pregroups were introduced by Lambek [5] as an algebraic tool for the syntactical analysis of sentences. The idea is that words have syntactical properties that can be described by a finite set of pregroup types. The problem of parsing pregroup grammars has been a subject of study in recent years (see the references).

Oehrle [8] proposed a recognising algorithm based on graph theoretic representations of lexical types and their complexes. The algorithm can be amended to become a parsing algorithm. Béchet [1] presents a recognition algorithm based on a restricted form of partial composition called functional composition. However, the algorithm employs a construction of a context-free grammar whose size is exponential with respect to the given pregroup grammar. A polynomial construction is given in [4]. Another approach to parsing pregroup grammars was proposed by Preller [9]. It yields a linear algorithm for unambiguous pregroup grammars with some restrictions imposed on the type lexicon.

Savateev [10] provides a characterisation of derivability in $\mathbf{L}^{\backslash}$ (the Lambek calculus with one residual $\backslash$ and without product), which is, essentially, a translation of $\mathbf{L}^{\backslash}$ in the right pregroup calculus with some additional constraint. Then, he presents a recognition algorithm for $\mathbf{L}^{\backslash}$-grammars, which uses this translation. Our algorithm modifies the latter. There are essential differences: (1) we regard poset rules and both left and right adjoints, while Savateev considers right adjoints only and no poset rules, (2) we drop the extra constraint needed for the translation of $\mathbf{L}^{\backslash}$. (3) Savateev's algorithm depends on some special properties of $\mathbf{L}^{\backslash}$-types, which is not the case for our algorithm. (Actually, Savateev does not explicitly refer to pregroups, but his translation can be interpreted in terms of pregroups.) Further we present briefly the algorithm of Savateev. We provide a complete proof of the correctness of our algorithm. At the end, we show how the

algorithm can be extended to return the reduction, thus becoming a full parsing algorithm. We also present some examples.

A *pregroup* is a structure $M = (M, \leqslant, \cdot, l, r, 1)$ such that $(M, \leqslant, \cdot, 1)$ is a partially ordered monoid, and l, r are unary operations on M, satisfying the adjoint laws:

$$(Al)\ a^l a \leqslant 1 \leqslant aa^l, \quad (Ar)\ aa^r \leqslant 1 \leqslant a^r a,$$

for all $a \in M$. Pregroups are a generalization of partially ordered groups. a^l (resp. a^r) is called the *left* (resp. *right*) *adjoint of a*.

A *right pregroup* is a partially ordered monoid in which for each element a there exists a *right adjoint* a^r satisfying the law (Ar).

It is worth noticing that the following laws are easily derivable from (Al), (Ar):

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

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

In a given pregroup we can define $a\backslash b = a^r b, a/b = ab^l$, and prove the *residuation law*:

$$ab \leqslant c \text{ iff } b \leqslant a\backslash c \text{ iff } a \leqslant c/b,$$

for all elements $a, b, c \in M$. Consequently, pregroups are a special class of residuated monoids, i.e. models of the Lambek calculus **L1** [2, 3]

One defines *iterated adjoints*: $a^{(n)} = a^{rr\ldots r}$ and $a^{(-n)} = a^{ll\ldots l}$, where n is a non-negative integer and a any element of $\mathbf{M}$. The following laws are provable:

$$a^{(n)} a^{(n+1)} \leqslant 1 \leqslant a^{(n+1)} a^{(n)},$$

for any integer n.

Lambek's approach to syntactic analysis is based on the notion of a *free pregroup*, generated by a finite poset $(P, \leqslant)$.

Elements of P are called *atoms* and denoted by letters p, q, r, s.

Expressions of the form $p^{(n)}$, for any $p \in P$ and any integer n are *simple terms*.

A *term* is a finite sequence (string) of simple terms. Terms are also called *types*. They are denoted by $\mathbb{A}, \mathbb{B}, \mathbb{C}$. If $p \leqslant q$ in P, then we write $p^{(n)} \leqslant q^{(n)}$, if n is even, and $q^{(n)} \leqslant p^{(n)}$, 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) $\mathbb{A}, p^{(n)}, p^{(n+1)}, \mathbb{B} \Rightarrow \mathbb{A}, \mathbb{B}$,
(EXP) $\mathbb{A}, \mathbb{B} \Rightarrow \mathbb{A}, p^{(n+1)}, p^{(n)}, \mathbb{B}$,
(IND) $\mathbb{A}, p^{(n)}, \mathbb{B} \Rightarrow \mathbb{A}, q^{(n)}, \mathbb{B}$, if $p^{(n)} \leqslant q^{(n)}$.

(CON), (EXP), (IND) are called Contraction, Expansion and Induced Step, respectively. They can be treated as rules of a term rewriting system. $\mathbb{A} \Rightarrow \mathbb{B}$ is true iff $\mathbb{A}$ can be rewritten into $\mathbb{B}$ 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**).

We can also define a *Generalized Contraction* that combines Contraction with Inducesd Step:

(GCON) $\mathbb{A}p^{(n)}q^{(n+1)}\mathbb{B} \Rightarrow \mathbb{A}\mathbb{B}$ if $p \leqslant q$ and n is even, or $q \leqslant p$ and n is odd.

Lambek [5] shows an important property called *the switching lemma*:

Lemma 1. *If* $\mathbb{A} \Rightarrow \mathbb{B}$ *then there exists* $\mathbb{C}$ *such that* $\mathbb{A} \Rightarrow \mathbb{C}$ *without (EXP) and* $\mathbb{C} \Rightarrow \mathbb{B}$ *without (CON).*

One can easily see as a consequence of this theorem that if $\mathbb{A} \Rightarrow t$, where t is a simple term or ε, then $\mathbb{A}$ 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** [2, 3].

A *pregroup grammar* is a quintuple $G = (\Sigma, P, \leqslant, s, I)$ such that Σ is a nonempty, finite alphabet, $(P, \leqslant)$ 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 $\mathbb{A}$ such that $(a, \mathbb{A}) \in I$. Let $x \in \Sigma^+$, $x = a_1...a_n$ $(a_i \in \Sigma)$. One says that the grammar G assigns type $\mathbb{B}$ to x, if there exist types $\mathbb{A}_i \in I(a_i)$, $i = 1, ..., n$, such that $\mathbb{A}_1, ..., \mathbb{A}_n \Rightarrow \mathbb{B}$ in **CBL**; we write $x \rightarrow_G \mathbb{B}$. The set $L(G) = \{x \in \Sigma^+ : x \rightarrow_G s\}$ is called the language of G.

It is useful to present reductions by the set of *links*. A link indicates a pair of simple terms that reduce to 1. If a whole string reduces to 1 then any simple term appearing in the string is connected by a link with another simple term. Each term can be a member of only one link and links cannot cross. We can see the links in the following example [6]:

$$\underbrace{I}_{\pi_1)}\ \underbrace{(\pi^r}\ \underbrace{s_1}\ \underbrace{j^l)}\ \underbrace{(i}\ \underbrace{o^l)}\ \underbrace{(o)}\ \underbrace{s^r}$$

will *meet* *him.*

where there are the following basic types:

π_1 - first person singular subject
π - subject
i - infinitive of intransitive verb
j - infinitive of any complete verb phrase
o - direct object
s_1 - declarative sentence in present tense
s - sentence

and the following partial order:

$\pi_1 \leqslant \pi, \quad i \leqslant j, \quad s_1 \leqslant s$

2 Parsing algorithm

Our goal is to check whether a given string x is a member of the language generated by the pregroup grammar G: $x \in L(G)$. If this is the case, we want to obtain the appropriate derivation. We should notice that the condition $A_1...A_n \Rightarrow s$ in **CBL** is equivalent to the condition $\mathbb{A}_1...\mathbb{A}_n s^{(1)} \Rightarrow \varepsilon$. In what follows, we write 1 for ε.

We define the algorithm in a style proposed by Savateev [10] 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.

2.1 Definition of the function M

We fix $G = (\Sigma, P, \leqslant, s, I)$ and $x \in \Sigma^+, \quad x = a_1...a_n$. We use special symbols $*, <, >$. Let us denote:
- $\mathcal{Z}$ - the set of integers
- $T = \{p^{(n)} : p \in P, n \in \mathcal{Z}\}$ - the set of simple terms
- $\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{D}, ..$ - elements of T^*
- $k^a = |I(a)|$
- $\mathbb{A}^a_j$ - the j-th possible assignment of type to a, $1 \leqslant j \leqslant k^a$ (hence $I(a) = \{\mathbb{A}^a_1, ...\mathbb{A}^a_{k^a}\}$)
- $Q^a = \langle *\mathbb{A}^a_1 * \mathbb{A}^a_2 * ... * \mathbb{A}^a_{k^a} *\rangle$
- $W^x \in (T \cup \{*, \langle, \rangle\})^* = Q^{a_1}...Q^{a_n}\langle * s^{(1)}$
- $W^x_i, 1 \leqslant i \leqslant |W^x|$ - the i-th symbol of the string W^x
- $W^x_{[i,j]}, 1 \leqslant i < j \leqslant |W^x|$ - the substring of W^x: $W^x_i W^x_{i+1}...W^x_j$

Let $M(i,j), 1 \leqslant i < j \leqslant |W^x|$ be a function such that $M(i,j) = 1$ iff one of the following conditions holds:

- **M1.** $W^x_{[i,j]} \in T^+$ and it reduces to 1.
- **M2a.** $W^x_{[i,j]}$ is of the form $\langle ...\rangle ...\langle V * \mathbb{C}$, where:
  - $\mathbb{C} \in T^+$
  - V contains no angle brackets
  - in $W^x_{[i,j]}$ there are g ($g \geqslant 0$) pairs of matched angle brackets; for the h-th pair of them there is a substring of the form $*\mathbb{D}_h*$ in between them such that $\mathbb{D}_h \in T^+$ and the string $\mathbb{D}_1...\mathbb{D}_g\mathbb{C}$ reduces to 1
- **M2b.** $W^x_{[i,j]}$ is of the form $\mathbb{C} * U\rangle ...\langle ...\rangle$, where:
  - $\mathbb{C} \in T^+$
  - U contains no angle brackets
  - in $W^x_{[i,j]}$ there are g ($g \geqslant 0$) pairs of matched angle brackets; for the h-th pair of them there is a substring of the form $*\mathbb{D}_h*$ in between them such that $\mathbb{D}_h \in T^+$ and the string $\mathbb{C}\mathbb{D}_1...\mathbb{D}_g$ reduces to 1
- **M3.** $W^x_{[i,j]}$ is of the form $\mathbb{D} * U\rangle ...\langle V * \mathbb{C}$, where:
  - $\mathbb{C}, \mathbb{D} \in T^+$
  - U,V contains no angle brackets
  - in $W^x_{[i,j]}$ there are g ($g \geq 0$) pairs of matched angle brackets; for the h-th

pair of them there is a substring of the form $*\mathbb{E}_h*$ in between them such that $\mathbb{E}_h \in T^+$ and the string $\mathbb{D}\mathbb{E}_1...\mathbb{E}_g\mathbb{C}$ reduces to 1

- **M4.** $W^x_{[i,j]}$ is of the form $\langle...\rangle...\langle...\rangle$, where:
  - in $W^x_{[i,j]}$ there are g ($g \geq 1$) pairs of matched angle brackets; for the h-th pair of them there is a substring of the form $*\mathbb{E}_h*$ in between them such that $\mathbb{E}_h \in T^+$ and the string $\mathbb{E}_1...\mathbb{E}_g$ reduces to 1

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

2.2 The recognition algorithm

We compute $M(i,j)$ dynamically. $M(i,i+1) = 1$ only if $W^x_i = p^{(m)}$, $W^x_{i+1} = q^{(m+1)}$ and the following condition holds

(GC) $p, q \in P, m \in \mathcal{Z}$ and $p \leqslant q$ in P if m is even or $q \leqslant p$ in P if m is odd.

The other initial case is when $W^x_{[i,j]}$ is of the form $p^{(m)} * U\rangle\langle V * q^{(m+1)}$, the condition (GC) holds and strings U, V contain no angle brackets. Then we put $M(i,j) = 1$.

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

- **A1a.** $W^x_i, W^x_j \in T$. If there exists k such that $i < k < j-1$ and $W^x_k \in T$, $W^x_{(k+1)} \in T$ and $M(i,k) = 1$ and $M(k+1,j) = 1$, then we put $M(i,j) = 1$.
- **A1b.** $W^x_i, W^x_j \in T$. If there exists k such that $i < k < j-1$ and $W^x_k = \rangle$, $W^x_{(k+1)} = \langle$ and $M(i,k) = 1$ and $M(k+1,j) = 1$, then we put $M(i,j) = 1$.
- **A2.** $W^x_i = p^{(m)}$, $W^x_j = q^{(m+1)}$ and the condition (GC) holds.
  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 $i < k < j$ and $W^x_k = *, W^x_{[i+1,k]}$ 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 $i < k < j$ and $W^x_k = *, W^x_{[k,j-1]}$ 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 $p^{(m)} * ...\rangle...\langle...q^{(m+1)}$ and the condition (GC) holds. 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 $p^{(m)}...\rangle...\langle...*q^{(m+1)}$ and the condition (GC) holds. 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 $p^{(m)} * ...\rangle...\langle... * q^{(m+1)}$ where the string "..." in between the angle brackets is not empty and the condition (GC) holds. 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 $M(i,j) = 1$

– **A5.** $W^x_{[i,j]}$ is of the form $\langle ...\rangle ...\langle ...\rangle$. If $M(k,k') = 1$, where W^x_k is a simple term in between the first pair of angle brackets, $W^x_{k'}$ is a simple 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 $M(i,j) = 1$.

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

Theorem 1. *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.

For strings of length 2, the algorithm computes $M(i,j) = 1$ only in case when $W^x_i = p^{(m)}$ and $W^x_{i+1} = q^{(m+1)}$ and the condition (GC) holds. $W^x_{[i,j]}$ is then of the form **(M1)** since $W^x_{[i,j]} \in T^+$ and the string $W^x_{[i,j]}$ reduces to 1. So $M(i,j) = 1$ according to the definition of **M**.

The other initial case when the algorithm computes $M(i,j) = 1$ is when $W^x_{[i,j]}$ is of the form $p^{(m)} * U\rangle\langle V * q^{(m+1)}$, the condition (GC) holds and the strings U and V contain no angle brackets. $W^x_{[i,j]}$ is then of the form **(M3)** since we can assume $\mathbb{D} = p^{(m)}$ and $\mathbb{C} = q^{(m+1)}$ and $g = 0$. So $\mathbb{DC}$ reduces to 1. Hence $M(i,j) = 1$ according to the definition of **M**.

Let us consider now the recursive cases when the algorithm computes $M(i,j) = 1$ (all cases of the description of the algorithm).

A1a. $W^x_i, W^x_j \in T$ and there exists k such that $i < k < j-1$ and $W^x_k \in T$, $W^x_{(k+1)} \in T$, $M(i,k) = 1$ and $M(k+1,j) = 1$. Then the substrings $W^x_{[i,k]}$ and $W^x_{[k+1,j]}$ are shorter than $W^x_{[i,j]}$ therefore by the induction hypothesis both $M(i,k)$ and $M(k+1,j)$ are equal to 1 according to the definition of **M**. $W^x_{[i,k]}$ and $W^x_{[k+1,j]}$ can be of the form **(M1)** or **(M3)**. There are the following cases.

- If both substrings $W^x_{[i,k]}$ and $W^x_{[k+1,j]}$ are of the form **(M1)** then $W^x_{[i,j]}$ also consists of simple terms and reduces to 1, as both $W^x_{[i,k]}$ and $W^x_{[k+1,j]}$ reduce to 1. $W^x_{[i,j]}$ is therefore of the form **(M1)**. Hence $M(i,j) = 1$ in accordance with the definition of **M**.
- $W^x_{[i,k]}$ is of the form **(M1)**. Therefore it consists of simple terms and reduces to 1 and $W^x_{[k+1,j]}$ is of the form **(M3)**. Then $W^x_{[i,j]}$ is also of the form **(M3)**. Hence $M(i,j) = 1$ according to the definition of **M**.
- $W^x_{[i,k]}$ is of the form **(M3)** and $W^x_{[k+1,j]}$ is of the form **(M1)**. So it consists of simple terms and reduces to 1. Then $W^x_{[i,j]}$ is also of the form **(M3)**. Hence $M(i,j) = 1$ according to the definition of **M**.
- $W^x_{[i,k]}$ and $W^x_{[k+1,j]}$ are of the form **(M3)**. Hence the whole string $W^x_{[i,j]}$ is also of the form **(M3)**. Then $M(i,j) = 1$ in accordance with the definition of **M**.

A1b. $W^x_i, W^x_j \in T$ and there exists k such that $i < k < j-1$ and W^x_k is a right angle bracket, $W^x_{(k+1)}$ is a left angle bracket, $M(i,k) = 1$ and $M(k+1,j) = 1$.

Then the substrings $W^x_{[i,k]}$ and $W^x_{[k+1,j]}$ are shorter than $W^x_{[i,j]}$ therefore by the induction hypothesis both $M(i,k)$ and $M(k+1,j)$ are equal to 1 according to the definition of **M**. $W^x_{[i,k]}$ is of the form **(M2b)** so there is a string $\mathbb{C}\mathbb{D}_1...\mathbb{D}_{g_1}$ that reduces to 1 (such that the first simple term of $\mathbb{C}$ is W^x_i, $\mathbb{D}_h \in T^+$ is a substring in between the h-th pair of matching angle brackets in the string $W^x_{[i,k]}$). The substring $W^x_{[k+1,j]}$ is of the form **(M2a)** so there is a string $\mathbb{E}_1...\mathbb{E}_{g_2}\mathbb{F}$ that reduces to 1 (such that the last simple term of $\mathbb{F}$ is W^x_j, $\mathbb{E}_h \in T^+$ is a substring in between the h-th pair of matching angle brackets in the string $W^x_{[k+1,j]}$). So the whole string $W^x_{[i,j]}$ is of the form **(M3)** (with the string $\mathbb{C}\mathbb{D}_1...\mathbb{D}_{g_1}\mathbb{E}_1...\mathbb{E}_{g_2}\mathbb{F}$ reducing to 1).

A2. $W^x_i = p^{(m)}$, $W^x_j = q^{(m+1)}$, the condition (GC) holds and $M(i+1,j-1) = 1$. $M(i+1,j-1) = 1$ is computed according to the definition of **M** by the induction hypothesis, as $W^x_{[i+1,j-1]}$ is shorter than $W^x_{[i,j]}$. Then the substring $W^x_{[i+1,j-1]}$ can be:

- of the form **(M1)**, then the whole string $W^x_{[i,j]}$ consists of simple terms and reduces to 1, so it is also of the form **(M1)**. Hence $M(i,j) = 1$ according to the definition of **M**.
- of the form **(M3)**, that is $\mathbb{D} * U\rangle...\langle V * \mathbb{C}$, where U and V contain no angle brackets and the string $\mathbb{D}\mathbb{E}_1...\mathbb{E}_g\mathbb{C}$ reduces to 1. We can assume $\mathbb{D}' = p^{(m)}\mathbb{D}$ and $\mathbb{C}' = \mathbb{C}q^{(m+1)}$. Then $W^x_{[i,j]} = \mathbb{D}' * U\rangle...\langle V * \mathbb{C}'$ and the string $\mathbb{D}'\mathbb{E}_1...\mathbb{E}_g\mathbb{C}'$ reduces to 1. Hence $M(i,j) = 1$ according to the definition of **M**.

We should notice that the string $W^x_{[i+1,j-1]}$ cannot be of the form **(M2a)**, **(M2b)** or **(M4)** since a simple term can be followed (preceded) only by another simple term or an asterisk.

A3a. $W^x_{[i,j]}$ is of the form $\langle...\rangle...\langle...p^{(m)}$, $p \in P, m \in \mathcal{Z}$ and there exists k such that $i < k < j$ and $W^x_k = *$, the string $W^x_{[i+1,k]}$ contains no angle brackets and $M(k+1,j) = 1$. $W^x_{[k+1,j]}$ is shorter than $W^x_{[i,j]}$ so $M(k+1,j) = 1$ is computed according to the definition of **M** by the induction hypothesis. The string $W^x_{[k+1,j]}$ ends with a simple term, so it can be of the form **(M1)**, **(M2a)** or **(M3)**. But it cannot begin with a left angle bracket and it contains angle brackets, so it must be of the form **(M3)**. So there is a string $\mathbb{C}\mathbb{E}_1...\mathbb{E}_g\mathbb{D}$ reducing to 1. Hence $W^x_{[i,j]}$ is of the form **(M2a)** with the string $\mathbb{C}\mathbb{E}_1...\mathbb{E}_g\mathbb{D}$ reducing to 1. Therefore $M(i,j) = 1$ according to the definition of **M**.

A3b. $W^x_{[i,j]}$ is of the form $p^{(m)}...\rangle...\langle...\rangle$, where $p \in P, m \in \mathcal{Z}$ and there exists k such that $i < k < j$, $W^x_k = *, W^x_{[k,j-1]}$ contains no angle brackets and $M(i,k-1) = 1$. The string $W^x_{[i,k-1]}$ is shorter than $W^x_{[i,j]}$ so M(i,k-1)=1 is computed according to the definition of **M**, by the induction hypothesis. The string $W^x_{[i,k-1]}$ begins with a simple term, so it can be of the form **(M1)**, **(M2b)** or **(M3)**. But it cannot end with a right angle bracket and it contains angle brackets, so it must be of the form **(M3)**. So there is a string $\mathbb{C}\mathbb{E}_1...\mathbb{E}_g\mathbb{D}$ reducing to 1. Hence $W^x_{[i,j]}$ is of the form **(M2b)** with the string $\mathbb{C}\mathbb{E}_1...\mathbb{E}_g\mathbb{D}$ reducing to 1. Therefore $M(i,j) = 1$ according to the definition of **M**.

A4a. . $W^x_{[i,j]}$ is of the form $p^{(m)} * ...\rangle ...\langle ... q^{(m+1)}$, the condition (GC) holds and $M(k, j-1) = 1$, where k is the position of the first left angle bracket in the string $W^x_{[i,j]}$. $W^x_{[k,j-1]}$ is shorter than $W^x_{[i,j]}$. Hence, by the induction hypothesis, $M(k, j-1) = 1$ according to the definition of **M**. The string $W^x_{[k,j-1]}$ must be therefore of the form **(M2a)** since it is the only form starting with an angle bracket and ending with something different from a bracket. Hence $W^x_{[k,j-1]} = \langle ...\rangle ...\langle V * \mathbb{C}$, where $\mathbb{C}$ is the string of simple terms, V contains no angle brackets and there are g pairs of matched angle brackets, for the h-th of them there is the substring $*\mathbb{E}_h*$ in between them, such that $\mathbb{E}_h \in T^+$ and $\mathbb{E}_1 ... \mathbb{E}_g \mathbb{C}$ reduces to 1. Let $\mathbb{C}' = \mathbb{C}q^{(m+1)}$, $\mathbb{D} = p^{(m)}$. Then $\mathbb{D}\mathbb{E}_1 ... \mathbb{E}_g \mathbb{C}'$ 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**.

A4b. $W^x_{[i,j]}$ is of the form $p^{(m)} ...\rangle ...\langle ... * q^{(m+1)}$, the condition (GC) holds, and $M(i+1, k) = 1$, where k the position of the last right angle bracket in the string $W^x_{[i,j]}$. $W^x_{[i+1,k]}$ is shorter than $W^x_{[i,j]}$. Hence, by the induction hypothesis, $M(i+1, k) = 1$ according to the definition of **M**. The string $W^x_{[i+1,k]}$ must be therefore of the form **(M2b)** since it is the only form ending with an angle bracket but starting with something different from a bracket. Hence $W^x_{[i+1,k]} = \mathbb{D} * U\rangle ...\langle ...\rangle$, where $\mathbb{D}$ is the string of simple terms, U contains no angle brackets and there are g pairs of matched angle brackets, for the h-th of them there is the substring $*\mathbb{E}_h*$ in between them, such that $\mathbb{E}_h \in T^+$ and $\mathbb{D}\mathbb{E}_1 ... \mathbb{E}_g$ reduces to 1. Let $\mathbb{D}' = p^{(m)}\mathbb{D}$, $\mathbb{C} = q^{(m+1)}$. Then $\mathbb{D}'\mathbb{E}_1 ... \mathbb{E}_g \mathbb{C}$ 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**.

A4c. $W^x_{[i,j]}$ is of the form $p^{(m)} * ...\rangle ...\langle ... * q^{(m+1)}$, where the string "..." in between the angle brackets is not empty and the condition (GC) holds. $M(k, k') = 1$, where k is the position of the first left angle bracket in the string $W^x_{[i,j]}$ and k' the position of the last right angle bracket in the string $W^x_{[i,j]}$. $W^x_{[k,k']}$ is shorter than $W^x_{[i,j]}$. Hence, by the induction hypothesis, $M(k, k') = 1$ according to the definition of **M**. The string $W^x_{[k,k']}$ must be therefore of the form **(M4)**. Hence $W^x_{[k,k']} = \langle ...\rangle ...\langle ...\rangle$, where there are g pairs of matched angle brackets, for the h-th of them there is the substring $*\mathbb{E}_h*$ in between them, such that $\mathbb{E}_h \in T^+$ and $\mathbb{E}_1 ... \mathbb{E}_g$ reduces to 1. Let $\mathbb{D} = p^{(m)}$, $\mathbb{C} = q^{(m+1)}$, $U = W^x_{[i+2,k-1]}$ (if $k > i+3$ else $U = \varepsilon$) and $V = W^x_{[k'+1,j-2]}$ (if $k' < j-3$ else $V = \varepsilon$). Then $\mathbb{D}\mathbb{E}_1 ... \mathbb{E}_g \mathbb{C}$ reduces to 1, and the string $W^x_{[i,j]}$ is therefore of the form **(M3)**. Then $M(i, j) = 1$ according to the definition of **M**.

A5 $W^x_{[i,j]}$ is of the form $\langle ...\rangle ...\langle ...\rangle$, and $M(k, k') = 1$, where W^x_k is a simple term in between the first pair of angle brackets and $W^x_{k'}$ is a simple term in between last pair of angle brackets in the string $W^x_{[i,j]}$ and $W^x_{k-1} = *$ and $W^x_{k'+1} = *$. $W^x_{[k,k']}$ is shorter than $W^x_{[i,j]}$ hence $M(k, k') = 1$ according to the definition of **M**. $W^x_{[k,k']}$ must be then of the form **(M3)**, so the whole string $W^x_{[i,j]}$ is of the form **(M4)** and therefore $M(i, j) = 1$ in accordance with the definition of **M**.

We will prove that the algorithm finds correctly all substrings for which the function $M(i,j) = 1$ by induction on the length of the substring. Let us notice that there are no such substrings that contain asterisk but no angle brackets.

The only strings of length 2 for which $M(i,j) = 1$ are of the form $p^{(m)}q^{(m+1)}$ where the condition (GC) holds (that is of the form **(M1)**), and the algorithm finds them correctly.

Let us consider now the substrings of the length $l > 2$ such that for all $l' < l$ the algorithm finds the substrings of the length l' correctly. (all forms of the definition of **M**)

- 1. $W^x_{[i,j]}$ is of the form **(M1)** that is $W^x_{[i,j]} \in T^+$ and it reduces to 1. There are 2 possible cases:
  - $W^x_i W^x_j$ does not reduce to 1 in the whole reduction of the string $W^x_{[i,j]}$ to 1. Then $W^x_{[i,j]}$ can be divided into 2 substrings. Each of them reduces to 1, is shorter than $W^x_{[i,j]}$, so they are found by the algorithm correctly (by the induction hypothesis). Hence by (**A1a**), $M(i,j) = 1$.
  - $W^x_i W^x_j$ reduces to 1 in the whole reduction of $W^x_{[i,j]}$ to 1. Then $W^x_{[i+1,j-1]}$ is shorter substring reducing to 1. By the induction hypothesis the string is found by the algorithm correctly, so $M(i,j) = 1$ (case (**A2**)).
- 2a. $W^x_{[i,j]}$ is of the form **(M2a)**. If $M(i,j) = 1$ then there exists a substring of the form $*\mathbb{D}_1*$ in between the first pair of angle brackets such that $\mathbb{D}_1$ takes part in some reduction to 1 (if $g > 0$). Let i' be the position of the first simple term in $\mathbb{D}_1$. The substring $W^x_{[i',j]}$ is of the form **(M3)**, is shorter than $W^x_{[i,j]}$ and $M(i',j) = 1$. By the induction hypothesis, the string is found by the algorithm correctly, so $M(i,j) = 1$ (case (**A3a**)). If $g = 0$, then $\mathbb{C}$ reduces to 1 and it is of the form **(M1)** and shorter than $W^x_{[i,j]}$. So, by the induction hypothesis it is found by the algorithm correctly. Then $M(i,j) = 1$ by (**A3a**).
- 2b. $W^x_{[i,j]}$ is of the form **(M2b)**. If $M(i,j) = 1$ then there exists the substring of the form $*\mathbb{D}_g*$ (if $g > 0$) in between the last pair of angle brackets such that $\mathbb{D}_g$ takes part in some reduction to 1. Let j' be the position of the last simple term in $\mathbb{D}_g$. The substring $W^x_{[i,j']}$ is of the form **(M3)**, is shorter than $W^x_{[i,j]}$ and $M(i,j') = 1$. By the induction hypothesis the string is found by the algorithm correctly, so $M(i,j) = 1$ (case (**A3b**)). If $g = 0$ then $\mathbb{C}$ reduces to 1 and it is of the form **(M1)** and shorter than $W^x_{[i,j]}$, so by the induction hypothesis it is found by the algorithm correctly. Then $M(i,j) = 1$ by (**A3b**).
- 3. $W^x_{[i,j]}$ is of the form **(M3)**. $W^x_i W^x_j$ takes part in the reduction to 1 in $W^x_{[i,j]}$. Let us assume: $W^x_i W^x_{i'}$ reduce to 1 where $i' \neq j$. Then $W^x_{i'+1}$ can be:
  - simple 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)** 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**)).

Let us assume $i' = j$ so $W^x_i W^x_j$ reduces to 1. There are the following cases:

- W^x_{i+1}, W^x_{j-1} are simple terms. Then the substring $W^x_{[i+1,j-1]}$ is of the form **(M3)**, it is shorter than $W^x_{[i,j]}$ hence $M(i+1,j-1) = 1$. By the induction hypothesis that string is found by the algorithm correctly so $M(i,j) = 1$ (case (**A2**)).
- $W^x_{i+1} = *, W^x_{j-1} \in T$. So $W^x_{[i,j]}$ is of the form $p^{(m)} * ...\rangle ... \langle ... q^{(m+1)}$. Let j' be the position of the first left angle bracket in $W^x_{[i,j]}$. We know there exists $j' < k < j-1$ such that $W^x_k W^x_{j-1}$ reduces to 1. Hence $W^x_{[j',j-1]}$ is of the form **(M2a)**, it is shorter than $W^x_{[i,j]}$ hence $M(j', j-1) = 1$. By the induction hypothesis that string is found by the algorithm correctly so $M(i,j) = 1$ (case (**A4a**)).
- $W^x_{i+1} \in T$, $W^x_{j-1} = *$. So $W^x_{[i,j]}$ is of the form $p^{(m)} ... \rangle ... \langle ... * q^{(m+1)}$. Let i' be the position of the last right angle in $W^x_{[i,j]}$. We know there exists $i+1 < k < i'$ such that $W^x_{i+1} W^x_k$ reduces to 1. Hence $W^x_{[i+1,i']}$ is of the form **(M2b)**, it is shorter than $W^x_{[i,j]}$ hence $M(i+1,i') = 1$. By the induction hypothesis that string is found by the algorithm correctly so $M(i,j) = 1$ (case (**A4b**)).
- $W^x_{i+1} = *, W^x_{j-1} = *$. Then the string $W^x_{[i,j]}$ can be of the form $p^{(m)} * U \rangle \langle V * q^{(m+1)}$ and then $M(i,j) = 1$ by the initial case of the description of the algorithm. If $W^x_{[i,j]}$ contains more brackets, then there is a string $W^x_{[k,k']}$, where k is the index of the first left angle bracket in the string $W^x_{[i,j]}$ and k' is the index of the last right angle bracket in the string $W^x_{[i,j]}$. $W^x_{[k,k']}$ is of the form **(M4)**, it is shorter than $W^x_{[i,j]}$ therefore by the induction hypothesis $M(k,k') = 1$. So $M(i,j) = 1$ by (**A4c**).

– 4. $W^x_{[i,j]}$ is of the form **(M4)**. Then let k be the index of the first simple term that participates in the reduction (that is the first simple term in the substring $\mathbb{D}_1$, so it is obviously in between the first pair of matched angle brackets) and k' be the index of the last simple term that participates in the reduction (that is the last simple term in the substring $\mathbb{D}_g$, so it is obviously in between the last pair of matched angle brackets). The substring $W^x_{[k,k']}$ is of the form **(M3)** is shorter than $W^x_{[i,j]}$ so by induction hypothesis $M(k,k') = 1$. Therefore $M(i,j) = 1$ by (**A5**).

The algorithm is polynomial. One can observe that all conditions for $W^x_{[i,j]}$ can be checked in $O(j-i)$ steps. The number of the substrings is equal to $O(|W^x|^2)$. So one can compute the function $M(1, |W^x|)$ in time $O(|W^x|^3)$.

2.3 Obtaining the reduction

The algorithm can be easily modified to become a parsing algorithm. Each obtained reduction is described by the set of the links that take part in it. 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 $Prev(i,j)$ to help us follow the backtracking (as the value of the function $M(i,j)$ does not tell 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:

- for any initial case, $Prev(i,j) = ((i,j),(0,0),(0,0))$
- if it is a computation by one of the cases: (**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 the function M was 1 in the current computation (that is e.g. in (**A2**) a pair $(k,l) = (i+1,j-1)$)
- if it is a computation by one of the cases: (**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 the function M was 1 in the current computation
- if it is a computation by one of the cases: (**A1a**), (**A1b**), then $Prev(i,j) = ((0,0),(i,k),(k+1,j))$

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 $L(i,j) = L(i,k) \cup L(k+1,j)$.

3 Examples

We run our algorithm on some linguistic examples from [6], [8].

Example 1. Let us take a sample dictionary:

I: π_1
will: $\pi^r s_1 j^l,\ q_1 j^l \pi^l$
meet: io^l
him: o

with

$\pi_1 \leqslant \pi,\quad i \leqslant j,\quad s_1 \leqslant s$

Let us consider the sentence:

I will meet him.

We construct a string W_x:

$$\underset{1}{\langle} \underset{2}{*} \underset{3}{\pi_1^{(0)}} \underset{4}{*} \underset{5}{\rangle} \underset{6}{\langle} \underset{7}{*} \underset{8}{\pi^{(1)}} \underset{9}{s_1^{(0)}} \underset{10}{j^{(-1)}} \underset{11}{*} \underset{12}{q_1^{(0)}} \underset{13}{j^{(-1)}} \underset{14}{\pi^{(-1)}} \underset{15}{*} \underset{16}{\rangle} \underset{17}{\langle} \underset{18}{*} \underset{19}{i^{(0)}} \underset{20}{o^{(-1)}} \underset{21}{*} \underset{22}{\rangle} \underset{23}{\langle} \underset{24}{*} \underset{25}{o^{(0)}} \underset{26}{*} \underset{27}{\rangle} \underset{28}{\langle} \underset{29}{*} \underset{30}{s^{(1)}}$$

We can compute the function $M(i,j)$. There are no substrings of the length 2 such that $M(i,j) = 1$. We start with the following $M(i,j)$:

1. $M(3,8) = 1$
2. $M(10,19) = 1$
3. $M(20,25) = 1$

Then we compute:

4. $M(10,25) = 1$ by the lines 2,3 and the case (**A1a**)
5. $M(10,27) = 1$ by the line 4 and the case (**A3b**)
6. $M(9,30) = 1$ by the line 5 and the case (**A4b**)
7. $M(3,30) = 1$ by the lines 1,6 and the case (**A1a**)
8. $M(1,30) = 1$ by the line 7 and the case (**A3a**)

Therefore there exists a type assignment for a given string of words which reduces to s.

The output of the parsing algorithm is:
{(3,8), (9,30), (10,19) (20,25)}

Example 2. The dictionary:

did: $q_2 i^l \pi^l,\ \pi^r s_2 i^l$
he: π_3
give: $io^l, i\overline{n}^l o'^l$
books: n
to: $i^r io^l, \overline{j} i^l$
her: o

with

$\pi_j \leqslant \pi,\ s_i \leqslant s,\ q_k \leqslant q \leqslant s,\ \overline{j} \leqslant \pi_3,\ o' \leqslant o,\ n \leqslant \overline{n} \leqslant o$

We consider the sentence:

Did he give books to her?

We construct a string W_x:

$< *\pi^r s_2 i^l * q_2 i^l \pi^l * >< *\pi_3 * >< * io^l * i\overline{n}^l o'^l * >< *n* >< *i^r io^l * \overline{j} i^l * >< *o* >< *s^r$

1 7 8 9 14 16 19 20 29 34 36 44 49

We can compute the function $M(i, j)$. There are no substrings of the length 2 such that $M(i, j) = 1$. The algorithm computes $M(i, j) = 1$:

1. $M(9, 14) = 1$ by initial case
2. $M(9, 16) = 1$ by line 1 and (**A3b**)
3. $M(8, 19) = 1$ by line 2 and (**A4b**)-useless
4. $M(8, 22) = 1$ by line 2 and (**A4b**) - useless
5. $M(20, 29) = 1$ by initial case
6. $M(8, 29) = 1$ by lines 3,5 and (**A1a**)-useless
7. $M(20, 31) = 1$ by line 5 and (**A3b**)
8. $M(8, 31) = 1$ by line 6 and (**A3b**)-useless
9. $M(19, 34) = 1$ by line 7 and (**A4b**)
10. $M(17, 34) = 1$ by line 9 and (**A4a**)
11. $M(9, 34) = 1$ by lines 2,10 and (**A1b**)
12. $M(8, 35) = 1$ by line 11 and (**A2**)
13. $M(36, 44) = 1$ by initial case
14. $M(8, 44) = 1$ by lines 12,13 and (**A1a**)
15. $M(36, 46) = 1$ by line 13 and (**A3b**)-useless
16. $M(8, 46) = 1$ by line 14 and (**A3b**)
17. $M(7, 49) = 1$ by line 16 and (**A4b**)
18. $M(1, 49) = 1$ by line 17 and (**A3a**)

Therefore there exists a type assignment for a given string which reduces to s. The output of the parsing algorithm is:
{(7,49), (8,35), (9,14), (19,34), (20,29), (36,44)}

Example 3. An example from [8].

The dictionary:

Kim: np
mailed: $np^r\ s\ pp^l\ np^l, np^r\ s\ np^l$
the: $np\ n^l$
letter: $n\ pp^l,\ n$
to: $pp\ np^l$
Sandy: np

We consider the sentence:

Kim mailed the letter to Sandy.

We construct a string W_x:

$< *np* >< *np^r\ s\ pp^l\ np^l * np^r\ s\ np^l * >< *np\ n^l * >< *n\ pp^l * n* >$

1 3 8 9 10 11 17 20 21 29

$< *pp\ np^l * >< *np* >< *s^r$

32 34 35 40 45

1. $M(3,8) = 1$ by initial case -(r1)
2. $M(1,8) = 1$ by line 1 and (**A3a**) - useless
3. $M(3,13) = 1$ by initial case -(r2)
4. $M(1,13) = 1$ by line 3 and (**A3a**) - useless
5. $M(15,20) = 1$ by initial case - (r2)
6. $M(11,20) = 1$ by initial case - (r1)
7. $M(21,26) = 1$ by initial case - (r2)
8. $M(15,26) = 1$ by lines 5,7 and (**A1a**) - (r2')
9. $M(11,26) = 1$ by line 6,7 and (**A1a**) - useless
10. $M(21,29) = 1$ by initial case - (r1)
11. $M(15,29) = 1$ by lines 5,10 and (**A1a**)
12. $M(11,29) = 1$ by lines 6,10 and (**A1a**)- (r1)
13. $M(21,31) = 1$ by line 10 and (**A3b**) - useless
14. $M(15,31) = 1$ by line 11 and (**A3b**) - useless
15. $M(11,31) = 1$ by line 12 and (**A3b**) - (r1)
16. $M(27,34) = 1$ by initial case - (r2)
17. $M(21,34) = 1$ by lines 7,16 and (**A1a**) - (r2")
18. $M(15,34) = 1$ by lines 5,17 and (**A1a**) or by line 8,16 and (**A1a**) - both leading to (r2)
19. $M(11,34) = 1$ by lines 6,17 and (**A1a**) - useless
20. $M(10,34) = 1$ by line 15 and (**A4b**) - (r1)
21. $M(35,40) = 1$ by initial case - (r1) and (r2)
22. $M(27,40) = 1$ by lines 16,21 and (**A1a**) - (r2)
23. $M(21,40) = 1$ by lines 7,22 and (**A1a**) - (r2) or by lines 17,20 and (**A1a**)
24. $M(15,40) = 1$ by lines 5,23 and (**A1a**) - (r2) or by lines 8,22 - (r2') and (**A1a**) or by lines 18,21 and (**A1a**) - (r2")
25. $M(11,40) = 1$ by lines 6,23 and (**A1a**) - useless
26. $M(10,40) = 1$ by lines 20,21 and (**A1a**) - (r1)
27. $M(35,42) = 1$ by line 21 and (**A3b**) - useless
28. $M(27,42) = 1$ by line 22 and (**A3b**) - useless
29. $M(21,42) = 1$ by line 23 and (**A3b**) - useless
30. $M(15,42) = 1$ by line 24 and (**A3b**) - (r2)
31. $M(11,42) = 1$ by line 25 and (**A3b**) - useless
32. $M(10,42) = 1$ by line 26 and (**A3b**) - (r1)
33. $M(14,45) = 1$ by line 30 and (**A4b**) - (r2)
34. $M(9,45) = 1$ by line 32 and (**A4b**) - (r1)
35. $M(3,45) = 1$ by lines 1,34 and (**A1a**) - (r1) or lines 3,33 and (**A1a**) - (r2)
36. $M(1,45) = 1$ by line 35 and (**A3a**) - (r1) and (r2)

There are 2 possible reductions:
{(3,8), (9,45), (10,34), (11,20), (21,29), (35,40)} and
{(3,13), (14,45), (15,20), (21,26), (27,34), (35,40)}

4 Algorithms of Oehrle and Savateev

For a better readability of this paper we briefly discuss related algorithms of Oehrle [8] and Savateev [10].

The run of Oehrle's algorithm on the last example is presented on pp. 67-71 of [8], so it takes much more space than our presentation. Oehrle's algorithm is essentially different. It represents the entry $a_1...a_n$ as a directed graph. Each type A assigned to a_i by G is represented as a linear graph of simple terms; one draws arcs from the source 1 to the first simple terms of these types and from the last simple terms of these types to the target 1. Then one identifies the target of the graph of a_i with the source of the graph of a_{i+1}. The algorithm adds new arcs to the resulting graph. For any path $t_1 \to 1 \to t_2$, one adds $t_1 \to t_2$, and for any path $t_1 \to p^(n) \to p^(n+1) \to t_2$, one adds $t_1 \to t_2$ ([8] does not regard poset rules). This procedure is recursively executed, running the graph from the left to the right. At the end, it returns the so-called winners, i.e. simple terms t such that the final graph contains the path $1 \to t \to 1$ from the source of a_1 to the target of a_n.

Oehrle [8] shows that the time is $O(n^3)$, but the size of G is treated as a constant. It can be shown that the time is cubic also in our sense (depending on $|W^x|$). However, the work of the algorithm seems more complicated than ours; at each step, it has to remember all predecessors and successors of every vertice and update some of them. Also, our algorithm can easily be modified to admit an arbitrary type $\mathbb{B}$ in the place of s: it suffices to replace $s^{(1)}$ by $\mathbb{B}^r$, where $(p_1^{(n_1)}...p_k^{(n_k)})^r = p_k^{(n_k+1)}...p_1^{(n_1+1)}$. This generalization is not directly applicable to Oehrle's algorithm.

The algorithm of Savateev [10] is a recognition algorithm for $\mathbf{L}^{\backslash}$-grammars. Recall that $\mathbf{L}^{\backslash}$ is the $\backslash$-fragment of the Lambek calculus. Types are formed out of atomic types $p, q, r, ...$ by the rule: if A, B are types, then $(A\backslash B)$ is a type. An $\mathbf{L}^{\backslash}$-grammar assigns a finite number of types to each $a \in \Sigma$. $L(G)$ is defined as for pregroup grammars except that **CBL** is replaced by $\mathbf{L}^{\backslash}$ (a comma separates types, whereas for pregroup grammars it is interpreted as the concatenation of strings).

Atoms are expressions p^n, where p is an atomic type and n is a positive integer. For each type A, $\gamma(A)$ is a finite string of atoms, defined as follows: $\gamma(p) = p^1$, $\gamma(A\backslash B) = (\gamma(A))^\top \gamma(B)$, where $(p^{n_1}...p^{n_k})^\top = p^{n_k+1}...p^{n_1+1}$. This is precisely the translation of types in the formalism of free pregroups except that Savateev writes p^{n+1} for our $p^{(n)}$.

A string of atoms has a good pairing, if there exists a system of non-crossing links (as in pregroup reductions) such that:

(1) each atom is a left or right end of exactly one link,

(2) if p^n is the left end of a link, then its right end is p^{n+1},

(3) if p^n is the left end of a link and n is even, then the interval between p^n and the right end p^{n+1} contains an atom p^m with $m < n$.

The key lemma states that $A_1, ..., A_n \to B$ is provable in $\mathbf{L}^{\backslash}$ if and only if the string $\gamma(A_1)...\gamma(A_n)(\gamma(B))^{\top}$ has a good pairing. Clearly (1) and (2) yield the reducibility of the string to 1 in **CBL** (after replacing p^{n+1} by $p^{(n)}$), and (3) is an additional constraint (necessary, since $\mathbf{L}^{\backslash}$ is weaker than **CBL**).

The recognition algorithm for $\mathbf{L}^{\backslash}$-grammars represents the entry $x = a_1...a_n$ like in our algorithm; each type A assigned to a_i is replaced by $\gamma(A)$. The algorithm checks whether there exists a string $A_1, ..., A_n$ of types assigned to $a_1, ..., a_n$, respectively, such that the string $\gamma(A_1)...\gamma(A_n)(\gamma(B))^{\top}$ has a good pairing. (B is the designated type of G.)

We will not write details of the algorithm and the proof of correctness, given in [10]. As mentioned in Section 1, although the algorithm is generally similar to ours (we follow it!), several details are essentially different. For instance, Savateev's algorithm and the proof of its correctness rely upon the fact that the right-most atom in $\gamma(A)$ has the form p^1, and other peculiarities of the translation. Thus, Savateev's algorithm does not correctly check whether an arbitrary string of atoms has a good pairing; it is correct only for strings which are related to $\mathbf{L}^{\backslash}$-sequents in the way described above. Our algorithm handles arbitrary strings of terms, admits both positive and negative integers n (i.e. both right and left adjoints), and needs no constraint like (3).

References

1. D. Béchet, Parsing Pregroup Grammars and Lambek Calculus using Partial Composition, *Studia Logica* 87.2-3 (2007), 199-224.
2. W. Buszkowski, Lambek Grammars Based on Pregroups, in: P. de Groote, G. Morrill, and C. Retorè (eds.), *Logical Aspects of Computational Linguistics*, LNAI 2099, 2001, 95–109.
3. W. Buszkowski, Type Logics and Pregroups, *Studia Logica* 87.2-3 (2007), 145–169.
4. W. Buszkowski and K. Moroz, Pregroup grammars and context-free grammars, in: C. Casadio and J. Lambek (eds.), *Computational Algebraic Approaches to Natural Language*, Polimetrica, 2008, 1–21.
5. J. Lambek, Type Grammars Revisited, in: A.Lecomte, F. Lamarche, and G. Perrier (eds.), *Logical Aspects of Computational Linguistics*, LNAI 1582, 1999, 1–27
6. J. Lambek, From Word to Sentence: a computational algebraic approach to grammar, Polimetrica, 2008.
7. M. Mootrgat and R. Oehrle, Pregroups and type-logical grammar: searching for convergence, in C. Casadio, P.J. Scott, R.A.G.Seely (eds.), *Language and grammar*, CSLI Publications, 2005, 141-160.
8. R. Oehrle, A parsing algorithm for pregroup grammars, *Proceedings of Categorial Grammars 2004*, Montpellier France, 2004, 59–75.
9. A.Preller, Linear Processing with Pregroups, *Studia Logica* 87.2-3 (2007), 171-197.
10. Y. Savateev, Unidirectional Lambek Grammars in Polynomial Time, *Theory of Computing Systems*, to appear.