

Regular realizability problems and context-free languages

A. Rubtsov^{*23} and M. Vyalyi^{**123}

¹ Computing Centre of RAS

² Moscow Institute of Physics and Technology

³ National Research University Higher School of Economics

rubtsov99@gmail.com

vyalyi@gmail.com

We investigate regular realizability (RR) problems, which are the problems of verifying whether the intersection of a regular language – the input of the problem – and a fixed language, called a filter, is non-empty. In this paper we focus on the case of context-free filters. The algorithmic complexity of the RR problem is a very coarse measure of the complexity of context-free languages. This characteristic respects the rational dominance relation. We show that a RR problem for a maximal filter under the rational dominance relation is **P**-complete. On the other hand, we present an example of a **P**-complete RR problem for a non-maximal filter. We show that RR problems for Greibach languages belong to the class **NL**. We also discuss RR problems with context-free filters that might have intermediate complexity. Possible candidates are the languages with polynomially-bounded rational indices. We show that RR problems for these filters lie in the class **NSPACE**($\log^2 n$).

1 Introduction

The context-free languages form one of the most important classes for formal language theory. There are many ways to characterize complexity of context-free languages. In this paper we propose a new approach to classification of context-free languages based on the algorithmic complexity of the corresponding regular realizability (RR) problems.

By ‘regular realizability’ we mean the problem of verifying whether the intersection of a regular language – the input of the problem – and a fixed language, called a filter, is non-empty. The filter F is a parameter of the problem. Depending on the representation of a regular language, we distinguish the deterministic RR problems $RR(F)$ and the nondeterministic ones $NRR(F)$, which correspond to the description of the regular language either by a deterministic or by a non-deterministic finite automaton.

^{*} Supported in part by RFBR grant 14–01–00641.

^{**} Supported in part RFBR grant 14–01–93107 and the scientific school grant NSh4652.2012.1.

The relation between algorithmic complexities of $\text{RR}(F)$ and $\text{NRR}(F)$ is still unknown. For our purpose – the characterization of the complexity of a context-free language – the nondeterministic version is more suitable. One of the reasons for this choice is a rational dominance relation \leq_{rat} (defined in Section 2). We show below that the dominance relation on filters $F_1 \leq_{\text{rat}} F_2$ implies the log-space reduction $\text{NRR}(F_1) \leq_{\log} \text{NRR}(F_2)$. So our classification is a very coarse version of the well-known classification of **CFL** by the rational dominance relation (see the book [2] for a detailed exposition of this topic).

Depending on a filter F , the algorithmic complexity of the regular realizability problem varies drastically. There are **RR** problems that are complete for complexity classes such as **L**, **NL**, **P**, **NP**, **PSPACE** [1,11]. In [12] a huge range of possible algorithmic complexities of the deterministic **RR** problems was presented. We prove below that for context-free nonempty filters the possible complexities are in the range between **NL**-complete problems and **P**-complete problems. Examples of **P**-complete **RR** problems are provided in Section 3. The filter consisting of all words provides an easy example of an **NL**-complete **RR** problem. In this case, the problem is exactly the reachability problem for digraphs. The upper bound by the class **P** follows from the reduction of an arbitrary **NRR**-problem specified by a context-free filter to the problem of verifying the emptiness of a language generated by a context-free grammar. We prove it in Section 3.

We will call a context-free language L *easy* if $\text{NRR}(L) \in \mathbf{NL}$ and *hard* if $\text{NRR}(L)$ is **P**-complete. In Section 3 we present an example of a non-generator of the CFLs cone, which is hard in this sense. In Section 4 we provide examples of easy languages. They cover a rather wide class – the so-called Greibach languages introduced in [7].

The exact border between hard and easy languages is unknown. Moreover, there are candidates for an intermediate complexity of **RR** problems. They are languages with polynomially-bounded rational indices.

The rational index was introduced in [5]. Recall that *rational index* $\rho_L(n)$ of a language L is a function that returns the maximum length of the shortest word from the intersection of the language L and a language $L(\mathcal{A})$ recognized by an automaton \mathcal{A} with n states, provided $L(\mathcal{A}) \cap L \neq \emptyset$:

$$\rho_L(n) = \max_{\mathcal{A}: |Q_{\mathcal{A}}|=n, L(\mathcal{A}) \cap L \neq \emptyset} \min\{|u| \mid u \in L(\mathcal{A}) \cap L\}. \quad (1)$$

The growth rate of the language's rational index is another measure of the complexity of a language. This measure is also related to the rational dominance (see Section 5 for details).

In Section 5 we prove that the **RR** problem for a context-free filter having polynomially-bounded rational index is in the class **NSPACE**($\log^2 n$). Note also that there are many known CFLs having polynomially-bounded rational indices [10]. But the **RR** problems for these languages are in **NL**. It would be interesting to find more sophisticated examples of CFLs having polynomially-bounded rational indices.

2 Preliminaries

The main point of our paper is investigation of the complexity of the NRR-problem for filters from the class of context-free languages **CFL**.

Definition 1. The regular realizability problem $\text{NRR}(F)$ is the problem of verifying non-emptiness of the intersection of the filter F with a regular language $L(\mathcal{A})$, where \mathcal{A} is an NFA. Formally

$$\text{NRR}(F) = \{\mathcal{A} \mid \mathcal{A} \text{ is an NFA and } L(\mathcal{A}) \cap F \neq \emptyset\}.$$

It follows from the definition that the problem $\text{NRR}(A^*)$ for the filter consisting of all words under alphabet A is the well-known **NL**-complete problem of digraph reachability. We will show below that $\text{NRR}(L) \in \mathbf{P}$ for an arbitrary context-free filter L . So it is suitable to use deterministic log-space reductions in the analysis of algorithmic complexity of the RR problems specified by CFL filters. We denote the deterministic log-space reduction by \leq_{\log} .

Let us recall some basic notions and fix notation concerning the CFLs. For a detailed exposition see [2,3]. We will refer to the empty word as ε . Let A_n and \bar{A}_n be the n -letter alphabets consisting of the letters $\{a_1, a_2, \dots, a_n\}$ and $\{\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n\}$ respectively. A well-known example of a context-free language, the *Dyck language* D_n , is defined by the grammar

$$S \rightarrow SS \mid \varepsilon \mid a_1 S \bar{a}_1 \mid \dots \mid a_n S \bar{a}_n.$$

Fix alphabets A and B . A language $L \subseteq A^*$ is *rationally dominated* by $L' \subseteq B^*$ if there exists a rational relation R such that $L = R(L')$, where $R(X) = \{u \in A^* \mid \exists v \in X (v, u) \in R\}$. We denote rational domination as \leq_{rat} . We say that languages L, L' are *rationally equivalent* if $L \leq_{\text{rat}} L'$ and $L' \leq_{\text{rat}} L$.

A rational relation is a graph of a multivalued mapping τ_R . We will call the mapping τ_R with a rational graph as a rational transduction. So $L \leq_{\text{rat}} L'$ means that $L = \tau_R(L')$. Such a transduction can be realized by a *rational transducer* (or finite-state transducer) T , which is a nondeterministic finite automaton with input and output tapes, where ε -moves are permitted. We say that u belongs to $T(v)$ if for the input v there exists a path of computation on which T writes the word u on the output tape and halts in the accepting state. Formally, a rational transducer is defined by the 6-tuple $T = (A, B, Q, q_0, \delta, F)$, where A is the input alphabet, B is the output alphabet, Q is the (finite) state set, q_0 is the initial state, $F \subseteq Q$ is the set of accepting states and $\delta: Q \times (A \cup \varepsilon) \times (B \cup \varepsilon) \times Q$ is the transition relation.

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

Define the composition of transducer T and automaton \mathcal{A} in the same way: automaton $\mathcal{B} = T \circ \mathcal{A}$ recognizes the language $\{w \mid \exists y \in L(\mathcal{A}) (w, y) \in R\}$.

The following proposition is an algorithmic version of the Elgot-Mezei theorem (see, e.g., [2, Th. 4.4]).

Proposition 1. *The composition of transducers and the composition of a transducer and an automaton are computable in deterministic log space.*

A *rational cone* is a class of languages closed under rational dominance. Let $\mathcal{T}(L)$ denote the least rational cone that includes language L and call it the *rational cone generated by L* . Such a cone is called *principal*. For example, the cone **Lin** of linear languages (see [2] for 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

$$S \rightarrow x_1 S \bar{x}_1 \mid x_2 S \bar{x}_2 \mid \varepsilon.$$

For a mapping $a \mapsto L_a$ the *substitution* σ is the morphism from A^* to the power set 2^{B^*} such that $\sigma(a) = L_a$. The image $\sigma(L)$ of a language $L \subseteq A^*$ is defined in the natural way. The *substitution closure* of a class of languages \mathcal{L} is the least class containing all substitutions of languages from \mathcal{L} to the languages from \mathcal{L} . We need two well-known examples of the substitution closure. The class **Qrt** of the *quasirational languages* is the substitution closure of the class **Lin**. The class of *Greibach languages* [7] is the substitution closure of the rational cone generated by the Dyck language D_1 and the symmetric language S .

It is important for our purposes that rational dominance implies a reduction for the corresponding RR problems.

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

Proof. Let T be a rational transducer such that $F_1 = T(F_2)$ and let \mathcal{A} be an input of the $\text{NRR}(F_1)$ problem. Construct the automaton $\mathcal{B} = T \circ \mathcal{A}$ and use it as an input of the $\text{NRR}(F_2)$ problem. It gives the log-space reduction due to Proposition 1.

In particular, this lemma implies that if a problem $\text{NRR}(F)$ is complete in a complexity class \mathcal{C} , then for any filter F' from the rational cone $\mathcal{T}(F)$ the problem $\text{NRR}(F')$ is in the class \mathcal{C} .

We will use the following reformulation of the Chomsky-Schützenberger theorem.

Theorem (Chomsky, Schützenberger). $\text{CFL} = \mathcal{T}(D_2)$.

In the next section, we prove that $\text{NRR}(D_2)$ is **P**-complete under deterministic log-space reductions. Thus, it follows from the Chomsky-Schützenberger theorem and Lemma 1 that any problem $\text{NRR}(F)$ for a CFL filter F lies in the class **P**.

3 Hard RR problems with CFL filters

In this section we present examples of hard context-free languages. The first example is the Dyck language D_2 .

By use of Lemma 1 and the Chomsky-Schützenberger theorem, we conclude that any generator of the CFL cone is hard. But there are additional hard languages. We provide such an example, too.

We start with some technical lemmas. The intersection of a CFL and a rational language is a CFL. We need an algorithmic version of this fact.

Lemma 2. *Let $G = (N, \Sigma, P, S)$ be a fixed context-free grammar. Then there exists a deterministic log-space algorithm that takes 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 grammar size is polynomial in $|Q_{\mathcal{A}}|$.*

This fact is well-known. We provide the proof because the construction will be used in the proof of Theorem 5 below.

Proof (of Lemma 2). First, to make the construction clearer, we assume that automaton \mathcal{A} has no ε -transitions. Let N' consist of the axiom S' and nonterminals $[qAp]$, where $A \in N$ and $q, p \in Q_{\mathcal{A}}$. Construct P' by adding for each rule $A \rightarrow X_1X_2 \cdots X_n$ from P the set of rules

$$\{[qAp] \rightarrow [qX_1r_1][r_1X_2r_2] \cdots [r_{n-1}X_np] \mid q, p, r_1, r_2, \dots, r_{n-1} \in Q_{\mathcal{A}}\}$$

to P' . Also add to P' rules $[q\sigma p] \rightarrow \sigma$ if $\delta_{\mathcal{A}}(q, \sigma) = p$ and $S' \rightarrow [q_0Sq_f]$ for each q_f from $F_{\mathcal{A}}$.

Now we prove that $L(G') = L(G) \cap L(\mathcal{A})$. Let G derive the word $w = w_1w_2 \cdots w_n$. Then grammar G' derives all possible sentential forms

$$[q_0w_1r_1][r_1w_2r_2] \cdots [r_{n-1}w_nq_f],$$

where $q_f \in F_{\mathcal{A}}$ and $r_i \in Q_{\mathcal{A}}$. And $[q_0w_1r_1][r_1w_2r_2] \cdots [r_{n-1}w_nq_f] \Rightarrow^* w_1w_2 \cdots w_n$ iff there is a successful run for the automaton \mathcal{A} on w . If G' derives a word w then each symbol w_i of the word has been derived from some nonterminal $[qw_ip]$. Due to the construction of the grammar G' the word w has been derived from some sentential form $[q_0w_1r_1][r_1w_2r_2] \cdots [r_{n-1}w_nq_f]$, which encodes a successive run of \mathcal{A} on w . Thus G' derives the word w only if G does as well.

The size of G' is polynomial in $Q_{\mathcal{A}}$. The size of N' is $|N| \cdot |Q_{\mathcal{A}}|^2 + 1$. Let k be the length of the longest rule in P . Then for each rule from P there are at most $|Q_{\mathcal{A}}|^{k+1}$ rules in P' and for rules in the form $[q\sigma p] \rightarrow \sigma$ or $S' \rightarrow [q_0Sq_f]$ there are at most $O(|Q_{\mathcal{A}}|^2)$ rules in P' .

Finally, the grammar G' is log-space constructible, because the rules of P' corresponding to the particular rule from P can be generated by inspecting all $(k+1)$ -tuples of states of \mathcal{A} and $k = O(1)$. Adding ε -transitions just increases $k+1$ to $2k$. For each rule $A \rightarrow X_1 \cdots X_n$ we add rules $[qAp] \rightarrow [qX_1q_1][q_1X_2q_2] \cdots [q_{n-1}X_np]$, where $q_i = q_{i+1}$ or $q_i \xrightarrow{\varepsilon} q_{i+1}$ for all i . In the case of $[q\sigma p] \rightarrow \sigma$ rules we add all such rules that $q \xrightarrow{\varepsilon} q'$, $p' \xrightarrow{\varepsilon} p$ and $\delta(q', \sigma) = p'$.

Note that if grammar G is in Chomsky normal form, then the number of nonterminals of the grammar G' is $O(|Q_{\mathcal{A}}|^2)$. Recall that for a grammar in the Chomsky normal form, the right-hand side of each rule consists of either two nonterminals, or one terminal. The empty word may be produced only by the axiom and the axiom does not appear in a right-hand side of any rule.

Also we need an algorithmic version of the Chomsky-Schützenberger theorem.

Lemma 3. *There exists a deterministic log-space algorithm that takes a description of a context-free grammar $G = (N, \Sigma, P, S)$ and produces a rational transducer T such that $T(D_2) = L(G)$.*

Now we are ready to prove hardness of the Dyck language D_2 .

Theorem 1. *The problem $\text{NRR}(D_2)$ is \mathbf{P} -complete.*

Proof. To prove \mathbf{P} -hardness we reduce the well-known \mathbf{P} -complete problem of verifying whether a context-free grammar generates an empty language [6] to $\text{NRR}(D_2)$. Based on a grammar G , construct a transducer T such that $T(D_2) = L(G)$ using Lemma 3. Let \mathcal{A} be a nondeterministic automaton obtained from the transducer T by ignoring the output tape. Then $L(\mathcal{A}) \cap D_2$ is nonempty iff $L(G)$ is nonempty. The mapping $G \rightarrow \mathcal{A}$ is the required reduction.

To prove that $\text{NRR}(D_2)$ lies in \mathbf{P} we reduce this problem to the problem of non-emptiness of a language generated by a context-free grammar.

For an input \mathcal{A} construct the grammar G such that $L(G) = L(\mathcal{A}) \cap D_2$ using Lemma 2.

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

Now we present another example of a hard language. Boasson proved in [4] that there exists a principal rational cone of non-generators of the CFL cone containing the family **Qrt** of the quasirational languages.

Below we establish \mathbf{P} -completeness of the nondeterministic RR problem for a generator of this cone. The construction follows the exposition in [3].

For brevity we denote the alphabet of the Dyck language D_1 by $A = \{a, \bar{a}\}^*$. Recall that the syntactic substitution of a language M into a language L is

$$L \uparrow M = \{m_1 x_1 m_2 x_2 \cdots m_r x_r \mid m_1, \dots, m_r \in M, x_1 x_2 \cdots x_r \in L\} \cup (\{\varepsilon\} \cap L).$$

We also use the language $S_\# = S \uparrow \#^*$ which is the *syntactic substitution* of the language $\#^*$ in the symmetric language S .

Let $M = aS_\# \bar{a} \cup \varepsilon$. The language $M^{(\infty)}$ is defined recursively in the following way: $x \in M^{(\infty)}$ iff either $x \in M$ or

$$x = ay_1 az_1 \bar{a} y_2 az_2 \bar{a} \cdots y_{n-1} az_{n-1} \bar{a} y_n \bar{a},$$

where $y_1, y_n \in X^*$, $y_i \in X^+$ for $2 \leq i \leq n-1$, $az_i \bar{a} \in M^{(\infty)}$ and $ay_1 y_2 \cdots y_n \bar{a} \in M$.

Let $\pi_X: (X \cup A)^* \rightarrow A^*$ be the morphism that erases symbols from the alphabet X . The language $M^{(+)}$ is defined to be $\pi_X^{-1}(A^* \setminus D_1)$.

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

Note that the languages S and $S_\#$ are rationally equivalent. So $S_\#$ is a generator of the cone **Lin** of the linear languages.

By combining this observation with Propositions 3.19 and 3.20 from [3], we get the following fact.

Theorem 2. *$S_\#^\uparrow$ is not a generator of the CFL cone, but the cone generated by $S_\#^\uparrow$ contains all quasirational languages.*

The language $S_{\#}^{\uparrow}$ is the union of two languages. In the proof of the **P**-completeness for the problem $\text{NRR}(S_{\#}^{\uparrow})$, we will use automata that do not accept words from the language $M^{(+)}$. For this purpose we need a notion of a marked automaton.

Definition 2. An NFA \mathcal{A} over the alphabet $A_n \cup \bar{A}_n$ is *marked* if there exists a function $h: Q_{\mathcal{A}} \rightarrow \mathbb{Z}$ satisfying the relations

$$\begin{aligned} h(q') &= h(q) + 1, & \text{if there exists a transition } q \xrightarrow{a_j} q' \text{ in } \mathcal{A}, \\ h(q') &= h(q) - 1, & \text{if there exists a transition } q \xrightarrow{\bar{a}_j} q' \text{ in } \mathcal{A}, \\ h(q) &= 0, & \text{if } q \text{ is either the initial state or an accepting state of } \mathcal{A}. \end{aligned} \tag{2}$$

In what follows we will identify for brevity the (directed) paths along the graph of an NFA and the corresponding words in the alphabet of the automaton. The vertices of the graph, i.e., the states of the automaton, are identified in this way with the *positions* of the word.

The *height* of a position is the difference between the number of the symbols a_i and the number of the symbols \bar{a}_i preceding the position. In terms of the position heights, the words in D_1 are characterized by two conditions: the height of any position is nonnegative and the height of the final position is 0.

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

Proof. The heights of positions are upperbounded by the height of the derivation tree in the grammar generating the language $D_2 \cap L(\mathcal{A}) \neq \emptyset$.

It is easy to see that for any grammar generating a non-empty language there is a word such that the height of a derivation tree for the word is at most the number of nonterminals in the grammar.

To finish the proof, we use the grammar constructed by Lemma 2 from the grammar generating D_2 in the Chomsky normal form. This grammar has $O(|Q_{\mathcal{A}}|^2)$ nonterminals.

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

Proposition 3. *There exists a transformation μ that takes 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 (i) $L(\mathcal{A}) \cap D_2 \neq \emptyset$ iff $L(\mathcal{A}') \cap D_2 \neq \emptyset$ and (ii) for any $w \in L(\mathcal{A}')$ the height of any position is nonnegative and the height of the final position is 0. The transformation μ is computed in deterministic log space.*

Proof. Let m be an upper bound on the heights of the positions in a word $w \in L(\mathcal{A}) \cap D_2$. By Proposition 2, m is $O(|Q_{\mathcal{A}}|^2)$. Note that m can be computed in deterministic log space.

The state set of the automaton \mathcal{A}' is $Q_{\mathcal{A}} \times \{0, \dots, m\} \cup \{r\}$, where r is the specific absorbing rejecting state.

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

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

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

It is clear that the description of the automaton \mathcal{A}' is constructed in deterministic log space.

Condition (ii) is forced by the construction of the automaton \mathcal{A}' . It remains to prove that condition (i) holds.

Note that if $L(\mathcal{A}) \cap D_2 = \emptyset$ then $L(\mathcal{A}') \cap D_2 = \emptyset$ too. In the other direction, if $L(\mathcal{A}) \cap D_2 \neq \emptyset$, then by Proposition 2 there exists a word $w \in L(\mathcal{A}) \cap D_2$ such that the height of any position in the word does not exceed m . So the word is accepted by the automaton \mathcal{A}' .

Theorem 3. $\text{NRR}(S_{\#}^{\uparrow})$ is **P**-complete under deterministic log space reductions.

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

Let \mathcal{A} be an input of the problem $\text{NRR}(D_2)$ and $\mathcal{A}' = \mu(\mathcal{A})$ be the marking transformation of the automaton \mathcal{A} .

We are going to construct the automaton \mathcal{B} over the alphabet $A \cup X \cup \{\#\}$ such that $L(\mathcal{A}') \cap D_2 \neq \emptyset$ iff $L(\mathcal{B}) \cap S_{\#}^{\uparrow} \neq \emptyset$.

The morphism $\varphi: (A_2 \cup \bar{A}_2)^* \rightarrow (A \cup X \cup \{\#\})^*$ is defined as follows:

$$\begin{aligned} \varphi: a_1 &\mapsto ax_1, \\ \varphi: \bar{a}_1 &\mapsto \bar{x}_1 \bar{a} \# \# \#, \\ \varphi: a_2 &\mapsto ax_2, \\ \varphi: \bar{a}_2 &\mapsto \bar{x}_2 \bar{a} \# \# \#. \end{aligned} \tag{3}$$

The automaton \mathcal{B} accepts words of the form $ax_1x_2w\bar{x}_2\bar{x}_1\bar{a}$, where $w = \varphi(u)$. It simulates the behavior of the automaton \mathcal{A}' on the word u and accepts iff \mathcal{A}' accepts the word u .

It follows from the definitions that if $u \in D_2$ then $ax_1x_2\varphi(u)\bar{x}_2\bar{x}_1\bar{a} \in M^{(\infty)}$. So if $L(\mathcal{A}') \cap D_2 \neq \emptyset$ then $L(\mathcal{B}) \cap S_{\#}^{\uparrow} \neq \emptyset$.

Now we are going to prove the opposite implication. Let

$$w = ax_1x_2\varphi(u)\bar{x}_2\bar{x}_1\bar{a} \in S_{\#}^{\uparrow} \cap L(\mathcal{B}).$$

The automaton \mathcal{A}' is marked and \mathcal{B} simulates the behavior of \mathcal{A}' on u . So the heights of positions in w are nonnegative and the height of the final position is 0.

Thus $w \notin M^{(+)} = \pi_X^{-1}(A^* \setminus D_1)$. Take a pair of the corresponding parentheses a, \bar{a} in the word w :

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

If $i \neq j$ then $w \notin M^{(\infty)}$. So $i = j$ for all pairs of the corresponding parentheses. This implies $u \in D_2 \cap L(\mathcal{A}')$.

We just have proved the correctness of the reduction. It can be computed in log space due to the following observations. To produce the automaton \mathcal{B} from the automaton \mathcal{A} we need to extend the state set by a finite number of pre- and postprocessing states to operate with the prefix $a x_1 x_2$ and with the suffix $\bar{x}_2 \bar{x}_1 \bar{a}$. Also we need to split all states in $Q_{\mathcal{A}'}$ in pairs to organize the simulation of \mathcal{A}' while reading the pairs of symbols $a x_i$ and $\bar{x}_i \bar{a}$. The transitions by the symbol $\#$ are trivial: $q \xrightarrow{\#} q$ for all q .

4 Easy RR problems with CFL filters

Now we present examples of easy languages. The simplest example is rational languages. Next we prove that the symmetric language and the language D_1 are easy. A simple observation shows that a substitution of easy languages into an easy language is easy. Thus we conclude that Greibach languages are easy.

Lemma 4. $\text{NRR}(S) \in \text{NL}$.

The proof of Lemma 4 is a slight modification of the arguments from [1] that prove a similar result for the language of palindromes.

Lemma 5. *Let L_c be a context-free language recognizable by a counter automaton. Then problem $\text{NRR}(L_c)$ lies in NL .*

In the proof we will use the following fact.

Lemma 6 ([13]). *Let M be a counter automaton with n states. Then the shortest word w from the language $L(M)$ has length at most n^3 and the counter of M on processing the word w doesn't exceed the value n^2 .*

We now return to the proof of Lemma 5.

Proof. Let M be a counter automaton that accepts by reaching the final state such that M recognizes the language L_c . Let \mathcal{A} be an automaton on the input of the regular realizability problem.

Construct the counter automaton $M_{\mathcal{A}}$ with the set of states $Q_M \times Q_{\mathcal{A}}$, the initial state $(q_0^M, q_0^{\mathcal{A}})$, with the set of accepting states $F_M \times F_{\mathcal{A}}$ and with the transition relation $\delta_{M_{\mathcal{A}}}$ such that $\delta_M(q, \sigma, z) \vdash (q', z')$, $\delta_{\mathcal{A}}(p, \sigma) = p'$ implies $\delta_{M_{\mathcal{A}}}((q, p), \sigma, z) \vdash ((q', p'), z')$. This is the standard composition construction.

The automaton $M_{\mathcal{A}}$ is a counter automaton with $|Q_M| \cdot |Q_{\mathcal{A}}| = c \times n$ states. Using Lemma 6 we obtain that the value of $M_{\mathcal{A}}$'s counter does not exceed $(cn)^2$ on the shortest word from $L(M_{\mathcal{A}})$. Then construct automaton \mathcal{B} such that $L(\mathcal{B})$

contains all such words from $L(M_A)$ such that the counter of M_A does not exceed $(cn)^2$. The automaton \mathcal{B} has $O(n^3)$ states and can be constructed in log space in the straightforward way similar to the proof of Proposition 3. Note that $L(M_A) \neq \emptyset$ iff $L(\mathcal{B}) \neq \emptyset$. So the map $\mathcal{A} \rightarrow \mathcal{B}$ gives a reduction of the problem $\text{NRR}(L_c)$ to the problem $\text{NRR}(\Sigma^*)$, which is in **NL**.

The language D_1 is recognized by a counter automaton in the obvious way.

Corollary 2. $\text{NRR}(D_1) \in \text{NL}$.

Lemma 7. *If L, L_a for all $a \in A$, are easy languages then $\sigma(L)$ is also easy.*

Proof. Let \mathcal{A} be an input for the problem $\text{NRR}(\sigma(L))$. Define the automaton \mathcal{A}' over the alphabet A with the state set $Q_{\mathcal{A}'} = Q_{\mathcal{A}}$. There is a transition $q \xrightarrow{a} q'$ in the automaton \mathcal{A}' iff there exists a word $w \in L_a$ such that $q \xrightarrow{w} q'$ in automaton \mathcal{A} .

It is clear from the definition that $L(\mathcal{A}) \cap \sigma(L) \neq \emptyset$ iff $L(\mathcal{A}') \cap L \neq \emptyset$. To apply an **NL**-algorithm for $\text{NRR}(L)$ one needs the transition relation of \mathcal{A}' . The transition relation is not a part of the input now. But it can be computed by **NL**-algorithms for $\text{NRR}(L_a)$. It is clear that the resulting algorithm is in **NL**.

Applying Lemma 7, Lemma 4 and Corollary 2, we deduce with the theorem.

Theorem 4. *Greibach languages are easy.*

5 The case of polynomially-bounded rational index

We do not know whether there exists a CFL that is neither hard nor easy. In this section we indicate one possible class of candidates for an intermediate complexity: the languages with polynomially-bounded rational indices.

Rational index appears to be a very useful characteristic of a context-free language because rational index does not increase significantly under rational transductions.

Theorem (Boasson, Courcelle, Nivat, 1981, [5]). *If $L' \leq_{\text{rat}} L$ 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 w.r.t. the rational dominance relation. Note that the rational index of a generator of the **CFL** cone has rather good estimations.

Theorem (Pierre, 1992, [9]). *The rational index of any generator of the rational cone of **CFL** belongs to $\exp(\Theta(n^2/\log n))$.*

The examples of easy languages in Section 4 have polynomially-bounded rational indices. Moreover, context-free languages with rational index $\Theta(n^\gamma)$ for any positive algebraic number $\gamma > 1$ were presented in [10]. All of them are easy. The proof is rather technical and is skipped here. Thus it is quite natural to suggest that any language with polynomially-bounded rational index is easy.

Unfortunately we are able to give only a weaker bound on the algorithmic complexity in the case of polynomially-bounded rational index.

Theorem 5. *For a context-free filter F with polynomially-bounded rational index, the problem $\text{NRR}(F)$ lies in $\mathbf{NSPACE}(\log^2 n)$.*

We use a technique quite similar to the technique from [8]. First we need an auxiliary result.

Lemma ([8]). *For a grammar G in the Chomsky normal form and for an arbitrary string $w = xyz$ from $L(G)$ of length n there is a nonterminal A in the derivation tree, such that A derives y and $n/3 \leq |y| \leq 2n/3$.*

Let us return to the proof of the theorem.

Proof (of Theorem 5). Consider a grammar G' in the Chomsky normal form such that $L(G') = F$. Fix an automaton \mathcal{A} with n states such that the minimal length of w from $L(\mathcal{A}) \cap F$ equals $\rho_F(n)$. The length of the word w is polynomial in n . Consider the grammar G such that $L(G) = L(\mathcal{A}) \cap F$ obtained from the grammar G' by the construction from Lemma 2.

The algorithm does not construct the grammar G itself, since such a construction expands the size of grammar G' up to n^3 times. Instead, the algorithm nondeterministically guesses the derivation tree of the word w in the grammar G , if it exists. Informally speaking, it restores the derivation tree starting from its ‘central’ branch.

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

If a nonterminal is $[q\sigma p]$, where σ is a terminal then the procedure should check that $q \xrightarrow{\sigma} p$ in the automaton \mathcal{A} .

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

1. Nondeterministically guess the ancestor $A_{i+1} = [\ell_{i+1} A'_{i+1} r_{i+1}]$ of A_i in the derivation tree. There are two possible cases:
 - (i) either $A_{i+1} \rightarrow [q'C'\ell_{i+1}]A_i$ in the grammar G (set up $C := [q'C'\ell_{i+1}]$)
 - (ii) or $A_{i+1} \rightarrow A_i[r_{i+1}C'p']$ (set up $C := [r_{i+1}C'p']$).
2. Recursively apply the procedure of checking correctness to the nonterminal C .
3. If successful set up $i := i + 1$.

Repetitions are finished and the procedure returns success if $A_j = A$. If any call of the procedure of checking correctness returns failure then the whole procedure returns failure.

In recursive calls the lengths of words to be checked diminish by a factor at most $2/3$. So the total number of recursive calls is $O(\log n)$, where n is the input length. Data to be stored during the process form a list of triples (an automaton

state, a nonterminal of the grammar G' , a automaton state). Each automaton state description requires $O(\log n)$ space and nonterminal description requires a constant size space since grammar G' is fixed. Thus the total space for the algorithm is $O(\log^2 n)$.

Acknowledgments

We are acknowledged to Abuzer Yakaryilmaz for pointing on the result of Lemma 5 and for reference to a lemma similar to Lemma 6.

References

1. Anderson, T., Loftus, J., Rampersad, N., Santean, N., Shallit, J.: Detecting palindromes, patterns and borders in regular languages. *Information and Computation* 207, 1096–1118 (2009)
2. Berstel, J.: Transductions and context-free languages. Teubner Verlag, Stuttgart / Leipzig / Wiesbaden (1979)
3. Berstel, J., Boasson, L.: Context-Free Languages. In: Leeuwen, van J. (ed.) *Handbook of Theoretical Computer Science*, Vol. B, pp. 59–102. Elsevier, Amsterdam (1990)
4. Boasson, L.: Non-générateurs algébriques et substitution. *RAIRO Informatique théorique* 19, 125–136 (1985)
5. Boasson, L., Courcelle, B., Nivat, M.: The rational index, a complexity measure for languages. *SIAM J. Comput.* 10(2), 284–296 (1981)
6. Greenlaw, R., Hoover, H. J., Ruzzo, L.: *Limits to Parallel Computation: P-completeness Theory*. Oxford Univ. Press, Oxford (1995)
7. Greibach, Sh.A.: An infinite hierarchy of context-free languages. *J. of the ACM* 16, 91–106 (1969)
8. Lewis, P.M., Stearns, R.E., Hartmanis, J.: Memory bounds for recognition of context-free and context-sensitive languages. In: *Switching Circuit Theory and Logical Design*, pp. 191–202. IEEE, New York (1965)
9. Pierre L.: Rational indexes of generators of the cone of context-free languages. *Theoretical Computer Science* 95, 279–305 (1992)
10. Pierre, L., Farinone, J.M.: Rational index of Context-free languages with rational index in $\Theta(n^\gamma)$ for algebraic numbers γ . *Informatique théorique et applications* 24(3), 275–322 (1990)
11. Vyalyi M.N.: On regular realizability problems. *Problems of Information Transmission* 47(4), 342–352 (2011)
12. Vyalyi M.N.: Universality of regular realizability problems. In: Bulatov, A.A., Shur, A.M. (eds) *CSR 2013. LNCS*, vol. 7913, pp. 271–282 Springer, Heidelberg (2013)
13. Yakaryilmaz, A.: One-counter verifiers for decidable languages. In: Bulatov, A.A., Shur, A.M. (eds) *CSR 2013. LNCS*, vol. 7913, pp. 366–377 Springer, Heidelberg (2013)