ISSN 0081-5438, Proceedings of the Steklov Institute of Mathematics, 2016, Vol. 294, pp. 129–138. © Pleiades Publishing, Ltd., 2016.
Published in Russian in Trudy Matematicheskogo Instituta imeni V.A. Steklova, 2016, Vol. 294, pp. 141–151.

On Translating Lambek Grammars with One Division into Context-Free Grammars

S. L. Kuznetsov[a]

Received April 13, 2016

Abstract—We describe a method of translating a Lambek grammar with one division into an equivalent context-free grammar whose size is bounded by a polynomial in the size of the original grammar. Earlier constructions by Buszkowski and Pentus lead to exponential growth of the grammar size.

DOI: 10.1134/S0081543816060080

1. LAMBEK GRAMMARS AND CONTEXT-FREE GRAMMARS

Lambek grammars were introduced by J. Lambek [12] for the mathematical description of the syntax of natural language fragments. They form a branch of the categorial grammar framework. Generally, grammars define *formal languages* (further we will omit the word "formal"), i.e., sets of *words* built from elements (*letters*, or *symbols*) of a finite set Σ called the *alphabet*. The set of all words over a given alphabet Σ is denoted by Σ^*; Σ^+ stands for the set of all nonempty words. The empty word is denoted by ε. We consider only grammars describing languages without the empty word (i.e., subsets of Σ^+). However, the empty word can be used inside the grammar formalism.

A language can be infinite (as a set), and therefore the task of describing it by means of a finite grammar is nontrivial (in particular, due to cardinality issues, not every language can be defined by a formal grammar of a particular kind).

A *categorial grammar* is a binary correspondence between letters of the alphabet and logical expressions called *syntactic types*. Each letter is associated with a finite number of syntactic types (possibly with more than one). A word[1] $w = a_1 \dots a_n$ belongs to the language defined by the grammar if and only if there exist syntactic types $A_1, \dots, A_n$ such that A_i is in the given correspondence with a_i (for all i from 1 to n) and the sequent $A_1 \dots A_n \to H$ is derivable in a specific logical calculus. Here H is a designated type (one for the whole grammar).

In Lambek grammars, the calculus used to determine whether a word belongs to the language is the *Lambek calculus*, denoted by L. Syntactic types for the Lambek calculus are built from a set of *primitive types* (variables) Pr using three binary connectives: *multiplication* $\cdot$ and *left* and *right divisions* $\backslash$ and $/$. The set of all types is denoted by Tp. We use capital Latin letters $(A, B, C, \dots)$ for types and capital Greek letters $(\Gamma, \Delta, \Pi, \dots)$ for finite (possibly empty) linearly ordered sequences of types. Following the linguistic tradition, we do not use commas to separate types in the sequence.

The Lambek calculus L derives objects of the form $\Pi \to B$ called *sequents*. Here B is a syntactic type and Π is a nonempty sequence of syntactic types.

[a] Steklov Mathematical Institute of Russian Academy of Sciences, ul. Gubkina 8, Moscow, 119991 Russia.
E-mail address: sk@mi.ras.ru

[1] Here we should note some difference in terminology between the studies of "formal" and "real" languages. In linguistic applications, *letters* of Σ correspond not to letters but rather to *words* (word forms) of the natural language; *words* over Σ correspond to *sentences*. Thus grammars describe not the lexical but the syntactic level of a language structure. In this paper we use the "letter–word" terminology, not the "word–sentence" one.

The axioms of L are sequents of the form $A \to A$, and the rules of inference are as follows:

$$\frac{A\Pi \to B}{\Pi \to A \setminus B}, \quad \text{where } \Pi \text{ is nonempty,} \qquad \frac{\Pi \to A \quad \Gamma B \Delta \to C}{\Gamma \Pi (A \setminus B) \Delta \to C},$$

$$\frac{\Pi A \to B}{\Pi \to B \,/\, A}, \quad \text{where } \Pi \text{ is nonempty,} \qquad \frac{\Pi \to A \quad \Gamma B \Delta \to C}{\Gamma (B \,/\, A) \Pi \Delta \to C},$$

$$\frac{\Gamma \to A \quad \Delta \to B}{\Gamma\Delta \to A \cdot B}, \qquad \frac{\Gamma A B \Delta \to C}{\Gamma (A \cdot B) \Delta \to C}.$$

In addition to these rules, L admits the *cut rule* [12]

$$\frac{\Pi \to A \quad \Gamma A \Delta \to C}{\Gamma \Pi \Delta \to C}$$

(this means that adding the cut rule does not lead to new derivable sequents).

If $\Pi \to B$ is derivable in L, we denote this fact by $\mathrm{L} \vdash \Pi \to B$.

Finally, a *Lambek grammar* is a triple $\mathcal{G} = \langle \Sigma, \rhd, H \rangle$, where Σ is an alphabet, $H \in \mathrm{Tp}$ is a designated type, and $\rhd \subset \Sigma \times \mathrm{Tp}$ is a finite binary correspondence between letters of the alphabet and syntactic types. As said before, a word $w = a_1 \dots a_n$ is accepted by $\mathcal{G}$ iff there exist types $A_1, \dots, A_n$ such that $a_i \rhd A_i$ (for all i from 1 to n) and $\mathrm{L} \vdash A_1 \dots A_n \to H$.

Note that Lambek grammars as defined above are explicitly disallowed to generate the empty word. However, one can drop the constraint of Π being nonempty from the rules, which leads to a modified calculus L*. Grammars based on L* can define languages containing the empty word.

Another way of describing the language syntax, introduced by Chomsky [4], is the framework of context-free grammars. A *context-free grammar* is a quadruple $G = \langle N, \Sigma, P, S \rangle$, where Σ is the alphabet over which we define the language, N is an auxiliary alphabet called the set of *nonterminal symbols* (N and Σ are required to be disjoint), $S \in N$ is the *start symbol*, and P is a finite set of *rules* of the form $A \Rightarrow \alpha$, where $A \in N$ and α is a word in the alphabet $N \cup \Sigma$. In our definition, α in every rule is required to be nonempty. The symbol A and the word α are called the *left-* and *right-hand sides* of the rule, respectively. Let η and θ be arbitrary (possibly empty) words over $N \cup \Sigma$. Then if $(A \Rightarrow \alpha) \in P$, the word $\eta\alpha\theta$ is *directly derivable* from the word $\eta A \theta$ in G (which is written as $\eta A \theta \Rightarrow_G \eta\alpha\theta$). The relation $\Rightarrow_G^*$ ("derivable in G") is defined as the reflexive–transitive closure of $\Rightarrow_G$. The *language defined by the grammar* G is the set $\{w \in \Sigma^+ \mid S \Rightarrow_G^* w\}$. Such languages are called *context-free* (recall that we consider only languages without the empty word).

If G is a grammar (a Lambek grammar or a context-free one), then the language defined by G will be denoted by $\mathcal{L}(G)$.

Theorem 1. *Each language defined by a Lambek grammar is context-free. Each context-free language is defined by a Lambek grammar.*

The first statement of this theorem was proved by Pentus [17]; for the case where only one division operation is used, this result was proved earlier by Buszkowski [3]. The second statement was proved by Gaifman [2] for a weaker formalism called basic categorial grammars or Ajdukiewicz–Bar-Hillel grammars; Buszkowski [3] noticed that Gaifman's construction also works for Lambek grammars. In this construction only one division operation ($\setminus$ or, symmetrically, $/$) is used.

Theorem 1 states the equivalence of Lambek grammars and context-free grammars *in the weak sense.* That is, the classes of languages defined by these two grammar formalisms coincide as sets of words. Actually, both Lambek grammars and context-free ones can solve a more sophisticated task than just determining whether a word belongs to the language. To a word that belongs to the language, they can assign an extra structure encoding the *semantics* ("meaning") of the word. If this extra structure is also preserved under the transformation, the grammars are *equivalent in the strong sense* [9, 10]. In this paper we consider only equivalence in the weak sense.

Apart from the Lambek calculus L itself, its extensions and fragments are also broadly studied. Many extensions of the Lambek calculus are important for linguistic applications [8, 13, 7, 14]. Connectives of the Lambek calculus have a natural interpretation as operations (multiplication and two divisions) on formal languages [18]; therefore, it seems very natural to extend the Lambek calculus to cover other well-known language-theoretic operations. An interesting open problem here is connected with the iteration ("Kleene star"). Although the question of extending the Lambek calculus with iteration looks easy, no "good" axiomatization is known; however, there exist complete systems with an ω-rule or infinite branches in derivation trees [11]. Interestingly enough, in analogous systems for the Gödel–Löb modal logic GL and its extensions, infinite derivations can be reduced to finite (cyclic) ones [26, 27]. Possibly, a variant of this strategy could also be successful for the Lambek calculus with iteration.

For the fragments of L, the cut elimination theorem makes their axiomatization an easy task: one just leaves only the rules operating the chosen connectives.

In this paper we consider the fragment of the Lambek calculus with only one division, $\mathrm{L}(\setminus)$. As opposed to the full Lambek calculus (L) and its fragments with two operations ($\mathrm{L}(\setminus,\cdot)$, $\mathrm{L}(/,\cdot)$, and $\mathrm{L}(/,\setminus)$) with NP-complete derivability problems [19, 25, 24], the derivability problem for $\mathrm{L}(\setminus)$ is decidable in polynomial time [23]. On the other hand, any context-free language (see above) can be defined by an $\mathrm{L}(\setminus)$-grammar. In other words, only one Lambek operation, namely, one of the two divisions, is sufficient for modeling context-free derivations.

2. SAVATEEV'S DERIVABILITY CRITERION FOR $\mathrm{L}(\setminus)$

In this section we formulate a graph-theoretic criterion for derivability in $\mathrm{L}(\setminus)$ that was proved by Savateev [22]. We will use this criterion in our construction.

Let $\mathrm{Atn} = \{p^{(i)} \mid p \in \mathrm{Pr},\ i \in \mathbb{N}\}$ be the set of *atoms* (an atom is a primitive type with a numerical superscript). We are going to consider finite nonempty sequences of atoms; the set of all such sequences is denoted by Atn^+.

For $\mathbb{A} = p_1^{(i_1)} p_2^{(i_2)} \dots p_k^{(i_k)} \in \mathrm{Atn}^+$ let $\mathbb{A}^{+2} = p_1^{(i_1+2)} p_2^{(i_2+2)} \dots p_k^{(i_k+2)}$.

We define two mappings $\gamma, \bar\gamma\colon \mathrm{Tp} \to \mathrm{Atn}^+$ of $\mathrm{L}(\setminus)$-types to sequences of atoms:

$$\gamma(p) = p^{(1)}, \qquad \bar\gamma(p) = p^{(2)},$$

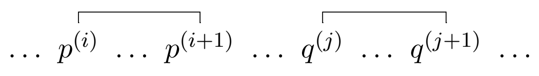
$$\gamma(A \setminus B) = \bar\gamma(A)\gamma(B), \qquad \bar\gamma(A \setminus B) = \bar\gamma(B)(\gamma(A))^{+2}.$$

Let $\mathbb{A}$ be a finite sequence of atoms. A *proof net* on $\mathbb{A}$ is a pairing on $\mathbb{A}$ such that

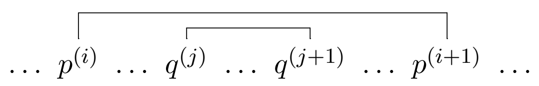
(1) every element belongs to exactly one pair;

(2) each pair consists of $p^{(i)}$ and $p^{(i+1)}$, where $p \in \mathrm{Pr}$, $i \in \mathbb{N}$, and $p^{(i)}$ lies to the left of $p^{(i+1)}$ in the sequence;

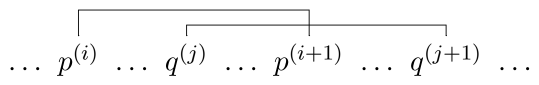
(3) one can draw the pairing as lines in the upper half-plane without intersections; in other words, the pair dispositions


$\dots\ p^{(i)}\ \dots\ p^{(i+1)}\ \dots\ q^{(j)}\ \dots\ q^{(j+1)}\ \dots$

and


$\dots\ p^{(i)}\ \dots\ q^{(j)}\ \dots\ q^{(j+1)}\ \dots\ p^{(i+1)}\ \dots$

are allowed, while


$\dots\ p^{(i)}\ \dots\ q^{(j)}\ \dots\ p^{(i+1)}\ \dots\ q^{(j+1)}\ \dots$

is prohibited;

(4) if the left atom in a pair has an even superscript, then between the atoms of the pair there exists an atom with the superscript less than the superscripts of both elements of the pair.

Note that in another paper [23] Savateev uses a slightly different criterion: for a pair in which the left atom has an even superscript 2ℓ, there should be an atom with superscript *precisely* $2\ell - 1$ in between. This criterion is equivalent to the one described above.

Theorem 2 [22]. *A sequent $A_1, \ldots, A_n \to B$ is derivable in* $\mathrm{L}(\,\backslash\,)$ *if and only if there exists a proof net on $\gamma(A_1)\ldots\gamma(A_n)\bar{\gamma}(B)$.*

3. THE GROWTH OF THE LAMBEK GRAMMAR SIZE UNDER TRANSLATION INTO A CONTEXT-FREE GRAMMAR

By Theorem 1, between the two grammar formalisms (Lambek grammars and context-free grammars) there exist translations in both directions that support weak equivalence of the formalisms. In such situations, however (see, for example, [21]), one is usually interested not only in the existence of such translations but also in the moderate *change* (usually growth) *of the size* of the object (in our case, the grammar) under such a transformation. If the size grows dramatically (e.g., exponentially), the translation becomes practically useless.

Unfortunately, the translation of Lambek grammars into context-free ones that was presented by Pentus [17] leads to exponential growth of the grammar size. In the general case such inefficiency appears to be inevitable, because the derivability problem for L is NP-complete (see above), while the parsing problem for context-free grammars is decidable in polynomial time.

For the one-division case another method for translating $\mathrm{L}(\,\backslash\,)$-grammars into context-free grammars was proposed earlier by Buszkowski [3], but it also leads to exponential growth of the grammar size. Here we present a method that translates an $\mathrm{L}(\,\backslash\,)$-grammar into a context-free grammar of size bounded by a polynomial in the original grammar size.

First we define the notion of *size* for Lambek grammars and context-free grammars more accurately.

Let $A \in \mathrm{Tp}$. We define the size $|A|$ of A as the number of primitive type occurrences in A. The formal definition of $|A|$ is recursive: $|p_i| = 1$; $|A \setminus B| = |B \,/\, A| = |A \cdot B| = |A| + |B|$. The size of a Lambek grammar $G = \langle \Sigma, \rhd, H \rangle$ is defined as follows: $|G| = |H| + \sum_{\langle a, A\rangle \in \rhd} |A|$.

By the size of a context-free grammar $G = \langle N, \Sigma, P, S \rangle$, we will mean the number $\sum_{(A \Rightarrow \alpha) \in P} |\alpha|$, where $|\alpha|$ is the number of letters in α. Note that $|\Sigma| \le |G|$ (we assume that Σ does not contain useless symbols not appearing in $\mathcal{L}(G)$), $|N| \le |G|$, and $|P| \le |G|$. The size of any grammar G is denoted by $|G|$ (this applies both to Lambek grammars and to context-free ones).

Theorem 3. *For any* $\mathrm{L}(\,\backslash\,)$*-grammar there exists an equivalent* (*in the weak sense*) *context-free grammar whose size is bounded by a polynomial in the size of the original grammar.*

4. SAVATEEV'S CONDITIONS AS CONTEXT-FREE RULES

The main idea of our translation of grammars is to write the proof net conditions as a context-free grammar. Unfortunately, the sequences of atoms used in the definition of a proof net are words over an infinite alphabet (Atn); therefore, it is formally impossible to construct a context-free grammar for a set of them. To overcome this issue, we restrict Atn to a finite set.

Let G be an $\mathrm{L}(\,\backslash\,)$-grammar. First we remove from Pr all primitive types not occurring in G. After that Pr becomes a finite set. Moreover, it has at most $|G|$ elements.

Now we define the *depth* of a syntactic type from $\mathrm{Tp}(\,\backslash\,)$ recursively: $d(p_i) = 1$; $d(A \setminus B) = \max\{d(A) + 1, d(B)\}$. On the other hand, we denote the accordingly restricted set of atoms, $\{p^{(i)} \mid p \in \mathrm{Pr},\ i \le m\}$, by Atn_m ($m \in \mathbb{N}$).

Lemma 1. *If* $A \in \mathrm{Tp}(\,\backslash\,)$, *then* $\gamma(A) \in \mathrm{Atn}^{+}_{d(A)}$ *and* $\bar{\gamma}(A) \in \mathrm{Atn}^{+}_{d(A)+1}$.

Proof. We proceed by structural induction on A. If $A = p_i$, then we have $d(A) = 1$, $\gamma(A) = p_i^{(1)} \in \mathrm{Atn}_1^+$, and $\bar\gamma(A) = p_i^{(2)} \in \mathrm{Atn}_2^+$.

For the case of $A = B \setminus C$ we preliminarily notice some obvious properties of the sets Atn_m^+. First, if $\mathbb{B} \in \mathrm{Atn}_{m_1}^+$ and $\mathbb{C} \in \mathrm{Atn}_{m_2}^+$, then $\mathbb{B}\mathbb{C} \in \mathrm{Atn}_{\max\{m_1,m_2\}}^+$. Second, if $\mathbb{B} \in \mathrm{Atn}_m^+$, then $\mathbb{B}^{+2} \in \mathrm{Atn}_{m+2}^+$.

Finally,

$$\gamma(B \setminus C) = \bar\gamma(B)\gamma(C) \in \mathrm{Atn}^+_{\max\{d(B)+1,d(C)\}} = \mathrm{Atn}^+_{d(B\setminus C)},$$

$$\bar\gamma(B \setminus C) = \bar\gamma(C)(\gamma(B))^{+2} \in \mathrm{Atn}^+_{\max\{d(C)+1,d(B)+2\}} = \mathrm{Atn}^+_{d(B\setminus C)+1}. \qquad \square$$

By the *depth* of an $\mathrm{L}(\setminus)$-grammar $G = \langle \Sigma, \rhd, H\rangle$ we will mean the number $d(G) = \max\{d(A) \mid$ either $A = H$ or $a \rhd A$ for some $a \in \Sigma\}$. Note that since $d(A) \le |A|$ for any type A (this is easily checked by induction), for any $\mathrm{L}(\setminus)$-grammar G we have $d(G) \le |G|$.[2]

Let $m = d(G) + 1$. Then for any sequent used for checking whether a word belongs to the language defined by G, the corresponding sequence of atoms lies in Atn_m^+. On the other hand, $|\mathrm{Atn}_m| = |\mathrm{Pr}|m \le |G|(d(G)+1) \le |G|(|G|+1)$; therefore, the sequences of atoms to be considered are words in an alphabet of polynomially bounded (with respect to the size of the original grammar) cardinality.

Now let us define a context-free grammar $\mathbf{S}_m$ that formalizes the existence of a proof net on a given word over the alphabet Atn_m. Let $\mathbf{S}_m = \langle N, \mathrm{Atn}_m, P, S\rangle$, where $N = \{S, R_1, \ldots, R_{m-1}\}$ and P consists of the following rules:

$$\begin{array}{rl}
S \Rightarrow R_k & \text{for all } k \text{ from 1 to } m-1,\\
R_{k_1} \Rightarrow R_{k_2} & \text{if } k_1 > k_2,\\
R_k \Rightarrow p^{(2\ell-1)} R_k p^{(2\ell)} & \text{if } k < m \text{ and } 2\ell < m,\\
R_k \Rightarrow p^{(2\ell)} R_k p^{(2\ell+1)} & \text{if } 2\ell+1 < m \text{ and } k < 2\ell,\\
R_{2\ell-1} \Rightarrow p^{(2\ell-1)} S p^{(2\ell)} & \text{if } 2\ell < m,\\
R_{2\ell-1} \Rightarrow p^{(2\ell-1)} p^{(2\ell)} & \text{if } 2\ell < m,\\
R_k \Rightarrow R_k S & \text{for all } k \text{ from 1 to } m-1,\\
R_k \Rightarrow S R_k & \text{for all } k \text{ from 1 to } m-1.
\end{array}$$

The total number of rules is at most $m + m^2 + 2|\mathrm{Pr}|m^2 + 2|\mathrm{Pr}|m + 2m \le 8|\mathrm{Pr}|m^2 \le 8|G|^3$. The right-hand side of each rule is of length at most 3. Thus, $|\mathbf{S}_m| \le 24|G|^3$, i.e., the size of $\mathbf{S}_m$ is bounded by a polynomial in the size of the original grammar G.

Lemma 2. *A sequence of atoms $\mathbb{A} \in \mathrm{Atn}_m^+$ has a proof net if and only if it belongs to the language described by $\mathbf{S}_m$.*

Proof. In order to use induction, we augment the statement of this lemma ($S \Rightarrow^*_{\mathbf{S}_m} \mathbb{A}$ if and only if $\mathbb{A}$ has a proof net) with analogous statements about other nonterminal symbols: $R_k \Rightarrow^*_{\mathbf{S}_m} \mathbb{B}$ if and only if $\mathbb{B}$ has a proof net and contains an atom with superscript not greater than k.

Now these statements are easily proved by simultaneous induction: in the "if" part the induction parameter is the derivation length in $\mathbf{S}_m$; for the "only if" part we proceed by induction on the length of the sequence of atoms. $\square$

[2]At the same time, if G contains a "very deep" complicated syntactic type, $d(G)$ can be close to $|G|$.

Now let us describe a method to obtain a context-free grammar weakly equivalent to G from $\mathbf{S}_m$. We will need the notion of *homomorphism* of formal languages.

Let Σ_1 and Σ_2 be two alphabets. A homomorphism is a mapping $h\colon \Sigma_1^* \to \Sigma_2^*$ such that $h(uv) = h(u)h(v)$ for any $u, v \in \Sigma_1^*$. Clearly, one can arbitrarily define h on the elements of Σ_1, and then it is uniquely extended to longer words from Σ_1^*; $h(\varepsilon)$ is always ε (otherwise the equality $h(a) = h(a\varepsilon) = h(a)h(\varepsilon)$ is violated).

An important particular case of a homomorphism is a *nonextending homomorphism*, which maps every letter $a \in \Sigma_1$ either to the empty word or to one letter from Σ_2 (but not to a longer word).

Further we will denote the alphabet Atn_m by Σ_1.

Recall that we are constructing a context-free grammar for the language defined by the $\mathrm{L}(\,\backslash\,)$-grammar $G = \langle \Sigma, \rhd, H\rangle$. Let us introduce an auxiliary alphabet $\Sigma_2 = \{\langle a, A\rangle \mid a \rhd A\}$ and a fresh symbol **\$** that does not belong to the alphabets introduced before. Define two homomorphisms $g\colon \Sigma_2 \cup \{\mathbf{\$}\} \to \Sigma_1$ and $h\colon \Sigma_2 \cup \{\mathbf{\$}\} \to \Sigma$ as follows:

$$g(\langle a, A\rangle) = \gamma(A), \qquad g(\mathbf{\$}) = \bar{\gamma}(H), \qquad h(\langle a, A\rangle) = a, \qquad h(\mathbf{\$}) = \varepsilon.$$

It is easy to see that if $\mathbf{S}_m$ defines a language M, then the language defined by G is equal to $h(g^{-1}(M) \cap \{u\mathbf{\$} \mid u \in \Sigma_2^+\})$. One can construct a context-free grammar for this language, because the operations of taking the preimage of a homomorphism, intersecting with a regular language, and applying a homomorphism preserve the context-free property of the language. These facts were independently noticed by several authors [5, 15, 28] and nowadays are included into standard textbooks in formal language theory [1, 6, 16]. However, we will have to carefully analyze the proofs in order to get an estimation of the size of the resulting context-free grammar.

5. ESTIMATION OF THE SIZE OF THE CONTEXT-FREE GRAMMAR

For the "inverse homomorphism" construction (the step from M to $g^{-1}(M)$) we will use an auxiliary notion of *pushdown automaton* (PDA). A PDA is an automaton equipped with a pushdown memory structure (or *stack*). We are going to use the following version of the definition: a PDA is a sextuple $\mathfrak{M} = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z_0\rangle$, where Q, Σ, and Γ are finite sets, $q_0 \in Q$, $Z_0 \in \Gamma$, and Δ is a finite subset of the Cartesian product $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$. The set Q is called the set of *states*, Σ and Γ are the *input* and *stack alphabets*, respectively, q_0 is the *initial state*, and Z_0 is the *starting stack symbol.* We will write elements of Δ (*transitions* of the automaton) of the form $\langle p, a, Z, q, \beta\rangle$ as follows: $p \xrightarrow{a,\, Z\colon \beta} q$.

A *configuration* of a PDA is a triple $\langle q, v, \gamma\rangle$, where $q \in Q$, $v \in \Sigma^*$, and $\gamma \in \Gamma^*$. Informally this means that the automaton is now in the state q with the word v not yet read from the input (v is a suffix of the word originally given to the automaton), and the stack contains γ.

The initial configuration of the PDA has the form $\langle q_0, w, Z_0\rangle$, where w is the word on which we run the automaton.

Applying a transition $p \xrightarrow{a,\, Z\colon \beta} q$ changes the configuration as follows: $\langle p, aw, Z\gamma\rangle \to_{\mathfrak{M}} \langle q, w, \beta\gamma\rangle$. Informally, if the automaton was in state p with Z on top of the stack,[3] the automaton reads a from the input (or reads nothing if $a = \varepsilon$), removes Z from the top of the stack, puts the word β onto the stack, and changes the state to q. Note that this definition requires the automaton to pop exactly one symbol from the stack (of course, it can always immediately put it back by adding it to β). As usual, $\to_{\mathfrak{M}}^*$ is the reflexive–transitive closure of $\to_{\mathfrak{M}}$.

Finally, a word w is *accepted* by a PDA $\mathfrak{M}$, or, in other words, belongs to the *language defined by* $\mathfrak{M}$, if $\langle q_0, w, Z_0\rangle \to_{\mathfrak{M}}^* \langle q, \varepsilon, \varepsilon\rangle$ for some[4] state $q \in Q$.

[3]In our formalism the stack "grows" to the left.

[4]As one can notice, in this definition the PDA halts when the stack becomes empty rather than when a special "halting" state is reached.

Note that in general $\mathfrak{M}$ operates *nondeterministically*; i.e., it can be possible to apply several different transitions from one configuration. A word is accepted by the automaton if at least one sequence of transitions succeeds (i.e., does not terminate in the middle of the input word and ends by a configuration with nothing on the stack).

The further plan of building a context-free grammar for $\mathcal{L}(G)$ is as follows. First we transform the grammar $\mathbf{S}_m$ into a PDA that defines the same language $M = \mathcal{L}(\mathbf{S}_m)$. Next we build a PDA for $g^{-1}(M) \cap \{u\$ \mid u \in \Sigma_2^+\}$. After that we return to context-free grammars (transform this PDA into a context-free grammar). Finally, by applying the nonextending homomorphism h, we obtain a context-free grammar for $\mathcal{L}(G) = h(g^{-1}(M) \cap \{u\$ \mid u \in \Sigma_2^+\})$.

The constructions of automata and grammars used in our transformations were taken from [1, 6]. In these books one can find correctness proofs (i.e., proofs of the facts that these automata and grammars define the needed languages). Here we focus on estimating the size of the resulting context-free grammar.

To estimate the complexity (size) of a PDA $\mathfrak{M} = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z_0 \rangle$, we will use the following parameters: first, the number of transitions ($|\Delta| = \delta(\mathfrak{M})$); second, the number $d(\mathfrak{M}) = \max\{|\beta| \mid (p \xrightarrow{a,\, Z:\, \beta} q) \in \Delta\}$, which characterizes the maximum number of letters put onto the stack at one transition; third, the number of states ($|Q| = q(\mathfrak{M})$).

Lemma 3. *There exists a PDA $\mathfrak{M}_1$ that defines the language $M = \mathcal{L}(\mathbf{S}_m)$ (see [1, Lemma 2.24]). Moreover, $\delta(\mathfrak{M}_1) \le 2|\mathbf{S}_m|$, $q(\mathfrak{M}_1) = 1$, and $d(\mathfrak{M}_1) = 3$.*

Proof. Let N be the set of all nonterminal symbols of $\mathbf{S}_m$.

The automaton $\mathfrak{M}_1$ contains only one state q_0 and is equal to $\langle \{q_0\}, \Sigma_1, N \cup \Sigma_1, \Delta_1, q_0, S \rangle$. Here we take the union of the main and auxiliary alphabets of $\mathbf{S}_m$ as the stack alphabet, and the starting stack symbol is the start symbol of the grammar.

The set of transitions Δ_1 is built in the following way:

(1) for each rule $A \Rightarrow \alpha$ of $\mathbf{S}_m$ we add a transition $q_0 \xrightarrow{\varepsilon,\, A:\, \alpha} q_0$;

(2) for each $a \in \Sigma_1$ we add a transition $q_0 \xrightarrow{a,\, a:\, \varepsilon} q_0$.

The number of transitions of type (1) is equal to the number of rules in $\mathbf{S}_m$. The number of transitions of type (2) is equal to $|\Sigma_1|$. Both numbers are less than or equal to $|\mathbf{S}_m|$. Therefore, $\delta(\mathfrak{M}_1) \le 2|\mathbf{S}_m|$.

The equality $d(\mathfrak{M}_1) = 3$ is due to the fact that the right-hand sides of all rules in $\mathbf{S}_m$ contain at most three symbols. □

Lemma 4. *Let $\mathfrak{M}_1$ be a PDA for M as constructed in Lemma 3, and let $g\colon \Sigma_2 \cup \{\$\} \to \Sigma_1$ be a homomorphism such that $|g(a)| \le n$ for every $a \in \Sigma_2$. Then there exists a PDA $\mathfrak{M}_2$ that defines the language $g^{-1}(M) \cap \{u\$ \mid \Sigma_2^+\}$ (see [6, Theorem 7.30] and [1, Lemma 2.22, Theorem 2.26]). Moreover, $q(\mathfrak{M}_2) \le 2(n+1)|\Sigma_2| + 2$, $\delta(\mathfrak{M}_2) \le (2n+6)|\mathbf{S}_m| \cdot |\Sigma_2| + 2|\mathbf{S}_m| + 2|\Sigma_2| + 2$, and $d(\mathfrak{M}_2) = 3$.*

Proof. Let us build the PDA $\mathfrak{M}_2 = \langle Q_2, \Sigma_2 \cup \{\$\}, \Gamma_2, \Delta_2, q_0, S \rangle$ in the following way. Let $\Gamma_2 = N \cup \Sigma_1 \cup \{\#\}$. The new symbol $\#$ will play the role of a *stack bottom marker*, forbidding the automaton to halt too early, even if the stack is empty and the input word is read up to the end.

For Q_2 we take the set of all pairs of the form $\langle i, x \rangle$, where $i \in \{0, 1\}$ and $x \in \Sigma_1^*$ is a suffix of a word $g(a)$ for some letter $a \in \Sigma_2$ (in particular, x can be empty; the states $\langle 0, \varepsilon \rangle$ and $\langle 1, \varepsilon \rangle$ will play a special role in $\mathfrak{M}_2$), and two additional states q_0 (the initial one) and q_{F} (the final one).

Finally, the transitions of $\mathfrak{M}_2$ (elements of Δ_2) are as follows:

(1) $q_0 \xrightarrow{\varepsilon,\, S:\, S\#} \langle 0, x \rangle$;

(2) $\langle 0, \varepsilon \rangle \xrightarrow{a,\, X:\, X} \langle 0, g(a) \rangle$ for all $X \in \Gamma_2$ and $a \in \Sigma_2$;

(3) $\langle 0, \varepsilon\rangle \xrightarrow{a,\, X\colon X} \langle 1, g(a)\rangle$ for all $X \in \Gamma_2$ and $a \in \Sigma_2$;

(4) $\langle 1, \varepsilon\rangle \xrightarrow{\$,\, X\colon X} \langle 1, g(\$)\rangle$ for all $X \in \Gamma_2$;

(5) $\langle i, bv\rangle \xrightarrow{\varepsilon,\, X\colon \alpha} \langle i, v\rangle$ if $q_0 \xrightarrow{b,\, X\colon \alpha} q_0$ is a transition of $\mathfrak{M}_1$;

(6) $\langle 1, \varepsilon\rangle \xrightarrow{\varepsilon,\, \#\colon \varepsilon} q_{\mathrm{F}}$.

The workflow of $\mathfrak{M}_2$ can be informally interpreted as follows. Transition (1) puts a special symbol # to the bottom of the stack. Since the only way to remove it is transition (6), the automaton can finish its operation only in the state q_{F}. Next, by transition (2), the automaton reads the ongoing symbol a and stores the word $g(a)$ in the "internal memory" (the second component of the state). Several applications of transition (5) emulate the operation of $\mathfrak{M}_1$ on this word. After this emulation the "internal memory" is again empty, and $\mathfrak{M}_2$ is ready for reading the next letter of the input. The penultimate letter is read by transition (3), changing the first component of the state from 0 to 1. This guarantees that the word is nonempty and does not consist of only one symbol **\$**. After that we again apply transition (5) several times. Finally, transition (4) and several applications of transition (5) do the same job for **\$**. This ensures that this symbol appears only once and is located at the end of the word (i.e., this handles the intersection with $\{u\$ \mid u \in \Sigma_2^+\}$).

The correctness of this construction follows easily from the proofs of the propositions from [1, 6] mentioned in the statement of the lemma.

It is easy to see that $d(\mathfrak{M}_2) = d(\mathfrak{M}_1) = 3$. Let us estimate $\delta(\mathfrak{M}_2) = |\Delta_2|$. We have $|\Gamma_2| \cdot |\Sigma_2|$ transitions of type (2) and the same number of transitions of type (3). The number of transitions of type (4) is equal to $|\Gamma_2|$. Since the length of each word $g(a)$ ($a \in \Sigma_2$) is not greater than n, every such word has at most $n+1$ suffixes (from the empty word to the whole one). The total number of such suffixes is less than or equal to $(n+1)|\Sigma_2|$. Each transition of type (5) is defined by such a suffix and a transition of $\mathfrak{M}_1$; hence the number of such transitions is not greater than $(n+1)|\Sigma_2|\delta(\mathfrak{M}_1)$. Finally, transitions of types (1) and (6) are unique, so there are two of them.

Thus, the total number of transitions in $\mathfrak{M}_2$ is not greater than

$$2|\Gamma_2| \cdot |\Sigma_2| + |\Gamma_2| + (n+1)|\Sigma_2|\delta(\mathfrak{M}_1) + 2.$$

Recalling that $|\Gamma_2| = |N| + |\Sigma_1| + 1 \le 2|\mathbf{S}_m| + 1$ and $\delta(\mathfrak{M}_1) \le 2|\mathbf{S}_m|$, we get the required estimation $(2n+6)|\mathbf{S}_m| \cdot |\Sigma_2| + 2|\mathbf{S}_m| + 2|\Sigma_2| + 2$.

Finally, the estimation $q(\mathfrak{M}_2) = |Q_2| \le 2(n+1)|\Sigma_2| + 2$ also follows from the estimation of the number of suffixes of words of the form $g(a)$. $\square$

Lemma 5. *Let $\mathfrak{M}_2 = \langle Q_2, \Sigma_2 \cup \{\$\}, \Gamma_2, \Delta_2, q_0, S\rangle$ be a PDA that defines a language over the alphabet $\Sigma_2 \cup \{\$\}$. Then there exists a context-free grammar G_2 that defines the same language* (*see* [1, Lemma 2.26]). *Moreover, $|G_2| \le (d(\mathfrak{M}_2)+1)\delta(\mathfrak{M}_2)(q(\mathfrak{M}_2))^{d(\mathfrak{M}_2)} + q(\mathfrak{M}_2)$.*

Proof. The desired grammar $G_2 = \langle N_2, \Sigma_2, P_2, S_2\rangle$ is built as follows.

For nonterminal symbols of this grammar we use triples $\langle q, Z, r\rangle$, where $q, r \in Q_2$ and $Z \in \Gamma_2$, and also a special symbol S_2. Following [1], we denote $\langle q, Z, r\rangle$ by $[qZr]$ (note that this is *one* symbol of N_2).

The rules of G_2 (elements of P_2) are the following ones:

(1) $[qZr] \Rightarrow a[rX_1s_1][s_1X_2s_2]\ldots[s_{k-1}X_ks_k]$ if $q \xrightarrow{a,\, Z\colon X_1\ldots X_k} r$ is a transition of $\mathfrak{M}_2$, $a \in \Sigma_2 \cup \{\$, \varepsilon\}$, and $s_1, \ldots, s_k$ are arbitrary elements of Q_2; in particular, for a transition of the form $q \xrightarrow{a,\, Z\colon \varepsilon} r$ we add the rule $[qZr] \Rightarrow a$;

(2) $S_2 \Rightarrow [q_0Sq]$ for every $q \in Q_2$.

For each rule of type (1), we have $k \le d(\mathfrak{M}_2)$. Since every such rule corresponds to a transition of $\mathfrak{M}_2$ and k states $s_1, \ldots, s_k \in Q_2$, the total number of such rules is not greater than

$\delta(\mathfrak{M}_2)(q(\mathfrak{M}_2))^{d(\mathfrak{M}_2)}$, and the total length of the right-hand sides of these rules is less than or equal to $(d(\mathfrak{M}_2)+1)\delta(\mathfrak{M}_2)(q(\mathfrak{M}_2))^{d(\mathfrak{M}_2)}$.

The number of rules of type (2) is equal to $|Q_2| = q(\mathfrak{M}_2)$, and the right-hand side of every such rule contains only one letter. This yields the required inequality for $|G_2|$. $\square$

Let us simplify the estimation of $|G_2|$. By construction, $|\Sigma_2| \le |G|$, and for every $a \in \Sigma_2$ we have $|g(a)| \le |G|$, where G is the original L(\ )-grammar. Moreover, $|\mathbf{S}_m| \le 24|G|^3$. Let $n = |G|$. By Lemma 4, $d(\mathfrak{M}_2) = 3$, $q(\mathfrak{M}_2) \le 2(n+1)n+2 = 2n^2+2n+2$, and

$$\delta(\mathfrak{M}_2) \le (2n+6)\cdot 24n^3 \cdot n + 2\cdot 24n^3 + 2n + 2 = 48n^5 + 144n^4 + 48n^3 + 2n + 2.$$

Therefore,

$$|G_2| \le 4\big(48n^5+144n^4+48n^3+2n+2\big)(2n^2+2n+2)^3 + 2n^2+2n+2 = O(n^{11}).$$

Finally, the application of a nonextending homomorphism h to the right-hand sides of G_2 does not increase its size and gives a context-free grammar of size $O(|G|^{11})$ for the language $h(g^{-1}(M) \cap \{u\$ \mid u \in \Sigma_2^+\}) = \mathcal{L}(G)$. This completes the proof of Theorem 3.

6. CONCLUSION

Despite the fact that the full Lambek calculus L is NP-complete, and therefore Theorem 3 could hardly be generalized to all Lambek grammars, for grammars where all types have constantly bounded depth there exists a polynomial ($O(n^4)$) algorithm for checking whether a word belongs to the language defined by such a grammar [20]. Therefore, the question of translating a Lambek grammar with types of bounded depth into a context-free grammar of polynomial size is worth further investigation.

The polynomial ($O(n^3)$) algorithm for checking derivability in L(\ ) [23] is essentially the application of the standard dynamic programming technique (the Cocke–Younger–Kasami algorithm, CYK) to the grammar $\mathbf{S}_m$. Unfortunately, this grammar is ambiguous (for some words it yields more than one parsing tree), and so most of the efficient parsing algorithms are either inapplicable to this grammar or work slower than the CYK. Moreover, the size of $\mathbf{S}_m$ depends on the complexity of the original sequent; therefore, even Valiant's universal algorithm [29] on this grammar works slower than the CYK. Nevertheless, the presented interpretation of Savateev's criterion in terms of context-free rules could be useful in attempts of constructing a more efficient algorithm for checking derivability in L(\ ).

ACKNOWLEDGMENTS

The author is grateful to Prof. G. Morrill (Barcelona) and Prof. M.R. Pentus (Moscow) for fruitful discussions.

This work is supported by the Russian Science Foundation under grant 14-50-00005.

REFERENCES

1. A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation, and Compiling*, Vol. 1: *Parsing* (Prentice-Hall, Englewood Cliffs, NJ, 1972).
2. Y. Bar-Hillel, C. Gaifman, and E. Shamir, "On categorial and phrase-structure grammars," Bull. Res. Council Israel, Sect. F, **9F**, 1–16 (1960).
3. W. Buszkowski, "The equivalence of unidirectional Lambek categorial grammars and context-free grammars," Z. Math. Logik Grundlagen Math. **31**, 369–384 (1985).
4. N. Chomsky, "Three models for the description of language," IRE Trans. Inf. Theory **IT-2** (3), 113–124 (1956).
5. J. Evey, "Application of pushdown store machines," in *Proc. 1963 Fall Joint Comput. Conf.* (Spartan Books, Baltimore, MD, 1963), pp. 215–227.

6. J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 2nd ed. (Addison-Wesley, Reading, MA, 2001).
7. G. Jäger, "On the generative capacity of multi-modal categorial grammars," Res. Lang. Comput. **1** (1–2), 105–125 (2003).
8. M. Kanazawa, "The Lambek calculus enriched with additional connectives," J. Log. Lang. Inf. **1** (2), 141–171 (1992).
9. M. Kanazawa and S. Salvati, "The string-meaning relations definable by Lambek grammars and context-free grammars," in *Formal Grammar: Proc. 17th and 18th Int. Confs., FG 2012/2013*, Ed. by G. Morrill and M.-J. Nederhof (Springer, Berlin, 2013), Lect. Notes Comput. Sci. **8036**, pp. 191–208.
10. S. L. Kuznetsov, "On translating context-free grammars into Lambek grammars," Tr. Mat. Inst. im. V.A. Steklova, Ross. Akad. Nauk **290**, 72–79 (2015) [Proc. Steklov Inst. Math. **290**, 63–69 (2015)].
11. S. L. Kuznetsov and N. S. Ryzhkova, "A fragment of the Lambek calculus with iteration," in *Mal'tsev Meeting 2015: Coll. Abstr. Int. Conf.* (Novosibirsk, 2015), p. 213.
12. J. Lambek, "The mathematics of sentence structure," Am. Math. Mon. **65** (3), 154–170 (1958).
13. M. Moortgat, "Multimodal linguistic inference," J. Log. Lang. Inf. **5** (3–4), 349–385 (1996).
14. G. V. Morrill, *Categorial Grammar: Logical Syntax, Semantics, and Processing* (Oxford Univ. Press, Oxford, 2011).
15. A. G. Oettinger, "Automatic syntactic analysis and the pushdown store," in *Structure of Language and Its Mathematical Aspects* (Am. Math. Soc., Providence, RI, 1961), Proc. Symp. Appl. Math. **12**, pp. 104–129.
16. A. E. Pentus and M. R. Pentus, *Mathematical Theory of Formal Languages* (Binom, Moscow, 2006) [in Russian].
17. M. R. Pentus, "Lambek calculus and formal grammars," Fundam. Prikl. Mat. **1** (3), 729–751 (1995). Engl. transl. in *Provability, Complexity, Grammars* (Am. Math. Soc., Providence, RI, 1999), AMS Transl., Ser. 2, **192**, pp. 57–86.
18. M. R. Pentus, "Completeness of the Lambek syntactic calculus," Fundam. Prikl. Mat. **5** (1), 193–219 (1999); see also "Models for the Lambek calculus," Ann. Pure Appl. Log. **75** (1–2), 179–213 (1995).
19. M. Pentus, "Lambek calculus is NP-complete," Theor. Comput. Sci. **357**, 186–201 (2006).
20. M. Pentus, "A polynomial-time algorithm for Lambek grammars of bounded order," Linguist. Anal. **36**, 441–471 (2010).
21. V. V. Podolskii, "Circuit complexity meets ontology-based data access," in *Computer Science—Theory and Applications: Proc. 10th Int. Comput. Sci. Symp. Russia, 2015* (Springer, Cham, 2015), Lect. Notes Comput. Sci. **9139**, pp. 7–26.
22. Yu. Savateev, "Lambek grammars with one division are decidable in polynomial time," in *Computer Science—Theory and Applications: Proc. 3rd Int. Comput. Sci. Symp. Russia, 2008* (Springer, Berlin, 2008), Lect. Notes Comput. Sci. **5010**, pp. 273–282.
23. Yu. V. Savateev, "Recogintion of derivability for the Lambek calculus with one division," Vestn. Mosk. Univ., Ser. 1: Mat. Mekh., No. 2, 59–62 (2009) [Moscow Univ. Math. Bull. **64** (2), 73–75 (2009)].
24. Yu. V. Savateev, "Algorithmic complexity of fragments of the Lambek calculus," Cand. Sci. (Phys.–Math.) Dissertation (Moscow State Univ., Moscow, 2009).
25. Yu. Savateev, "Product-free Lambek calculus is NP-complete," Ann. Pure Appl. Log. **163** (7), 775–788 (2012).
26. D. S. Shamkanov, "Circular proofs for the Gödel–Löb provability logic," Mat. Zametki **96** (4), 609–622 (2014) [Math. Notes **96**, 575–585 (2014)].
27. D. Shamkanov, "Nested sequents for provability logic GLP," Log. J. IGPL **23** (5), 789–815 (2015).
28. M. P. Schützenberger, "On context-free languages and push-down automata," Inf. Control **6** (3), 246–264 (1963).
29. L. G. Valiant, "General context-free recognition in less than cubic time," J. Comput. Syst. Sci. **10**, 308–315 (1975).

Translated by the author