DOI 10.1007/s10469-021-09657-5
Algebra and Logic, Vol. 60, No. 5, November, 2021
(Russian Original Vol. 60, No. 5, September-October, 2021)

COMPLEXITY OF LAMBEK CALCULI WITH MODALITIES AND OF TOTAL DERIVABILITY IN GRAMMARS

S. M. Dudakov,[1*] **B. N. Karlov,**[2*]
S. L. Kuznetsov,[3*,**] **and E. M. Fofanova**[4]

UDC 510.649

Keywords: *Lambek calculus, substructural logic, exponential, relevant modality, algorithmic complexity, context-free grammars, total derivability.*

The Lambek calculus with the unit can be defined as the atomic theory (algebraic logic) of the class of residuated monoids. This calculus, being a theory of a broader class of algebras than Heyting ones, is weaker than intuitionistic logic. Namely, it lacks structural rules: permutation, contraction, and weakening. We consider two extensions of the Lambek calculus with modalities—the exponential, under which all structural rules are permitted, and the relevant modality, under which only permutation and contraction rules are allowed. The Lambek calculus with a relevant modality is used in mathematical linguistics. Both calculi are algorithmically undecidable. We consider their fragments in which the modality is allowed to be applied to just formulas of Horn depth not greater than 1. *We prove that these fragments are decidable and belong to the* NP *class. To show this, in the case of a relevant modality, we introduce a new notion of* $\mathcal{R}$*-total derivability in context-free grammars, i.e., existence of a derivation in which each rule is used at least a given number of times. It is stated that the* $\mathcal{R}$*-totality problem is* NP*-hard for context-free grammars. Also we pinpoint algorithmic complexity of* $\mathcal{R}$*-total derivability for more general classes of generative grammars.*

*Supported by RFBR, project No. 20-01-00435.

**Supported by the Council for Grants (under RF President), grant MK-1184.2021.1.1.

[1]Tver State University, Tver, Russia; sergeydudakov@yandex.ru. [2]Tver State University, Tver, Russia; bnkarlov@gmail.com. [3]Steklov Mathematical Institute, Russian Academy of Sciences, Moscow, Russia; skuzn@inbox.ru. [4]Lomonosov Moscow State University, Moscow, Russia; evgeniya.f.20.02@yandex.ru.
Translated from *Algebra i Logika*, Vol. 60, No. 5, pp. 471-496, September-October, 2021. Russian DOI: 10.33048/alglog.2021.60.502. Original article submitted April 2, 2021; accepted November 29, 2021.

INTRODUCTION

A *residuated monoid* is a partially ordered monoid $(A, \cdot, \mathbf{1}, \preceq)$ with additionally defined left ($\backslash$) and right ($/$) division operations, such that $b \preceq a\backslash c \iff a \cdot b \preceq c \iff a \preceq c/b$. Note that this condition is bound not to the equality (as, e.g., for division in groups), but rather to the partial order; the idea of partially ordered residuated algebraic structures goes back to Krull [1].

The Lambek calculus with the unit, $\mathbf{L_1}$, can be defined as the atomic theory (algebraic logic) of the class of residuated monoids (see [2]). We will consider the set of formulas built from variables and a constant $\mathbf{1}$ using operations $\cdot$, $\backslash$, and $/$. Theorems of $\mathbf{L_1}$ are exactly all statements of the form $A \preceq B$ (where A and B are formulas) which are generally true in all residuated monoids.

We describe a sequent calculus for $\mathbf{L_1}$. Formulas are denoted by capital Latin letters; capital Greek letters denote sequences of formulas; Λ stands for the empty sequence. Sequents of the $\mathbf{L_1}$ calculus are expressions of the form $\Gamma \vdash A$, where Γ is a sequence of formulas (possibly empty) and A is a formula. Axioms of our calculus are $A \vdash A$ and $\Lambda \vdash \mathbf{1}$. Inference rules are defined as follows:

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, \Pi, A\backslash B, \Delta \vdash C}\ (\backslash \vdash), \quad \frac{A, \Pi \vdash B}{\Pi \vdash A\backslash B}\ (\vdash \backslash), \quad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \cdot B, \Delta \vdash C}\ (\cdot \vdash),$$

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, B/A, \Pi, \Delta \vdash C}\ (/ \vdash), \quad \frac{\Pi, A \vdash B}{\Pi \vdash B/A}\ (\vdash /), \quad \frac{\Pi \vdash A \quad \Delta \vdash B}{\Pi, \Delta \vdash A \cdot B}\ (\vdash \cdot),$$

$$\frac{\Gamma, \Delta \vdash C}{\Gamma, \mathbf{1}, \Delta \vdash C}\ (\mathbf{1} \vdash), \qquad \frac{\Pi \vdash A \quad \Gamma, A, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ (\text{cut}).$$

A natural example of a residuated monoid is an algebra of formal languages over an alphabet. An interpretation on the algebra of formal languages corresponds to the linguistic understanding of the Lambek calculus.

A particular case of residuated monoids are Heyting algebras—more precisely, their reducts in the language of $\wedge$, $\Rightarrow$, $\top$, up to change of notation: $a \wedge b = a \cdot b$, $\top = \mathbf{1}$, $a \Rightarrow b = a\backslash b$. The $\mathbf{L_1}$ calculus, being the theory of a broader class of algebras, is itself weaker than intuitionistic logic. Namely, it lacks *structural rules* of permutation, contraction, and weakening. We will consider an extension of the $\mathbf{L_1}$ calculus with a modality under which structural rules are allowed. The idea of such a modality, which is called the *exponential* and is denoted by !, goes back to Girard [3] who introduced it in linear logic. For the exponential, we have the following inference rules:

introduction rules

$$\frac{\Gamma, A, \Delta \vdash C}{\Gamma, !A, \Delta \vdash C}\ (! \vdash), \qquad \frac{!A_1, \ldots, !A_n \vdash B}{!A_1, \ldots, !A_n \vdash !B}\ (\vdash !),$$

and structural rules

$$\frac{\Gamma, B, !A, \Delta \vdash C}{\Gamma, !A, B, \Delta \vdash C}\ (!\text{perm}_1), \qquad \frac{\Gamma, !A, B, \Delta \vdash C}{\Gamma, B, !A, \Delta \vdash C}\ (!\text{perm}_2),$$

$$\frac{\Gamma, !A, !A, \Delta \vdash C}{\Gamma, !A, \Delta \vdash C}\ (!\text{contr}), \qquad \frac{\Gamma, \Delta \vdash C}{\Gamma, !A, \Delta \vdash C}\ (!\text{weak}).$$

The extension of $\mathbf{L_1}$ with the exponential will be denoted by $\mathbf{!L_1}$.

Let us consider one more calculus, $\mathbf{!^r L_1}$, which is obtained from $\mathbf{!L_1}$ by removing the weakening rule (!weak). Such ! modality is said to be *relevant* since the behavior of formulas under this modality corresponds to a relevant logic [4]. A relevant modality is meaningful from a linguistic point of view [5].

Cut-eliminability in the $\mathbf{!L_1}$ and $\mathbf{!^r L_1}$ calculi follows from a more general result [6]. The derivability problem for $\mathbf{L_1}$ belongs to the NP complexity class. On the other hand, systems $\mathbf{!L_1}$ and $\mathbf{!^r L_1}$ have undecidable derivability problems [5, 7]. However, if we allow applying the ! modality only to variables, both problems (for $\mathbf{!L_1}$ and for $\mathbf{!^r L_1}$) become decidable and belong to the NP class [5].

We study the frontier between decidable and undecidable fragments of $\mathbf{!L_1}$ and $\mathbf{!^r L_1}$ in more detail. For formulas which are built from variables using only $\backslash$ and / operations, we define the notion of *depth*: $d(p_i) = 0$, where p_i is a variable, and $d(A\backslash B) = d(B/A) = \max\{d(A) + 1, d(B)\}$. We say that a sequent is 1-*bounded* if in this sequent ! is applied only to formulas of depth not greater than 1. Due to the permutation rule, each formula of the form $!A$, where $d(A) \leqslant 1$, is equivalent to a formula of the form $!(\dots(p/q_n)/\dots/q_1)$, which we will write as $!(p/(q_1 \dots q_n))$.

THEOREM 1. Derivability problems in the $\mathbf{!L_1}$ and $\mathbf{!^r L_1}$ calculi for 1-bounded sequents are algorithmically decidable and belong to NP.

This strengthening of the result from [5] is in a sense sharp: if one allows formulas of depth 2 under !, then the derivation problem becomes undecidable [5]. For a relevant modality, we also obtain a new upper bound. Namely, the following holds:

THEOREM 2. The derivability problem in the $\mathbf{!^r L_1}$ calculus for 1-bounded sequents built using only $\backslash$ and ! is NP-hard.

This result does not follow from NP-completeness of $\mathbf{L_1}$ (see [8]) because its fragment with only one operation $\backslash$ is polynomially decidable [9].

In order to prove the above results, we introduce a new notion of $\mathcal{R}$-*total derivability* in context-free grammars (meaning that there exists a derivation which uses each rule at least a given number of times) and establish NP-completeness of the corresponding algorithmic problem. We also pinpoint algorithmic complexity of $\mathcal{R}$-total derivability for more general classes of generative grammars.

1. AUXILIARY CALCULI $\mathbf{!_{\leqslant 1} L_1}$ AND $\mathbf{!^r_{\leqslant 1} L_1}$

In order to construct NP-algorithms for checking derivability in the given fragments of the $\mathbf{!L_1}$ and $\mathbf{!^r L_1}$ calculi, we introduce equivalent auxiliary calculi denoted by $\mathbf{!_{\leqslant 1} L_1}$ and $\mathbf{!^r_{\leqslant 1} L_1}$.

We will need the notion of a *multiset*, an analog of a set in which elements may have multiplicities. Formally, a multiset is a pair $\mathbf{X} = (X, \mu)$, where X is a set (*support*) and $\mu\colon X \to \mathbb{N}$ is the multiplicity function. A multiset is *finite* if $\mu(x) \neq 0$ only for a finite number of values of $x \in X$ (while the X itself could be infinite). For two multisets $\mathbf{X} = (X, \mu)$ and $\mathbf{Y} = (X, \nu)$ with the same support, we define their *sum*: $\mathbf{X} \uplus \mathbf{Y} = (X, \zeta)$, where $\zeta(x) = \mu(x) + \nu(x)$ for each $x \in X$, and the *"submultiset" relation*: $\mathbf{X} \subseteq \mathbf{Y}$ if $\mu(x) \leqslant \nu(x)$ for each $x \in X$. We say that an element $x \in X$ *belongs to the multiset* $\mathbf{X} = (X, \mu)$ if $\mu(x) > 0$.

Recall that a *context-free grammar* (cf-grammar for short) is a quadruple $G = (N, \Sigma, \mathcal{P}, s)$, where N and Σ are two disjoint alphabets whose letters are called *nonterminals* and *terminals*, respectively, $s \in N$, and $\mathcal{P}$ is a finite set of rules of the form $A \to \alpha$, where $A \in N$ and $\alpha \in (N \cup \Sigma)^*$. We write $\beta \Rightarrow \gamma$ if $\beta = \eta A \theta$, $\gamma = \eta \alpha \theta$, and $A \to \alpha \in \mathcal{P}$ for some $\eta, \theta \in (N \cup \Sigma)^*$.

A *derivation* in a cf-grammar G is a chain $\beta_0 \Rightarrow \ldots \Rightarrow \beta_k$, denoted $\beta_0 \Rightarrow^k \beta_k$, or $\beta_0 \Rightarrow^* \beta_k$ if the number of steps does not matter. If $k > 0$, we also write $\beta_0 \Rightarrow^+ \beta_k$. If $w \in \Sigma^*$ (i.e., does not contain nonterminals) and $s \Rightarrow^* w$, then w is derivable in a cf-grammar G.

Now let $\mathcal{R}$ be a multiset of rules in G, i.e., $\mathcal{R} = (\mathcal{P}, \rho)$. We say that a derivation is $\mathcal{R}$-*total* if, for each rule, the number of times it is used in the derivation is not less than the multiplicity of that rule in $\mathcal{R}$. And we write $\beta_0 \Rightarrow^*_{\mathcal{R}} \beta_k$. In particular, if, for $\mathcal{R}$, we take the whole $\mathcal{P}$ with multiplicity 1 for each rule, we obtain the notion of a *total* derivation.

Now we are ready to formulate the calculi $\mathbf{!_{\leqslant 1}L_1}$ and $\mathbf{!^{r}_{\leqslant 1}L_1}$. Sequents of these calculi are expressions of the form $!\Phi; \Pi \vdash B$, where $!\Phi$ is a multiset of formulas of the form $!(p/(q_1 \ldots q_n))$, Π is a sequence of formulas, B is a formula, and $\Pi \vdash B$ is 1-bounded.

Sets of axioms of the $\mathbf{!_{\leqslant 1}L_1}$ and $\mathbf{!^{r}_{\leqslant 1}L_1}$ calculi are defined as follows. Given a multiset $!\Phi$ and a variable s, we define a cf-grammar $G_{!\Phi,s} = (N, \Sigma, \mathcal{P}, s)$, where N is the set of variables occurring in $!\Phi$, with s added, and $\Sigma = \{\hat{r} \mid r \in N\}$. The rules in $\mathcal{P}$ are as follows: $p \to q_1 \ldots q_m$ for each formula $!(p/(q_1 \ldots q_m))$ from $!\Phi$ and $r \to \hat{r}$ for each $r \in N$.

As axioms of the $\mathbf{!_{\leqslant 1}L_1}$ calculus, we take all sequents of the form $!\Phi; r_1, \ldots, r_n \vdash s$, where the word $\hat{r}_1 \ldots \hat{r}_n$ is derivable in the cf-grammar $G_{!\Phi,s}$, and sequents $!\Phi; \Lambda \vdash \mathbf{1}$ for arbitrary $!\Phi$.

The multiset $\mathcal{R}_{!\Phi}$ of rules of the cf-grammar $G_{!\Phi,s}$ is defined in the following way. Each rule of the form $p \to q_1 \ldots q_m$ has the same multiplicity as the corresponding formula $!(p/(q_1 \ldots q_m))$ has in the multiset $!\Phi$. Rules of the form $r \to \hat{r}$ have multiplicity 0. As axioms of the $\mathbf{!^{r}_{\leqslant 1}L_1}$ calculus, we take all sequents of the form $!\Phi; r_1, \ldots, r_n \vdash s$, where the word $\hat{r}_1 \ldots \hat{r}_n$ has an $\mathcal{R}_{!\Phi}$-total derivation in the cf-grammar $G_{!\Phi,s}$, and the sequent $\varnothing; \Lambda \vdash \mathbf{1}$. This class of axioms reflects the idea of relevance: each rule, which corresponds to a formula from $!\Phi$, should be used.

Inference rules in both calculi, $\mathbf{!_{\leqslant 1}L_1}$ and $\mathbf{!^{r}_{\leqslant 1}L_1}$, are the same:

$$\frac{!\Psi; \Pi \vdash A \quad !\Phi; \Gamma, B, \Delta \vdash C}{!\Theta; \Gamma, \Pi, A \backslash B, \Delta \vdash C}\ (\backslash \vdash), \qquad \frac{!\Psi; \Pi \vdash A \quad !\Phi; \Gamma, B, \Delta \vdash C}{!\Theta; \Gamma, B/A, \Pi, \Delta \vdash C}\ (/ \vdash),$$

$$\frac{!\Phi; A, \Pi \vdash B}{!\Phi; \Pi \vdash A\backslash B}\ (\vdash \backslash), \qquad \frac{!\Phi; \Pi, A \vdash B}{!\Phi; \Pi \vdash B/A}\ (\vdash /), \qquad \frac{!\Phi; \Gamma, \Delta \vdash C}{!\Phi; \Gamma, \mathbf{1}, \Delta \vdash C}\ (\mathbf{1} \vdash),$$

$$\frac{!\Phi; \Gamma, A, B, \Delta \vdash C}{!\Phi; \Gamma, A \cdot B, \Delta \vdash C}\ (\cdot \vdash), \qquad \frac{!\Phi; \Pi \vdash A \quad !\Psi; \Delta \vdash B}{!\Theta; \Pi, \Delta \vdash A \cdot B}\ (\vdash \cdot),$$

$$\frac{!\Phi \uplus \{!A\}; \Gamma, \Delta \vdash C}{!\Phi; \Gamma, !A, \Delta \vdash C}\ (! \vdash), \qquad \frac{!\Phi; \Lambda \vdash B}{!\Phi; \Lambda \vdash !B}\ (\vdash !),$$

$$\frac{!\Psi; \Pi \vdash A \quad !\Phi; \Gamma, A, \Delta \vdash C}{!\Theta; \Gamma, \Pi, \Delta \vdash C}\ (\text{cut}), \qquad \frac{!\Psi; \Lambda \vdash !A \quad !\Phi \uplus \{!A\}; \Pi \vdash C}{!\Theta; \Pi \vdash C}\ (!\text{cut}).$$

Each two-premise rule should obey the following *correctness conditions*: $!\Phi \subseteq !\Theta$, $!\Psi \subseteq !\Theta$, $!\Theta \subseteq !\Phi \uplus !\Psi$.

THEOREM 3. A 1-bounded sequent $\Pi \vdash B$ is derivable in $\mathbf{!L_1}$ if and only if a sequent $\varnothing; \Pi \vdash B$ is derivable in $\mathbf{!_{\leqslant 1}L_1}$. The same holds for calculi $\mathbf{!^r L_1}$ and $\mathbf{!^r_{\leqslant 1}L_1}$ respectively.

Proof. All rules of the original calculi $\mathbf{!L_1}$ and $\mathbf{!^r L_1}$, which do not relate to ! (but including (cut)), are translated into $\mathbf{!_{\leqslant 1}L_1}$ and $\mathbf{!^r_{\leqslant 1}L_1}$ by adding empty !-parts. Rules $(! \vdash)$, $(\vdash !)$, $(!\text{perm}_1)$, (!contr), and (!weak) in the case of $\mathbf{!L_1}$ are simulated using a cut with appropriate sequents:

$$\{!A\}; \Lambda \vdash A, \quad \{!A\}; \Lambda \vdash !A, \quad \{!A\}; \Lambda \vdash !A \cdot !A, \quad \{!A\}; \Lambda \vdash \mathbf{1}.$$

For the opposite direction, we suppose that the sequent $\{!A_1, \ldots, !A_n\}; \Pi \vdash C$ is derivable in the new calculus and construct a derivation of $!A_1, \ldots, !A_n, \Pi \vdash C$ in the old one by induction on the derivation. In the base (axiom) case, we add an extra induction on a derivation in a cf-grammar. □

2. CUT ELIMINATION IN $\mathbf{!_{\leqslant 1}L_1}$ AND $\mathbf{!^r_{\leqslant 1}L_1}$

In order to prove cut-elimination theorems in $\mathbf{!_{\leqslant 1}L_1}$ and $\mathbf{!^r_{\leqslant 1}L_1}$, we will need the following abstract lemma on multisets.

LEMMA 4. Let $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ be four multisets, and let $\mathbf{D} \subseteq \mathbf{A} \uplus \mathbf{B} \uplus \mathbf{C}$, $\mathbf{A} \subseteq \mathbf{D}$, $\mathbf{B} \subseteq \mathbf{D}$, and $\mathbf{C} \subseteq \mathbf{D}$. Then there exists a multiset $\mathbf{E}$ such that $\mathbf{E} \subseteq \mathbf{A} \uplus \mathbf{C}$, $\mathbf{A} \subseteq \mathbf{E}$, $\mathbf{C} \subseteq \mathbf{E}$, $\mathbf{E} \subseteq \mathbf{D}$, and $\mathbf{D} \subseteq \mathbf{E} \uplus \mathbf{B}$.

Proof. Let some element belong to $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ with multiplicities a, b, c, d respectively. If $a + c \leqslant d$, then we add this element to $\mathbf{E}$ with multiplicity $e = a + c$. Conditions $e \leqslant a + c$, $a \leqslant e$, and $c \leqslant e$ are obviously satisfied, $e \leqslant d$ by our presupposition, and $d \leqslant a + b + c = e + b$ because $\mathbf{D} \subseteq \mathbf{A} \uplus \mathbf{B} \uplus \mathbf{C}$. If, however, $a + c > d$, then we put $e = \max\{a, c, d - b\}$. Clearly, $a \leqslant e$, $c \leqslant e$, and $d \leqslant e + b$. Next, $a \leqslant d$ and $c \leqslant d$ by hypothesis, and $d - b \leqslant d$, whence $e \leqslant d$. Finally, $e \leqslant a + c$ since $d < a + c$. □

First we eliminate the (!cut) rule.

LEMMA 5. If sequents $!\Psi; \Lambda \vdash !A$ and $!\Phi \uplus \{!A\}; \Pi \vdash C$ are derivable without using (cut) and (!cut), and $!\Theta$ obeys $!\Phi \subseteq !\Theta$, $!\Psi \subseteq !\Theta$, and $!\Theta \subseteq !\Phi \uplus !\Psi$, then the sequent $!\Theta; \Pi \vdash C$ is also derivable without using (cut) and (!cut). This statement holds both for $\mathbf{!_{\leqslant 1}L_1}$ and for $\mathbf{!^r_{\leqslant 1}L_1}$.

Proof. Notice that the first sequent, $!\Psi; \Lambda \vdash !A$, where $A = p/(q_1 \dots q_n)$, can be derived without using a cut in a unique way, by applying rules $(\vdash /)$ and $(\vdash !)$ to the $!\Psi; q_1, \dots, q_n \vdash p$ axiom.

Now we proceed by induction on the complexity of the derivation of $!\Phi \uplus \{!A\}; \Pi \vdash C$. The most interesting case is the base case, where this sequent is an axiom of the form $!\Phi \uplus \{!A\}; r_1, \dots, r_m \vdash s$. For the $\mathbf{!^r_{\leqslant 1}L_1}$ calculus, this means that the word $\hat{r}_1 \dots \hat{r}_m$ has an $\mathcal{R}_{!\Phi \uplus \{!A\}}$-total derivation π_1 in the cf-grammar $G_{!\Phi \uplus \{!A\}, s}$. Let $A = p/(q_1 \dots q_n)$. The $!\Psi; q_1, \dots, q_n \vdash p$ axiom gives an $\mathcal{R}_{!\Psi}$-total derivation of the word $\hat{q}_1 \dots \hat{q}_n$ in the cf-grammar $G_{!\Psi, p}$. Thus, we also obtain such a derivation for a nonterminal word $q_1 \dots q_n$. Denote this derivation by π_2. Consider two cases.

Case 1. Let $!A$ not occur in $!\Phi$. In the π_1 derivation, we replace each application of the $p \to q_1 \dots q_n$ rule with a copy of the π_2 derivation. Since $!\Phi \subseteq !\Theta$ and $!\Psi \subseteq !\Theta$, the new derivation π is a correct derivation in $G_{!\Theta, s}$ (the rule corresponding to the $!A$ formula was removed from π_1). On the other hand, a calculation of rule applications shows that this derivation is $\mathcal{R}_{!\Phi \uplus !\Psi}$-total, and therefore it is $\mathcal{R}_{!\Theta}$-total.

Case 2. Now let $!A$ occur (with some multiplicity) in $!\Phi$. In the π_1 derivation, we replace only one (e.g., the first) application of the $p \to q_1 \dots q_n$ rule with a copy of the π_2 derivation. The derivation π constructed in this way would be a correct derivation in $G_{!\Theta, s}$: now $!A$ belongs to $!\Theta$. On the other hand, π is $\mathcal{R}_{!\Theta}$-total.

In both cases the target sequent happens to be an axiom. For $\mathbf{!_{\leqslant 1}L_1}$, we use the same argument, but without counting rule applications.

There is one more induction base case, where the right premise is an axiom of the form $!\Phi \uplus \{!A\}; \Lambda \vdash \mathbf{1}$. This case is possible only for the $\mathbf{!_{\leqslant 1}L_1}$ calculus, and here the target sequent $!\Theta; \Lambda \vdash \mathbf{1}$ is also an axiom.

In order to prove the induction step, we consider the lowermost rule in the derivation of $!\Phi \uplus \{!A\}; \Pi \vdash C$.

Case 1. Consider $(! \vdash)$. The derivation is rebuilt by interchanging (!cut) and $(! \vdash)$. Under this transformation, the correctness conditions for (!cut) remain valid: both parts of the inclusions are extended by the $!B$ formula introduced by the $(! \vdash)$ rule. The new application of (!cut) is eliminable by the induction hypothesis.

Case 2. Consider any other one-premise rule, i.e., $(\vdash \backslash)$, $(\vdash /)$, $(\cdot \vdash)$, $(\mathbf{1} \vdash)$, or $(\vdash !)$. Denote this rule by P. The derivation is rebuilt by interchanging (!cut) and P. The P rule is kept valid because P and (!cut) operate on different sides of "$;$." Conditions on $!\Theta$ are also respected.

Case 3. Consider a two-premise rule, i.e., $(\backslash \vdash)$, $(/ \vdash)$, or $(\vdash \cdot)$. We will focus on the $(\vdash \cdot)$ rule; the others can be treated similarly. A derivation has the form

$$\dfrac{!\Psi; \Lambda \vdash !A \quad \dfrac{!\Upsilon_1; \Pi_1 \vdash C_1 \quad !\Upsilon_2; \Pi_2 \vdash C_2}{!\Phi \uplus \{!A\}; \Pi_1, \Pi_2 \vdash C_1 \cdot C_2}\,(\vdash \cdot)}{!\Theta; \Pi_1, \Pi_2 \vdash C_1 \cdot C_2}\,(!\text{cut})$$

Since $!\Phi \uplus \{!A\} \subseteq !\Upsilon_1 \uplus !\Upsilon_2$, one of the multisets $!\Upsilon_i$ contains $!A$. Let it be $!\Upsilon_2$. Then $!\Upsilon_2 =$

$!\Upsilon'_2 \uplus \{!A\}$. We have $!\Upsilon'_2 \subseteq !\Phi$ since $!\Upsilon'_2 \uplus \{!A\} \subseteq !\Phi \uplus \{!A\}$, whence $!\Upsilon'_2 \subseteq !\Theta$. There are two cases to consider.

Case 3.1. Let $!\Upsilon_1 \subseteq !\Theta$. Since $!\Theta \subseteq !\Phi \uplus !\Upsilon_1 \uplus !\Upsilon'_2$ (because $!\Theta \uplus \{!A\} \subseteq !\Psi \uplus !\Phi \uplus \{!A\} \subseteq !\Psi \uplus !\Upsilon_1 \uplus !\Upsilon'_2 \uplus \{!A\}$), the conditions of Lemma 4 hold. By this lemma, there exists $!\Xi$ such that $!\Psi \subseteq !\Xi$, $!\Upsilon'_2 \subseteq !\Xi$, $!\Xi \subseteq !\Psi \uplus !\Upsilon'_2$, $!\Xi \subseteq !\Theta$, and $!\Theta \subseteq !\Xi \uplus !\Upsilon_1$. Now we have

$$\dfrac{!\Upsilon_1; \Pi_1 \vdash C_1 \qquad \dfrac{!\Psi; \Lambda \vdash !A \quad !\Upsilon'_2 \uplus \{!A\}; \Pi_2 \vdash C_2}{!\Xi; \Pi_2 \vdash C_2}\ (!\mathrm{cut})}{!\Theta; \Pi_1, \Pi_2 \vdash C_1 \cdot C_2}\ (\vdash \cdot)$$

The new application of (!cut) is eliminable by the induction hypothesis.

Case 3.2. Let $!\Upsilon_1 \not\subseteq !\Theta$. On the other hand, $!\Upsilon_1 \subseteq !\Phi \uplus \{!A\} \subseteq !\Theta \uplus \{!A\}$. This means that $!\Upsilon_1$ outsteps $!\Theta$ using an extra copy of $!A$. In other words, $!\Upsilon_1 = !\Upsilon'_1 \uplus \{!A\}$, and multiplicities of $!A$ in $!\Upsilon'_1$ and in $!\Theta$ coincide. We prove that $!\Theta \subseteq !\Phi \uplus !\Upsilon'_1 \uplus !\Upsilon'_2$. As in Case 3.1, we have $!\Theta \subseteq !\Phi \uplus !\Upsilon_1 \uplus !\Upsilon'_2 = !\Phi \uplus !\Upsilon'_1 \uplus !\Upsilon'_2 \uplus \{!A\}$. In addition, the multiplicity of $!A$ in $!\Theta$ is the same the one in $!\Upsilon'_1$ and is therefore not greater than the one in $!\Phi \uplus !\Upsilon'_1 \uplus !\Upsilon'_2$. Thus, $!\Theta \subseteq !\Phi \uplus !\Upsilon'_1 \uplus !\Upsilon'_2$, $!\Psi \subseteq !\Theta$, $!\Upsilon'_1 \subseteq !\Theta$, $!\Upsilon'_2 \subseteq !\Theta$ (because $!\Upsilon'_i \uplus \{!A\} \subseteq !\Theta \uplus \{!A\}$). Let us apply Lemma 4 twice, with $\mathbf{B} \subseteq !\Upsilon_2$ and with $\mathbf{B} \subseteq !\Upsilon_1$. This yields two multisets $!\Xi_1$ and $!\Xi_2$, where $!\Xi_1$ satisfies the following conditions: $!\Psi \subseteq !\Xi_1$, $!\Upsilon'_1 \subseteq !\Xi_1$, $!\Xi_1 \subseteq !\Psi \uplus !\Upsilon'_1$, $!\Xi_1 \subseteq !\Theta$, and $!\Theta \subseteq !\Xi_1 \uplus !\Upsilon'_2$, and similar properties of $!\Xi_2$ are obtained by swapping $!\Upsilon'_1$ and $!\Upsilon'_2$. We rebuild the derivation in the following way:

$$\dfrac{\dfrac{!\Psi; \Lambda \vdash !A \quad !\Upsilon'_1 \uplus \{!A\}; \Pi_1 \vdash C_1}{!\Xi_1; \Pi_1 \vdash C_1} \quad \dfrac{!\Psi; \Lambda \vdash !A \quad !\Upsilon'_2 \uplus \{!A\}; \Pi_1 \vdash C_2}{!\Xi_2; \Pi_2 \vdash C_1}\ (!\mathrm{cut})}{!\Theta; \Pi_1, \Pi_2 \vdash C_1 \cdot C_2}\ (\vdash \cdot)$$

All correctness conditions for rules (!cut) and $(\vdash \cdot)$, except for $!\Theta \subseteq !\Xi_1 \uplus !\Xi_2$, have been already obtained using Lemma 4. The latter condition follows from $!\Theta \subseteq !\Xi_1 \uplus$ and $!\Upsilon'_2 \subseteq !\Xi_2$. Both applications of (!cut) here have simpler derivations of their right premises and are therefore eliminable by the induction hypothesis. □

LEMMA 6. If sequents $!\Psi; \Pi \vdash A$ and $!\Phi; \Gamma, A, \Delta \vdash C$ are derivable without using (cut) and (!cut), and if $!\Theta$ obeys $!\Phi \subseteq !\Theta$, $!\Psi \subseteq !\Theta$, and $!\Theta \subseteq !\Phi \uplus !\Psi$, then the sequent $!\Theta; \Pi \vdash C$ is also derivable without using (cut) and (!cut). This statement holds both for $\mathbf{!_{\leqslant 1}L_1}$ and for $\mathbf{!^{r}_{\leqslant 1}L_1}$.

Proof. We proceed by nested induction. The outer induction parameter is the complexity of a formula A, and the inner induction parameter is the sum of sizes of derivations of $!\Psi; \Pi \vdash A$ and $!\Phi; \Gamma, A, \Delta \vdash C$.

In the induction base case, the above sequents are axioms. Moreover, they are axioms of the form $!\Psi; p_1, \ldots, p_n \vdash q$ and $!\Phi; r_1, \ldots, r_i, q, r_{i+1}, \ldots, r_m \vdash s$. Hence, there exist two derivations: a derivation π_1 of the word $\hat{p}_1 \ldots \hat{p}_n$ in the cf-grammar $G_{!\Psi,q}$ and a derivation π_2 of the word $\hat{r}_1 \ldots \hat{r}_i q \hat{r}_{i+1} \ldots \hat{r}_m$ in the cf-grammar $G_{!\Phi,s}$ (in the second derivation, we have removed the

application of the rule $q \to \hat{q}$). By placing π_1 after π_2, we obtain a derivation π of the word $\hat{r}_1 \dots \hat{r}_i \hat{p}_1 \dots \hat{p}_n \hat{r}_{i+1} \dots \hat{r}_m$ in the united cf-grammar $G_{!\Theta,s}$ (since $!\Phi \subseteq !\Theta$ and $!\Psi \subseteq !\Theta$, this cf-grammar contains all rules from both $G_{!\Phi,s}$ and $G_{!\Psi,q}$). Moreover, in the case of $\mathbf{!}^{\mathbf{r}}_{\leqslant 1}\mathbf{L_1}$, the derivation π_1 is $\mathcal{R}_{!\Psi}$-total and the derivation π_2 is $\mathcal{R}_{!\Phi}$-total. Since $!\Theta \subseteq !\Phi \uplus !\Psi$, π is an $\mathcal{R}_{!\Theta}$-total derivation. Thus, the target sequent $!\Theta; r_1, \dots, r_i, p_1, \dots, p_n, r_{i+1}, \dots, r_m \vdash s$ is an axiom of the given calculus.

In order to prove the induction step, we consider the lowermost rules applied in the derivations of the given sequents. Such an application will be referred to as *principal* if it introduces the formula A which is removed by the cut rule. Below are three possible (and interleaving) cases.

Case 1. Suppose that the derivation of the left premise is concluded by a nonprincipal application of some rule. This rule should introduce a connective to the left-hand side of the sequent.

Case 1.1. Consider a rule $(\setminus \vdash)$ or $(/ \vdash)$.

$$\dfrac{\dfrac{!\Upsilon_1; \Pi_2 \vdash E \quad !\Upsilon_2; \Pi_1, F, \Pi_3 \vdash A}{!\Psi; \Pi_1, \Pi_2, E\backslash F, \Pi_3 \vdash A}\,(\setminus \vdash) \qquad !\Phi; \Gamma, A, \Delta \vdash C}{!\Theta; \Gamma, \Pi_1, \Pi_2, E\backslash F, \Pi_3, \Delta \vdash C}\,(\text{cut})$$

We have $!\Upsilon_1 \subseteq !\Theta$, $!\Upsilon_2 \subseteq !\Theta$, $!\Phi \subseteq !\Theta$, and $!\Theta \subseteq !\Upsilon_1 \uplus !\Upsilon_2 \uplus !\Phi$. By Lemma 4, there exists $!\Xi$ such that $!\Phi \subseteq !\Xi$, $!\Upsilon_2 \subseteq !\Xi$, $!\Xi \subseteq !\Phi \uplus !\Upsilon_2 \subseteq !\Xi$, $!\Xi \subseteq !\Theta$, and $!\Theta \subseteq !\Upsilon_1 \uplus !\Xi$. The following derivation is correct:

$$\dfrac{!\Upsilon_1; \Pi_2 \vdash E \qquad \dfrac{!\Upsilon_2; \Pi_1, F, \Pi_3 \vdash A \quad !\Phi; \Gamma, A, \Delta \vdash C}{!\Xi; \Gamma, \Pi_1, F, \Pi_3, \Delta \vdash C}\,(\text{cut})}{!\Theta; \Gamma, \Pi_1, \Pi_2, E\backslash F, \Pi_3, \Delta \vdash C}\,(\setminus \vdash)$$

The new application of (cut) has a smaller summary depth of derivations of premises and is therefore eliminable by the induction hypothesis.

Case 1.2. Consider a rule $(\cdot \vdash)$. The derivation is rebuilt (decreasing the summary depth of derivations of premises of (cut)) by interchanging (cut) and $(\cdot \vdash)$. The correctness conditions for (cut) remain valid.

Case 1.3. Consider $(! \vdash)$. The derivation is rebuilt similarly to Case 1.2, by interchanging (cut) and $(! \vdash)$. The new (cut) is correct because adding $\{!B\}$ keeps multiset inclusions true.

Case 2. The derivation of the right premise is concluded by a nonprincipal application of some rule. This rule cannot be $(\vdash !)$ because the right premise should contain a formula A to the right of the " ; " symbol.

Case 2.1. Consider a one-premise rule P. As in Cases 1.2 and 1.3, we interchange P and (cut), decreasing the depth. The correctness of (cut) is preserved.

Case 2.2. Consider a two-premise rule (i.e., $(\vdash \cdot)$, $(\setminus \vdash)$, or $(/ \vdash)$). The derivation is rebuilt similarly to Case 1.1.

Case 3. Finally, each of $!\Psi; \Pi \vdash A$ and $!\Phi; \Gamma, A, \Delta \vdash C$ either is an axiom or is derived by a principal application of a rule. The case where A is a variable and both sequents are axioms has

already been considered (induction base). Let us consider other possibilities for A.

Case 3.1. Let $A = \mathbf{1}$. In the $\mathbf{!^r_{\leqslant 1}L_1}$ calculus, we see that the target sequent coincides with the premise of the $(\mathbf{1} \vdash)$ rule, i.e., it has already been derived, and the cut is unnecessary. For $\mathbf{!_{\leqslant 1}L_1}$, the argument is a bit more complicated. The premise of $(\mathbf{1} \vdash)$ is of the form $!\Phi; \Gamma, \Delta \vdash C$, and correctness conditions yield only the inclusion $!\Phi \subseteq !\Theta$, i.e., $!\Theta = !\Phi \uplus !\Psi$ for some $!\Psi$. However, adding $!\Psi$ to all sequents in the derivation of $!\Phi; \Gamma, \Delta \vdash C$ does not spoil the rules and, in the case of $\mathbf{!_{\leqslant 1}L_1}$, also keeps the axioms valid. (In other words, the weakening rule is admissible in $\mathbf{!_{\leqslant 1}L_1}$ with no cut.) This gives cut-free derivability of $!\Theta; \Gamma, \Delta \vdash C$.

Case 3.2. Let $A = E\backslash F$ or $A = F/E$. We have the following:

$$\dfrac{\dfrac{!\Psi; E, \Pi \vdash F}{!\Psi; \Pi \vdash E\backslash F}\,(\vdash \backslash) \quad \dfrac{!\Upsilon_1; \Gamma_2 \vdash E \quad !\Upsilon_2; \Gamma_1, F, \Delta \vdash C}{!\Phi; \Gamma_1, \Gamma_2, E\backslash F, \Delta \vdash C}\,(\backslash \vdash)}{!\Theta; \Gamma_1, \Gamma_2, \Pi, \Delta \vdash C}\,(\text{cut})$$

Moreover, $!\Psi \subseteq !\Theta$, $!\Upsilon_1 \subseteq !\Theta$, $!\Upsilon_2 \subseteq !\Theta$, and $!\Theta \subseteq !\Psi \uplus !\Upsilon_1 \uplus !\Upsilon_2$. Let us apply Lemma 4 and construct $!\Xi$ such that $!\Psi \subseteq !\Xi$, $!\Upsilon_2 \subseteq !\Xi$, $!\Xi \subseteq !\Psi \uplus !\Upsilon_2$, $!\Xi \subseteq !\Theta$, and $!\Theta \subseteq !\Xi \uplus !\Upsilon_1$. Now we rebuild the derivation, splitting the cut into two cuts with smaller complexity of the formula being removed (outer induction hypothesis):

$$\dfrac{!\Upsilon_1; \Gamma_2 \vdash E \qquad \dfrac{!\Psi; E, \Pi \vdash F \quad !\Upsilon_2; \Gamma_1, F, \Delta \vdash C}{!\Xi; \Gamma_1, E, \Pi, \Delta \vdash C}\,(\text{cut})}{!\Theta; \Gamma_1, \Gamma_2, \Pi, \Delta \vdash C}\,(\text{cut})$$

Case 3.3. Let $A = E \cdot F$. Similar to Case 3.2.

Case 3.4. Let $A = !B$. The (cut) rule can be replaced by (!cut) with the same correctness condition. The latter is eliminable by Lemma 5. □

3. $\mathcal{R}$-TOTAL DERIVABILITY IN CF-GRAMMARS

The $\mathbf{!_{\leqslant 1}L_1}$ and $\mathbf{!^r_{\leqslant 1}L_1}$ calculi have nontrivial sets of axioms: checking whether a given sequent is an axiom involves solving derivability problems in cf-grammars. For $\mathbf{!_{\leqslant 1}L_1}$, this is the usual problem of derivability of a word in a cf-grammar. It is well-known that it is solvable in polynomial time. For $\mathbf{!^r_{\leqslant 1}L_1}$, we need to construct an algorithm for checking $\mathcal{R}$-total derivability.

THEOREM 7. The problem of $\mathcal{R}$-total derivability of a given word w in a given cf-grammar G for a given multiset $\mathcal{R}$ of rules belongs to the NP class.

We will consider a cf-grammar $G = (N, \Sigma, \mathcal{P}, S)$, where $N = \{A_1, \ldots, A_m\}$ and $\mathcal{P} = \{p_1, \ldots, p_r\}$ (i.e., $m = |N|$ and $r = |\mathcal{P}|$). In this section we denote terminals by lowercase letters and nonterminals by uppercase ones (in particular, we will use S instead of s). Let l be the length of the longest right-hand side of a rule from $\mathcal{P}$. Rules of the form $A \to \alpha$ will be called A-rules.

By $|w|$ we denote the length of a word w; $|w|_a$ stands for the number of symbols a in this word. The *projection* $\mathrm{Pr}_\Sigma(\alpha)$ of a word α onto an alphabet Σ is the word which is obtained by removing all symbols not belonging to Σ from α.

It is convenient to represent derivations in a cf-grammar (starting from the initial nonterminal S) as *trees:* inner vertices correspond to applications of rules and the yield is the word derived. Vertices located at a distance k from the root form the kth *level* of the tree.

If the cf-grammar has no rules of the form $A \to B$ (chain rules) or $A \to \varepsilon$ (empty rules), then the length of any derivation $S \Rightarrow^* w$, where $|w| = n$, is not greater than $2n - 1$. Therefore, the problem of $\mathcal{R}$-total derivability is solved by the following simple algorithm: one has to guess a derivation of length not more than $2n-1$ and check it for correctness. For an arbitrary cf-grammar, there are two difficulties.

The first challenge is that there could be cycles in the grammar, that is, derivations of the form $A \Rightarrow^+ A$. In the presence of cycles, an algorithm which just guesses the sequence of rules might enter a dead loop. Removing cycles keeps derivability but could violate totality. For example, in a cf-grammar with rules $S \to T$, $T \to S$, and $S \to a$, the word a has a total derivation $S \Rightarrow T \Rightarrow S \Rightarrow a$, but each total derivation includes the $S \Rightarrow^+ S$ cycle.

The second challenge is that even if cycles are missing, some derivations could have exponential length. For example, let a grammar have m nonterminals, $A_1 = S$, A_2, …, A_m, and the following rules: $A_1 \to A_2A_2$, $A_2 \to A_3A_3$, …, $A_{m-1} \to A_mA_m$, $A_m \to \varepsilon$. Then $A_1 \Rightarrow^* \varepsilon$, but the length of this derivation is equal to $1 + 2 + 4 + \cdots + 2^{m-1} = 2^m - 1$, since each application of an $A_i \to A_{i+1}A_{i+1}$ rule doubles the number of nonterminals. Therefore, an algorithm which attempts to guess the derivation directly will have at least exponential complexity.

The examples given above show that a polynomial algorithm should somehow take into account empty rules and cycles.

For a nonterminal $A \in N$, let $\mathrm{E}(A)$ denote the set of rules which can be used in some derivation $A \Rightarrow^* \varepsilon$. Notice that we do not require all rules from $\mathrm{E}(A)$ to be used in one derivation. It is sufficient that each rule has its own derivation. In order to compute $\mathrm{E}(A)$, we define the following auxiliary sets:

$$\begin{aligned} \mathrm{E}_0(A) &= \varnothing, \\ \mathrm{E}_{i+1}(A) &= \mathrm{E}_i(A) \cup \bigcup \big\{\, \mathrm{E}_i(B_1) \cup \cdots \cup \mathrm{E}_i(B_k) \cup \{\, p_j \,\} : \\ &\qquad p_j = A \to B_1 \ldots B_k,\ \mathrm{E}_i(B_t) \neq \varnothing \text{ for } t = 1, \ldots, k \,\big\}. \end{aligned} \tag{1}$$

In the set $\mathrm{E}_i(A)$, we record all rules which can be used in a derivation $A \Rightarrow^* \varepsilon$ of length not greater than i, and $\mathrm{E}(A) = \bigcup\limits_{i=1}^{\infty} \mathrm{E}_i(A)$. Clearly, $\mathrm{E}(A) \neq \varnothing$ if and only if $A \Rightarrow^* \varepsilon$.

Algorithm E computation of sets $\mathrm{E}(A)$
Input: $G = (N, \Sigma, P, S)$ is a cf-grammar
Output: $\mathrm{E}(A)$ for all $A \in N$

1: Let $\mathrm{E}_0(A) = \varnothing$ for each $A \in N$; let $i = 0$
2: **do**
3: Compute $\mathrm{E}_{i+1}(A)$ for each $A \in N$ by formula (1); $i = i + 1$
4: **while** $\mathrm{E}_i(A) \neq \mathrm{E}_{i-1}(A)$ for some $A \in N$
5: Let $\mathrm{E}(A) = \mathrm{E}_i(A)$ for each $A \in N$

LEMMA 8. Algorithm E computes all sets $\mathrm{E}(A)$ in polynomial time.

Proof. That the above-given algorithm is correct follows from the two statements below, which are proved by induction on i.

—If $p_j \in \mathrm{E}_i(A)$, then there exists a derivation $A \Rightarrow^* \varepsilon$, in which the rule p_j is used.

—If there exists a derivation $A \Rightarrow^i \varepsilon$, in which the rule p_j is used, then $p_j \in \mathrm{E}(A)$.

These statements guarantee that when the algorithm stops, exactly the required sets $\mathrm{E}(A)$ have been computed. Polynomiality of the running time is checked directly: in the loop in lines 2-4, the sets $\mathrm{E}_i(A)$ could only increase, and so the number of iterations does not exceed mr. □

Next, we are going to show how to find rules which can be used in derivations of the form $A \Rightarrow^+ B$, e.g., in cycles $A \Rightarrow^+ A$. For nonterminals $A, B \in N$, let $\mathrm{C}(A, B)$ denote the set of rules which may be used in some derivation $A \Rightarrow^+ B$. As in the case of sets $\mathrm{E}(A)$, we do not require all rules from the set $\mathrm{C}(A, B)$ to be used in the same derivation. In order to compute $\mathrm{C}(A, B)$, we define the following auxiliary sets:

$$\begin{aligned}
\mathrm{C}_0(A, B) &= \varnothing, \\
\mathrm{C}_{i+1}(A, B) &= \mathrm{C}_i(A, B) \\
&\quad \cup \bigcup \big\{ \mathrm{E}(B_1) \cup \cdots \cup \mathrm{E}(B_{q-1}) \cup \mathrm{C}_i(B_q, B) \cup \mathrm{E}(B_{q+1}) \cup \ldots \\
&\quad \cup \mathrm{E}(B_k) \cup \{ p_j \} : \ p_j = A \to B_1 \ldots B_k, \ 1 \leqslant q \leqslant k, \\
&\quad \mathrm{C}_i(B_q, B) \neq \varnothing \text{ or } B_q = B, \\
&\quad \mathrm{E}(B_t) \neq \varnothing \text{ for } t = 1, \ldots, q-1, q+1, \ldots, k \big\}.
\end{aligned} \tag{2}$$

At each step, the set $\mathrm{C}_i(A, B)$ is extended by rules from longer derivations $A \Rightarrow^+ B$. Such derivations, up to the order of rules, are of the form $A \Rightarrow B_1 \ldots B_{q-1} B_q B_{q+1} \ldots B_k \Rightarrow^* B_q \Rightarrow^* B$. This derivation uses the rule $p_j = A \to B_1 \ldots B_{q-1} B_q B_{q+1} \ldots B_k$, all rules from derivations $B_t \Rightarrow^* \varepsilon$, i.e., from $\mathrm{E}(B_t)$ $(t \neq q)$, and also rules from $B_q \Rightarrow^* B$ if the latter derivation is nontrivial. Since that derivation is shorter, the rules used in it have already been registered in $\mathrm{C}_i(B_q, B)$. As the set of all rules is finite, the extension of sets $\mathrm{C}_i(A, B)$ stabilizes. Let $\mathrm{C}(A, B) = \bigcup\limits_{i=1}^{\infty} \mathrm{C}_i(A, B)$.

Algorithm C — computation of sets $\mathrm{C}(A, B)$
Input: $G = (N, \Sigma, P, S)$ is a cf-grammar
Output: $\mathrm{C}(A, B)$ for all $A, B \in N$
1: Compute $\mathrm{E}(A)$ for each $A \in N$ (Algorithm E)
2: Let $\mathrm{C}_0(A, B) = \varnothing$ for each $A, B \in N$. Let $i = 0$
3: **do**

4: Compute $\mathrm{C}_{i+1}(A,B)$ for each $A,B \in N$ by formula (2); $i = i+1$
5: **while** $\mathrm{C}_i(A,B) \neq \mathrm{C}_{i-1}(A,B)$ for some $A,B \in N$
6: Let $\mathrm{C}(A,B) = \mathrm{C}_i(A,B)$ for each $A,B \in N$

LEMMA 9. Algorithm C computes all sets $\mathrm{C}(A,B)$ in polynomial time.

Proof. That the above-given algorithm is correct follows from the following two statements.

—If $p_j \in \mathrm{C}_i(A,B)$, then there exists a derivation $A \Rightarrow^+ B$, in which the rule p_j is used.

—If there exists a derivation $A \Rightarrow^i B$, $i > 0$, where the rule p_j is used, then $p_j \in \mathrm{C}(A,B)$.

Both statements are verified by induction on i. These statements guarantee that when the algorithm stops, exactly the required sets $\mathrm{C}(A,B)$ have been computed. In the loop in lines 3-5, the sets $\mathrm{C}_i(A,B)$ could only increase, hence the number of iterations of this loop does not exceed $m^2 r$. Applying formula (2) and checking the loop condition also require polynomial time. □

Algorithms E and C are based on standard algorithms for removing empty and chain rules [10].

In what follows, we will use *generalized* multisets, in which elements may have multiplicity ∞. For ∞, the following holds: $\infty \geqslant n$ and $\infty + n = \infty$ for each $n \in \mathbb{N}$. In the definition of $\mathcal{R}$-totality ($\alpha \Rightarrow^*_{\mathcal{R}} \beta$), however, $\mathcal{R}$ will still be a multiset without elements of multiplicity ∞. When implemented, ∞ is represented by a Boolean flag. For a set M, by M^∞ we denote the multiset which includes exactly the elements of M, each with multiplicity ∞.

The next lemma explains the meaning of sets $\mathrm{C}(A,A)$.

LEMMA 10. If $\alpha \Rightarrow^*_{\mathcal{R}} \beta$, and this derivation includes nonterminal A, $\mathcal{R}' \subseteq \mathcal{R} \uplus \mathrm{C}(A,A)^\infty$, and $\mathcal{R}'$ does not include elements with multiplicity ∞, then $\alpha \Rightarrow^*_{\mathcal{R}'} \beta$.

Proof. Let a derivation π_1 be given, and let it be of the form $\alpha \Rightarrow^* \gamma A \delta \Rightarrow^* \beta$. For each rule $p_i \in \mathrm{C}(A,A)$, there exists a derivation $A \Rightarrow^+ A$, in which p_i is used. Adding all these derivations to π_1 as many times as necessary, we obtain the desired derivation $\pi_2 : \alpha \Rightarrow^* \gamma A \delta \Rightarrow^* \gamma A \delta \Rightarrow^* \beta$. □

Let us formulate, without proofs, two simple properties of derivations yielding ε.

LEMMA 11. Let T be a derivation tree with yield ε, which does not contain cycles. Then the number of vertices on each branch of T does not exceed $m+1$, and the number of vertices at each level of T does not exceed l^m.

LEMMA 12. Let a word $\beta \in N^*$ be obtained from another word $\alpha \in N^*$ by the permutation of symbols. Then a derivation $\alpha \Rightarrow^*_{\mathcal{R}} \varepsilon$ exists if and only if $\beta \Rightarrow^*_{\mathcal{R}} \varepsilon$.

Now we can describe a nondeterministic polynomial algorithm which solves the $\mathcal{R}$-derivability problem for the empty word.

Algorithm RE — $\mathcal{R}$-total derivability of the empty word
Input: $G = (N, \Sigma, P, S)$ is a cf-grammar,
A_{i_0} is a nonterminal,
$\mathcal{R}$ is a multiset of rules
Output: "yes," if $A \Rightarrow^*_{\mathcal{R}} \varepsilon$, and "no" otherwise
1: **if** $\mathcal{R}$ includes a rule which contains a terminal, **then return** "no" **endif**
2: Compute sets $\mathrm{C}(A,A)$ for all $A \in N$ (Algorithm C)

Create counters $c_1, \ldots, c_m, c'_1, \ldots, c'_m$, let $c_{i_0} = 1$, initialize others with 0
Let $\mathcal{D}$ be a generalized multiset of rules, $\mathcal{D} = \varnothing$
Let N_1 be a set of nonterminals, $N_1 = \{ A_{i_0} \}$
for $t = 1$ **to** m **do**
    for $i = 1$ **to** m **do** $c'_i = 0$; **end for**
    for all $c_i \neq 0$ **do**
        Let $A_i \to \beta_1, \ldots, A_i \to \beta_q$ be all A_i-rules
        Guess numbers $n_1, \ldots, n_q \geqslant 0$ such that $n_1 + \cdots + n_q = c_i$
        for all $n_k \neq 0$ **do** $\mathcal{D} = \mathcal{D} \uplus \{ A_i \to \beta_k \}^{n_k}$ **end for**
        for $j = 1$ **to** m **do** $c'_j = c'_j + \sum_{k=1}^{q} n_k \cdot |\beta_k|_{A_j}$ **end for**
    end for
    for $i = 1$ **to** m **do** $c_i = c'_i$ **end for**
    for $c_i \neq 0$ **do** $N_1 = N_1 \cup \{ A_i \}$ **end for**
end for
$\mathcal{D} = \mathcal{D} \uplus \left(\bigcup \{ \mathrm{C}(A, A) : A \in N_1 \} \right)^{\infty}$
if $\mathcal{R} \subseteq \mathcal{D}$ **and** $c_i = 0$ for $i = 1, \ldots, m$ **then**
    return "yes"
else
    return "no"
end if

Here $\{p\}^{n_k}$ is a multiset which includes only one rule p, with multiplicity n_k.

First, the algorithm builds a derivation tree $A_{i_0} \Rightarrow^* \varepsilon$ without cycles. During this procedure, it registers the rules used in $\mathcal{D}$ and the nonterminals that occur in N_1. The tree is built level by level. The number of vertices at a level may be exponential, and the algorithm does not keep the level in an explicit form. Instead, for each nonterminal A_i it keeps, in counter c_i, the number of its occurrences at the current level (due to Lemma 12, the order of nonterminals does not matter), and at each step, for each A_i-rule the algorithm guesses how much times that rule should be used. Next, the algorithm adds applications of all rules from sets $\mathrm{C}(A, A)$, where $A \in N_1$, with unbounded multiplicity (by Lemma 10). Finally, the algorithm checks that it has guessed a correct derivation. The condition $\mathcal{R} \subseteq \mathcal{D}$ guarantees that the derivation is $\mathcal{R}$-total, and $c_i = 0$ ensures that the resulting word is empty.

LEMMA 13. Algorithm RE accepts its input if and only if there exists a derivation $A_{i_0} \Rightarrow^*_{\mathcal{R}} \varepsilon$.

Proof. Suppose that there exists a derivation $A_{i_0} \Rightarrow^*_{\mathcal{R}} \varepsilon$ and let T be the corresponding derivation tree. If there is a branch in T which includes a vertex u and its successor v marked with the same nonterminal A, then we transform T by replacing the subtree with root u by the subtree with root v. Repeating this procedure, we obtain a tree T' on each branch of which all nonterminals are different. By Lemma 11, each branch of T' contains at most $m + 1$ vertices, and each level contains at most l^m vertices. The derivation T' may no longer be $\mathcal{R}$-total, but the rules used in the $A \Rightarrow^+ A$ subderivations will be taken into account in the $\mathrm{C}(A, A)$ sets.

We have to prove that there exists an accepting run of Algorithm RE. Let the tth level of the

tree T' contain a word α_t. By induction on t, we show that for each $t = 0, 1, \ldots, m$ there exists a run of Algorithm RE such that after t iterations of the loop in lines 6-16, the following properties hold:

—$c_i = |\alpha_t|_{A_i}$ for each $i = 1, \ldots, m$;

—$\mathcal{D}$ includes rules used in the derivation $A_{i_0} \Rightarrow^* \alpha_t$, with correct multiplicities;

—N_1 includes exactly the nonterminals which occur in the derivation $A_{i_0} \Rightarrow^* \alpha_t$.

When the loop in lines 6-16 terminates, the set N_1 contains all nonterminals which occur in the tree T', and the multiset $\mathcal{D}$ contains all rules used in T', with correct multiplicities. In line 17, the algorithm adds to $\mathcal{D}$, with multiplicity ∞, all rules which could have been used due to cycles (by Lemma 10). This includes rules from $\mathcal{R}$ which were removed when T was replaced by T'. Hence, in line 18, the condition $\mathcal{R} \subseteq \mathcal{D}$ holds. Each counter c_i stores the number of occurrences of a nonterminal A_i in the yield of T'. Since the yield is empty, $c_i = 0$ for all i. The algorithm accepts its input.

Now suppose that there exists an accepting run of the algorithm. Then for each $t = 0, 1, \ldots, m$ there exists a word α_t, and we can construct a partial derivation tree with the root A_{i_0} up to level t, the concatenation of labels at which is equal to α_t. Moreover, after t iterations of the loop in lines 6-16, the above properties of c_i, $\mathcal{D}$, and N_1 hold (induction on t). Since N_1 includes only those nonterminals that occur in the derivation, in line 17 we add to $\mathcal{D}$ all the rules which could augment the derivation (by Lemma 10). The algorithm accepts its input, so we have $\mathcal{R} \subseteq \mathcal{D}$, and also $c_i = 0$ for all i. This means that the derivation $A_{i_0} \Rightarrow^*_{\mathcal{R}} \varepsilon$ has been constructed.

The algorithm constructs m levels of the tree (except the root), and by Lemma 11, each level contains at most l^m vertices. Thus, the length of a binary representation for c_i is $O(m \log_2 l)$, and the loop in line 12 is executed in polynomial time. The loop in lines 6-16 performs m iterations. Hence the overall running time of the algorithm is also polynomial. □

We formulate yet another property of derivations.

LEMMA 14. Suppose a derivation $A \Rightarrow^* \alpha$ has no cycles and does not use empty rules. Then the length of this derivation does not exceed $2mn - m - n$, where $n = |\alpha|$.

In order to prove this property, it is sufficient to obtain an upper bound on the number of vertices in a derivation tree with more than one child $(n-1)$ and of those with one child $((2n-1)(m-1))$.

Now we describe a nondeterministic polynomial algorithm which solves the $\mathcal{R}$-total derivability problem in the general case.

Algorithm R — $\mathcal{R}$-total derivability

Input: $G = (N, \Sigma, P, S)$ is a cf-grammar,
w is a word in alphabet Σ,
$\mathcal{R}$ is a multiset of rules

Output: "yes," if there exists a derivation $S \Rightarrow^*_{\mathcal{R}} w$,
"no" otherwise

1: Guess a number $k \leqslant 2mn - m - n$
2: Let $\alpha = S$

3: Let $\mathcal{D}$ be a generalized multiset of rules, $\mathcal{D} = \varnothing$
4: Let N_1 be a set of nonterminals, $N_1 = \{ S \}$
5: Compute sets $\mathrm{C}(A, A)$ for all $A \in N$ (Algorithm C)
6: **for** $i = 1$ **to** k **do**
7: **if** $\alpha \in \Sigma^*$ **then** Break the loop **endif**
8: Guess a nonterminal A and its occurrence in word α
9: Guess a rule $p_j = A \to \beta$ and
10: Replace the guessed occurrence A in α by β
11: $\mathcal{D} = \mathcal{D} \uplus \{ p_j \}$. Add all non-terminals from β to N_1
12: **end for**
13: Let $\alpha = X_1 X_2 \dots X_t$, where $X_i \in \Sigma \cup N$
14: **for** $i = 1$ **to** t **do**
15: **if** $X_i \in N$ **then**
16: Guess a multiset of rules $\mathcal{R}' \subseteq \mathcal{R}$
17: **if** $X_i \Rightarrow^*_{\mathcal{R}'} \varepsilon$ (Algorithm RE) **then**
18: Remove X_i from α
19: $\mathcal{D} = \mathcal{D} \uplus \mathcal{R}'$
20: Add nonterminals occurring in the derivation $X_i \Rightarrow^*_{\mathcal{R}'} \varepsilon$ to N_1
21: **else**
22: **return** "no"
23: **end if**
24: **end if**
25: **end for**
26: $\mathcal{D} = \mathcal{D} \cup \left(\bigcup \{ \mathrm{C}(A, A) : A \in N_1 \} \right)^\infty$
27: **if** $\alpha = w$ **and** $\mathcal{R} \subseteq \mathcal{D}$ **then return** "yes" **else return** "no" **endif**

First, the algorithm guesses a part of the derivation without empty rules and cycles. Next, it derives empty words from the remaining nonterminals. The rules used are registered in $\mathcal{D}$, and the nonterminals that occur are registered in N_1. Finally, the algorithm augments the derivation with cycles for nonterminals from N_1. The input is accepted if and only if the word w is derived and $\mathcal{R} \subseteq \mathcal{D}$.

LEMMA 15. Algorithm R accepts its input if and only if $S \Rightarrow^*_{\mathcal{R}} w$.

Proof. Let $S \Rightarrow^*_{\mathcal{R}} w$. We prove that there exists an accepting run of the algorithm. The derivation will be transformed so that all derivations of the empty word are performed at the end. That is, the derivation is of the form $S \Rightarrow^* \gamma \Rightarrow^* w$, where $\mathrm{Pr}_\Sigma(\gamma) = w$, and if the empty word is derived from some nonterminal A, then the $A \Rightarrow^* \varepsilon$ part is entirely inside the $\gamma \Rightarrow^* w$ derivation.

If there are cycles in the $S \Rightarrow^* \gamma$ derivation, then we remove them. After this operation, the word γ is still derived from S, but not necessarily with the same multiset of rules. By Lemma 14, the length of the new derivation is not greater than $2mn - m - n$. In line 1, the algorithm will guess this length and store it in k.

Let a given derivation $S \Rightarrow^k \gamma$ be of the form $S \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \cdots \Rightarrow \gamma_k$, where $\gamma_k = \gamma$. By induction on i, we show that there exists a run of the algorithm such that after i iterations of the loop in lines 6-12, the following three conditions hold:

—$\alpha = \gamma_i$;

—$\mathcal{D}$ contains exactly the rules from $S \Rightarrow^* \gamma_i$, with correct multiplicities;

—N_1 is the set of nonterminals used in the derivation $S \Rightarrow^* \gamma_i$.

When the loop terminates, we obtain a word $\alpha = \gamma$ such that $\mathrm{Pr}_\Sigma(\alpha) = w$ and the empty word is derivable from each nonterminal occurring in α. After that, in the loop in lines 14-25, the algorithm will register the rules used in derivations of empty words. Indeed, the derivation $\gamma \Rightarrow^* w$ can be rebuilt into $\gamma \Rightarrow^* \gamma'_1 \Rightarrow^* \gamma'_2 \Rightarrow^* \cdots \Rightarrow^* \gamma'_t = w$, where $\gamma'_i = \eta X_i \theta \Rightarrow^* \eta\theta = \gamma'_{i+1}$ if $X_i \in N$ (otherwise $\gamma'_{i+1} = \gamma'_i$). In line 16, the algorithm guesses the multiset $\mathcal{R}'$ necessary to perform the derivation $X_i \Rightarrow^*_{\mathcal{R}'} \varepsilon$. There exists a computation such that after the loop in lines 14-25 terminates, the multiset $\mathcal{D}$ contains, as a minimum, all rules from the derivation $S \Rightarrow^* \gamma \Rightarrow^* w$ with required multiplicities.

Finally, in line 26, the algorithm augments $\mathcal{D}$ by using cycles. After that, $\alpha = w$, $\mathcal{R} \subseteq \mathcal{D}$, and the algorithm accepts its input.

The converse statement (if the algorithm accepts its input, then $S \Rightarrow^*_R w$) is checked similarly to Lemma 13. Polynomiality of the running time of the algorithm is checked directly by applying Lemma 14. □

4. COMPLEXITY OF THE TOTAL DERIVABILITY PROBLEM

It turns out that for cf-grammars the total derivability problem is algorithmically harder then the usual derivability problem.

THEOREM 16. The total derivability problem for a given cf-grammar G and a given word w is NP-hard. Therefore, the $\mathcal{R}$-total derivability problem (where $\mathcal{R}$ is an input parameter) is also NP-hard.

Proof. We construct a reduction to the total derivability problem of a well-known NP-complete problem 3-PARTITION [11]: For a tuple of natural numbers (given in unary notation) $a_1, \ldots, a_{3n}$, whose sum is nb, determine whether there exists a partition of the tuple into triples, the sum in each of which equals b.

For a given tuple of numbers, we construct a cf-grammar with the following rules:

$$S \to AAAdS, \quad S \to \varepsilon, \quad A \to A_i \text{ (for } i = 1, \ldots, 3n),$$
$$A_i \to c^{a_i} \text{ (for } i = 1, \ldots, 3n).$$

The word $(c^b d)^n$ has a total derivation in this grammar if and only if there exists the required partition into triples. □

However, if we pass on to more general classes of generative grammars, then the total derivability problem and the usual derivability problem will have the same complexity. Recall that in an arbitrary generative grammar, $G = (N, \Sigma, \mathcal{P}, S)$ rules from $\mathcal{P}$ are of the form $\alpha \to \beta$, where

$\alpha, \beta \in (N \cup \Sigma)^*$, $\alpha \neq \varepsilon$. A grammar is said to be *noncontracting* if it is always true that $|\alpha| \leqslant |\beta|$, and *context-sensitive* if $\alpha = \eta A \theta$ and $\beta = \eta\gamma\theta$, where $A \in N$, $\eta, \theta, \gamma \in (N \cup \Sigma)^*$, $\gamma \neq \varepsilon$.

THEOREM 17. The total derivability problem is m-complete for arbitrary generative grammars and is PSACE-complete for noncontracting and context-sensitive grammars.

Proof. Below we present a nondeterministic algorithm which solves the problem of $\mathcal{R}$-total derivability.

Algorithm RG $\mathcal{R}$-total derivability
Input: $G = (N, \Sigma, P, S)$ is a generative grammar,
$a_1 a_2 \ldots a_n$ is a word in alphabet Σ,
$\mathcal{R}$ is a multiset of rules
Output: "yes," if there exists a derivation $S \Rightarrow^*_{\mathcal{R}} w$,
"no" otherwise

1: $v = w$; let $\mathcal{R}_1$ be a multiset of rules, $\mathcal{R}_1 = \varnothing$
2: **while** $v \neq S$ **do**
3: Guess a number $i \in \{ 1, \ldots, r \}$
4: **if** v does not contain occurrences of β_i **then return** "no" **endif**
5: Guess an occurrence of β_i in v and replace this occurrence with α_i
6: $\mathcal{R}_1 = \mathcal{R}_1 \uplus \{ \alpha_i \to \beta_i \}$
7: **end while**
8: **if** $\mathcal{R} \subseteq \mathcal{R}_1$ **then return** "yes" **else return** "no" **endif**

It is straightforward to verify that if $S \Rightarrow^*_{\mathcal{R}} w$ then the algorithm returns "yes," and otherwise either it returns "no" or does not halt. In the case of noncontracting (in particular, context-sensitive) grammars, we have $|v| \leqslant |w|$. In addition, we can bound the size of $\mathcal{R}_1$ by the size of $\mathcal{R}$, in order to exclude too big multiplicities in $\mathcal{R}_1$. Therefore, the $\mathcal{R}$-totality problem in this case belongs to the class NPSPACE and, hence, to the class PSPACE.

We prove that the problem of total (and, therefore, $\mathcal{R}$-total) derivability is m-hard in the general case and is PSPACE-hard for context-sensitive grammars. The appropriate complexity estimations are known for the usual derivability problem, and we will reduce it to the total derivability one.

Let $G = (N, \Sigma, \mathcal{P}, S)$ be an arbitrary generative grammar, in which $\Sigma = \{ a_1, \ldots, a_k \}$ and $\mathcal{P} = \{ \alpha_1 \to \beta_1, \ldots, \alpha_r \to \beta_r \}$. Given the grammar G, we construct a new grammar $G' = (N', \Sigma', \mathcal{P}', S')$ in the following way.

First, let $N' = N \cup \{ A_a : a \in \Sigma \} \cup \{ S' \}$, where S' is a new initial nonterminal. For an arbitrary word $\alpha \in (N \cup \Sigma)^*$, by α' we denote the word which is obtained from α by replacing each terminal $a \in \Sigma$ with a nonterminal A_a. Second, put $\Sigma' = \Sigma \cup \{ b, c \}$, where $b, c \notin \Sigma$. Third, we define the set of rules $\mathcal{P}'$ as follows.

—For each old rule $\alpha \to \beta \in \mathcal{P}$, we introduce a new rule $\alpha' \to \beta'$.

—For each old rule $\alpha \to \beta \in \mathcal{P}$, where $\beta = B_1 \ldots B_l$ and $l > 0$, we introduce rules which implement the derivation $\beta' \Rightarrow^* b^{|\beta|}$: namely,

$$\beta' \to bB_2 \ldots B_l,\ bB_2 \ldots B_l \to bbB_3 \ldots B_l,\ \ldots,\ b^{l-1}B_l \to b^l.$$

—For each terminal $a \in \Sigma$, we add the rule $A_a \to a$ to $\mathcal{P}'$.

—Finally, we add to $\mathcal{P}'$ the rule $S' \to Sc\alpha'_1 c\alpha'_2 c \dots c\alpha'_r cA_{a_1} cA_{a_2} c \dots cA_{a_k}$.

We prove that $S \Rightarrow^* w$ in G if and only if $S' \Rightarrow^*_{\mathcal{P}'} wcb^{|\beta_1|}cb^{|\beta_2|}c \dots cb^{|\beta_r|}ca_1ca_2c \dots ca_k$ in G'. If $S \Rightarrow^* w$ in G, then the desired derivation is the following:

$$\begin{aligned} S' &\Rightarrow Sc\alpha'_1 \dots c\alpha'_r cA_{a_1} \dots cA_{a_k} \Rightarrow^* w'c\alpha'_1 \dots c\alpha'_r cA_{a_1} \dots cA_{a_k} \\ &\Rightarrow^* w'c\beta'_1 \dots c\beta'_r cA_{a_1} \dots cA_{a_k} \Rightarrow^* wcb^{|\beta_1|} \dots cb^{|\beta_r|}ca_1 \dots ca_k. \end{aligned}$$

Now let $S \Rightarrow^*_{\mathcal{P}'} wcb^{|\beta_1|}c \dots cb^{|\beta_r|}ca_1c \dots ca_k$ in G'. Since $c \notin \Sigma$ and $w \in \Sigma^*$, we conclude that $S \Rightarrow^* w$ in G'. We have $b \notin \Sigma$, so rules of the third group could not have been applied in this derivation, and hence $S \Rightarrow^* w$ in G.

If G is a context-sensitive grammar, then so is G'. Thus, we have proved that the total derivability problem is m-complete for arbitrary grammars and that it is PSPACE-complete for context-sensitive and noncontracting grammars. □

5. PROOFS OF THEOREMS 1 AND 2

Now we are ready to prove the announced results on algorithmic complexity of derivability problems for 1-bounded sequents. Using Theorem 3, we pass from $\mathbf{!L_1}$ or $\mathbf{!^r L_1}$ to $\mathbf{!_{\leqslant 1} L_1}$ or $\mathbf{!^r_{\leqslant 1} L_1}$ respectively. In these calculi, there are no structural rules, and each rule introduces exactly one connective. Hence, the size of a derivation (the number of rule applications) is bounded by the length of a target sequent. A nondeterministic algorithm for verifying derivability guesses a potential derivation and then checks its correctness. Obviously, the correctness of rules can be checked in polynomial time. For checking whether a sequent in a leaf of a derivation tree is an axiom, in the case of $\mathbf{!_{\leqslant 1} L_1}$ we use the standard algorithm for checking derivability in a cf-grammar, and for $\mathbf{!^r_{\leqslant 1} L_1}$, we apply Algorithm R for checking $\mathcal{R}$-total derivability, as described in Section 3. Theorem 1 is proved.

In order to prove Theorem 2, we note that a sequent of the form $!\Phi; r_1, \dots, r_n \vdash s$ is derivable in $\mathbf{!^r_{\leqslant 1} L_1}$ if and only if it is an axiom, i.e., $\hat{r}_1 \dots \hat{r}_n$ is derivable in $G_{!\Phi,s}$, and its derivation is $\mathcal{R}_{!\Phi}$-total. Now let an arbitrary cf-grammar G and an arbitrary word $a_1 \dots a_n$ be given. We rename each terminal a to $\hat{a}$ and add the a itself as a nonterminal. Also add rules $a_i \to \hat{a}_i$ for each $i = 1, \dots, n$. Let a multiset $\mathcal{R}$ include rules of G, each with multiplicity 1. The word $a_1 \dots a_n$ is totally derivable in the grammar G if and only if $\hat{a}_1 \dots \hat{a}_n$ is $\mathcal{R}$-totally derivable in the new grammar. Indeed, rules of G are included in $\mathcal{R}$ with multiplicity 1, and new rules $a_i \to \hat{a}_i$ will certainly be used.

Now we construct $!\Phi$ such that $\mathcal{R}_{!\Phi} = \mathcal{R}$. Then $\mathcal{R}$-total derivability of the word $\hat{a}_1 \dots \hat{a}_n$ is equivalent to derivability of the sequent $!\Phi; a_1, \dots, a_n \vdash S$ in $\mathbf{!^r_{\leqslant 1} L_1}$. We translate this sequent to an equivalent sequent of the $\mathbf{!^r L_1}$ calculus, which contains only $\setminus$ and ! operations. The reduction we have constructed establishes NP-hardness of the derivability problem for 1-bounded sequents built by using $\setminus$ and ! in the $\mathbf{!^r L_1}$ calculus. Theorem 2 is proved.

REFERENCES

1. W. Krull, Axiomatische Begründung der allgemeinen Idealtheorie, Sitzber. d. phys.-med. Soc. Erlangen, **56**, 47-63 (1924).
2. J. Lambek, "Deductive systems and categories. II: Standard constructions and closed categories," *Lect. Notes Math.*, **86**, Springer, Berlin (1969), pp. 76-122.
3. J.-Y. Girard, "Linear logic," *Theor. Comput. Sci.*, **50**, No. 1, 1-102 (1987).
4. L. L. Maksimova, "On system of axioms of the calculus of rigorous implication," *Algebra Logika*, **3**, No. 3, 59-68 (1964).
5. M. Kanovich, S. Kuznetsov, and A. Scedrov, "Undecidability of the Lambek calculus with a relevant modality," *Lect. Notes Comput. Sci.*, **9804**, Springer, Berlin (2016), pp. 240-256.
6. M. Kanovich, S. Kuznetsov, V. Nigam, and A. Scedrov, "Subexponentials in non-commutative linear logic," *Math. Struct. Comput. Sci.*, **29**, No. 8, 1217-1249 (2019).
7. P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar, "Decision problems for propositional linear logic," *Ann. Pure Appl. Log.*, **56**, Nos. 1-3, 239-311 (1992).
8. M. Pentus, "Lambek calculus is NP-complete," *Theor. Comput. Sci.*, **357**, Nos. 1-3, 186-201 (2006).
9. Yu. V. Savateev. "Algorithmic complexity of fragments of the Lambek calculus," Cand. Sci. Dissertation, Moscow State Univ., Moscow (2009).
10. A. Aho and J. Ullman, *The Theory of Parsing, Translation, and Compiling*, Vol. 1, *Parsing*, Prentice-Hall, 1972.
11. M. R. Garey and D. S. Johnson, "Complexity results for multiprocessor scheduling under resource constraints," *SIAM J. Comput.*, **4**, No. 4, 397-411 (1975).