Categorial Grammars Based on Variants of the Lambek Calculus

Candidate of Science (Ph. D.) Thesis Summary

by Stepan Lvovich Kuznetsov,

Department of Mathematical Logic and Theory of Algorithms,
Faculty of Mathematics and Mechanics, Moscow State University

Defended on June 8, 2012 at Moscow State University (dissertation council No. D 501.001.84).
Confirmed by the Ministry of Education and Science of Russia on March 11, 2013 (order No. 130/nk).
Supervisor: Prof., Dr. Sc. M. R. Pentus.
Official opponents:
Dr. Sc. I. G. Lysionok (Steklov Mathematical Institute, RAS);
C. Sc. (Ph. D.) A. V. Kudinov (Institute for Information Transmission Problems, RAS);
Tver State University.

Historical Background

The Lambek calculus L was introduced by J. Lambek [13] for defining natural language syntax using categorial grammars [17][16]. This calculus uses syntactic types built from primitive ones using three binary connectives: multiplication, left and right division.

Chomsky [6] suggested another family of grammars, called the Chomsky hierarchy. The most well-known class grammars from this hierarchy is the class of context-free grammars. Context-free grammars are widely used for parsing artificial languages (e.g., programming languages [1]), but for natural languages categorial grammars have significant advantages. In particular, they enjoy the lexicalisation property: all the information about syntax is kept in the categorial dictionary, and the analyser needs only that part of the dictionary, which is relevant to the text being parsed. Categorial grammars also allow to do semantic analysis, e.g., using Montague semantics [5].

We focus on comparing classes of languages generated by different classes of grammars just as sets of words, without syntactic structure. In this (so called *weak*) sense Lambek grammars are equivalent to context-free ones: the class of languages generated by grammars, based on L, coincides with the class of all context-free languages without the empty word [18]. Similar questions can be posed for grammars based on variants of the Lambek calculus (its fragments and extensions). It is known [4] that all context-free languages without the empty word can be generated by grammars based on the fragment of the Lambek calculus with only one division. Kanazawa [12] considered languages generated by grammars based on the Lambek calculus with additive conjunction and disjunction. This class of languages strictly contains finite intersections of context-free languages and lies inside the class of all context-sensitive languages. Moortgat [15] enriched the Lambek calculus with two modalities; as shown by Jäger [11], this extension does not enlarge the class of languages. Dikovsky and Dekhtyar [7] considered categorial dependency grammars (CDGs) based on a fragment of the Lambek calculus without multiplication, but with additional connectives for nonlocal dependencies, with a restriction: all the denominators are primitive types. Languages generated by CDGs form a special class that strictly contains the class context-free languages and is closed under finite unions, intersections with regular languages, and taking image or preimage of a homomorphism. As shown by Buszkowski [3], any recursively enumerable language can be generated by a grammar based on an extension of the Lambek calculus with a finite set of axioms.

The Lambek calculus is complete with respect to models interpreting Lambek types as formal languages over an alphabet (connectives of L correspond to multiplication, left and right divisions of languages) [20]. Such models are called L-models.

Derivability problems for L and its fragments $\mathrm{L}(\backslash, /)$ and $\mathrm{L}(\cdot, \backslash)$ are NP-complete as shown by Pentus [21] for L and by Savateev [23] for $\mathrm{L}(\backslash, /)$ and $\mathrm{L}(\cdot, \backslash)$. On the other hand, if the complexity of types is bounded (this is the usual case in practice), then there are polynomial ($O(n^5)$) time algorithms. These algorithms were independently presented by Pentus [22] and Fowler [9]. Fowler used such algorithms for parsing English sentences from CCGBank [8]. Derivability problem for $\mathrm{L}(\backslash)$ is decideable in polynomial time [23].

Main Results

1. The class of languages generated by $\mathrm{L}_{\mathbf{1}}$-grammars coincides with the class of all context-free languages.
2. The derivability problem for $\mathrm{L}_{\mathbf{1}}(\backslash)$ is decidable in polynomial time.
3. The class of languages generated by $\mathrm{L}(\backslash; p_1)$-grammars coincides with the class of all context-free languages without the empty word.
4. The derivability problem for $\mathrm{L}(\cdot, \backslash; p_1)$ is NP-complete.
5. We present a calculus L^{R} for the unary operation of language reversal and prove its completeness with respect to models on subsets of free semigroups (language models); the class of languages generated by L^{R}-grammars coincides with the class of all context-free languages without the empty word.

Extended Abstract

In **Chapter 1** we define notions to be used further and state known results.

Define the Lambek calculus L. A countable set $\mathrm{Pr} = \{p_1, p_2, p_3, \dots\}$ is called the set of *primitive types;* p_1 is also denoted by p. *Types* of the Lambek calculus are defined recursively:
(1) every primitive type is a type;
(2) if A and B are types, then $(A \backslash B)$, $(B \mathbin{/} A)$, and $(A \cdot B)$ are also types.
The set of all types is denoted by Tp. For types we use capital Latin letters, and capital Greek ones denote finite (possibly empty) sequences of types. A *sequent* is an expression of the form $\Pi \to C$. Sequents are the objects to be derived in L.

The axioms and rules of L are as follows: $p_i \to p_i$ (ax);

$$\frac{A\,\Pi \to B}{\Pi \to A \backslash B}\ (\to \backslash),\ \text{where } \Pi \text{ is not empty;} \qquad \frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma\, \Pi\, (A \backslash B)\, \Delta \to C}\ (\backslash \to);$$

$$\frac{\Pi\, A \to B}{\Pi \to B \mathbin{/} A}\ (\to /),\ \text{where } \Pi \text{ is not empty;} \qquad \frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma\, (B \mathbin{/} A)\, \Pi\, \Delta \to C}\ (/ \to);$$

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

$$\frac{\Pi \to A \quad \Gamma\, A\, \Delta \to C}{\Gamma\, \Pi\, \Delta \to C}\ (\text{cut}).$$

The set $\mathrm{Tp}(\backslash)$ is the set of types without $\cdot$ and $/$. The calculus with axioms (ax) and rules $(\backslash \to)$ and $(\to \backslash)$ is called $\mathrm{L}(\backslash)$. (The $\backslash$ connective can be replaced by $/$, and the resulting theory will be symmetric to the $\backslash$ one). Calculi $\mathrm{L}(\cdot, \backslash)$ and $\mathrm{L}(\backslash, /)$ are defined in a similar way.

$\mathrm{Tp}(p_1, \dots, p_N)$ stands for the set of types that use only primitive types from $\{p_1, \dots, p_N\}$. Restricting the Lambek calculus to this set of types yields the calculus $\mathrm{L}(p_1, \dots, p_N)$; its fragments with restricted sets of connectives are denoted by $\mathrm{L}(\backslash; p_1, \dots, p_N)$, $\mathrm{L}(\cdot, \backslash; p_1, \dots, p_N)$, $\mathrm{L}(\backslash, /; p_1, \dots, p_N)$.

The calculus L^* is obtained from L by removing the $\Pi \neq \Lambda$ constraint of the $(\to \backslash)$ and $(\to /)$ rules. Adding a constant $\mathbf{1}$ as a special primitive type gives the new set of types $\mathrm{Tp}_{\mathbf{1}}$. The calculus $\mathrm{L}_{\mathbf{1}}$ is obtained by adding a new axiom $\to \mathbf{1}$ ($(\to \mathbf{1})$) and a new rule

$$\frac{\Gamma\, \Delta \to A}{\Gamma\, \mathbf{1}\, \Delta \to A}\ (\mathbf{1} \to);$$

to L^*. For the set of types we now use $\mathrm{Tp}_{\mathbf{1}}$.

The substitution of type A for z ($z \in \mathrm{Pr} \cup \{\mathbf{1}\}$) into $\mathscr{A}$ is denoted by $\mathscr{A}[z := A]$. Here $\mathscr{A}$ can be any syntactic object: a type, a sequence of types, a sequent, or (see further) a categorial grammar. The expression $\mathscr{A}[z_1 := A_1, z_2 := A_2, \dots]$ (or $\mathscr{A}[z_i := A_i]$) means that all substitutions are performed simultaneously.

Now we define the central notion of the thesis.

Definition. Let $\mathscr{L}$ be one of the variants of the Lambek calculus (L, $\mathrm{L}_{\mathbf{1}}$, L^{R} (defined further), or a fragment of one of those calculi) and $\mathrm{Tp}_{\mathscr{L}}$ be the corresponding set of types. A *grammar based on* $\mathscr{L}$ (an *$\mathscr{L}$-grammar*) is a triple $\mathcal{G} = \langle \Sigma, H, \rhd \rangle$, where Σ is a non-empty finite set (alphabet), $H \in \mathrm{Tp}_{\mathscr{L}}$, and $\rhd \subset \mathrm{Tp}_{\mathscr{L}} \times \Sigma$ is a finite correspondence between types and elements of Σ. The language generated by $\mathcal{G}$ is $\mathfrak{L}(\mathcal{G}) = \{a_1 \dots a_k \in \Sigma^* \mid \exists B_1, \dots, B_k \colon B_i \rhd a_i \text{ and } \mathscr{L} \vdash B_1 \dots B_k \to H\}$. Such languages are called *$\mathscr{L}$-languages.*

Theorem 2–3 [1] (C. Gaifman, 1960; W. Buzkowski, 1985; M. Pentus, 1993)**.** *The class of* L*-languages coincides with the class of all context-free languages without the empty word.* [2][4][18]

Introduce the calculi MCLL$'$, MCLL, and MCLL$_{\mathbf{1},\perp}$, that are variants of multiplicative linear logic.

The set $\mathrm{Var} = \{p_1, p_2, \dots\}$ is now called the set of *variables;* $\mathrm{At} = \mathrm{Var} \cup \{\bar{q} \mid q \in \mathrm{Var}\} \cup \{\mathbf{1}, \perp\}$ is the set of *atoms. Formulae* of MCLL$_{\mathbf{1},\perp}$ are built from atoms using two binary connectives $⅋$ and $\otimes$. Sequents of MCLL$_{\mathbf{1},\perp}$ are of the form $\to \Gamma$, where Γ is a finite non-empty sequence of formulae. *Negation* is defined externally: $p_i^\perp = \bar{p}_i$, $\bar{p}_i^\perp = p_i$, $\mathbf{1}^\perp = \perp$, $\perp^\perp = 1$, $(A ⅋ B)^\perp = B^\perp \otimes A^\perp$, $(A \otimes B)^\perp = B^\perp ⅋ A^\perp$. Sequents $\to p_i\, \bar{p}_i$ and $\to \mathbf{1}$ are axioms of MCLL$_{\mathbf{1},\perp}$. The rules are as follows:

$$\frac{\to \Gamma\, A\, B\, \Delta}{\to \Gamma\, (A ⅋ B)\, \Delta}\ (\to ⅋); \quad \frac{\to \Gamma\, A \quad \to B\, \Delta}{\to \Gamma\, (A \otimes B)\, \Delta}\ (\to \otimes);$$

$$\frac{\to \Gamma\, \Delta}{\to \Gamma \perp \Delta}\ (\to \perp); \quad \frac{\to \Gamma\, \Delta}{\to \Delta\, \Gamma}\ (\mathrm{rot}).$$

The *standard translation* of Lambek types into MCLL$_{\mathbf{1},\perp}$ formulae is defined recursively: $\widehat{p_i} = p_i$ $(p_i \in \mathrm{Pr})$, $\widehat{\mathbf{1}} = \mathbf{1}$, $\widehat{A \cdot B} = \widehat{A} \otimes \widehat{B}$, $\widehat{A \setminus B} = \widehat{A}^\perp ⅋ \widehat{B}$, $\widehat{B\,/\,A} = \widehat{B} ⅋ \widehat{A}^\perp$. If $B_1, \dots, B_m, C \in \mathrm{Tp}_{\mathbf{1}}$, then $\mathrm{L}_{\mathbf{1}} \vdash B_1 \dots B_m \to C \iff \mathrm{MCLL}_{\mathbf{1},\perp} \vdash \to \widehat{B}_m^\perp \dots \widehat{B}_1^\perp \widehat{C}$. The fragment of MCLL$_{\mathbf{1},\perp}$ without constants $\mathbf{1}$ and $\perp$ is called MCLL and corresponds to L*. The original Lambek calculus L corresponds to the calculus MCLL$'$, obtained from MCLL by adding the restriction $\Gamma\Delta \neq \Lambda$ to the $(\to ⅋)$ rule.

We use Pentus-style *proof nets* to study derivability in MCLL and MCLL$'$, and, using the standard translation, in L* and L. Let $\Gamma = A_1 \dots A_m$. Write it as $\diamond A_1 \diamond \dots \diamond A_m$ and enumerate all the symbols in this string except brackets (an atom is counted as one symbol). Denote the set of pairs $\langle \text{symbol}, \text{number} \rangle$, called *occurrences,* by Ω_Γ. For $\alpha, \beta \in \Omega_\Gamma$ let $\alpha <_\Gamma \beta$ if the number of α is less than the number of β. The $\prec_\Gamma$ relation is defined as the transitive closure of parse trees of the formulae $A_1, \dots, A_m$ (if α is in the subtree of a tree in which β is the root, then $\alpha \prec_\Gamma \beta$). A occurrence of a subformula is the corresponding subset of Ω_Γ. If X is such an occurrence, then $\mathrm{l}(X)$ is the occurrence of the nearest connective or $\diamond$ to the left of X, and $\mathrm{r}(X)$ is the occurrence of the nearest connective of $\diamond$ to the right of X (if there is no such occurrence, then $\mathrm{r}(X) = \langle \diamond, 1 \rangle$). The set of all $⅋$ (resp., $\otimes$, $\diamond$) occurrences is denoted by $\Omega_\Gamma^{⅋}$ (resp., $\Omega_\Gamma^{\otimes}$, $\Omega_\Gamma^{\diamond}$); occurrences of atoms of the form p_i (resp., $\bar{p}_i$) form the set $\Omega_\Gamma^{\mathrm{At}^+}$ (resp., $\Omega_\Gamma^{\mathrm{At}^-}$); $\Omega_\Gamma^{⅋\diamond} = \Omega_\Gamma^{⅋} \cup \Omega_\Gamma^{\diamond}$.

Definition. A *proof net* is a structure $\mathfrak{N} = \langle \langle \Omega_\Gamma, <_\Gamma, \prec_\Gamma \rangle, \mathcal{A}, \mathcal{E} \rangle$, where $\mathcal{A}, \mathcal{E} \subset \Omega_\Gamma \times \Omega_\Gamma$, which satisfies the following axioms:
(1) $|\Omega_\Gamma^{⅋\diamond}| - |\Omega_\Gamma^{\otimes}| = 2$;
(2) $\mathcal{A}$ is a total function from $\Omega_\Gamma^{\otimes}$ to $\Omega_\Gamma^{⅋\diamond}$;
(3) $\mathcal{E}$ is a bijective function from $\Omega_\Gamma^{\mathrm{At}^+}$ to $\Omega_\Gamma^{\mathrm{At}^-}$, and if α is an occurrence of p_i, then $\mathcal{E}(\alpha)$ is an occurrence of $\bar{p}_i$;
(4) the graph $\langle \Omega_\Gamma, \mathcal{A} \cup \mathcal{E} \rangle$ can be drawn in the upper semiplane without intersections with its vertices lying on the bound of the semiplane in the $<_\Gamma$ order;
(5) the graph $\langle \Omega_\Gamma, \mathcal{A} \cup \prec_\Gamma \rangle$ is acyclic.

Definition. A proof net is called *strong,* if it satisfies one more axiom:
(6) for any occurrence $X \subset \Omega_\Gamma$ of a subformula $\widetilde{\mathcal{A}}(\mathrm{l}(X)) \neq \widetilde{\mathcal{A}}(\mathrm{r}(X))$, where

$$\widetilde{\mathcal{A}}(\alpha) = \begin{cases} \alpha, & \text{if } \alpha \in \Omega_\Gamma^{⅋\diamond}, \\ \mathcal{A}(\alpha), & \text{if } \alpha \in \Omega_\Gamma^{\otimes}. \end{cases}$$

Theorem 13 (M. Pentus, 1996)**.** *A sequent* $\to \Gamma$ *is provable in* MCLL *if and only if there exists a proof net for* Γ*.* [19]

Theorem 14. *A sequent* $\to \Gamma$ *is provable in* MCLL$'$ *if and only if there exists a strong proof net for* Γ*.* [25]

Next, we present a method of building an L*-grammar for a context-free language M containing the empty word [25]. Consider the language $M - \{\epsilon\}$. Due to Theorem 2–3 there exists an L-grammar $\mathcal{G} = \langle \Sigma, \rhd, H \rangle$. In $\mathcal{G}$, the type H is primitive (let $H = p_1$) and all the types occurring in $\rhd$, are of one of the forms p_i, $p_j \setminus p_i$, or $p_k \setminus (p_j \setminus p_i)$, $k, j \neq 1$. If we consider $\mathcal{G}$ an L*-grammar, it will still generate the same language. Perform the following substitution in $\mathcal{G}$: substitute $\quad \mathcal{G}\ D = \bigl((r \setminus r) \setminus \bigl((s \setminus s) \setminus q\bigr)\bigr) \setminus q$ for p_1. By means of proof nets one can prove that this grammar generates the language $\mathfrak{L}(\mathcal{G}) \cup \{\varepsilon\} = M$. Thus we have established that any context-free language is an L*-language. This result is very natural in view of Theorem 2–3, but yet has not been proven before.)

In **Chapter 2** we consider the Lambek calculus with the unit and its fragments.

We introduce a substitution which reduces derivability in L$_{\mathbf{1}}$ to derivability in L*:

Theorem 16. *If* $\Pi \to C$ *is a sequent with types from* $\mathrm{Tp}_{\mathbf{1}}$ *and* q *is a new primitive type, then*

$$\mathrm{L}_{\mathbf{1}} \vdash \Pi \to C \iff \mathrm{L}^* \vdash \bigl(\Pi \to C\bigr)[p_i := (\mathbf{1} \cdot p_i) \cdot \mathbf{1}][\mathbf{1} := q \setminus q].$$

This theorem yields a characterisation of L$_{\mathbf{1}}$-languages:

[1]We keep theorem numeration from the original Russian text of the thesis. Theorems 2 and 3 are here formulated as one theorem.

Theorem 17. *Every* $\mathrm{L}_{\mathbf{1}}$*-language is context-free.*

Next, we introduce another substitution to reduce $\mathrm{L}_{\mathbf{1}}(\backslash)$ to $\mathrm{L}^*(\backslash)$:

Theorem 18. *If* $\Pi \to C$ *is a sequent with types from* $\mathrm{Tp}_{\mathbf{1}}(\backslash)$ *and* q *is a new primitive type, then*

$$\mathrm{L}_{\mathbf{1}}(\backslash) \vdash \Pi \to C \iff \mathrm{L}^*(\backslash) \vdash \big(\Pi \to C\big)[\mathbf{1} := q \,\backslash\, q,\ p_i := (q \,\backslash\, q) \,\backslash (p_i \,\backslash\, q)].$$

[26]

Using this substitution and Savateev's results, we prove decideability of $\mathrm{L}_{\mathbf{1}}(\backslash)$ in polynomial time.

Theorem 19. *There exists an algorithm, that checks derivability of sequents in* $\mathrm{L}_{\mathbf{1}}(\backslash)$*, in* $O(n^3)$ *steps, where* n *is the length of the sequent (the number of connective, primitive type and* $\mathbf{1}$ *occurrences).*

In **Chapter 3** we consider the Lambek calculus with one primitive type ($\mathrm{L}(p)$) and its fragments ($\mathrm{L}(\backslash; p)$ and others).

We introduce a type substitution that reduces derivability in $\mathrm{L}(\backslash)$ to derivability in $\mathrm{L}(\backslash; p)$. Earlier a similar substitution for the whole MCLL calculus was presented by Métayer [14]. For $\mathrm{L}(\backslash)$ and $\mathrm{L}(\backslash; p)$ there exists a substitution invented by Hendriks [10], but the types there are of exponential size and the proof is more complicated than ours. The next theorem is proved using proof nets.

Theorem 20. *For every sequent* $\Pi \to C$*, where* $\Pi = B_1 \dots B_m$*,* $m \geqslant 1$ *, and* $B_1, \dots, B_m, C \in \mathrm{Tp}(\backslash; p_1, \dots, p_N)$*,*

$$\mathrm{L}(\backslash; p_1, \dots, p_N) \vdash \Pi \to C \iff \mathrm{L}(\backslash; p) \vdash (\Pi \to C)[p_1 := A_1, \dots, p_N := A_N],$$

$$\textit{where } A_k = \big(p^{k+1} \cdot \big(((p \cdot p) \,\backslash\, p) \,\backslash\, p\big) \cdot p^{N-k+1}\big) \,\backslash\, p, \qquad k = 1, \dots, N.$$

[25]

The notation $(D_1 \cdot \ldots \cdot D_m) \,\backslash\, C$ here is a shortcut for $D_m \,\backslash (D_{m-1} \,\backslash (D_{m-2} \,\backslash \dots \backslash (D_1 \,\backslash\, C) \dots))$; p^m means $p \cdot \ldots \cdot p$ (m times).

The same construction works for L^* [24]. This substitution depends on the set of primitive types used in the sequent; there also exists a uniform substitution, but with exponential growth of type complexity.

Theorem 24. *The class of all* $\mathrm{L}(\backslash; p)$*-languages coincides with the class of all context-free languages without the empty word. The class of all* $\mathrm{L}^*(\backslash; p)$*-languages coincides with the class of all context-free languages.*

Theorem 25. *Derivability problems in* $\mathrm{L}(\cdot, \backslash; p)$ *and* $\mathrm{L}^*(\cdot, \backslash; p)$ *are NP-complete.*

In **Chapter 4** we introduce the Lambek calculus with the reversal operation (L^{R}).

Define the notion of L-model (language model). Now let Σ be an arbitrary non-empty finite *or countable* set. Formal languages over Σ not containing the empty word form the set $\mathcal{P}(\Sigma)$ with the following three operations (multiplication, left and right division): $M \cdot N = \{uv \mid u \in M, v \in N\}$; $M \,\backslash\, N = \{u \in \Sigma^+ \mid (\forall v \in M)\, vu \in N\}$; $N \,/\, M = \{u \in \Sigma^+ \mid (\forall v \in M)\, uv \in N\}$.

Definition. An *L-model* is a structure $\mathcal{M} = \langle \Sigma, w \rangle$, where w is a mapping from Tp to $\mathcal{P}(\Sigma^+)$, such that $w(A \cdot B) = w(A) \cdot w(B)$, $w(A \,\backslash\, B) = w(A) \,\backslash\, w(B)$, $w(B \,/\, A) = w(B) \,/\, w(A)$ for any two types A and B. A sequent $F_1 \dots F_m \to G$ is true in $\mathcal{M}$ if $w(F_1) \cdot \ldots \cdot w(F_m) \subseteq w(G)$.

Theorem 27 (M. Pentus, 1995)**.** *A sequent is derivable in* L *if and only if it is true in all L-models.* [20]

Introduce the calculus L^{R}. Types of L^{R} are built from primitive types using three binary connectives $\backslash$, $/$, $\cdot$ (as in L) and one unary connective $^{\mathrm{R}}$ (written in postfix form, A^{R}). Additional rules of inference are as follows:

$$\frac{\Gamma \to C}{\Gamma^{\mathrm{R}} \to C^{\mathrm{R}}}\ ({}^{\mathrm{R}} \to {}^{\mathrm{R}}) \qquad \frac{\Gamma\, A\, \Delta \to C}{\Gamma\, A^{\mathrm{RR}}\, \Delta \to C}\ ({}^{\mathrm{RR}} \to) \qquad \frac{\Gamma \to C}{\Gamma \to C^{\mathrm{RR}}}\ (\to {}^{\mathrm{RR}})$$

The notion of L-model is naturally extended to L^{R} by adding the condition $w(A^{\mathrm{R}}) = w(A)^{\mathrm{R}} = \{a_n \dots a_1 \mid a_1 \dots a_n \in w(A)$ to the definition of $w\colon \mathrm{Tp}^{\mathrm{R}} \to \mathcal{P}(\Sigma^+)$.

Theorem 28. *A sequent with types from* Tp^{R} *is derivable in* L^{R} *if and only if it is true in all L-models.* [27]

This completeness theorem shows that the extension is chosen correctly.

Any type from Tp^{R} is equivalent to a type in normal form, i.e. a type where the $^{\mathrm{R}}$ connective is applied only to primitive types.

Now we prove Theorem 28. Let a sequent whose types are in normal form be not derivable in L^{R}. Then is is not derivable without applying rules for $^{\mathrm{R}}$, therefore by Theorem 27 there exists a countermodel $\mathcal{M}$ for this sequent, but this model can violate the condition $w(A^{\mathrm{R}}) = w(A)^{\mathrm{R}}$ for some types A. We construct a countermodel $\mathcal{M}_2$ satisfying this condition for every A.

Let Φ be the set of all subtypes of the sequent in question. Define recursively a counter $f(A)$, $A \in \Phi$: $f(p_i) = f(p_i^{\mathrm{R}}) = 1$, $f(A \cdot B) = f(A) + f(B) + 10$, $f(A \,\backslash\, B) = f(B \,/\, A) = f(B)$; let $K = \max\{f(A) \mid A \in \Phi\}$, $N = 2K + 25$. Introduce a new alfabet $\Sigma_1 = \Sigma \times \{1, \dots, N\}$ and consider the homomorphism $h\colon \Sigma^+ \to \Sigma_1^+$, $a \mapsto \langle a, 1 \rangle \dots \langle a, N \rangle$. Let $T = \{u \in \Sigma_1^+ \mid u$ is not a subword of a word of the form $h(v)$ or $h(v)^{\mathrm{R}}$ ($v \in \Sigma^+$) or u starts or ends with a symbol of the form $\langle a, 2k\rangle\}$. Put $w_2(p_i) = h(w(p_i)) \cup h(w(p_i^{\mathrm{R}}))^{\mathrm{R}} \cup T$ and propagate the w_2 mapping to all types from Tp^{R} according to the definition of L-model for L^{R}. The model $\mathcal{M}_2 = \langle \Sigma_1, w_2\rangle$ is the one we need.

Finally, we state some properties of L^{R} and its fragments:

Theorem 29. *The class of all L^{R}-languages coincides with the class of all context-free languages without the empty word.*

Theorem 30. *Derivability problems for calculi $\mathrm{L}^{\mathrm{R}}(\backslash)$, L^{R}, and all intermediate fragments of L^{R} are NP-complete.*

Acknowledgements

The author thanks Prof. Mati Pentus for guiding him into the subject and constant attention to his work. He is also grateful to the members of the Department of Mathematical Logic and Theory of Algorithms, Moscow State University for the friendly atmosphere and constructive discussions.

References

[1] A. V. Aho, R. Sethi, J. D. Ullman. Compilers: principles, techniques and tools. — Reading, Mass.: Addison-Wesley, 1985.

[2] Y. Bar-Hillel, C. Gaifman, E. Shamir. On the categorial and phrase-structure grammars. *Bulletin of the Research Council of Israel, Section F.* **9F**, 1960. — P. 1–16.

[3] W. Buszkowski. Some decision problems in the theory of syntactic categories. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik.* **28**, 1982, 539–548.

[4] W. Buszkowski. The equivalence of unidirectional Lambek categorial grammars and context-free grammars. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik.* **31**, 1985, 369–384.

[5] B. Carpenter. Type-logical semantics. — Cambridge, Mass.: The MIT Press, 1997.

[6] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory.* **I T-2**, No. 3, 1956, 113–124.

[7] M. Dekhtyar, A. Dikovsky. Generalized categorial dependency grammars. *Trakhtenbrot/Festschrift,* ed. by A. Avron et al. *Lecture Notes in Computer Science,* Vol. **4800**. Berlin etc.: Springer, 2008, 230–255.

[8] T. A. D. Fowler. Parsing CCGBank with the Lambek calculus. *Parsing with Categorial Grammar Workshop, ESSLLI 2009.* Bordeaux, 2009, 38–42.

[9] T. A. D. Fowler. A polynomial time algorithm for parsing with the bounded order Lambek grammars. *The Mathematics of Language: proceedings of the 10-th and 11-th Biennial Conference, MOL 2010 and MOL 2011,* ed. by Ch. Ebert, G. Jäger and J. Michaelis. *Lecture Notes in Computer Science,* Vol. **6149**. Berlin etc.: Springer, 2009, 36–43.

[10] H. Hendriks. Studied flexibility. *ILLC Dissertation Series,* DS-1993-05, Amsterdam, 1993.

[11] G. Jäger. On the generative capacity of multi-modal categorial grammars. *Research on Language and Computation,* **1**, No. 1–2, 2003, 105–125.

[12] M. Kanazawa. The Lambek calculus enriched with additional connectives. *Journal of Logic, Language and Information,* **1**, 1992, 141–171.

[13] J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly.* **65**, No. 3, 1958, 154–170.

[14] F. Métayer. Polynomial equivalence among systems LLNC, LLNC_a and LLNC_0. *Theoretical Computer Science.* **227**, No. 1, 1999. — P. 221–229.

[15] M. Moortgat. Multimodal linguistic inference. *Journal of Logic, Language and Information,* **5**, No. 3–4, 1996, 349–385.

[16] M. Moortgat. Categorial type logic. *Handbook of Logic and Language,* ed. by J. van Benthem and A. ter Meulen. Elsevier, 1997.

[17] G. Morrill. Categorial grammar: logical syntax, semantics and processing. Oxford: Oxford University Press, 2011.

[18] M. R. Pentus. Lambek calculus and formal grammars (in Russian). *Fundamental and Applied Mathematics*, **1**, No. 3, 1995, 729–751.

[19] M. Pentus. Free monoid completeness of the Lambek calculus allowing empty premises. *Proceedings of Logic Colloquium '96,* ed. by J. M. Larrazabal, D. Lascar and G. Mints. *Lecture Notes in Logic,* Vol. **12**. Berlin etc.: Springer, 1998. — P. 171–209.

[20] M. R. Pentus. Completeness of the Lambek syntactic calculus (in Russian). *Fundamental and Applied Mathematics*, **5**, No. 1, 1999, 193–219.

[21] M. Pentus. Lambek calculus is NP-complete. *Theoretical Computer Science,* **357**, No. 1–3, 2006, 186–201.

[22] M. Pentus. A polynomial-time algorithm for Lambek grammars of bounded order. *Linguistic Analysis.* **36**, No. 1–4, 2010, 441–471.

[23] Yu. V. Savateev. Algorithmic complexity of fragments of the Lambek calculus (in Russian). Ph. D. Thesis, Moscow State University, 2009.

Author's Papers

[24] S. L. Kuznetsov. Lambek calculus with one division and one primitive type permitting empty antecedents. *Moscow University Math. Bulletin.* **64**, No. 2, 2009. — P. 76–79.

[25] S. Kuznetsov. Lambek grammars with one division and one primitive type. *Logic Journal of the IGPL.* **20**, No. 1, 2012. — P. 207–221.

[26] S. L. Kuznetsov. Lambek calculus with a unit and one division. *Moscow University Math. Bulletin.* **66**, No. 4, 2011. — P. 173–175.

[27] S. L. Kuznetsov. Lambek calculus with the reversal operation (in Russian). VINITI preprint No. 152-V2012, 17.04.2012.
Revised English version: S. Kuznetsov. Lambek calculus with the reversal operation. *Logical Aspects of Computational Linguistics,* ed. by D. Béchet and A. Dikovsky, *Lecture Notes in Computer Science* **7351**, 2012, P. 151–160.