ISSN 0032-9460, Problems of Information Transmission, 2015, Vol. 51, No. 4, pp. 349–360. © Pleiades Publishing, Inc., 2015.
Original Russian Text © M.N. Vyalyi, A.A. Rubtsov, 2015, published in Problemy Peredachi Informatsii, 2015, Vol. 51, No. 4, pp. 47–59.

LARGE SYSTEMS

On Regular Realizability Problems for Context-Free Languages

M. N. Vyalyi[a,1] **and A. A. Rubtsov**[b,2]

[a] *Dorodnitsyn Computing Center of the Russian Academy of Sciences, Moscow, Russia*
Moscow Institute of Physics and Technology (State University), Moscow, Russia
National Research University—Higher School of Economics, Moscow, Russia
e-mail: vyalyi@gmail.com

[b] *Moscow Institute of Physics and Technology (State University), Moscow, Russia*
National Research University—Higher School of Economics, Moscow, Russia
e-mail: alex@rubtsov.su

Received September 9, 2014; in final form, March 13, 2015

Abstract—We consider regular realizability problems, which consist in verifying whether the intersection of a regular language which is the problem input and a fixed language (filter) which is a parameter of the problem is nonempty. We study the algorithmic complexity of regular realizability problems for context-free filters. This characteristic is consistent with the rational dominance relation of CF languages. However, as we prove, it is more rough. We also give examples of both **P**-complete and **NL**-complete regular realizability problems for CF filters. Furthermore, we give an example of a subclass of CF languages for filters of which the regular realizability problems can have an intermediate complexity. These are languages with polynomially bounded rational indices.

DOI: 10.1134/S0032946015040043

1. INTRODUCTION

Context-free (CF) languages are one of important classes in the theory of formal languages. There are many various ways to characterize the complexity of a CF language. In the present paper we propose a new complexity classification for the CF languages, which is based on the algorithmic complexity of the regular realizability problem corresponding to the language.

A regular realizability problem consists in verifying whether the intersection of a regular language which is the problem input and a fixed filter language which is a parameter of the problem is nonempty. Depending on a way of describing the input regular language, two types of problems are distinguished. In the case where the regular language is specified by a description of a deterministic finite automaton, the problem is said to be a (deterministic) regular realizability problem; we denote it by $\mathrm{RR}(F)$. In the case where the regular language is specified by a description of a nondeterministic finite automaton (with ε-transitions), the problem is said to be a nondeterministic regular realizability problem and is denoted by $\mathrm{NRR}(F)$.

The interrelation between the algorithmic complexity of the deterministic and nondeterministic versions of the problem remains an open question. The only separation was obtained in [1] in the case of regular filters modulo the $\mathbf{L} \neq \mathbf{NL}$ conjecture: for any nonempty regular language R, the $\mathrm{NRR}(R)$ problem is **NL**-complete, but for the 0^* language, the $\mathrm{RR}(0^*)$ problem is **L**-complete.

[1] Supported in part by the Russian Foundation for Basic Research, project no. 12-01-00864, and the President of the Russian Federation Council for State Support of Leading Scientific Schools, project no. NSh-4652.2012.1.

[2] Supported in part by the Russian Foundation for Basic Research, project no. 14-01-00641.

Depending on a filter, regular realizability problems (both versions) are found to be complete in various complexity classes. There are known examples of problems complete in the **NL**, **P**, **NP**, and **PSPACE** classes [2, 3]. In [4], results are obtained on universality of deterministic regular realizability problems under rather strong reductions.

However, the filters used in the universality proofs are of a highly artificial form. Therefore, an interest arises to studying the complexity of regular realizability problems for natural classes of filters, in particular, for CF languages. In this case the regular realizability problems belong to the **P** class, and their classification with respect to log-space reductions is of interest.

For the analysis of the algorithmic complexity of regular realizability problems with CF filters, we choose the nondeterministic version of the problem. In this case there is a direct connection between reductions of regular realizability problems in logarithmic space and rational transductions on the corresponding filters.

In what follows, unless otherwise specified, by a regular realizability problem we always mean a nondeterministic regular realizability problem.

Classification of CF languages with respect to rational transductions (or, equivalently, with respect to the rational dominance relation) plays an important role in the theory of CF languages (for details, see [5]). However, the structure of equivalence classes of CF languages with respect to rational dominance is found to be highly complicated. Restricting ourselves to only the algorithmic complexity of regular realizability problems with CF filters, we want to obtain a more rough and convenient classification.

This paper presents first results in this direction. Though the obtained results are far from being complete, they provide a new viewpoint on the classification problem for CF languages and suggest new formulations for a number of old conjectures related to this problem.

Regular realizability problems for CF filters that are maximal with respect to rational dominance (in other words, for generators of the rational cone of CF languages) are **P**-complete with respect to deterministic log-space reductions (see Section 3). On the other hand, many classes important for the theory of CF languages consist of languages corresponding to regular realizability problems in the **NL** class. Such are, for instance, the Greibach languages (see Section 4). Note that if a regular realizability problem belongs to the **NL** class, it is necessarily **NL**-complete.

We say that a CF language L is *easy* if $\mathrm{NRR}(L) \in \mathbf{NL}$, and if an $\mathrm{NRR}(L)$ problem is **P**-complete, we say that the CF language L is *hard*.

In Section 3 we present an example of a CF language which is hard in this sense but is not a generator of the whole cone of CF languages. Thus, the classification of CF languages with respect to algorithmic complexity of regular realizability problems is more rough than the classification with respect to the rational dominance relation.

On the other hand, this language rationally dominates all quasirational languages, i.e., is a counterexample to the well-known conjecture formulated in [6]. Based on our result, we may refine this conjecture as follows: *If a CF language rationally dominates all quasirational languages, it is hard*.

An exact boundary between easy and hard languages is not known. Moreover, there is a class which possibly contains regular realizability problems of an intermediate complexity. These are languages with polynomially bounded rational indices.

Recall that the *rational index* $\rho_L(n)$ of a language L is a function that returns the maximum (over n-state automata) length of the shortest word in the intersection of L with a language $L(\mathcal{A})$ recognized by a nondeterministic automaton $\mathcal{A}$ with n states such that $L(\mathcal{A}) \cap L \neq \varnothing$:

$$\rho_L(n) = \max_{\substack{\mathcal{A}:\, |Q_{\mathcal{A}}|=n \\ L(\mathcal{A})\cap L\neq\varnothing}} \min\{|u| \mid u \in L(\mathcal{A}) \cap L\}.$$

The rational index is also a complexity measure on languages. It is consistent with rational transductions (for a precise statement, see Section 5) and is convenient for proving that one language strictly rationally dominates another. In Section 5 we show that the regular realizability problem for CF filters with polynomially bounded rational indices belongs to the class $\mathbf{NSPACE}(\log^2 n)$. Note that numerous examples of CF languages with polynomially bounded rational indices are known [7], and the rational dominance relation on them has a rather complicated structure. However, the regular realizability problems for these languages belong to the **NL** class. The question on more complicated constructions of CF languages with polynomially bounded rational indices remains open.

2. PRELIMINARIES

The main goal of this paper is analyzing the algorithmic complexity of the NRR problem for context-free filters.

The definitions of complexity classes and reductions between them used in the present paper are standard. They can be found in most of modern textbooks on complexity theory, e.g., in [8,9].

Definition 1. A regular realizability problem $\mathrm{NRR}(F)$ is an algorithmic problem of verifying whether the intersection of a filter language F with a regular language $L(\mathcal{A})$ as the problem input is nonempty, where $\mathcal{A}$ is a nondeterministic finite automaton (with ε-transitions).

Formally,

$$\mathrm{NRR}(F) = \{\mathcal{A} \mid \mathcal{A} \in \mathrm{NFA},\ L(\mathcal{A}) \cap F \neq \varnothing\}.$$

It follows from the definition that the $\mathrm{NRR}(A^*)$ problem for a filter consisting of all words is the well-known **NL**-complete directed graph reachability problem. Below we show that the $\mathrm{NRR}(L)$ problem for an arbitrary CF language L belongs to the **P** class. Thus, the most suitable for analyzing the algorithmic complexity is the (deterministic) reduction in logarithmic space, which will be denoted by $\leq_{\log}$.

Recall basic notions and definitions from theory of CF languages (for details, see [5, 10]). The empty string will be denoted by ε. Let A_n and $\bar{A}_n$ be alphabets of cardinality n consisting of symbols $\{a_1, a_2, \ldots, a_n\}$ and $\{\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_n\}$, respectively. The CF language most important for us is the *Dyck language* D_n^*, i.e., the language of correctly nested sequences of brackets of n types defined by the grammar

$$D_n^* = \langle S \to TS + \varepsilon;\ T \to a_1 S \bar{a}_1 + \ldots + a_n S \bar{a}_n \rangle.$$

In some proofs we will need grammars in *Chomsky normal form*. Recall that such grammars obey the following properties: the right-hand side of any rule contains either a terminal, or two nonterminals, or the empty string; the empty string can be derived from an axiom only, and in this case the axiom does not enter the right-hand sides of the rules.

Recall also that a *sentential form* is a word consisting of terminals and nonterminals of a grammar. The transitive closure of the derivation relation on the set of sentential forms will be denoted by $u \Rightarrow^* v$.

A relation $R \subset A^* \times B^*$ is a *rational relation* if there exist a morphism $\varphi\colon A^* \times B^* \to M$ to a finite monoid and a subset $N \subseteq M$ such that $R = \varphi^{-1}(N)$.

We say that a language $L' \subseteq B^*$ *rationally dominates* a language $L \subseteq A^*$ if there exists a rational relation R such that $L = \{u \in A^* \mid \exists\, v \in L'\ (v, u) \in R\}$. Denote the rational dominance relation by $\leq_{\mathrm{rat}}$. We say that languages L and L' are *rationally equivalent* if $L \leq_{\mathrm{rat}} L'$ and $L' \leq_{\mathrm{rat}} L$.

A rational transduction is a multivalued partially defined map τ_R whose graph is a rational relation R. Thus, $L \leq_{\mathrm{rat}} L'$ means that $L = \tau_R(L')$. A rational transduction can be realized by by a *finite-state transducer* T, which is a nondeterministic finite automaton (NFA) with output.

In each operation step, the transducer T, depending on its state, reads a symbol (or ε) from the input tape, writes a symbol (or ε) on the output tape, and changes the state. We say that a word u belongs to $T(v)$ if for the input v there exists a path of the transducer on which the word v is read, the word u is written on the output tape, and the transducer T passes to an accepting state. Thus, the domain of a rational transduction is a regular language.

Formally, a finite-state transducer is defined by a tuple $T = (A, B, Q, q_0, \delta, F)$, where A is an input tape alphabet, B an output tape alphabet, Q a finite set of states, q_0 an initial state, $F \subseteq Q$ a set of accepting states, and $\delta\colon Q \times (A \cup \varepsilon) \times (B \cup \varepsilon) \times Q$ a transition relation.

In algorithmic applications we will assume that any finite-state transducer, as well as an automaton, is defined by lists of alphabets and states and a table of the transition relation.

Consider two finite-state transducers T_1 and T_2 realizing rational relations R_1 and R_2. We say that a transducer $T = T_1 \circ T_2$ is the composition of T_1 and T_2 if the relation R corresponding to T is of the form $R = \{(u, v) \mid \exists\, y (u, y) \in R_1, (y, v) \in R_2\}$.

The composition of a finite-state transducer T and an automaton $\mathcal{A}$ is defined similarly: we say that the automaton $\mathcal{B} = T \circ \mathcal{A}$ recognizes the language

$$\{w \mid \exists\, y \in L(\mathcal{A})\ (w, y) \in R\}.$$

The following statement is an algorithmic version of the Elgot–Mezei theorem (for a formulation of the theorem, see, e.g., [5, Theorem III.4.4]).

Proposition 1. *The composition of finite-state transducers and the compositions of a finite-state transducer and a finite automaton are computable in logarithmic space.*

Explicit constructions for the composition of finite-state transducers, as well as for the composition of a transducer and a finite automaton, are well known (see, e.g., [11, p. 88, 206]). One can easily see that these constructions are computable in log space.

A *rational cone* is a class of languages closed under the rational dominance relation. Denote by $\mathcal{T}(L)$ the minimal rational cone containing a language L. We call it the rational cone generated by L, and the language L, in turn, is called a *generator* of $\mathcal{T}(L)$. Such cones are said to be *principal*. For instance, the cone **Lin** of linear languages (see [5] for a definition) is principal: $\mathbf{Lin} = \mathcal{T}(S)$, where the symmetric language S over the alphabet $X = \{x_1, x_2, \bar{x}_1, \bar{x}_2\}$ is defined by the grammar

$$S = \langle S \to x_1 S \bar{x}_1 + x_2 S \bar{x}_2 + \varepsilon \rangle.$$

We will need the operation of *substitution* of languages for symbols. Let a map $\sigma\colon A \to 2^{B^*}$ be given; i.e., to a symbol a of the alphabet there corresponds a language L_a. Then the image $\sigma(L)$ of a language $L \subseteq A^*$ is defined in a natural way, namely, as a set of words that can be obtained by substituting a word in the language L_a for each symbol a. The *substitution closure* of a language class $\mathcal{L}$ is the minimal class that contains all substitutions of languages in $\mathcal{L}$ to languages in $\mathcal{L}$. The class **Qrt** of *quasirational languages* is by definition the substitution closure of the class **Lin** of linear languages. *Greibach languages* [12] are the substitution closure of the rational cone generated by the Dyck language with a single pair of brackets D_1^* and the symmetric language S.

By **CFL** we denote the cone of all context-free languages.

It is important for our purposes that rational dominance implies the reduction between the corresponding regular realizability problems.

Lemma 1. *If* $F_1 \leq_{\mathrm{rat}} F_2$, *then* $\mathrm{NRR}(F_1) \leq_{\log} \mathrm{NRR}(F_2)$.

Proof. Let T be a finite-state transducer such that $F_1 = T(F_2)$, and let an automaton $\mathcal{A}$ be an input of the $\mathrm{NRR}(F_1)$ problem. Construct the automaton $\mathcal{B} = T \circ \mathcal{A}$ to be the input of the $\mathrm{NRR}(F_2)$ problem. This construction is possible in log space due to Proposition 1. $\triangle$

In particular, this implies that if an NRR(F) problem is complete in a class $\mathcal{C}$, then for an arbitrary filter F' in the rational cone $\mathcal{T}(F)$ the NRR(F') problem belongs to $\mathcal{C}$.

We will use the following version of the Chomsky–Schützenberger theorem.

Theorem 1. *We have* $\mathbf{CFL} = \mathcal{T}(D_2^*)$.

In Section 3 we will prove that the NRR(D_2^*) problem is **P**-complete under log-space reductions. Thus, from the Chomsky–Schützenberger theorem and Lemma 1 it follows that the NRR(F) problem for any CF filter F belongs to the **P** class.

As was already mentioned in Section 1, a CF language L is easy if NRR(L) $\in$ **NL** and is hard if the NRR(L) problem is **P**-complete under log-space reductions.

3. HARD RR PROBLEMS FOR CF FILTERS

Let us give some examples of hard CF languages. The first of them is the Dyck language D_2^*.

Using Lemma 1 and the Chomsky–Schützenberger theorem, we obtain that any generator of the cone of CF languages is a hard language. However, as is shown below, hard languages are not confined to generators of the **CFL** cone only.

To start with, we need several technical lemmas. It is well known that the intersection of a CF language and a regular language is a CF language. We need an algorithmic version of this fact.

Lemma 2. *Let $G = (N, \Sigma, P, S)$ be a fixed CF grammar. Then there exists a log-space algorithm that inputs a description of an NFA $\mathcal{A} = (Q_\mathcal{A}, \Sigma, \delta_\mathcal{A}, q_0, F_\mathcal{A})$ and constructs a grammar $G' = (N', \Sigma, P', S')$ generating the language $L(G) \cap L(\mathcal{A})$. The size of this grammar is polynomial in $|Q_\mathcal{A}|$.*

Since the proof of this lemma employs a construction that we need below, we present the proof for completeness.

Proof. First assume that $\mathcal{A}$ has no ε-transitions. Let the nonterminal set N' consist of an axiom S' and nonterminals of the form $[qAp]$ with $A \in N$ and $q, p \in Q_\mathcal{A}$. We construct a set P' of rules by adding for each rule $A \to X_1 X_2 \dots X_n$ in P the set of rules

$$\{[qAp] \to [qX_1r_1][r_1X_2r_2]\dots[r_{n-1}X_np] \mid (q, p, r_1, r_2, \dots, r_{n-1}) \in Q_\mathcal{A}^{n+1}\}$$

to P'. We also add to P' the rules $[q\sigma p] \to \sigma$ if $p \in \delta_\mathcal{A}(q, \sigma)$ and the rules $S' \to [q_0 S q_f]$ for each state q_f in $F_\mathcal{A}$.

Now let us prove that $L(G') = L(G) \cap L(\mathcal{A})$. Let G generate a word $w = w_1 w_2 \dots w_n$. Then G' generates all possible sentential forms

$$[q_0 w_1 r_1][r_1 w_2 r_2]\dots[r_{n-1} w_n q_f]$$

with $q_f \in F_\mathcal{A}$ and $r_i \in Q_\mathcal{A}$. The derivation $[q_0 w_1 r_1][r_1 w_2 r_2]\dots[r_{n-1} w_n q_f] \Rightarrow^* w_1 w_2 \dots w_n$ is realized if and only if $\mathcal{A}$ accepts the word w, i.e., when w is read, there is a sequence of transitions from the initial state to the accepting state. This sequence is what is encoded in the sentential form given above. Thus, if a word w is generated by G and is accepted by $\mathcal{A}$, it is also generated by G'.

Conversely, if G' generates a word w, then each symbol w_i is derived from some nonterminal $[qw_ip]$. By the construction of G', the word w was derived from some sentential form

$$[q_0 w_1 r_1][r_1 w_2 r_2]\dots[r_{n-1} w_n q_f],$$

which encodes a sequence of transitions of $\mathcal{A}$ from the initial state to the accepting state when w is read. Thus, if G' generates a word, this word is accepted by $\mathcal{A}$. It is also clear that G' generates only the words that are generated by G. Thus, $L(G') = L(G) \cap L(\mathcal{A})$.

The size of G' is polynomial in $|Q_\mathcal{A}|$. The nonterminal set N' is of size $O(|N||Q_\mathcal{A}|^2)$. Let k be the length of the longest rule in P. Then for any rule in P there are constructed at most $|Q_\mathcal{A}|^{k+1}$

corresponding rules in P', and for the rules $[q\sigma p] \to \sigma$ or $S' \to [q_0 S q_f]$, at most $O(|Q_{\mathcal{A}}|^2)$ rules are constructed.

Therefore, the grammar G' can be constructed in logarithmic space: the set P' of rules is split into subsets corresponding to the rules in P, and each of the subsets is constructed by considering all possible tuples of $k+1$ states of $\mathcal{A}$, where $k = O(1)$.

Allowing for ε-transitions merely increases the constant $k+1$ to $2k$. For each rule $A \to X_1 \dots X_n$ we add the rules

$$[qAp] \to [qX_1q_1][q_2X_2q_3]\dots[q_{2n-1}X_np],$$

where for all $i < 2n-1$ we have $q_i = q_{i+1}$ or $q_i \xrightarrow{\varepsilon} q_{i+1}$. For rules of the form $[q\sigma p] \to \sigma$, we add all rules such that $q \xrightarrow{\varepsilon} q'$, $p' \xrightarrow{\varepsilon} p$, and $p' \in \delta_{\mathcal{A}}(q', \sigma)$. $\triangle$

We also need an algorithmic version of the Chomsky–Schützenberger theorem.

Lemma 3. *There exists a log-space algorithm that inputs a description of a CF grammar $G = (N, \Sigma, P, S)$ and constructs a finite-state transducer T such that $T(D_2^*) = L(G)$.*

A proof of this statement can easily be obtained from appropriate proofs of the Chomsky–Schützenberger theorem. For example, constructions from the proof of [5, Theorem II.3.10] are realized in logarithmic space in an obvious way. (Note that any morphism can be realized by a finite-state transducer.)

Lemmas 2 and 3 easily imply the **P**-hardness of the Dyck language D_2^*.

Theorem 2. *The* NRR(D_2^*) *problem is* **P***-complete.*

Proof. We reduce the well-known **P**-complete problem of verifying whether a language generated by a CF grammar is empty [13] to NRR(D_2).

Given a grammar G, we construct a finite-state transducer T such that $T(D_2^*) = L(G)$ by using the construction from Lemma 3.

Let $\mathcal{A}$ be a nondeterministic automaton obtained from the finite-state transducer T by ignoring the output tape. The intersection $L(\mathcal{A}) \cap D_2^*$ is nonempty if and only if the language $L(G)$ is nonempty. The map $G \to \mathcal{A}$ is the desired reduction.

It remains to prove that the NRR(D_2^*) problem belongs to **P**. We reduce this problem to the problem of verifying whether a language generated by a CF grammar is nonempty: given the automaton $\mathcal{A}$ at the input, we construct a grammar G such that $L(G) = L(\mathcal{A}) \cap D_2^*$ by using Lemma 2. $\triangle$

Corollary 1. *Any generator of the rational cone* **CFL** *is a hard language.*

Let us give an example of a hard language other than a generator of the **CFL**. In [14] it is proved that there exists a principal rational cone distinct from the whole cone of CF languages but containing the family **Qrt** of quasirational languages.

For some generator of this cone, we will prove **P**-completeness of the nondeterministic regular realizability problem. Our construction is based on the work [10].

For brevity, we denote the alphabet of the D_1^* language by $A = \{a, \bar{a}\}$. Define the *padding* $S_{\#}$ of a symmetric language S as a language over the alphabet $\{x_1, x_2, \bar{x}_1, \bar{x}_2, \#\}$ consisting of words of the form $m_1y_1m_2y_2\dots m_ny_n$, where $y_i \in \{x_1, x_2, \bar{x}_1, \bar{x}_2\}$, $m_i \in \#^*$, and $y_1 \dots y_n \in S$ (this is a particular case of a syntactic substitution).

Let $M = aS_{\#}\bar{a} \cup \varepsilon$. Recursively define the language $M^{(\infty)}$ as follows: $w \in M^{(\infty)}$ if and only if either $w \in M$ or

$$w = ay_1az_1\bar{a}y_2az_2\bar{a}\dots y_{n-1}az_{n-1}\bar{a}y_n\bar{a},$$

where $y_1, y_n \in X^*$, $y_i \in X^+$ $2 \le i \le n-1$, $az_i\bar{a} \in M^{(\infty)}$, and $ay_1y_2\dots y_n\bar{a} \in M$.

Remark 1. The padding using # symbols is purely a technicality required only because of the condition $y_i \in X^+$ in the definition of $M^{(\infty)}$.

Let $\pi_A\colon (X \cup A)^* \to A^*$ be the morphism that erases the symbols of an alphabet X. Define the language $M^{(+)}$ as $\pi_A^{-1}(A^* \setminus D_1^*)$.

Finally, define $S_\#^\uparrow = M^{(\infty)} \cup M^{(+)}$.

Note that the languages S and $S_\#$ are rationally equivalent. Therefore, $S_\#$ generates the cone **Lin** of linear languages. From the results of [10, Propositions 3.19 and 3.20] and from this observation, we obtain the following fact.

Theorem 3. *The language $S_\#^\uparrow$ is a CF language. It is not a generator of the cone* **CFL**, *but the cone generated by $S_\#^\uparrow$ includes all quasirational languages.*

Now we expose a plan of the proof of **P**-completeness of $S_\#^\uparrow$. The language $S_\#^\uparrow$ is the union of the languages $M^{(\infty)}$ and $M^{(+)}$, and the language $M^{(\infty)}$ is expressible enough to reduce $\mathrm{NRR}(D_2^*)$ to $\mathrm{NRR}(M^{(\infty)})$: we will encode i-brackets as ax_i and $\bar{x}_i\bar{a}$. To apply this idea, in constructing the reduction we should confine ourselves with automata that do not accept words of $M^{(+)}$.

Let us describe a class of such automata which is wide enough for our purposes.

Definition 2. We say that an NFA $\mathcal{A}$ over the alphabet $A_n \cup \bar{A}_n$ is *marked* if there exists a function $h\colon Q_\mathcal{A} \to \mathbb{Z}$ such that

$$\begin{aligned}
&h(q') = h(q) + 1 && \text{if there exists a transition } q \xrightarrow{a_j} q' \text{ in } \mathcal{A},\\
&h(q') = h(q) - 1 && \text{if there exists a transition } q \xrightarrow{\bar{a}_j} q' \text{ in } \mathcal{A},\\
&h(q) = 0 && \text{if } q \text{ is either an initial or an accepting state of } \mathcal{A}.
\end{aligned}$$

By a position in a word we mean a place between consecutive symbols of the word and also the beginning and end of the word (formally, positions are prefixes of the word ordered with respect to the length). The *height* of a position is the difference between the number of symbols of A_n and those of $\bar{A}_n$ located before this position. In these terms, words of D_1^* are specified by two conditions: the height of any position is nonnegative and the height of the final position is zero.

Proposition 2. *Let an NFA $\mathcal{A}$ satisfy the condition $D_2^* \cap L(\mathcal{A}) \neq \varnothing$. Then there exists a word $w \in D_2^* \cap L(\mathcal{A}) \neq \varnothing$ such that the height of any position of w is $O(|Q_\mathcal{A}|)^2$.*

Proof. For the language D_2^*, we will use the grammar in Chomsky normal form, and for $D_2^* \cap L(A)$, the grammar described in the proof of Lemma 2.

The height of positions is upper bounded by the height of the derivation tree in the grammar generated by the language $D_2^* \cap L(A)$, which is nonempty.

One can easily see that any grammar generating a nonempty language has a word whose derivation tree height is not greater than the number of nonterminals in this grammar.

It remains to note that the grammar for $D_2^* \cap L(A)$ has $O(|Q_\mathcal{A}|^2)$ nonterminals. $\triangle$

In the following we need a syntactic transformation of an automata defined over the alphabet $A_2 \cup \bar{A}_2$.

Proposition 3. *There exists a transformation μ that inputs a description of an automaton $\mathcal{A}$ over the alphabet $A_2 \cup \bar{A}_2$ and produces a description of a marked automaton $\mathcal{A}' = \mu(\mathcal{A})$ such that*

- $L(\mathcal{A}) \cap D_2^* \neq \varnothing \iff L(\mathcal{A}') \cap D_2^* \neq \varnothing$;
- *For any word $w \in L(\mathcal{A}')$ the height of any position is nonnegative and the height of the final position is* 0.

The transformation μ is computable in logarithmic space.

Remark 2. Since the height is the difference between the numbers of opening and closing brackets of both types, the language $L(\mathcal{A}')$ is not necessarily a subset of D_2^*.

Proof of Proposition 3. By Proposition 2 there is an upper bound $m = O(|Q_{\mathcal{A}}|^2)$ on the minimum of the maximum height of a position in a word of $L(\mathcal{A}) \cap D_2^*$ provided that this intersection is nonempty. Note that the estimate m is computable in logarithmic space.

The set of states of $\mathcal{A}'$ is $Q_{\mathcal{A}} \times \{0, \ldots, m\} \cup \{r\}$, where r is a specially assigned rejecting state.

If $q \xrightarrow{\alpha} q'$, where $\alpha \in \{a_1, a_2\}$ is a transition in $\mathcal{A}$, then in $\mathcal{A}$ there exist the transitions $(q, i) \xrightarrow{\alpha} (q', i+1)$ for all $0 \le i < m$ and the transition $(q, m) \xrightarrow{\alpha} r$.

If $q \xrightarrow{\alpha} q'$, where $\alpha \in \{\bar{a}_1, \bar{a}_2\}$ is a transition in $\mathcal{A}$, then in $\mathcal{A}$ there exist the transitions $(q, i) \xrightarrow{\alpha} (q', i-1)$ for all $0 < i \le m$ and the transition $(q, 0) \xrightarrow{\alpha} r$.

The initial state of $\mathcal{A}'$ is $(q_0, 0)$, where q_0 is the initial state of $\mathcal{A}$. The set of accepting states of $\mathcal{A}'$ is $F \times \{0\}$, where F is the set of accepting states of $\mathcal{A}$.

It is clear that a description of $\mathcal{A}'$ is computable in logarithmic space.

The construction of $\mathcal{A}'$ implies that position heights are nonnegative and that the height of the initial state is zero. It remains to prove the equivalence of the two nonempty intersection conditions.

Note that if $L(\mathcal{A}) \cap D_2^* = \varnothing$, then $L(\mathcal{A}') \cap D_2^* = \varnothing$. Conversely, if $L(\mathcal{A}) \cap D_2^* \ne \varnothing$, then by Proposition 2 there exists a word $w \in L(\mathcal{A}) \cap D_2^*$ such that the height of any position of w is upper bounded by m. Thus, this word is accepted by $\mathcal{A}'$. $\triangle$

Now we pass to the proof of the theorem.

Theorem 4. *The* $\mathrm{RR}(S_{\#}^{\uparrow})$ *problem is* **P***-complete under log-space reductions.*

Proof. We reduce $\mathrm{NRR}(D_2^*)$ to $\mathrm{NRR}(S_{\#}^{\uparrow})$.

Let an automaton $\mathcal{A}$ be the input of $\mathrm{NRR}(D_2^*)$, and let $\mathcal{A}' = \mu(\mathcal{A})$ be obtained from $\mathcal{A}$ via the marking procedure.

Let us construct an automaton $\mathcal{B}$ over the alphabet $A \cup X \cup \{\#\}$ such that $L(\mathcal{A}') \cap D_2^* \ne \varnothing$ if and only if $L(\mathcal{B}) \cap S_{\#}^{\uparrow} \ne \varnothing$.

Define a morphism $\varphi\colon (A_2 \cup \bar{A}_2)^* \to (A \cup X \cup \{\#\})^*$ as follows:

$$\begin{aligned} &\varphi\colon a_1 \mapsto a x_1, \\ &\varphi\colon \bar{a}_1 \mapsto \bar{x}_1 \bar{a} \#\#, \\ &\varphi\colon a_2 \mapsto a x_2, \\ &\varphi\colon \bar{a}_2 \mapsto \bar{x}_2 \bar{a} \#\#. \end{aligned} \tag{1}$$

This morphism is injective, since images of the symbols in (1) form a prefix code. Denote by T the finite-state transducer corresponding to the inverse morphism, and by $\mathcal{B}' = T \circ \mathcal{A}'$ the composition of this transducer and $\mathcal{A}'$.

Let $\mathcal{B}$ accept precisely the words of the form $a x_1 x_2 w \bar{x}_2 \bar{x}_1 \bar{a}$ with $w = \varphi(u)$ and $u \in L(\mathcal{A}')$.

It follows from the definition of $M^{(\infty)}$ that if $u \in D_2^*$, then $a x_1 x_2 \varphi(u) \bar{x}_2 \bar{x}_1 \bar{a} \in M^{(\infty)}$. Hence, if $L(\mathcal{A}') \cap D_2^* \ne \varnothing$, then $L(\mathcal{B}) \cap S_{\#}^{\uparrow} \ne \varnothing$.

Now let us prove the converse inclusion. Let

$$w = a x_1 x_2 \varphi(u) \bar{x}_2 \bar{x}_1 \bar{a} \in S_{\#}^{\uparrow} \cap L(\mathcal{B}).$$

Note that $w \in L(\mathcal{B})$ implies $u \in L(\mathcal{A}')$. By the construction of $\mathcal{A}'$, the heights of positions in u are nonnegative and the height of the final position is 0. It is clear from the form of morphism (1) that the same holds for w. Hence, $w \notin M^{(+)} = \pi_A^{-1}(A^* \setminus D_1^*)$; i.e., $\pi_A(w) \in D_1^*$. The brackets in

a correctly nested sequence of brackets are split into pairs of matching brackets. Take a pair of matching brackets a and $\bar{a}$ in w:

$$w = w_0 a x_i w_1 \bar{x}_j \bar{a} w_2.$$

By the definition of $M^{(\infty)}$, after deleting all subwords of the form $az\bar{a}$ in the word w_1, the word $ax_i w_1 \bar{x}_j \bar{a}$ is transformed into a word $ax_i v \bar{x}_j \bar{a}$ of the language M. Thus, $x_i v \bar{x}_j \in S_{\#}$, so that $i = j$.

Hence it follows that $u \in D_2^* \cap L(\mathcal{A}')$.

Thus, we have proved correctness of the reduction. Now consider the algorithmic complexity. The automaton $\mathcal{B}' = T \circ \mathcal{A}'$ is constructible in logarithmic space by Proposition 1. The automaton $\mathcal{B}$ differs from $\mathcal{B}'$ by preprocessing and postprocessing states required to process the prefix ax_1x_2 and suffix $\bar{x}_2\bar{x}_1\bar{a}$. $\triangle$

4. EASY RR PROBLEMS FOR CF FILTERS

Now we pass to examples of easy languages. The simplest example are regular languages. Below we will show that the symmetric language and the Dyck language D_1^* are easy. A rather simple observation shows that easy language are closed under substitutions. Thus, we will prove that Greibach languages are also easy.

Lemma 4. *We have* $\mathrm{NRR}(S) \in \mathbf{NL}$.

The proof of Lemma 4 can be obtained by slightly modifying the arguments in [2], where a similar result for the language of palindromes is proved.

Recall that a *counter automaton* is a pushdown automaton with stack alphabet consisting of a single symbol.

Lemma 5. *Let L_c be a CF language recognizable by a counter automaton. Then the regular realizability problem* $\mathrm{NRR}(F)$ *belongs to the* **NL** *class.*

In the proof of the lemma we use the following fact.

Lemma 6 [15]. *Let M be a counter automaton with n states. Then the shortest word w accepted by M is of length no more than n^3, and the value of the counter of M when processing the word w never exceeds n^2.*

Proof of Lemma 5. Let M be a counter automaton that accepts when reaching the final state, and let M recognize L_c. Let $\mathcal{A}$ be the automaton at the input of the regular realizability problem.

We construct a counter automaton $M_{\mathcal{A}}$ recognizing the language $L(M) \cap L(\mathcal{A}) = L_c \cap L(\mathcal{A})$ using the standard product construction: the automaton has state set $Q_M \times Q_{\mathcal{A}}$, initial state $(q_0^M, q_0^{\mathcal{A}})$, set of accepting states $F_M \times F_{\mathcal{A}}$, and transition relation $\delta_{M_{\mathcal{A}}}$ such that $\delta_M(q, \sigma, z) \vdash (q', z')$ and $\delta_{\mathcal{A}}(p, \sigma) = p'$ implies $\delta_{M_{\mathcal{A}}}((q, p), \sigma, z) \vdash ((q', p'), z')$.

Thus, $M_{\mathcal{A}}$ is a counter automaton with $|Q_M||Q_{\mathcal{A}}| = cn$ states. By Lemma 6 we obtain that the value of the counter of $M_{\mathcal{A}}$ does not exceed $(cn)^2$ when processing the shortest word. Now we construct an NFA $\mathcal{B}$ such that the language $L(\mathcal{B})$ includes all words of $L(M_{\mathcal{A}})$ on which the counter of $M_{\mathcal{A}}$ does not exceed $(cn)^2$.

The automaton $\mathcal{B}$ has $O(n^3)$ states and is naturally constructible in logarithmic space, similarly to the construction used in the proof of Proposition 3. Note that $L(M_{\mathcal{A}}) \neq \varnothing$ if and only if $L(\mathcal{B}) \neq \varnothing$. Thus, the map $\mathcal{A} \to \mathcal{B}$ is a reduction of $\mathrm{NRR}(L_c)$ to the $\mathrm{NRR}(\Sigma^*)$ problem, which belongs to **NL**. $\triangle$

Clearly, the Dyck language D_1^* is recognizable by a counter automaton. Thus, we obtain the following result.

Corollary 2. *We have* $\mathrm{NRR}(D_1^*) \in \mathbf{NL}$.

Lemma 7. *If L and L_a are easy languages, then $\sigma(L)$ is also an easy language.*

Proof. Let $\mathcal{A}$ be an automaton at the input of the $\mathrm{RR}(\sigma(L))$ problem. Define an automaton $\mathcal{A}'$ over the alphabet A with state set $Q_{\mathcal{A}'} = Q_{\mathcal{A}}$. The automaton $\mathcal{A}'$ has a transition $q \xrightarrow{a} q'$ if and only if there exists a word $w \in L_a$ such that there is a path $q \xrightarrow{w} q'$ in $\mathcal{A}$.

It is clear from the definition that $L(\mathcal{A}) \cap \sigma(L) \neq \varnothing$ if and only if $L(\mathcal{A}') \cap L \neq \varnothing$. To apply an **NL** algorithm for solving the $\mathrm{RR}(L)$ problem, it is required to compute the transition relation of $\mathcal{A}'$. This transition relation is not a part of the problem input, but it can be computed by an **NL** algorithm for the $\mathrm{RR}(L_a)$ problem. Thus we obtain a resulting **NL** algorithm for $\mathrm{RR}(\sigma(L))$. $\triangle$

Lemmas 4 and 7 and Corollary 2 yield the main result of this section.

Theorem 5. *Greibach languages are easy.*

5. POLYNOMIAL RATIONAL INDEX

It is not known whether there exist languages other than hard and easy. In this section we describe a class of CF languages that might contain languages of intermediate complexity between hard and easy. These are languages with polynomially bounded rational index.

The rational index was introduced in [16]. It was found to be a rather useful characteristic of CF languages, since it does not significantly increase under transduction.

Theorem 6 [16]. *If $L' \leq_{\mathrm{rat}} L$, then there exists a constant c such that*

$$\rho_{L'}(n) \leq cn(\rho_L(cn) + 1).$$

Thus, the rational index can be used to separate languages with respect to strict rational dominance. Note that for the rational index of a generator of the **CFL** cone there is a rather tight estimate.

Theorem 7 [17]. *The rational index of any generator of the* **CFL** *cone lies in* $\exp(\Theta(n^2/\log n))$.

Examples of easy languages from Section 4 have polynomially bounded rational indices. It is but natural to suppose that any language with a polynomially bounded rational index is easy. There are some arguments in favor of this conjecture.

For example, CF languages with rational index $\Theta(n^\gamma)$ were proposed in [7] for any positive algebraic number $\gamma > 1$. All these languages are constructed from linear languages by specific substitutions using a construction similar to that of the $S^{\uparrow}_{\#}$ language described above. Therefore, all of them are easy. The proof is rather technical because of tedious definitions of these languages and is therefore omitted here. We note only that the proof employs Lemma 7 and a separate reasoning concerning easiness of the non-substitution part of a language.

Unfortunately, in the case of a polynomially bounded rational index we can only prove a weaker estimate for the algorithmic complexity.

Theorem 8. *Let F be a context-free filter with a polynomially bounded rational index. Then the* $\mathrm{NRR}(F)$ *problem belongs to the* $\mathbf{NSPACE}(\log^2 n)$ *class.*

We will use a technique close to that of [18]. We need the following auxiliary result.

Lemma 8 [18]. *Let G be a grammar in Chomsky normal form. Then for any word $w = xyz$ of length n in $L(G)$ there exists a nonterminal A in the derivation tree such that A generates y and $n/3 \leq |y| \leq 2n/3$.*

Proof of Theorem 8. Consider a grammar G' in Chomsky normal form such that $L(G') = F$. This means in particular that right-hand sides of rules in G' are of lengths at most 2.

By applying Lemma 2, to an automaton $\mathcal{A}$ with n states we assign a grammar G such that $L(G) = L(\mathcal{A}) \cap F$. If $L(\mathcal{A}) \cap F \neq \varnothing$, then there exists a word w generated by G of length

at most $\rho_F(n)$; i.e., its length is polynomial in n. The desired nondeterministic automaton guesses such a word and verifies whether a derivation in G exists.

An algorithm in quadratic logarithmic space cannot write the grammar G explicitly, since the construction in Lemma 2 makes the size of the grammar G' n^2 times as large. Instead, the algorithm nondeterministically guesses the derivation tree for w in G. Informally, it reconstructs the derivation tree starting from the "central" branch.

The main part of the algorithm is a recursive procedure that checks the correctness for a nonterminal $A = [qA'p]$ of G. We say that a nontermial $A = [qA'p]$ is correct if A generates a word w in G.

If a nontermial is of the form $[q\sigma p]$ where σ is a terminal, then the procedure must check that there exists a path $q \xrightarrow{\sigma} p$ in $\mathcal{A}$.

In the general case, the correctness checking procedure nondeterministically guesses a nonterminal $A_1 = [\ell_1 A'_1 r_1]$ such that $w = p_1 u_1 s_1$ and A_1 generates u_1 with $1/3|w| \le |u_1| \le 2/3|w|$. Then the procedure is recursively applied to the nonterminal A_1. In the case of success, the procedure sets $i := 1$ and repeats the following steps:

1. Nondeterministically guess a nonterminal $A_{i+1} = [\ell_{i+1} A'_{i+1} r_{i+1}]$ which is an ancestor of A_i in the derivation tree. Two cases are possible:
   (i) either $A_{i+1} \to [q'C'\ell_{i+1}]A_i$ in the grammar G (set $C := [q'C'\ell_{i+1}]$);
   (ii) or $A_{i+1} \to A_i[r_{i+1}C'p']$ (set $C := [r_{i+1}C'p']$);
2. Recursively apply the correctness checking procedure to the nonterminal C;
3. In the case of success, set $i := i + 1$.

The procedure terminates successfully if at some time we obtain $A_j = A$. If at least one correctness check was unsuccessful, the whole procedure returns failure.

In recursive calls, the word length reduces in each step at least by a factor of 2/3. Thus, the total number of recursive calls is of the order of $O(\log n)$, where n is the input length. In each recursive call one has to store a triple (automaton state, nontermial of G', automaton state). A description of an automaton state requires $O(\log n)$ space, and a description of a nonterminal of the grammar requires a constant space, since the grammar G' is fixed. Thus, he algorithm requires $O(\log^2 n)$ space in total. $\triangle$

The authors are grateful to Abuzer Yakaryılmaz for pointing out the result of Lemma 5 and for a reference to the proof of a lemma similar to Lemma 6.

REFERENCES

1. Rubtsov, A.A., On Prompting Regular Languages in the Model of Generalized Nondeterministic Automata, in *Matematicheskie modeli i zadachi upravleniya* (Mathematical Models and Control Problems), Moscow: Moscow Inst. of Physics and Technology, 2011, pp. 61–67.

2. Anderson, T., Loftus, J., Rampersad, N., Santean, N., and Shallit, J., Detecting Palindromes, Patterns and Borders in Regular Languages, *Inform. and Comput.*, 2009, vol. 207, no. 11, pp. 1096–1118.

3. Vyalyi, M.N., On Regular Realizability Problems, *Probl. Peredachi Inf.*, 2011, vol. 47, no. 4, pp. 43–54 [*Probl. Inf. Trans.* (Engl. Transl.), 2011, vol. 47, no. 4, pp. 342–352].

4. Vyalyi, M.N., On Expressive Power of Regular Realizability Problems, *Probl. Peredachi Inf.*, 2013, vol. 49, no. 3, pp. 86–104 [*Probl. Inf. Trans.* (Engl. Transl.), 2013, vol. 49, no. 3, pp. 276–291].

5. Berstel, J., *Transductions and Context-Free Languages*, Stuttgart: Teubner, 1979.

6. Autebert, J.-M., Beauquier, J., Boasson, L., and Nivat, M., Quelques problémes ouverts en théorie des langages algébriques, *RAIRO Inform. Théor.*, 1979, vol. 13, no. 4, pp. 363–378.

7. Pierre, L. and Farinone, J.-M., Context-Free Languages with Rational Index in $\Theta(n^\lambda)$ for Algebraic Numbers λ, *RAIRO Inform. Théor. Appl.*, 1990, vol. 24, no. 3, pp. 275–322.

8. Arora, S. and Barak, B., *Computational Complexity: A Modern Approach*, Cambridge, UK: Cambridge Univ. Press, 2009.

9. Sipser, M., *Introduction to the Theory of Computation*, Boston, MA: Course Technology Cengage Learning, 2012, 3rd ed.

10. Berstel, J. and Boasson, L., Context-Free Languages, *Handbook of Theoretical Computer Science*, van Leewen, J., Ed., Amsterdam: Elsevier, 1990, vol. B, pp. 59–102.

11. *Word, Language, Grammar*, vol. 1 of *Handbook of Formal Languages*, Rozenberg, G. and Salomaa, A., Eds., Berlin: Springer, 1997.

12. Greibach, S.A., An Infinite Hierarchy of Context-Free Languages, *J. ACM*, 1969, vol. 16, no. 1, pp. 91–106.

13. Greenlaw, R., Hoover, H.J., and Ruzzo, W.L., *Limits to Parallel Computation: P-Completeness Theory*, New York: Oxford Univ. Press, 1995.

14. Boasson, L., Non-générateurs algébriques et substitution, *RAIRO Inform. Théor.*, 1985, vol. 19, no. 2, pp. 125–136.

15. Yakaryılmaz, A., One-Counter Verifiers for Decidable Languages, *Computer Science—Theory and Applications (Proc. 8th Int. Computer Science Sympos. in Russia (CSR'2013), Ekaterinburg, Russia, June 25–29, 2013)*, Bulatov, A.A. and Shur, A.M., Eds., Lect. Notes Comput. Sci., vol. 7913, Heidelberg: Springer, 2013, pp. 366–377.

16. Boasson, L., Courcelle, B., and Nivat, M., The Rational Index: A Complexity Measure for Languages, *SIAM J. Comput.*, 1981, vol. 10, no. 2, pp. 284–296.

17. Pierre, L., Rational Indexes of Generators of the Cone of Context-Free Languages, *Theoret. Comput. Sci.*, 1992, vol. 95, no. 2, pp. 279–305.

18. Lewis, P.M., Stearns, R.E., and Hartmanis, J., Memory Bounds for Recognition of Context-Free and Context-Sensitive Languages, in *Proc. 6th Annual Sympos. on Switching Circuit Theory and Logical Design (SWCT'1965), Ann Arbor, MI, USA, Oct. 6–8, 1965*, New York: IEEE, 1965, pp. 191–202.