Complexity of the Lambek Calculus and Its Fragments

Mati Pentus

Department of Mathematical Logic and Theory of Algorithms
Faculty of Mechanics and Mathematics
Moscow State University
119991, Moscow, Russia

Abstract

We consider the original Lambek calculus and its natural modification called *the Lambek calculus allowing empty premises*. Both calculi have three binary connectives: an associative product operator and its two residuals, the left and right division. This paper contains a short survey of complexity results concerning fragments of these calculi obtained by restricting the set of connectives and/or the number of variables.

Keywords: Lambek calculus, complexity

Introduction

The Lambek syntactic calculus L (introduced in [8]) is one of the logical calculi used in the paradigm of categorial grammar for deriving reduction laws of syntactic types (also called "categories") in natural and formal languages. In categorial grammars based on the Lambek calculus (or its variants), an expression is assigned to category B / A (resp. $A \setminus B$) if and only if the expression produces an expression of category B whenever it is followed (resp. preceded) by an expression of category A. An expression is assigned to category $A \cdot B$ if and only if the expression can be obtained by concatenation of an expression of category A and an expression of category B. The reduction laws derivable in this calculus are of the form $A \to B$ (meaning "every expression of category A is also assigned to category B"). An overview of logical frameworks used in categorial grammars can be found, e.g., in [2] and [11].

There is a natural modification of the original Lambek calculus, which we call *the Lambek calculus allowing empty premises* and denote by L^* (see [21, p. 44]). Intuitively, the modified calculus allows the empty expression to be assigned to some categories.

The calculus L^* is in fact a fragment of noncommutative linear logic (introduced by V. M. Abrusci in [1]). Essentially the same logic was called BL2 by J. Lambek [9] (it was also studied by several other authors). Also the cyclic linear logic proposed by J.-Y. Girard and expounded by D. N. Yetter (see [5,22]) is conservative over L^*. In the propositional multiplicative fragments of all these logics, the cut rule can be eliminated and all cut-free proofs are of polynomial size. Thus, the derivability problem for these fragments is in NP.

In 2003, it was proved that for L (and L^*) the derivability problem is NP-hard and thus NP-complete. This was done by establishing a polynomial-time reduction from the Boolean satisfiability problem to the derivability problem for L (this reduction works also for L^*, and thus also for the multiplicative fragment of noncommutative linear logic and for the multiplicative fragment of cyclic linear logic). The derivability problem for all fragments obtained from the original Lambek calculus L (or L^*) by restricting the set of connectives (except for the trivial fragment with only the multiplication) remained open. All these complexity problems were solved in 2006–2009 by Yury Savateev who established polynomial-time reductions from the Boolean satisfiability problem to the derivability problems for those fragments that contain two of the three connectives of the Lambek calculus. It turned out that for the unidirectional Lambek calculus (the fragment that has only one division operator and no multiplication) the derivability problem is decidable in deterministic polynomial time. To establish this theorem, Y. Savateev invented an efficient method for testing derivability in the unidirectional Lambek calculus. At first, one decomposes the given types into atomic building blocks, labels them with natural numbers (which indicate the "Horn depth" in the original type), and puts them in a certain order (which reflects the group-theoretic interpretation of the division operator). Next, one evaluates an auxiliary predicate of 'acceptability' for all substrings of the string of labelled atomic building blocks. Doing this in the manner of dynamic programming we obtain a straightforward cubic algorithm for deciding derivability in the unidirectional Lambek calculus.

For each of the above fragments, the derivability problem remains in the same complexity class if we allow for only one variable.

The proofs of all these results are sketched in this survey. We also propose a modification of Y. Savateev's construction for the product-free fragments. The modified construction uses less variables and yields slightly shorter sequents.

This paper is organized as follows. The first section contains definitions of the calculi L and L^*. In Section 2, we define the calulus CMLL and show that it is conservative over L^*. In Section 3, we formulate a criterion for derivability in CMLL. In Section 4, we sketch the proof of NP-completeness for L and L^*. Complexity results for fragments of L and L^* are presented in the last two sections.

1 Lambek Calculus

First we define *the Lambek calculus allowing empty premises* (denoted by L^*).

Assume that an enumerable set of *variables* Var is given. The *types* of L^* are built of variables (also called *primitive types* in the context of the Lambek calculus) and three

binary connectives $\cdot$, $/$, and $\backslash$. The set of all types is denoted by Tp. The letters p, $q, \ldots$ range over the set Var, capital letters $A, B, \ldots$ range over types, and capital Greek letters range over finite (possibly empty) sequences of types. For notational convenience, we assume that $\cdot$ has higher priority than $\backslash$ and $/$.

The *sequents* of L^* are of the form $\Gamma \to A$ (note that Γ can be the empty sequence). The calculus L^* has the following axioms and rules of inference:

$$A \to A, \qquad \frac{\Phi \to B \quad \Gamma B \Delta \to A}{\Gamma \Phi \Delta \to A}\ (\mathrm{cut}),$$

$$\frac{\Pi A \to B}{\Pi \to B \,/\, A}\ (\to/), \qquad \frac{\Phi \to A \quad \Gamma B \Delta \to C}{\Gamma (B \,/\, A) \Phi \Delta \to C}\ (/\to),$$

$$\frac{A \Pi \to B}{\Pi \to A \setminus B}\ (\to\backslash), \qquad \frac{\Phi \to A \quad \Gamma B \Delta \to C}{\Gamma \Phi (A \setminus B) \Delta \to C}\ (\backslash\to),$$

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

As usual, we write $\mathrm{L}^* \vdash \Gamma \to A$ to indicate that the sequent $\Gamma \to A$ is derivable in L^*.

Example 1.1 The sequent $(p \setminus p) \setminus p \to p$ can be derived in L^* as follows:

$$\dfrac{\dfrac{\dfrac{p \to p}{\to p \setminus p}\ (\to\backslash) \qquad p \to p}{(p \setminus p) \setminus p \to p}\ (\backslash\to).}{}$$

The calculus L has the same axioms and rules with the only exception that in the rules $(\to\backslash)$ and $(\to/)$ we require Π to be nonempty. Thus, if $\mathrm{L} \vdash \Gamma \to A$, then Γ is nonempty. In fact, L is the original syntactic calculus introduced in [8]. Evidently, if $\mathrm{L} \vdash \Gamma \to A$, then $\mathrm{L}^* \vdash \Gamma \to A$.

It is known that the cut-elimination theorem holds for both L and L^*.

Example 1.2 The sequent $(p \setminus p) \setminus p \to p$ cannot be derived in L.

Example 1.3 The sequent $r \cdot (q \setminus s) \to (q \,/\, r) \setminus s$ can be derived in L as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{r \to r \quad q \to q}{(q \,/\, r)\, r \to q}\ (/\to) \qquad s \to s}{(q \,/\, r)\, r\, (q \setminus s) \to s}\ (\backslash\to)}{r\, (q \setminus s) \to (q \,/\, r) \setminus s}\ (\to\backslash)}{r \cdot (q \setminus s) \to (q \,/\, r) \setminus s}\ (\cdot\to).$$

The intended linguistic use of the Lambek calculus is demonstrated by the following example.

Example 1.4 Let s and np be two primitive types (s is intended to be the type of sentences and np is intended to be the type of noun phrases). In an example from [8], the English words *Jane* and *likes* are assigned the types np, and $(np \backslash s) / np$, respectively. The type of *likes* shows that *likes* yields a sentence if two noun phrases are added to

it, one on the right-hand side, another one on the left-hand side. Further, in this toy grammar the pronouns *he* and *she* are assigned the type $s \,/\, (np \setminus s)$, since any of them yields a sentence when combined with a sentence missing a noun phrase on the left-hand side. To check the grammaticality of a string of words, say *he likes Jane*, we attempt to derive a sequent whose left-hand side consists of types corresponding to these words. Since the sequent

$$(s \,/\, (np \setminus s))\,((np \setminus s) \,/\, np)\,np \to s$$

is derivable in L, we see that *he likes Jane* is accepted as a grammatically correct sentence by this grammar.

Similarly, *him* and *her* are assigned the type $(s \,/\, np) \setminus s$. In view of

$$\mathrm{L} \vdash np\,((np \setminus s) \,/\, np)\,((s \,/\, np) \setminus s) \to s,$$

we see that *Jane likes him* is a grammatically correct sentence. Moreover, *she likes him* is also accepted, since

$$\mathrm{L} \vdash (s \,/\, (np \setminus s))\,((np \setminus s) \,/\, np)\,((s \,/\, np) \setminus s) \to s.$$

The rules $(\to\setminus)$, $(\to/)$, and $(\cdot\to)$ are reversible in both L and L^* (the converse rules are easy to derive with the help of the cut rule).

If $\mathrm{L} \vdash A \to B$ and $\mathrm{L} \vdash B \to A$, then we write $A \underset{\mathrm{L}}{\leftrightarrow} B$ and say that A and B are equivalent. Replacing a type by an equivalent type in a sequent does not affect the derivability of the sequent. It is easy to verify that $(A \cdot B) \cdot C \underset{\mathrm{L}}{\leftrightarrow} A \cdot (B \cdot C)$ and $(A \setminus B) \,/\, C \underset{\mathrm{L}}{\leftrightarrow} A \setminus (B \,/\, C)$, which allows us to omit parentheses in certain types.

2 Cyclic Linear Logic

The cyclic linear logic was proposed by J.-Y. Girard and expounded by D. N. Yetter [5,22]. It is conservative over L^*. We consider its multiplicative fragment without the constants $\bot$ and $\mathbf{1}$. There are several equivalent sequent calculi for this fragment; here we consider only one of them and denote it by CMLL. The calculus CMLL may also be considered as a fragment of Lambek's bilinear logic BL3 from [9]; however, we use $\parr$ instead of $\oplus$.

In the definition of formulas of CMLL we shall employ the same enumerable set Var that was used in the definition of Lambek calculus types. It is well known that in CMLL every formula has an equivalent normal form, where the negation operator is only applied to variables. For simplicity, we shall only consider formulas in normal form. The negation of a variable p will be denoted by $\overline{p}$.

The set of formulas Fm of the calculus CMLL is defined as the smallest set satisfying the following conditions:

- if $p \in \mathrm{Var}$, then $p \in \mathrm{Fm}$ and $\overline{p} \in \mathrm{Fm}$;
- if $A \in \mathrm{Fm}$ and $B \in \mathrm{Fm}$, then $(A \otimes B) \in \mathrm{Fm}$ and $(A \parr B) \in \mathrm{Fm}$.

The binary connective $\otimes$ is called 'tensor', and $\parr$ is called 'par'. Variables and their negations (i.e., the formulas shown in the first item of the definition of formulas) are called *atoms*.

In the linear logic context, capital letters $A, B, \ldots$ range over the set Fm and capital Greek letters range over the set of finite (possibly empty) sequences of formulas from Fm.

The sequents of the calculus CMLL are of the form $\to \Gamma$.

We need an operation $(\,\cdot\,)^{\perp}\colon \mathrm{Fm} \to \mathrm{Fm}$ defined on the set Fm. It maps each formula to its *negation* as follows:

$$\begin{aligned}(p)^{\perp} &= \overline{p},\\ (\overline{p})^{\perp} &= p,\\ (A \otimes B)^{\perp} &= (B)^{\perp} \parr (A)^{\perp},\\ (A \parr B)^{\perp} &= (B)^{\perp} \otimes (A)^{\perp}\end{aligned}$$

(as usual, p ranges over Var). Evidently, $A^{\perp\perp} = A$.

Example 2.1 Let $s \in \mathrm{Var}$ and $np \in \mathrm{Var}$. According to the definition,

$$(s \parr (\overline{s} \otimes np))^{\perp} = (\overline{np} \parr s) \otimes \overline{s} \quad \text{and} \quad ((\overline{np} \parr s) \parr \overline{np})^{\perp} = np \otimes (\overline{s} \otimes np).$$

The axioms of the calculus CMLL are $\to \overline{p}\,p$ and $\to p\,\overline{p}$, where $p \in \mathrm{Var}$.

The calculus CMLL has the following rules of inference:

$$\frac{\to \Gamma A B \Delta}{\to \Gamma (A \parr B) \Delta}\ (\parr),$$

$$\frac{\to \Gamma A \Gamma' \quad \to B \Delta}{\to \Gamma (A \otimes B) \Delta \Gamma'}\ (\otimes_1), \qquad \frac{\to \Gamma A \quad \to \Delta B \Delta'}{\to \Delta \Gamma (A \otimes B) \Delta'}\ (\otimes_2),$$

$$\frac{\to \Gamma A \Gamma' \quad \to (A)^{\perp} \Delta}{\to \Gamma \Delta \Gamma'}\ (\mathrm{cut}_1), \qquad \frac{\to \Gamma A \quad \to \Delta (A)^{\perp} \Delta'}{\to \Delta \Gamma \Delta'}\ (\mathrm{cut}_2).$$

We shall write $\mathrm{CMLL} \vdash \to \Gamma$ if the sequent $\to \Gamma$ is derivable in CMLL.

The cut-elimination theorem holds for CMLL, i.e., the set of derivable sequents does not change if we drop the rules (cut_1) and (cut_2).

It can be proved by induction on derivation length that if $\to \Phi\Psi$ is derivable, then $\to \Psi\Phi$ is derivable too. Thus, only cyclic permutations of formulas in a sequent are allowed. This justifies the word 'cyclic' in the name of the logic.

To embed L^* into CMLL, we shall map each type $A \in \mathrm{Tp}$ to a formula $\widehat{A} \in \mathrm{Fm}$:

$$\begin{aligned}\widehat{p} &= p,\\ \widehat{A \,/\, B} &= \widehat{A} \parr (\widehat{B})^{\perp},\\ \widehat{A \setminus B} &= (\widehat{A})^{\perp} \parr \widehat{B},\\ \widehat{A \cdot B} &= \widehat{A} \otimes \widehat{B}.\end{aligned}$$

In [7], this mapping is denoted by $(\,\cdot\,)^\flat$.

Lemma 2.2 *Let $C_1, \ldots, C_n, D \in \mathrm{Tp}$. The sequent $C_1 \ldots C_n \to D$ is derivable in* L^* *if and only if the sequent $\to \widehat{C_n}^{\perp} \ldots \widehat{C_1}^{\perp} \widehat{D}$ is derivable in* CMLL.

Proof. This lemma was proved in [12, Sec. 7]. □

Example 2.3 Let $s \in \mathrm{Var}$ and $np \in \mathrm{Var}$. According to the definition, $\widehat{s/(np\backslash s)} = s \parr (\overline{s} \otimes np)$, $\widehat{(np\backslash s)/np} = (\overline{np} \parr s) \parr \overline{np}$, and $\widehat{(s/np)\backslash s} = (np \otimes \overline{s}) \parr s$. Thus, the sequent $(s \,/\, (np \setminus s))\,((np \setminus s) \,/\, np)\,((s \,/\, np) \setminus s) \to s$ from Example 1.4 corresponds to the CMLL-sequent

$$\to (\overline{s} \otimes (s \parr \overline{np}))\,(np \otimes (\overline{s} \otimes np))\,((\overline{np} \parr s) \otimes \overline{s})\,s,$$

which is indeed derivable in CMLL.

In view of Lemma 2.2, any algorithm for deciding derivability in CMLL also provides an algorithm for deciding derivability in L^*.

3 Proof Nets

To characterize derivability in CMLL, we shall use a proof-net-based criterion introduced in [13], where it was exposed for the multiplicative fragment of Abrusci's noncommutative linear logic (with the multiplicative constants). Essentially the same criterion is given in [3], where it is formulated for the constant-free multiplicative fragment of the cyclic linear logic. This criterion is quite similar to the one presented in Section 4 of the survey [7] and credited to Chapter 5 of [4]. In all these criteria, a planar graph consisting of a parse tree and axiom links divides the plane into regions, a binary relation is defined on the set of all regions (this is where the definitions from [13] and [7] differ), and the acyclicity condition is imposed on the graph corresponding to this binary relation.

In this section, we give an informal exposition of the criterion from [13] for CMLL.

Given a sequent $\to A_1 A_2 \ldots A_{k-1} A_k$, we replace it by the sequent

$$\to A_1 \parr (A_2 \parr \ldots \parr (A_{k-1} \parr A_k) \ldots)$$

(it is well known that they are either both derivable or both nonderivable in CMLL). Thus, we may assume that we have a sequent $\to A$, containing n occurrences of atoms. Evidently, it contains $n - 1$ occurrences of binary connectives.

On the set of all occurrences of connectives and atoms in A, we define the binary relation $\prec$ so that $\alpha \prec \beta$ if and only if α is in the scope of β.

Now we draw the parse tree of the formula A so that (1) moving from left to right we visit all nodes in the infix order, (2) if $\alpha \prec \beta$, then α is placed higher than β (thus, the root of the parse tree is at the bottom), and (3) the edges of the parse tree are drawn as straight line segments, and they do not intersect. We add one dummy occurrence of $\parr$ at the bottom left corner and connect it with the root by a dummy edge (again a straight

line segment). This dummy occurrence becomes the new root of the tree. The *extended parse tree* (i.e., the parse tree together with the dummy node and the dummy edge) is the first component of a proof net. It is uniquely determined by the given sequent.

Example 3.1 Let p, q, and s be different elements of Var. Let us consider the sequent

$$\rightarrow \big(\overline{p} \otimes \big(((q \parr (\overline{s} \otimes s)) \otimes \overline{s}) \parr s\big)\big) \parr \big(\big(((\overline{s} \otimes s) \parr (\overline{s} \parr s)) \otimes \overline{q}\big) \parr p\big).$$

Its extended parse tree is shown in Figure 1.

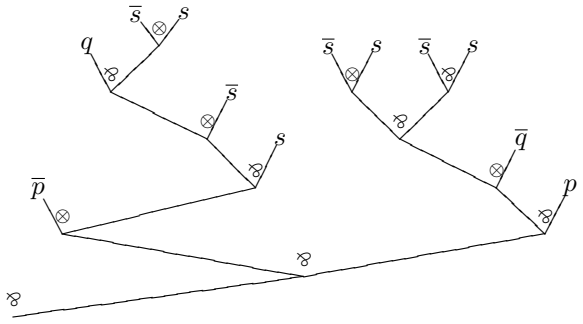


Fig. 1. An extended parse tree

The second component of a proof net is a set of axiom links. In general, axiom links are not uniquely determined by the given sequent. Each axiom link is an edge between two occurrences of atoms that are negations of each other. Each atom occurrence must be incident to exactly one axiom link. The axiom links must be drawn above the parse tree without intersecting each other or edges from the extended parse tree. Thus, the first two components together form a planar graph, which divides the plane into regions (in [7,3] they are called *faces*). It is easy to see that we obtain $\frac{n}{2} + 1$ regions.

Example 3.2 One possible set of axiom links for the sequent from Example 3.1 is shown in Figure 2 using dotted lines.

For each occurrence of a binary connective, we specify to which adjacent region it "belongs". The dummy occurrence belongs to its only adjacent region, the outer region of the graph (in [7] this region is called the *infinite face*). Each nondummy occurrence of $\parr$ or $\otimes$ belongs to the region adjacent to both edges leading from this occurrence to its children (for an occurrence of $\parr$, in [7] this region is called the *Inner face* of the occurrence). Thus, every occurrence of a binary connective belongs to the region immediately above it.

Now we come to the third component of a proof net. We require that in each region there must be exactly one occurrence of $\parr$ (i.e., there must be exactly $\frac{n}{2} + 1$ occurrences of $\parr$ and they must all belong to different regions). The third component of a proof tree consists of arcs leading from each tensor occurrence to the par occurrence in the same region. These arcs (we call them *region arcs*) together with the arcs specified by the

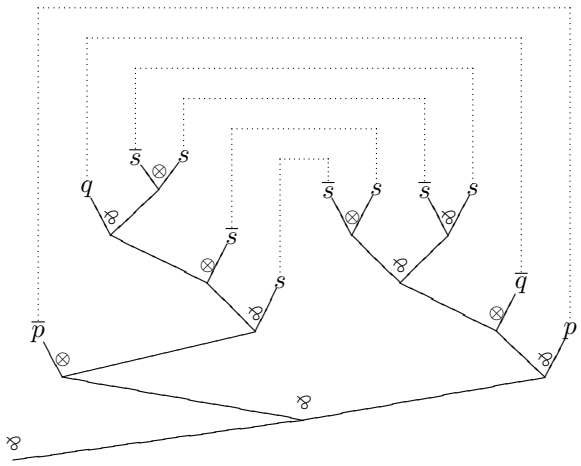


Fig. 2. Axiom links

partial order $\prec$ (directed towards the root) form a directed graph. This graph should not contain cycles (of course, we consider only directed cycles). In other words, if we denote the set of all region arcs by $\mathcal{A}$, then the binary relation $\prec \cup \mathcal{A}$ must be acyclic (we call a binary relation *acyclic* if its transitive closure is irreflexive). Note that if the first two components of a proof net are given, then the third component either does not exist or is uniquely determined.

If all the above conditions are satisfied, then we have a proof net for $\rightarrow A$. We shall call such proof nets *region proof nets*.

Example 3.3 Let us consider the extended proof tree and axiom links from Example 3.2. Indeed, in each of the seven regions there is exactly one occurrence of $\parr$. The region arcs are shown in Figure 3 (we indicate the left-to-right order of occurrences of connectives by subscripts). However, there is a cycle in the directed graph consisting of region arcs and arcs specified by the partial order $\prec$ (the cycle consists of the vertices denoted by $\parr_2$, $\otimes_4$, $\parr_8$, and $\otimes_{10}$). Thus, the structure shown in Figure 3 is not a region proof net.

Theorem 3.4 *A sequent $\rightarrow A$ is derivable in* CMLL *if and only if there exists a region proof net for $\rightarrow A$.*

Proof. The proof is similar to that of Theorem 7.12 from [13]. □

4 The Complexity of L^* and L

We give a sketch of the proof of the NP-completeness for L^* and L that can be found in [14,15].

In this and the next section, we construct mappings that take Boolean formulas in conjunctive normal form to sequents of the Lambek calculus. In both sections, we assume that we are given a Boolean formula in the conjunctive normal form $c_1 \wedge \ldots \wedge c_m$,

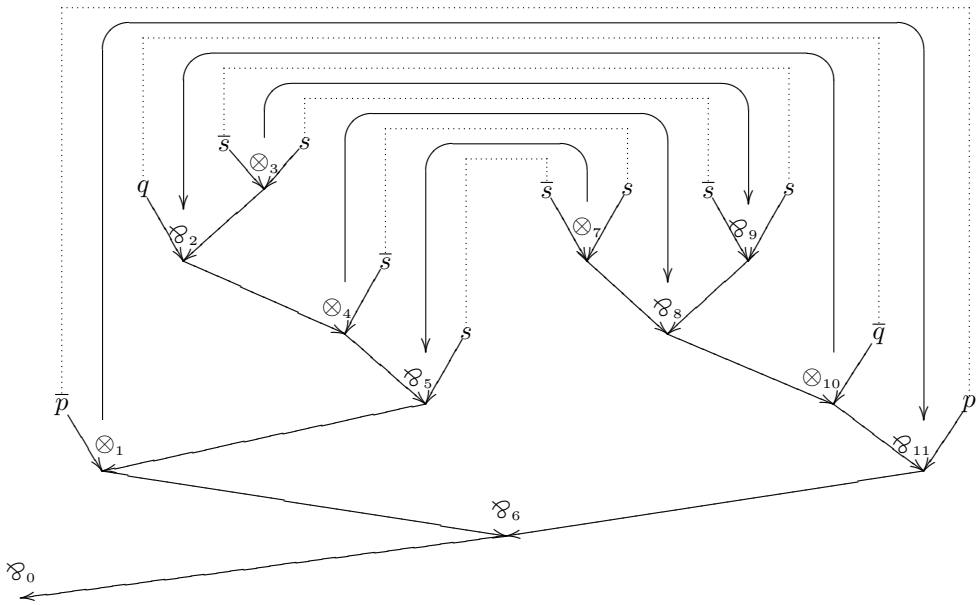


Fig. 3. Region arcs

with clauses $c_1, \ldots, c_m$ and variables $x_1, \ldots, x_n$. The reduction we are going to present in this section maps the formula to a sequent that is derivable in L^* (and in L) if and only if the formula $c_1 \wedge \ldots \wedge c_m$ is satisfiable.

For any Boolean variable x_i let $\neg_0 x_i$ stand for the literal $\neg x_i$ and $\neg_1 x_i$ stand for the literal x_i. Note that $\langle t_1, \ldots, t_n \rangle \in \{0,1\}^n$ is a satisfying assignment for the Boolean formula $c_1 \wedge \ldots \wedge c_m$ if and only if for every positive integer $j \leq m$ there exists a positive integer $i \leq n$ such that the literal $\neg_{t_i} x_i$ occurs in the clause c_j (as usual, 1 stands for "true" and 0 stands for "false").

Let p_i^j (where $0 \leq i \leq n$ and $0 \leq j \leq m$) be distinct primitive types from Var.

We define three families of types:

$$
\begin{aligned}
G_i^0 &\leftrightharpoons p_0^0 \setminus p_i^0 \quad \text{if } 1 \leq i \leq n,\\
G_i^j &\leftrightharpoons (p_0^j \setminus G_i^{j-1}) \cdot p_i^j \quad \text{if } 1 \leq i \leq n \text{ and } 1 \leq j \leq m,\\
H_i^0 &\leftrightharpoons p_{i-1}^0 \setminus p_i^0 \quad \text{if } 1 \leq i \leq n,\\
H_i^j &\leftrightharpoons p_{i-1}^j \setminus (H_i^{j-1} \cdot p_i^j) \quad \text{if } 1 \leq i \leq n \text{ and } 1 \leq j \leq m,\\
E_i^0(t) &\leftrightharpoons p_{i-1}^0 \setminus p_i^0 \quad \text{if } 1 \leq i \leq n \text{ and } t \in \{0,1\},\\
E_i^j(t) &\leftrightharpoons \begin{cases} (p_{i-1}^j \setminus E_i^{j-1}(t)) \cdot p_i^j & \text{if the literal } \neg_t x_i \text{ occurs in the clause } c_j,\\ p_{i-1}^j \setminus (E_i^{j-1}(t) \cdot p_i^j) & \text{otherwise} \end{cases}\\
&\qquad \text{if } 1 \leq i \leq n,\ 1 \leq j \leq m, \text{ and } t \in \{0,1\}.
\end{aligned}
$$

For convenience we introduce the following abbreviations:

$$\begin{aligned} G &\leftrightharpoons G_n^m, \\ H_i &\leftrightharpoons H_i^m \quad \text{if } 1 \le i \le n, \\ E_i(t) &\leftrightharpoons E_i^m(t) \quad \text{if } 1 \le i \le n \text{ and } t \in \{0,1\}, \\ F_i &\leftrightharpoons (E_i(1) \,/\, H_i) \cdot H_i \cdot (H_i \setminus E_i(0)) \quad \text{if } 1 \le i \le n. \end{aligned}$$

Example 4.1 Let $n = 1$, $m = 2$, $c_1 = x_1$, and $c_2 = \neg x_1$. Then

$$\begin{aligned} G &= (p_0^2 \setminus ((p_0^1 \setminus (p_0^0 \setminus p_1^0)) \cdot p_1^1)) \cdot p_1^2, \\ H_1 &= p_0^2 \setminus ((p_0^1 \setminus ((p_0^0 \setminus p_1^0) \cdot p_1^1)) \cdot p_1^2), \\ E_1(0) &= (p_0^2 \setminus (p_0^1 \setminus ((p_0^0 \setminus p_1^0) \cdot p_1^1))) \cdot p_1^2, \\ E_1(1) &= p_0^2 \setminus (((p_0^1 \setminus (p_0^0 \setminus p_1^0)) \cdot p_1^1) \cdot p_1^2), \\ F_1 &= (E_1(1) \,/\, H_1) \cdot H_1 \cdot (H_1 \setminus E_1(0)). \end{aligned}$$

The proofs of the following lemmas can be found in [14] and [15].

Lemma 4.2 *Let $1 \le i \le n$ and $t \in \{0,1\}$. Then $\mathrm{L} \vdash F_i \to E_i(t)$.*

Lemma 4.3 *Suppose $\langle t_1, \ldots, t_n \rangle$ is a satisfying assignment for the Boolean formula $c_1 \wedge \ldots \wedge c_m$. Then $\mathrm{L} \vdash E_1(t_1) \ldots E_n(t_n) \to G$.*

Example 4.4 Let $n = 2$, $m = 1$, $c_1 = x_1 \vee x_2$, $t_1 = 0$, and $t_2 = 1$. In view of Lemma 4.3, $\mathrm{L} \vdash E_1(0)E_2(1) \to G$, where

$$\begin{aligned} G &= (p_0^1 \setminus (p_0^0 \setminus p_2^0)) \cdot p_2^1, \\ E_1(0) &= p_0^1 \setminus ((p_0^0 \setminus p_1^0) \cdot p_1^1), \\ E_2(1) &= (p_1^1 \setminus (p_1^0 \setminus p_2^0)) \cdot p_2^1. \end{aligned}$$

Lemma 4.5 *If the formula $c_1 \wedge \ldots \wedge c_m$ is satisfiable, then $\mathrm{L} \vdash F_1 \ldots F_n \to G$.*

Proof. Suppose $\langle t_1, \ldots, t_n \rangle$ is a satisfying assignment for the formula $c_1 \wedge \ldots \wedge c_m$. According to Lemma 4.3 $\mathrm{L} \vdash E_1(t_1) \ldots E_n(t_n) \to G$. It remains to apply Lemma 4.2 and the cut rule n times. □

Example 4.6 Let $n = 1$, $m = 1$, and $c_1 = x_1$. In view of Lemma 4.5, we have $\mathrm{L} \vdash (E_1(1) \,/\, H_1) \cdot H_1 \cdot (H_1 \setminus E_1(0)) \to G$, where

$$\begin{aligned} G &= (p_0^1 \setminus (p_0^0 \setminus p_1^0)) \cdot p_1^1, \\ H_1 &= p_0^1 \setminus ((p_0^0 \setminus p_1^0) \cdot p_1^1), \\ E_1(0) &= p_0^1 \setminus ((p_0^0 \setminus p_1^0) \cdot p_1^1), \\ E_1(1) &= (p_0^1 \setminus (p_0^0 \setminus p_1^0)) \cdot p_1^1. \end{aligned}$$

Lemma 4.7 *If $\mathrm{L}^* \vdash F_1 \ldots F_n \to G$, then $\mathrm{L}^* \vdash E_1(t_1) \ldots E_n(t_n) \to G$ for some $\langle t_1, \ldots, t_n \rangle \in \{0,1\}^n$.*

Lemma 4.8 *Let* $\langle t_1, \ldots, t_n \rangle \in \{0,1\}^n$. *If* $\mathrm{L}^* \vdash E_1(t_1) \ldots E_n(t_n) \to G$, *then the assignment* $\langle t_1, \ldots, t_n \rangle$ *is a satisfying assignment for the Boolean formula* $c_1 \wedge \ldots \wedge c_m$.

Example 4.9 Let $n = 2$, $m = 1$, and $c_1 = x_1 \vee x_2$. We consider the assignment $\langle 0, 0 \rangle$, which is not a satisfying assignment for the Boolean formula $x_1 \vee x_2$. In view of Lemma 4.8, $\mathrm{L}^* \nvdash E_1(0)E_2(0) \to G$, where

$$\begin{aligned} G &= (p_0^1 \setminus (p_0^0 \setminus p_2^0)) \cdot p_2^1, \\ E_1(0) &= p_0^1 \setminus ((p_0^0 \setminus p_1^0) \cdot p_1^1), \\ E_2(0) &= p_1^1 \setminus ((p_1^0 \setminus p_2^0) \cdot p_2^1). \end{aligned}$$

Lemma 4.10 *If* $\mathrm{L}^* \vdash F_1 \ldots F_n \to G$, *then the formula* $c_1 \wedge \ldots \wedge c_m$ *is satisfiable.*

Proof. Immediate from Lemma 4.7 and Lemma 4.8. □

Theorem 4.11 *The* L*-derivability problem is* NP*-complete.*

Proof. The number of variable occurrences in a cut-free derivation in L can not exceed the square of the number of variable occurrences in the final sequent. Thus, the L-derivability problem is in NP.

According to Lemma 4.5 and Lemma 4.10, the mapping that takes $c_1 \wedge \ldots \wedge c_m$ to $F_1 \ldots F_n \to G$ yields a reduction from the classical satisfiability problem SAT to the L-derivability problem. The problem SAT is known to be NP-hard. Thus the L-derivability problem is NP-hard as well. □

Theorem 4.12 *The* L^**-derivability problem is* NP*-complete.*

Proof. The theorem follows immediately from Lemma 4.5 and Lemma 4.10. The proof is similar to that of the previous theorem. □

All the lemmas in Sections 4 and 5 can be proved using the region proof nets defined in Section 3. To reformulate any result concerning L^* in terms of proof nets, we use Lemma 2.2 and Theorem 3.4. When we deal with L, to reformulate $\mathrm{L} \vdash C_1 \ldots C_n \to D$, we need a variant of region proof nets with the additional stipulation that the connective occurrence preceding a (not necessarily proper) subformula of $\widehat{D}$ or $\widehat{C_i}^{\perp}$ is not in the same region as the connective occurrence succeeding this subformula (in [6] such proof nets are called *strong*).

Example 4.13 To illustrate Lemma 4.8, we consider the sequent $E_1(0)\, E_2(0) \to G$ from Example 4.9. This sequent corresponds to the Boolean formula $x_1 \vee x_2$ and the assignment where both x_1 and x_2 are false. Our aim is to show that $\mathrm{L}^* \nvdash E_1(0)\, E_2(0) \to G$. In view of Lemma 2.2, this is equivalent to

$$\mathrm{CMLL} \nvdash \to \widehat{E_2(0)}^{\perp}\, \widehat{E_1(0)}^{\perp}\, \widehat{G}.$$

Note that

$$\begin{aligned}
\widehat{G} &= (\overline{p_0^1} \parr (\overline{p_0^0} \parr p_2^0)) \otimes p_2^1,\\
\widehat{E_1(0)} &= \overline{p_0^1} \parr ((\overline{p_0^0} \parr p_1^0) \otimes p_1^1),\\
\widehat{E_2(0)} &= \overline{p_1^1} \parr ((\overline{p_1^0} \parr p_2^0) \otimes p_2^1),\\
\widehat{E_1(0)}^{\perp} &= (\overline{p_1^1} \parr (\overline{p_1^0} \otimes p_0^0)) \otimes p_0^1,\\
\widehat{E_2(0)}^{\perp} &= (\overline{p_2^1} \parr (\overline{p_2^0} \otimes p_1^0)) \otimes p_1^1.
\end{aligned}$$

In view of Theorem 3.4, it suffices to show that there is no proof net for the sequent $\rightarrow \widehat{E_2(0)}^{\perp}\, \widehat{E_1(0)}^{\perp}\, \widehat{G}$.

The extended parse tree for $\widehat{E_2(0)}^{\perp} \parr (\widehat{E_1(0)}^{\perp} \parr \widehat{G})$ is shown in Figure 4.

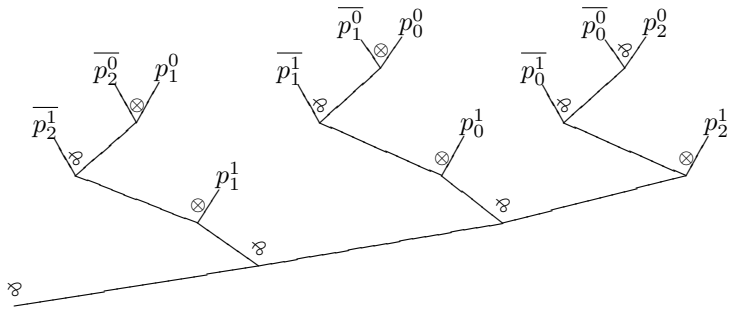


Fig. 4. The extended parse tree corresponding to $x_1 \vee x_2$ and $\langle 0, 0\rangle$

There is a unique way to establish axiom links between occurrences of atoms that are negations of each other. The axiom links are shown in Figure 5. We indicate the left-to-right order of occurrences of connectives by subscripts.

Finally, we add the region arcs. In order to demonstrate some regularities of extended parse trees and axiom links corresponding to sequents of the form $E_1(t_1) \ldots E_n(t_n) \rightarrow G$, we distort the graph layout in a continuous way, as shown in Figure 6. The occurrences denoted by $\parr_1$, $\otimes_3$, $\parr_5$, $\otimes_7$, $\parr_9$, and $\otimes_{11}$ form a cycle. Hence the sequent $\rightarrow \widehat{E_2(0)}^{\perp}\, \widehat{E_1(0)}^{\perp}\, \widehat{G}$ is not derivable in CMLL.

Example 4.14 To illustrate Lemma 4.3, we consider the sequent $E_1(0)\, E_2(1) \rightarrow G$ from Example 4.4. This sequent corresponds to the Boolean formula $x_1 \vee x_2$ and the assignment where x_1 is false and x_2 is true. Our aim is to show that $\mathrm{L} \vdash E_1(0)\, E_2(1) \rightarrow G$. It suffices to find a strong proof net for the sequent

$$\rightarrow \widehat{E_2(1)}^{\perp}\, \widehat{E_1(0)}^{\perp}\, \widehat{G}.$$

The types $\widehat{E_1(0)}^{\perp}$ and $\widehat{G}$ are the same as in Example 4.13, and

$$\widehat{E_2(1)}^{\perp} = \overline{p_2^1} \parr ((\overline{p_2^0} \otimes p_1^0) \otimes p_1^1).$$

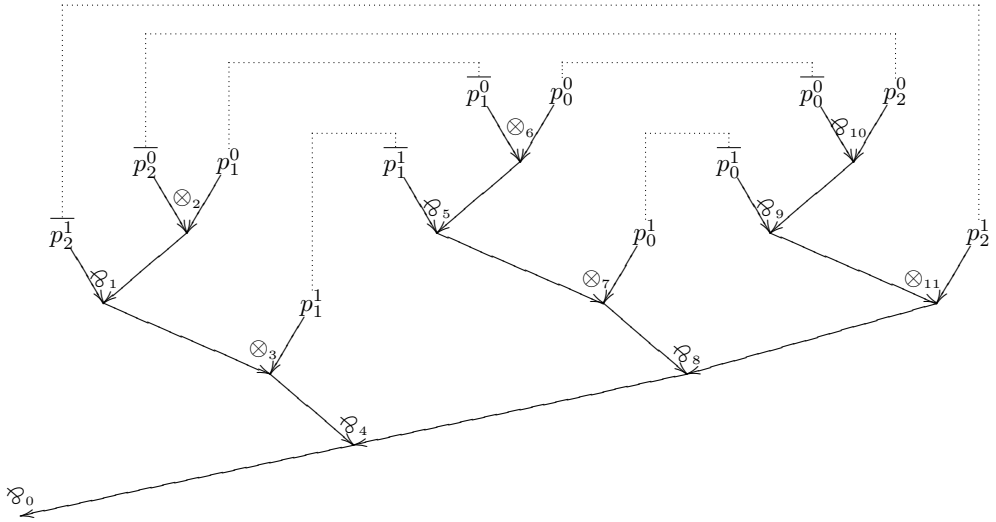


Fig. 5. The extended parse tree and axiom links corresponding to $x_1 \vee x_2$ and $\langle 0, 0 \rangle$

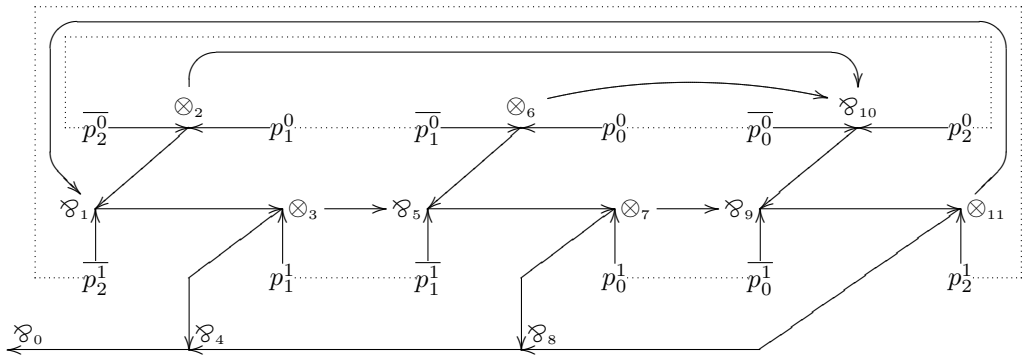


Fig. 6. The extended parse tree, axiom links, and region arcs corresponding to $x_1 \vee x_2$ and $\langle 0, 0 \rangle$

The proof net is shown in Figure 7. Evidently, it is strong.

Example 4.15 The Boolean formula $x_1 \wedge \neg x_1$ is not satisfiable. This means that $\mathrm{L} \nvdash E_1(0) \to G$ and $\mathrm{L} \nvdash E_1(1) \to G$ (the types G, $E_1(0)$, and $E_1(1)$ are shown in Example 4.1). For both assignments, we can draw the axiom links and region arcs, but there is a cycle consisting of region arcs and arcs specified by the partial order $\prec$. For the sequent $E_1(0) \to G$, this can be seen in Figure 8, where the occurrences denoted by $\parr_2$, $\otimes_4$, $\parr_8$, and $\otimes_{10}$ form a cycle.

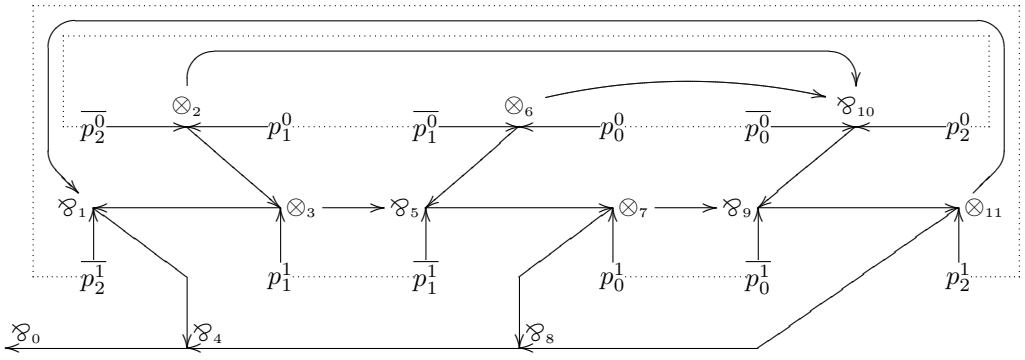


Fig. 7. The proof net corresponding to $x_1 \vee x_2$ and $\langle 0, 1\rangle$

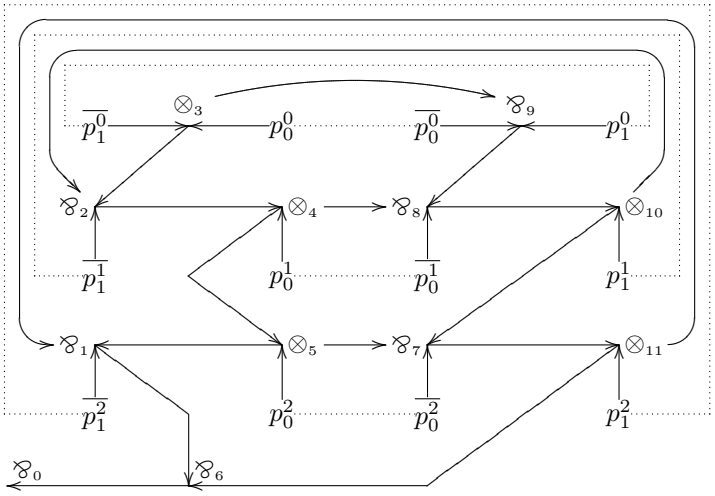


Fig. 8. The extended parse tree, axiom links, and region arcs corresponding to $x_1 \wedge \neg x_1$ and $\langle 0\rangle$

5 The Complexity of Fragments with Restricted Sets of Connectives

5.1 The Product-Free Fragments of L^* *and* L

A proof of the NP-completeness for $\mathrm{L}^*(\backslash, /)$ and $\mathrm{L}(\backslash, /)$ was discovered by Y. Savateev and published in [18]. Here we follow the exposition from [20], but slightly modify the notation in order to avoid name collision.

Let p_i^j, q_i^j, a_i^j, and b_i^j (where $0 \leq i \leq n$ and $0 \leq j \leq m$) be distinct primitive types from Var.

We define the following families of types:

$$
\begin{aligned}
\check{G}^0 &\leftrightharpoons (p_0^0 \setminus p_n^0),\\
\check{G}^j &\leftrightharpoons (q_n^j \,/\, ((q_0^j \setminus p_0^j) \setminus \check{G}^{j-1})) \setminus p_n^j,\\
\check{G} &\leftrightharpoons \check{G}^m\\
\check{A}_i^0 &\leftrightharpoons (a_i^0 \setminus p_i^0),\\
\check{A}_i^j &\leftrightharpoons (q_i^j \,/\, ((b_i^j \setminus a_i^j) \setminus \check{A}_i^{j-1})) \setminus p_i^j,\\
\check{A}_i &\leftrightharpoons \check{A}_i^m,\\
\check{E}_i^0(t) &\leftrightharpoons p_{i-1}^0,\\
\check{E}_i^j(t) &\leftrightharpoons \begin{cases} q_i^j \,/\, (((q_{i-1}^j \,/\, \check{E}_i^{j-1}(t)) \setminus p_{i-1}^j) \setminus p_i^{j-1}), & \text{if } \neg_t x_i \text{ occurs in } c_j,\\ (q_{i-1}^j \,/\, (q_i^j \,/\, (\check{E}_i^{j-1}(t) \setminus p_i^{j-1}))) \setminus p_{i-1}^j, & \text{otherwise,} \end{cases}\\
\check{F}_i(t) &\leftrightharpoons (\check{E}_i^m(t) \setminus p_i^m),\\
\check{B}_i^0 &\leftrightharpoons a_i^0,\\
\check{B}_i^j &\leftrightharpoons q_{i-1}^j \,/\, (((b_i^j \,/\, \check{B}_i^{j-1}) \setminus a_i^j) \setminus p_{i-1}^{j-1}),\\
\check{B}_i &\leftrightharpoons \check{B}_i^m \setminus p_{i-1}^m.
\end{aligned}
$$

Let Π_i denote the sequence of types $(\check{F}_i(0) \,/\, (\check{B}_i \setminus \check{A}_i))\, \check{F}_i(0)\, (\check{F}_i(0) \setminus \check{F}_i(1))$.

Lemma 5.1 *If the formula $c_1 \wedge \ldots \wedge c_m$ is satisfiable, then $\mathrm{L}(\setminus, /) \vdash \Pi_1 \ldots \Pi_n \to \check{G}$.*

Lemma 5.2 *If $\mathrm{L}^*(\setminus, /) \vdash \Pi_1 \ldots \Pi_n \to \check{G}$, then the formula $c_1 \wedge \ldots \wedge c_m$ is satisfiable.*

Theorem 5.3 *The derivability problems for $\mathrm{L}^*(\setminus, /)$ and $\mathrm{L}(\setminus, /)$ are* NP*-complete.*

The length of the sequent $\Pi_1 \ldots \Pi_n \to \check{G}$ (i.e., the total number of variable occurrences) equals $(6n + 1)(4m + 2)$.

We propose a modification of Savateev's construction. In this modification, a Boolean formula in conjunctive normal form with n clauses and m variables is mapped to a sequent of length $(5n + 1)(4m + 2)$.

We use the types $\check{G}$ and $\check{F}_i(t)$ defined above and introduce the following family of types:

$$
\begin{aligned}
\check{C}_i^0 &\leftrightharpoons p_i^0,\\
\check{C}_i^j &\leftrightharpoons (q_i^j \,/\, \check{C}_i^{j-1}) \setminus p_i^j.
\end{aligned}
$$

We denote $\check{C}_i \leftrightharpoons \check{C}_i^m$.

Let Γ_i denote the sequence of types $(\check{F}_i(0) \,/\, (\check{C}_{i-1} \setminus \check{C}_i))\, \check{F}_i(0)\, (\check{F}_i(0) \setminus \check{F}_i(1))$.

Lemma 5.4 *If the formula $c_1 \wedge \ldots \wedge c_m$ is satisfiable, then $\mathrm{L}(\setminus, /) \vdash \Gamma_1 \ldots \Gamma_n \to \check{G}$.*

Lemma 5.5 *If $\mathrm{L}^*(\setminus, /) \vdash \Gamma_1 \ldots \Gamma_n \to \check{G}$, then the formula $c_1 \wedge \ldots \wedge c_m$ is satisfiable.*

The above lemmas show that our modification of Savateev's construction provides another polynomial time reduction of the classical satisfiability problem SAT to the $\mathrm{L}^*(\setminus, /)$-derivability problem and the $\mathrm{L}(\setminus, /)$-derivability problem.

5.2 The Fragments with Multiplication and One Division

A proof of the NP-completeness for $\mathrm{L}^*(\cdot,\backslash)$, $\mathrm{L}^*(\cdot,/)$, $\mathrm{L}(\cdot,\backslash)$, and $\mathrm{L}(\cdot,/)$ can be found in Y. Savateev's thesis [19].

Using the types introduced in Section 4 and additional primitive types r_i (where $1 \le i \le n$), we define the following family of types:

$$\begin{aligned}
\dot{F}_1 &\leftrightharpoons E_1(0) \cdot ((E_1(0) \setminus E_1(1)) \cdot (H_1 \setminus r_1)),\\
\dot{F}_i &\leftrightharpoons ((E_{i-1}(0) \setminus r_{i-1}) \setminus E_i(0)) \cdot (E_i(0) \setminus E_i(1)) \cdot (H_i \setminus r_i) \quad \text{if } 1 < i \le n,\\
\dot{F}_{n+1} &\leftrightharpoons (E_n(0) \setminus r_n) \setminus H_{n+1}.
\end{aligned}$$

Lemma 5.6 *If the formula $c_1 \wedge \ldots \wedge c_m$ is satisfiable, then $\mathrm{L}(\cdot,\backslash) \vdash \dot{F}_1 \ldots \dot{F}_{n+1} \to G$.*

Lemma 5.7 *If $\mathrm{L}^*(\cdot,\backslash) \vdash \dot{F}_1 \ldots \dot{F}_{n+1} \to G$, then the formula $c_1 \wedge \ldots \wedge c_m$ is satisfiable.*

Theorem 5.8 *The derivability problems for $\mathrm{L}^*(\cdot,\backslash)$, $\mathrm{L}^*(\cdot,/)$, $\mathrm{L}(\cdot,\backslash)$, and $\mathrm{L}(\cdot,/)$ are* NP*-complete.*

5.3 The Fragments with One Connective

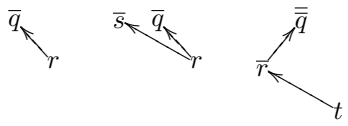
The fragments $\mathrm{L}^*(\cdot)$ and $\mathrm{L}(\cdot)$ are obviously decidable in polynomial time: a sequent is derivable if its antecedent and succedent yield the same sequence of primitive types after we remove all parentheses and all occurrences of the connective $\cdot$.

A deterministic polynomial-time algorithm for $\mathrm{L}(\backslash)$ and $\mathrm{L}(/)$ was discovered by Y. Savateev and published in [16] and [17]. It relies on a graph-based presentation of derivations. Here we shall call these graphs *strong unidirectional proof nets*. We give an informal exposition of the criterion from [16] in a slightly modified form.

With each type of $\mathrm{L}(\backslash)$ we associate a directed tree together with a linear order on its vertices; we draw the tree so that this linear order corresponds to the left-to-right direction. The vertices are the occurrences of primitive types in the given type. To obtain the tree corresponding to $A \setminus B$, we take the tree corresponding to B, add the left-right converse of the tree for A on the left, and draw an arc from the root of B to the root of A.

In a directed tree, each vertex has *depth*, the number of edges in the unique path from the root to this vertex. In the following diagrams, we shall indicate the depth of a vertex by the number of short horizontal lines over the primitive type.

Example 5.9 The following three trees correspond to $q \setminus r$, $s \setminus (q \setminus r)$, and $(q \setminus r) \setminus t$.



Now we can give the definition of a strong unidirectional proof net for a sequent $C_1 \ldots C_n \to D$. The first component of such a proof net is the tree that corresponds to $C_n \setminus (C_{n-1} \setminus \ldots \setminus (C_1 \setminus D) \ldots)$. The second component of the proof net is a set of axiom

links. Each axiom link is an edge between two occurrences of the same primitive type. Each vertex must be incident to exactly one axiom link. The axiom links must be drawn above the tree without intersecting each other or edges from the tree.

We require that for each axiom link the depth of its left end be one greater than the depth of its right end. Moreover, if the right end of an axiom link has non-zero even depth, then between the two ends of the axiom link there must be a vertex of smaller depth.

If all the above conditions are satisfied, then we have a strong unidirectional proof net for $C_1 \dots C_n \to D$.

Example 5.10 Consider the sequent $(r \setminus p)\,((s \setminus p) \setminus t) \to (s \setminus r) \setminus t$, which is derivable in L. A strong unidirectional proof net for this sequent is shown in Figure 9.

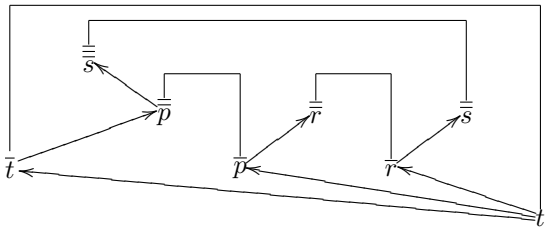


Fig. 9. A strong unidirectional proof net

Theorem 5.11 *A sequent $C_1 \dots C_n \to D$ is derivable in* L($\setminus$) *if and only if there exists a strong unidirectional proof net for $C_1 \dots C_n \to D$.*

The proof of the theorem can be found in [16].

Now a polynomial-time algorithm for the derivability problem in L($\setminus$) is easy to design. We construct the tree corresponding to a given sequent, and for each interval of vertices (in the sense of the left-to-right order), we find out whether there exists a set of axiom links that involves all the vertices of the interval and does not involve other vertices (each axiom link must satisfy the conditions from the definition of a strong unidirectional proof net). The maximal interval covers all the tree and gives an answer whether a strong unidirectional proof net exists.

A deterministic polynomial-time algorithm for L*($\setminus$) and L*($/$) was also discovered by Y. Savateev. To obtain a derivability criterion for L*($\setminus$) it suffices to modify the definition of a proof net. The only difference is in the last condition (about axiom links whose right end has non-zero even depth). For L*($\setminus$) the corresponding condition is the following. If the right end of an axiom link has depth $2k$, where $k > 0$, and the depth of the left predecessor of the left end is greater than $2k$, then there must be a vertex of depth less than $2k$ between the two ends of the axiom link. We shall call proof nets of this kind *unidirectional*. It is easy to see that each strong unidirectional proof net is a unidirectional proof net.

Example 5.12 Consider the sequent $(q \setminus q) \setminus s \to \setminus s$, which is derivable in L*, but not in L. A unidirectional proof net for this sequent is shown in Figure 10.

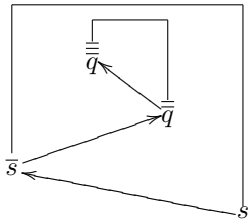


Fig. 10. A unidirectional proof net

Theorem 5.13 *A sequent $C_1 \dots C_n \to D$ is derivable in $\mathrm{L}^*(\backslash)$ if and only if there exists a unidirectional proof net for $C_1 \dots C_n \to D$.*

Theorem 5.13 provides a deterministic polynomial-time algorithm for the derivability problem for $\mathrm{L}^*(\backslash)$. We use the method of dynamic programming, similarly to the case of $\mathrm{L}(\backslash)$.

6 The Complexity of Fragments with Restricted Number of Variables

For each of the above fragments, the derivability problem remains in the same complexity class if we allow for only one variable (or restrict the number of variables by any positive integer).

We introduce the abbreviation

$$A^k \leftrightharpoons \underbrace{A \cdot \ldots \cdot A}_{k \text{ times}},$$

where A is a type and k is a positive integer.

If a fragment contains both $\backslash$ and $/$ and a sequent contains only the variables q_i, where $0 < i < N$, then we can replace each variable q_i by the type $p^i \setminus p \,/\, p^{N-i}$, where $p \in \mathrm{Var}$, and this does not affect derivability (this substitution is based on a construction from [10]). Since $(A \cdot B) \setminus C \underset{\mathrm{L}}{\leftrightarrow} B \setminus (A \setminus C)$ and $C \,/\, (A \cdot B) \underset{\mathrm{L}}{\leftrightarrow} (C \,/\, B) \,/\, A$, the above types can be replaced by equivalent types that do not contain $\cdot$.

Example 6.1 It is easy to see that $\mathrm{L} \nvdash q_1 \setminus q_2 \to q_1$. In the one-variable fragment, this corresponds to

$$\mathrm{L} \nvdash ((p \setminus p \,/\, p) \,/\, p) \setminus (p \setminus (p \setminus p \,/\, p)) \to (p \setminus p \,/\, p) \,/\, p.$$

Here $N = 3$.

To prove the NP-completeness of the one-variable fragments of $\mathrm{L}(\cdot, \backslash)$ and $\mathrm{L}^*(\cdot, \backslash)$, we use a substitution discovered by S. Kuznetsov (see [6]) involving only the left division (of course, there is a dual substitution, which fits for $\mathrm{L}(\cdot, /)$ and $\mathrm{L}^*(\cdot, /)$). Let in L or L^* a sequent contain only the variables q_i, where $0 < i < N$. Then we can replace each

variable q_i by the type

$$\big(p^{i+1} \cdot \big(((p \cdot p) \setminus p) \setminus p\big) \cdot p^{N-i}\big) \setminus p,$$

and this does not affect derivability. Obviously, these types can be replaced by equivalent types that do not contain $\cdot$ or $/$.

Example 6.2 If $N = 3$, then Kuznetsov's substitution maps q_1 to

$$p \setminus (p \setminus (((p \setminus (p \setminus p)) \setminus p) \setminus (p \setminus (p \setminus p))))$$

and q_2 to

$$p \setminus (((p \setminus (p \setminus p)) \setminus p) \setminus (p \setminus (p \setminus (p \setminus p)))).$$

Conclusion

The complexity results for the derivability problem for fragments of L and L^* are summarized in Table 1, where NP denotes NP-completeness. One-variable fragments are denoted by $\mathrm{L}(p_1)$ and $\mathrm{L}^*(p_1)$.

	L	L^*	$\mathrm{L}(p_1)$	$\mathrm{L}^*(p_1)$
$\cdot, \setminus, /$	NP	NP	NP	NP
$\cdot, \setminus$	NP	NP	NP	NP
$\cdot, /$				
$\cdot$	P	P	P	P
$\setminus, /$	NP	NP	NP	NP
$\setminus$	P	P	P	P
$/$				

Table 1
The complexity of fragments of L and L^*

References

[1] Abrusci, V. M., *Phase semantics and sequent calculus for pure noncommutative classical linear propositional logic*, Journal of Symbolic Logic **56** (1991), pp. 1403–1451.

[2] Buszkowski, W., *Mathematical linguistics and proof theory*, in: J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, Elsevier/MIT Press, Amsterdam/Cambridge, MA, 1997 pp. 683–736.

[3] de Groote, P., *A dynamic programming approach to categorial deduction*, in: H. Ganzinger, editor, *Proc. Automated deduction—CADE-16*, Springer, Berlin, 1999 pp. 1–15.

[4] Fleury, A., "La règle d'échange: logique linéaire multiplicative tréssée," Ph.D. thesis, Université Paris 7 (1996), thèse de Doctorat, spécialité Mathématiques.

[5] Girard, J.-Y., *Towards a geometry of interaction*, in: J. W. Gray and A. Scedrov, editors, *Categories in Computer Science and Logic*, American Mathematical Society, Providence, RI, 1989 pp. 69–108.

[6] Kuznetsov, S. L., *Lambek calculus with one division and one primitive type permitting empty antecedents*, Moscow University Mathematics Bulletin **64** (2009), pp. 76–79.

[7] Lamarche, F. and C. Retoré, *Proof nets for the Lambek calculus—an overview*, in: V. M. Abrusci and C. Casadio, editors, *Proofs and Linguistic Categories, Proc. 1996 Roma Workshop*, CLUEB, Bologna, 1996 pp. 241–262.

[8] Lambek, J., *The mathematics of sentence structure*, American Mathematical Monthly **65** (1958), pp. 154–170.

[9] Lambek, J., *From categorial grammar to bilinear logic*, in: K. Došen and P. Schroeder-Heister, editors, *Substructural Logics*, Oxford University Press, Oxford, 1993 pp. 207–237.

[10] Métayer, F., *Polynomial equivalence among systems* LLNC, LLNC_a *and* LLNC_0, Theoretical Computer Science **227** (1999), pp. 221–229.

[11] Moortgat, M., *Categorial type logics*, in: J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, Elsevier/MIT Press, Amsterdam/Cambridge, MA, 1997 pp. 93–177.

[12] Pentus, M., *Equivalent types in Lambek calculus and linear logic*, MIAN Prepublication Series for Logic and Computer Science LCS-92-02, Steklov Mathematical Institute, Moscow (1992).

[13] Pentus, M., *Free monoid completeness of the Lambek calculus allowing empty premises*, in: J. M. Larrazabal, D. Lascar and G. Mints, editors, *Proc. Logic Colloquium '96*, Springer, Berlin, 1998 pp. 171–209.

[14] Pentus, M., *Lambek calculus is* NP-*complete*, CUNY Ph.D. Program in Computer Science Technical Report TR-2003005, CUNY Graduate Center, New York (2003).

[15] Pentus, M., *Lambek calculus is* NP-*complete*, Theoretical Computer Science **357** (2006), pp. 186–201.

[16] Savateev, Y., *The derivability problem for Lambek calculus with one division*, Artificial Intelligence Preprint Series 56, Utrecht University (2006).

[17] Savateev, Y., *Lambek grammars with one division are decidable in polynomial time*, in: E. A. Hirsch, A. A. Razborov, A. L. Semenov and A. Slissenko, editors, *Computer Science—Theory and Applications, Third International Computer Science Symposium in Russia, CSR 2008, Moscow, Russia, June 7–12, 2008, Proceedings*, Springer, Berlin, 2008 pp. 273–282.

[18] Savateev, Y., *Product-free Lambek calculus is* NP-*complete*, CUNY Ph.D. Program in Computer Science Technical Report TR-2008012, CUNY Graduate Center, New York (2008).

[19] Savateev, Y., "Algorithmic Complexity of Fragments of the Lambek Calculus," Ph.D. thesis, Moscow State University (2009), (in Russian).

[20] Savateev, Y., *Product-free Lambek calculus is* NP-*complete*, in: S. N. Artemov and A. Nerode, editors, *Proc. Logical Foundations of Computer Science 2009*, Springer, Berlin, 2009 pp. 380–394.

[21] van Benthem, J., "Language in Action: Categories, Lambdas and Dynamic Logic," North-Holland, Amsterdam, 1991.

[22] Yetter, D. N., *Quantales and (noncommutative) linear logic*, Journal of Symbolic Logic **55** (1990), pp. 41–64.