ISSN 0081-5438, Proceedings of the Steklov Institute of Mathematics, 2015, Vol. 290, pp. 63–69. © Pleiades Publishing, Ltd., 2015.
Published in Russian in Trudy Matematicheskogo Instituta imeni V.A. Steklova, 2015, Vol. 290, pp. 72–79.

On Translating Context-Free Grammars into Lambek Grammars

S. L. Kuznetsov [a]

Received March 15, 2015

Abstract—We consider context-free grammars and Lambek grammars enriched with semantic labeling. Such grammars do not just answer whether a given word belongs to the language described by the grammar, but, if the answer is positive, also assign the word a λ-term that corresponds to the semantic value ("meaning") of the word. We present a modification of W. Buszkowski's direct translation of context-free grammars in the Chomsky normal form into Lambek grammars; this modification preserves semantic values of words.

DOI: 10.1134/S0081543815060061

1. FORMAL LANGUAGES AND CATEGORIAL GRAMMARS

A *formal language* (further we will omit the word "formal") is an arbitrary set of words built from elements (*letters*) of a finite set Σ (called the *alphabet*). Languages are usually infinite, and in order to describe them finitely one uses various *grammars*, including *categorial grammars*. A categorial grammar is a correspondence between letters of the alphabet and logical expressions called *syntactic types*. To each letter the grammar assigns a finite number of types (maybe more than one). A word $w = a_1 \dots a_n$ belongs to the language defined by such a grammar if and only if there exist types $A_1, \dots, A_n$ such that A_i is assigned to a_i (for all i from 1 to n) and the *sequent* $A_1 \dots A_n \to H$ is derivable by means of a special logical calculus. Here H is some fixed type, one for the whole grammar. In this case we say that w is *accepted* or *generated* by the given grammar.

Note some difference in terminology between studies of "formal" and "real" languages. In linguistic applications *letters* of Σ correspond not to letters but to *words* (word forms) of the natural language; *words* built from letters of Σ correspond to *sentences*. Thus grammars describe the language on the syntactic rather than lexical level. In this paper we use the "letter–word" terminology, not the "word–sentence" one.

One can use various calculi to describe formal languages using categorial grammars; below we accurately define a particular sort of categorial grammars, namely, grammars based on the *product-free Lambek calculus*, or just *Lambek grammars* [13].

Syntactic types of the Lambek calculus L are built from a set of *primitive types* (variables) Pr using two binary connectives $\setminus$ and $/$, called the *left* and *right division*, respectively. The set of all syntactic types is denoted by Tp (more precisely, $\mathrm{Tp}(\setminus, /)$).

The Lambek calculus derives expressions of the form $A_1 \dots A_n \to B$ called *sequents*. We will use Greek letters for finite (possibly empty) sequences of types.

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

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

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

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

The *cut rule*

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

is admissible in L [13]. (This means that adding the cut rule does not enlarge the set of derivable sequents.)

We write $\mathrm{L} \vdash \Pi \to B$ if the sequent $\Pi \to B$ is derivable in L.

Finally, a *Lambek grammar* is a 3-tuple $\mathcal{G} = \langle \Sigma, \rhd, H \rangle$, where Σ is an alphabet, $H \in \mathrm{Tp}$ is a distinguished type, and $\rhd \subset \Sigma \times \mathrm{Tp}$ is a finite binary correspondence between letters of the alphabet and syntactic types. As said earlier, the word $w = a_1 \dots a_n$ is accepted by the grammar $\mathcal{G}$ if and only if 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 never generate the empty word. However, one can drop the constraint of Π being nonempty from the rules, which yields a modified calculus L^*. Grammars based on L^* can describe languages containing the empty word.

2. LAMBEK GRAMMARS ENRICHED WITH SEMANTIC LABELS

Lambek grammars defined in the previous section describe languages in the *weak sense*, just as sets of words. However, they can be enriched to solve a more sophisticated problem: if a word belongs to the language, then the grammar not only just generates it but also yields an object, called the *semantic value*, that is assigned to the word and considered to encode its "meaning." Of course, since some letters can be assigned more than one syntactic type, and also some sequents have more than one derivation in L, there could be words with more than one semantic value. (From the linguistic point of view this models homonymy, both lexical and syntactic.)

In this paper semantic values for words and parts of words will be represented by *typed λ-terms*. The system of types used in the λ-formalism differs from the type system of the Lambek calculus. To avoid misunderstanding, we will call the types of the λ-terms *semantic types*.

Semantic types are built from a set $\mathbf{B}$ of *basic types* using one binary connective "$\to$." The set of all semantic types is denoted by $\mathbf{T}$. Types are *populated* with λ-terms: for every semantic type we have a countable number of *variables* of this type; we also allow *constants* (technically, the only difference between constants and variables is that substitution and λ-abstraction (see below) are permitted only for variables). If f is a term of type $A \to B$ and u is a term of type A, then (fu) is a term of type B; if v is a term of type B and x is a variable of type A, then $\lambda x.v$ is a term of type $A \to B$. If u is a term of type A, we write $u : A$. By $u[x := v]$ we denote the result of substituting the term v for all *free* (not under λ) occurrences of the variable x (x and v must be of the same type). The set of all λ-terms is denoted by Tm_λ.

Terms are considered modulo α-, β-, and η-equivalence: one can freely rename bounded variables and perform reductions on subterms: $(\lambda x.u)v \rightsquigarrow_\beta u[x := v]$ (if free variables of v are not bounded in u) and $\lambda x.(ux) \rightsquigarrow_\eta u$ if x is not a free variable of u.

Now we introduce a translation $\sigma \colon \mathrm{Tp} \to \mathbf{T}$ from syntactic to semantic types. On primitive types this function is defined arbitrarily (and the image of a primitive syntactic type is not necessarily a basic semantic type), and then σ is uniquely extended to Tp: $\sigma(A \setminus B) = \sigma(B / A) = \sigma(A) \to \sigma(B)$.

We denote both syntactic and semantic types by capital Latin letters ($A, B, C, \dots$). If A is a syntactic type and u is a term, we also write $u : A$, meaning $u : \sigma(A)$.

Now we enrich the Lambek calculus with *semantic labeling* [2] (see, e.g., [5]). This is done in a Curry–Howard style [10]. If $\Pi = A_1 \dots A_n$ and $\vec{x} = (x_1, \dots, x_n)$ is a sequence of variables, then by $\vec{x} : \Pi$ we denote the sequence $x_1 : A_1, \dots, x_n : A_n$. Sequents of the enriched calculus L_λ are expressions of the form $\vec{x} : \Pi \to u : A$.

Axioms of L_λ are sequents of the form $x : A \to x : A$. The rules of inference are as follows:

$$\frac{x : A, \vec{y} : \Pi \to u : B}{\vec{y} : \Pi \to (\lambda x.u) : (A \setminus B)}, \qquad \frac{\vec{y} : \Pi, x : A \to u : B}{\vec{y} : \Pi \to (\lambda x.u) : (B \,/\, A)}, \quad \text{where } \Pi \text{ is nonempty,}$$

$$\frac{\vec{x} : \Pi \to u : A \quad \vec{y} : \Gamma, t : B, \vec{z} : \Delta \to v : C}{\vec{y} : \Gamma, \vec{x} : \Pi, f : (A \setminus B), \vec{z} : \Delta \to v[t := (fu)] : C},$$

$$\frac{\vec{x} : \Pi \to u : A \quad \vec{y} : \Gamma, t : B, \vec{z} : \Delta \to v : C}{\vec{y} : \Gamma, f : (B \,/\, A), \vec{x} : \Pi, \vec{z} : \Delta \to v[t := (fu)] : C}.$$

The cut rule in L_λ is also admissible and is formulated as follows (actually it acts as the substitution operation on λ-terms):

$$\frac{\vec{x} : \Pi \to u : A \quad \vec{y} : \Gamma, t : A, \vec{z} : \Delta \to v : C}{\vec{y} : \Gamma, \vec{x} : \Pi, \vec{z} : \Delta \to v[t := u] : C}.$$

One can easily see (and prove by induction on the derivation length) that if $\mathrm{L} \vdash A_1 \dots A_n \to H$, then there exists a term $v : H$ such that $\mathrm{L}_\lambda \vdash x_1 : A_1, \dots, x_n : A_n \vdash v : H$ and the bounded variables of v are contained in $\{x_1, \dots, x_n\}$. Thus the term v somehow *encodes the derivation* of the sequent $A_1 \dots A_n \to H$ in L, and it will be used to build the semantic value of the corresponding word. This semantic value is composed from the semantic values assigned by the grammar to letters of the word, which are substituted for the corresponding variables x_i.

Let us define the notion of L_λ-grammar more accurately and formally. An L_λ-grammar is a 3-tuple $\mathcal{G} = \langle \Sigma, \mathcal{D}, H \rangle$. As in the definition of L-grammar, here Σ is an alphabet, $H \in \mathrm{Tp}$, but now instead of a binary correspondence $\rhd$ we use a ternary relation $\mathcal{D} \subset \Sigma \times \mathrm{Tp} \times \mathrm{Tm}_\lambda$, called the *categorial vocabulary*. The set $\mathcal{D}$ is finite, and if $\langle a, A, u \rangle \in \mathcal{D}$, then $u : A$. Implicitly, the mapping $\sigma \colon \mathrm{Tp} \to \mathbf{T}$ is also part of $\mathcal{G}$.

The word $w = a_1 \dots a_n$ is accepted by $\mathcal{G}$ if and only if there exist syntactic types $A_1, \dots, A_n$ and terms $u_1, \dots, u_n$ such that $\langle a_i, A_i, u_i \rangle \in \mathcal{D}$ $(i = 1, \dots, n)$ and $\mathrm{L}_\lambda \vdash x_1 : A_1, \dots, x_n : A_n \to v : H$ for some term v. In this case the grammar yields the term $v[x_1 := u_1, \dots, x_n := u_n]$ as a semantic value for w.

3. CONTEXT-FREE GRAMMARS AND LAMBEK GRAMMARS

Another, more common, class of grammars used for describing fragments of natural languages and also programming languages is the class of *context-free grammars* introduced by Chomsky [6]. A context-free grammar is a 4-tuple $\mathcal{G} = \langle N, \Sigma, \mathcal{P}, s \rangle$, where N and Σ are disjoint alphabets (elements of N are called *nonterminal symbols*), $s \in N$ is the *start symbol*, and $\mathcal{P}$ is a finite set of *rules*. Each rule is of the form $p \Rightarrow \alpha$, where $p \in N$ and α is a word constructed from letters of the united alphabet $N \cup \Sigma$. The rule $p \Rightarrow \alpha$ can be applied to a word of the form $\phi p \psi$ (in the united alphabet), yielding $\phi \alpha \psi$. A word w is accepted by G if it can be obtained from the one-letter word s by a finite number of rule applications and does not contain nonterminal symbols.

Languages described by context-free grammars are called *context-free languages*.

Further we mainly consider context-free grammars in *Chomsky normal form*. Rules of such grammars can be of two forms: either $p \Rightarrow qr$, where $q, r \in N$, or $p \Rightarrow a$, where $a \in \Sigma$. If a context-free language does not contain the empty word, it can be described by a context-free grammar in Chomsky normal form.

It is known [14] that context-free grammars and Lambek grammars are equivalent in the weak sense: every context-free language is generated by some Lambek grammar, and vice versa, any language generated by a Lambek grammar is also generated by a context-free one.

However, as shown earlier, besides judging whether a word belongs to the language, a Lambek grammar also assigns a λ-term (called the semantic value of the word) to every word that belongs to the language.

Context-free grammars can also be enriched with semantic labels, so they could assign semantic values (λ-terms) to the words accepted. For context-free grammars this enrichment is called *Montague semantics* (see, e.g., [7]). First, we associate a semantic type with every nonterminal symbol. This is done by a function $\sigma\colon N \to \mathbf{T}$. Now let $p \Rightarrow \alpha$ be a rule of the grammar G and α contains nonterminal symbols $q_1, \dots, q_k$ (in the order as listed). Then this rule is associated with a λ-term u of type $\sigma(p)$ with free variables from the list $x_1, \dots, x_k$. In particular, if the right-hand side of the rule does not contain any nonterminal symbol, the term u cannot contain any free variables (it must be *closed*). Finally, if a word w is generated from s by a sequence of rule applications: $s \Rightarrow \beta_1 \Rightarrow \ldots \Rightarrow \beta_m \Rightarrow w$, then, going from right to left, we assign a closed λ-term to each nonterminal symbol. If $\beta_i = \phi p \psi$, $\beta_{i+1} = \phi\alpha\psi$, and nonterminal symbols $q_1, \dots, q_k$ from α are already associated with terms $v_1, \dots, v_k$, then p is associated with the term $u[x_1 := v_1, \dots, x_k := v_k]$. Terms associated to nonterminal symbols from ϕ and ψ are kept unmodified.

Thus it is interesting to have translations between context-free grammars and Lambek grammars that preserve not only languages generated by the grammars as sets of words but also semantic values of the words.

Historically, the first method of translating context-free grammars into Lambek grammars, introduced by Gaifman [1],[1] proceeds in two steps: first the context-free grammar is transformed into Greibach normal form [8], and then this grammar in Greibach normal form is translated into a Lambek grammar. (Strictly speaking, only context-free grammars not generating the empty word can be transformed into Greibach normal form. For languages containing the empty word Gaifman's construction can also be used via an easy technical trick [12].)

Kanazawa and Salvati [11] presented a method to transform a context-free grammar into Greibach normal form while preserving semantic values of words. This method works only for grammars without ϵ-rules, i.e., rules with empty right-hand side (in particular, the generated language should not contain the empty word), and *cyclic* rules, i.e. rules of the form $p \Rightarrow q$, where $p, q \in N$. Then, given a context-free grammar in Greibach normal form, one can apply Gaifman's translation, which can also be naturally used to convey semantic labeling. The prohibition of ϵ-rules and cyclic rules is inevitable, because a context-free grammar with such rules can assign an infinite number of semantic values to the same word, which is impossible for Lambek grammars.

However, there also exists a direct transformation of a context-free grammar in Chomsky normal form into a Lambek grammar, suggested by Buszkowski [4]. For practical purposes this translation is preferable, because it does not increase (more precisely, increases only linearly) the grammar's size and keeps the structure of derivation trees. And it happens that this translation can be performed in such a way that it preserves semantic values of the words accepted by the grammar. In the next section, we will describe a modification of Buszkowski's translation that preserves semantic values.

A translation in the opposite direction—from Lambek grammars to context-free ones—was first constructed by Pentus [14]. Kanazawa and Salvati [11] showed that this translation can be extended to a translation of grammars with semantic labeling that keeps the semantic values of words. Unfortunately, Pentus's construction is inefficient: it leads to exponential growth of the grammar's size. This inefficiency is apparently unavoidable due to complexity reasons: the derivability problem in the product-free Lambek calculus is NP-complete [15], whereas one can decide whether a word belongs to a context-free language in polynomial time.

[1] Gaifman considered not Lambek grammars but a weaker formalism called *Ajdukiewicz–Bar-Hillel grammars* or *basic categorial grammars*. The fact that Gaifman's construction also works for Lambek grammars was noted by Buszkowski [3].

4. A MODIFICATION OF BUSZKOWSKI'S TRANSLATION THAT PRESERVES SEMANTIC VALUES

For simplicity we consider Montague semantics of a specific kind. Each rule of the form $t \Rightarrow a$ is associated with a closed λ-term, and rules of the form $p \Rightarrow qr$ can be of two sorts: *right-to-left*, for which $\sigma(r) = \sigma(q) \to \sigma(p)$, and *left-to-right*, for which $\sigma(q) = \sigma(r) \to \sigma(p)$, and the λ-term associated with the rule is (x_2x_1) or (x_1x_2), respectively. Intuitively, p is associated with the λ-term which is the application of the λ-terms associated with q and r to each other (in the right-to-left or left-to-right order). This case is most often used in practice [7] and is chosen in order to make the proof easier. The modification of Buszkowski's construction presented below also works in the general case.

Given a context-free grammar G with a Montague-style semantic labeling, we need to construct a Lambek grammar $\mathcal{G}$ that generates the same language and assigns the same semantic values to the words of this language as G.

Consider all nonterminal symbols of the grammar G as primitive types ($N \subseteq \mathrm{Pr}$); the mapping σ is also preserved. With every nonterminal symbol we associate a set $I(t) \subset \mathrm{Tp}$ in the following way:

- $t \in I(t)$;
- if $p \Rightarrow qt$ is a rule of G, then $(q \setminus p) \in I(t)$;
- if t is a nonterminal symbol and $p \Rightarrow qr$ is a rule of G, then $(q \setminus p)/(t \setminus r) \in I(t)$.

Now we define the categorial grammar $\mathcal{G}$. Let $H = s$. For every rule of the form $t \Rightarrow a$ associated with the term $u : \sigma(t)$ and for $A \in I(t)$, we add an entry to the categorial vocabulary $\mathcal{D}$ in the following way:

- if $A = t$, then we add $\langle a, A, u\rangle$;
- if $A = q \setminus p$ and $p \Rightarrow qt$ is a right-to-left rule, then we add $\langle a, A, u\rangle$ (in this case $\sigma(t) = \sigma(q) \to \sigma(p)$; thus u has the correct type $\sigma(A)$);
- if $A = q \setminus p$ and $p \Rightarrow qt$ is a left-to-right rule, we take a variable f of type $\sigma(q)$, let $\widetilde{u} = \lambda f.(fu)$, and add $\langle a, A, \widetilde{u}\rangle$ ($\widetilde{u}$ is a well-formed term of type $\sigma(q) \to \sigma(p) = \sigma(q \setminus p) = \sigma(A)$);
- if $A = (q \setminus p)/(t \setminus r)$ and $p \Rightarrow qr$ is a right-to-left-rule, we take a variable g of type $\sigma(t) \to \sigma(r)$, let $\widetilde{u} = \lambda g.(gu)$, and add $\langle a, A, \widetilde{u}\rangle$ ($\widetilde{u}$ is now of type $(\sigma(t) \to \sigma(r)) \to \sigma(r) = \sigma(A)$);
- finally, if $A = (q \setminus p)/(t \setminus r)$ and $p \Rightarrow qr$ is a left-to-right rule, we take two variables g and f of types $\sigma(t) \to \sigma(r)$ and $\sigma(r) \to \sigma(p)$, respectively, let $\widehat{u} = \lambda g.\lambda f.(f(gu))$, and add $\langle a, A, \widehat{u}\rangle$. As in the previous cases, $\widehat{u}$ has the correct type $\sigma(A)$.

Proposition 1. *If a word $w \in \Sigma^+$ is accepted by $\mathcal{G}$, then it is also generated by G.*

Proof. For every right-to-left rule of the form $p \Rightarrow qr$ consider the sequent $x : q, f : r \to (fx) : p$, and for every left-to-right rule $p \Rightarrow rq$ consider the sequent $f : r, x : q \to (fx) : p$. Denote the set of all such sequents by $\mathcal{A}$. Let us add them as new axioms to L_λ and also add the cut rule, thus getting a new calculus $\mathrm{L}_\lambda + \mathcal{A}$.

Now let $t \in N$ and $A \in I(t)$. One can easily check that then $\mathrm{L}_\lambda + \mathcal{A} \vdash x : t \to x' : A$, where x' is a term of type $\sigma(A)$ (either x, or $\widetilde{x}$, or $\widehat{x}$).

Let the word $w = a_1 \dots a_n$ be accepted by $\mathcal{G}$. Then $\mathrm{L}_\lambda \vdash x_1 : A_1, \dots, x_n : A_n \to v : s$. By the cut rule, we get

$$\mathrm{L}_\lambda + \mathcal{A} \vdash x_1 : t_1, \dots, x_n : t_n \to v[x_1 := x_1', \dots, x_n := x_n'] : s.$$

Now we reformulate the calculus $\mathrm{L}_\lambda + \mathcal{A}$ in an equivalent form. Instead of the set $\mathcal{A}$ of axioms we introduce a set $\mathcal{R}$ of the following rules of inference:

$$\frac{\vec{z}_1 : \Gamma_1 \to v : q \quad \vec{z}_2 : \Gamma_2 \to u : r}{\vec{z}_1 : \Gamma_1, \vec{z}_2 : \Gamma_2 \to (uv) : p} \qquad \text{for every right-to-left rule } p \Rightarrow qr,$$

$$\frac{\vec{z}_1 : \Gamma_1 \to u : r \quad \vec{z}_2 : \Gamma_2 \to v : q}{\vec{z}_1 : \Gamma_1, \vec{z}_2 : \Gamma_2 \to (uv) : p} \qquad \text{for every left-to-right rule } p \Rightarrow rq.$$

The calculi $\mathrm{L}_\lambda + \mathcal{A}$ and $\mathrm{L}_\lambda + \mathcal{R}$ derive the same sequents, but $\mathrm{L}_\lambda + \mathcal{R}$ enjoys cut elimination (this is proved in a standard way). Therefore,

$$\mathrm{L}_\lambda + \mathcal{R} \vdash x_1 : t_1, \dots, x_n : t_n \to v' : s$$

(for some v'), and the derivation does use the cut rule. Since types in this sequent do not contain logical connectives, the only rules that can be used in the derivation are rules from $\mathcal{R}$, and the derivation itself is actually a sequence of applications of G-rules. Hence, w is accepted by G. $\square$

This proposition shows that $\mathcal{G}$ does not generate any extra words in comparison with G. Moreover, the cut elimination procedure in $\mathrm{L}_\lambda + \mathcal{R}$ preserves the term on the right-hand side of the sequent (modulo α-, β-, and η-equivalence; this is proved in the same way as for L_λ itself [9]). Hence, v' is the same term as $v[x_1 := x'_1, \dots, x_n := x'_n]$, and at the same time it is the semantic value assigned to the word w by G. The choice of x_i, $\widehat{x}_i$, or $\widetilde{x}_i$ for x'_i is determined by the type A_i. Thus the semantic values coincide.

Proposition 2. *If a word w is generated by G, then it is accepted by $\mathcal{G}$ with the same semantic value.*

Proof. Proceed by induction on the length of the sequence of applications of G-rules. We decorate nonterminal symbols with syntactic types. The start symbol s is decorated with itself (as a primitive type). At the induction step we apply a rule of the form $p \Rightarrow qr$, decorate r with $(q \setminus p)$, and consider several cases:

- p is decorated with itself; then decorate q with itself as well;
- p is decorated with a type of the form $(q_1 \setminus p_1)$; then decorate q with $(q_1 \setminus p_1)/(q \setminus p)$;
- p is decorated with a type of the form $(q_1 \setminus p_1)/(p \setminus s)$; then decorate q with $(q_1 \setminus p_1)/(q \setminus s)$.

One can easily check by induction that every occurrence of a nonterminal symbol t is decorated by a type from $I(t)$.

All nonterminal symbols occurring in the sequence of applications of G-rules are assigned some λ-terms by the grammar G. Now we go along this sequence backwards and assign one more term to each nonterminal symbol according to the derivation in the Lambek calculus, so that if the old term was u, then the new term will be either u, or $\widetilde{u}$, or $\widehat{u}$. More precisely, we prove the following statement by induction: if after a number of applications of G-rules one obtains a word $t_1 \dots t_m$ ($t_i \in N$) from s and every t_i is assigned a term u_i by G, then there exist terms $u'_1, \dots, u'_m$ such that for every i the term u'_i is either u_i, or $\widetilde{u}_i$, or $\widehat{u}_i$, with $u'_i : A_i$, where A_i is the type that decorates the occurrence of t_i (the choice of one of the three alternatives for u_i is determined by this type), and, finally,

$$\mathrm{L}_\lambda \vdash x_1 : A_1, \dots, x_m : A_m \to \bar{v} : s,$$

where $\bar{v}[x_1 = u'_1, \dots, x_m = u'_m]$ is the term assigned to the starting symbol s (the semantic value of the word).

The induction base is evident: $m = 1$, $A_1 = s$, and we take v for u'_1.

If a right-to-left rule of the form $p \Rightarrow qr$ is applied at the induction step and the nonterminal symbol p is associated with a term u by G, then this term is of the form (hu_1), where h is associated with r and u_1 is associated with q. Also let p, q, and r be decorated with types A, B, and C, respectively. Considering possible cases for the types A, B, and C, one can easily check that

$$\mathrm{L}_\lambda \vdash u'_1 : B, h' : C \vdash u' : A,$$

and then it is sufficient to apply the cut rule.

The left-to-right case is considered symmetrically.

Finally, we apply the rules of the form $t_i \Rightarrow a$. Since the choice of the correct u'_i is determined by A_i, the corresponding 3-tuples $\langle a_i, A_i, u'_i \rangle$ exist in the categorial vocabulary $\mathcal{D}$. Hence, the word $a_1 \dots a_n$ is accepted by $\mathcal{G}$ with semantic value v. $\square$

These two propositions together yield the following result:

Theorem. *If G is a context-free grammar in Chomsky normal form with Montague semantics and $\mathcal{G}$ is the Lambek grammar built from G as described above, then $\mathcal{G}$ generates the same language as G and assigns the same semantic values to the words of this language as G.*

ACKNOWLEDGMENTS

The author is grateful to the participants of the workshop "Natural Language and Computer Science" held at Vienna Summer of Logic 2014 (Vienna, Austria), and especially to M. Kanazawa, for fruitful discussions. The author is also grateful to Prof. M.R. Pentus for his attention to this work.

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

REFERENCES

1. Y. Bar-Hillel, C. Gaifman, and E. Shamir, "On categorial and phrase-structure grammars," Bull. Res. Council Israel, Sect. F **9F**, 1–16 (1960).
2. J. van Benthem, "The semantics of variety in categorial grammar," in *Categorial Grammar* (J. Benjamins, Amsterdam, 1988), pp. 37–55.
3. W. Buszkowski, "The equivalence of unidirectional Lambek categorial grammars and context-free grammars," Z. Math. Logik Grundlagen Math. **31**, 369–384 (1985).
4. W. Buszkowski, "On the equivalence of Lambek categorial grammars and basic categorial grammars," Preprint LP-93-07 (Univ. Amsterdam, Inst. Logic Lang. Comput., Amsterdam, 1993), ILLC Prepubl. Ser.
5. B. Carpenter, *Type-Logical Semantics* (MIT Press, Cambridge, MA, 1997).
6. N. Chomsky, "Three models for the description of language," IRE Trans. Inf. Theory **IT-2** (3), 113–124 (1956).
7. D. R. Dowty, R. E. Wall, and S. Peters, *Introduction to Montague Semantics* (D. Reidel, Dordrecht, 1981).
8. S. A. Greibach, "A new normal-form theorem for context-free phrase structure grammars," J. Assoc. Comput. Mach. **12** (1), 42–52 (1965).
9. H. Hendriks, "Studied flexibility: Categories and types in syntax and semantics," PhD Dissert. (Univ. Amsterdam, Amsterdam, 1993).
10. W. A. Howard, "The formulae-as-types notion of construction," in *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Ed. by J. P. Seldin and H. J. Roger (Academic, London, 1980), pp. 479–490.
11. 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.
12. S. Kuznetsov, "Lambek grammars with one division and one primitive type," Log. J. IGPL **20** (1), 207–221 (2012).
13. J. Lambek, "The mathematics of sentence structure," Am. Math. Mon. **65** (3), 154–170 (1958).
14. M. R. Pentus, "Lambek calculus and formal grammars," Fundam. Prikl. Mat. **1** (3), 729–751 (1995).
15. Yu. Savateev, "Product-free Lambek calculus is NP-complete," in *Logical Foundations of Computer Science: Proc. Int. Symp. LFCS 2009*, Ed. by S. Artemov and A. Nerode (Springer, Berlin, 2009), Lect. Notes Comput. Sci. **5407**, pp. 380–394.

Translated by the author