# Simplest Non-Regular Deterministic Context-Free Language

## Petr Jančar<sup>1</sup> and Jiří Šíma<sup>2</sup>

<sup>1</sup>Dept of Computer Science, Faculty of Science, Palacký University Olomouc, Czechia petr.jancar@upol.cz

#### Abstract

We introduce a new notion of C-simple problems for a class C of decision problems (i.e. languages), w.r.t. a particular reduction. A problem is C-simple if it can be reduced to each problem in C. This can be viewed as a conceptual counterpart to C-hard problems to which all problems in C reduce. Our concrete example is the class of non-regular deterministic context-free languages (DCFL'), with a truth-table reduction by Mealy machines (which proves to be a preorder). The main technical result is a proof that the DCFL' language  $L_{\#} = \{0^n1^n \mid n \geq 1\}$  is DCFL'-simple, which can thus be viewed as the simplest problem in the class DCFL'.

This result has already provided an application, to the computational model of neural networks 1ANN at the first level of analog neuron hierarchy. This model was proven not to recognize  $L_{\#}$ , by using a specialized technical argument that can hardly be generalized to other languages in DCFL'. By the result that  $L_{\#}$  is DCFL'-simple, w.r.t. the reduction that can be implemented by 1ANN, we immediately obtain that 1ANN cannot accept any language in DCFL'.

It thus seems worthwhile to explore if looking for C-simple problems in other classes C under suitable reductions could provide effective tools for expanding the lower-bound results known for single problems to the whole classes of problems.

Keywords: deterministic context-free language, truth-table reduction, Mealy automaton, pushdown automaton

### 1 Introduction

We introduce a new notion of C-simple problems for a class C of decision problems (i.e. languages). A problem is C-simple if it can be reduced to each problem in C; if this problem is, moreover, in C, it can be viewed as a simplest problem in C. The C-simple problems are thus a conceptual counterpart to the common C-hard problems (like, e.g., NP-hard problems) to which conversely any problem in C reduces. These definitions (of C-simple and C-hard problems) are parametrized by a chosen reduction that does not have a higher computational complexity than the class C itself. Therefore, it may be said that if a C-hard problem has a (computationally) "easy" solution, then each problem in C has an "easy" solution. On the other hand, if we prove that a C-simple problem is not "easy", in particular that it cannot be solved by machines of a type M that can implement the respective reduction, then all problems in C are not "easy", that is, are not solvable by M; this extends a lower-bound result for one problem to the whole class of problems.

<sup>&</sup>lt;sup>2</sup>Institute of Computer Science of the Czech Academy of Sciences, Prague, Czechia sima@cs.cas.cz

In this paper, we consider  $\mathcal{C}$  to be the class of non-regular deterministic context-free languages, which we denote by DCFL'; we thus have DCFL' = DCFL \ REG (where REG denotes the class of regular languages). We use a truth-table reduction by Mealy machines (which is motivated below). Hence a DCFL'-simple problem is a language  $L_0 \subseteq \Sigma^*$  (over an alphabet  $\Sigma$ ) that can be reduced to each DCFL' language  $L \subseteq \Delta^*$  by a Mealy machine  $\mathcal{A}$  with an oracle L, denoted  $\mathcal{A}^L$ . More precisely, the finite-state transducer  $\mathcal{A}$  transforms a given input word  $w \in \Sigma^*$  to a prefix  $\mathcal{A}(w) \in \Delta^*$  of queries for the oracle L. In addition, each state q of  $\mathcal{A}^L$  is associated with a finite tuple  $\sigma_q = (s_{q1}, \ldots, s_{qr_q})$  of  $r_q$  query suffixes from  $\Delta^*$ , and with a truth table  $f_q : \{0,1\}^{r_q} \to \{0,1\}$ . After  $\mathcal{A}^L$  reads an input word w (translating it to  $\mathcal{A}(w)$ ), by which it enters a state q, for each  $i \in \{1,2,\ldots,r_q\}$  it queries whether or not the string  $\mathcal{A}(w) \cdot s_{qi}$  is in L (or, equivalently, whether or not  $\mathcal{A}(w)$  belongs to the quotient  $L/s_{qi} = \{v \in \Delta^* \mid v \cdot s_{qi} \in L\}$ ), and aggregates the answers by the truth table  $f_q$  for deciding if w is accepted.

This truth-table reduction by Mealy machines proves to be a preorder, denoted as  $\leq_{tt}^A$ . The main technical result of this paper is that the DCFL' language  $L_\# = \{0^n1^n \mid n \geq 1\}$  (over the binary alphabet  $\{0,1\}$ ) is DCFL'-simple, since  $L_\# \leq_{tt}^A L$  for each language L in DCFL'. The class DCFLS of DCFL'-simple languages comprises REG and is a strict subclass of DCFL; e.g., the DCFL' language  $L_R = \{wcw^R \mid w \in \{a,b\}^*\}$  over the alphabet  $\{a,b,c\}$  proves to be not DCFL'-simple. The closure properties of DCFLS are similar to that of DCFL as the class DCFLS is closed under complement and intersection with regular languages, while being not closed under concatenation, intersection, and union.

The above definition of DCFL'-simple problems has originally been motivated by the analysis of the computational power of neural network (NN) models which is known to depend on the (descriptive) complexity of their weight parameters [8, 11]. The so-called analog neuron hierarchy [9] of binary-state NNs with increasing number of  $\alpha$  extra analog-state neurons, denoted as  $\alpha$ ANN for  $\alpha \geq 0$ , has been introduced for studying NNs with realistic weights between integers (finite automata) and rational numbers (Turing machines). We use the notation  $\alpha$ ANN also for the class of languages accepted by  $\alpha$ ANNs, which can clearly be distinguished by the context. The separation  $1ANN \subseteq 2ANN$  has been witnessed by the DCFL' language  $L_{\#} \in 2$ ANN \ 1ANN. The proof of  $L_{\#} \notin 1$ ANN is rather technical (based on the Bolzano-Weierstrass theorem) which could hardly be generalized to other DCFL' languages, while it was conjectured that  $L \notin 1$ ANN for all DCFL' languages L, that is,  $DCFL' \subseteq (2ANN \setminus 1ANN)$  (implying  $1ANN \cap DCFL = 0ANN = REG$ ). An idea how to prove this conjecture is to show that  $L_{\#} \notin 1$ ANN is in some sense the simplest problem in the class DCFL', namely, to reduce  $L_{\#}$  to any DCFL' language L by using a reduction that can be carried out by 1ANNs, which are at least as powerful as finite automata. This would imply that L cannot be accepted by any 1ANN since it is at least as hard as  $L_{\#}$  that has been proven not to be recognized by 1ANNs.

The idea why  $L_{\#}$  should serve as the simplest language in the class DCFL' comes from the fact that any reduced context-free grammar G generating a non-regular language  $L \subseteq \Delta^*$  is self-embedding [3, Theorem 4.10]. This means that there is a so-called self-embedding nonterminal A admitting the derivation  $A \Rightarrow^* xAy$  for some non-empty strings  $x, y \in \Delta^+$ . Since G is reduced, there are strings  $v, w, z \in \Delta^*$  such that  $S \Rightarrow^* vAz$  and  $A \Rightarrow^* w$  where S is the start nonterminal in G, which implies  $S \Rightarrow^* vx^mwy^mz \in L$  for every  $m \geq 0$ . It is thus straightforward to suggest to reduce an input word  $0^m1^n \in \{0,1\}^*$  where  $m, n \geq 1$ , to the string  $vx^mwy^nz \in \Delta^*$  (while the inputs outside  $0^+1^+$  are mapped onto some fixed string outside L) since  $0^m1^n \in L_{\#}$  entails  $vx^mwy^nz \in L$ .

However, the suggested (one-one) reduction from  $L_\#$  to L is not consistent because  $vx^mwy^nz\in L$  does not necessarily imply  $0^m1^n\in L_\#$ . For example, consider the DCFL' language  $L_1=\{0^m1^n\mid 1\le m\le n\}$  over the binary alphabet  $\Delta=\{0,1\}$  for which there are no words  $v,x,w,y,z\in\Delta^*$  such that  $vx^mwy^nz\in L_1$  would ensure m=n. Nevertheless, we can pick two inputs  $0^m1^{n-1}$  and  $0^m1^n$  instead of one, that is, x=0,y=1, and  $v=w=z=\varepsilon$  ( $\varepsilon$  denoting the empty string), which satisfy  $0^m1^n\in L_\#$  iff m=n iff  $vx^mwy^{n-1}z\notin L_1$  and  $vx^mwy^nz\in L_1$ . It turns out that this can be generalized to any DCFL' language. Namely, we prove in this paper that for DCFL' language  $L\subseteq\Delta^*$  over any alphabet  $\Delta$ , there are non-empty words  $v,x,w,y,z\in\Delta^+$  and a language  $L'\in\{L,\overline{L}\}$ , where  $\overline{L}=\Delta^*\setminus L$  is the complement of L, such that  $0^m1^n\in L_\#$  iff  $vx^mwy^{n-1}z\notin L'$  and  $vx^mwy^nz\in L'$ .

Therefore, the simple many-one (in fact, one-one) reduction from  $L_{\#}$  with one query to the oracle L is replaced by a truth-table reduction, that is, by a special Turing reduction in which all its finitely many (in our case two) oracle queries are presented at the same time and there is a Boolean function (a truth table) which, when given the answers to the queries, produces the final answer of the reduction. This truth-table reduction from  $L_{\#}$  to L can be implemented by a deterministic finite-state transducer (a Mealy machine)  $\mathcal{A}$  with the oracle L: It transforms the input  $0^m1^n$  where  $m, n \geq 1$  (the inputs outside  $0^+1^+$  are rejected), to the output  $vx^mwy^{n-1} \in \Delta^+$  and carries out two queries to L that arise by concatenation of this output with two fixed suffixes z and yz; hence the queries are  $vx^mwy^{n-1}z \in L$  and  $vx^mwy^nz \in L$ . The truth table is defined so that the input  $0^m1^n$  is accepted by  $\mathcal{A}^L$  iff the two answers to these queries are distinct and at same time, the first answer is negative in the case L' = L, and positive in the case  $L' = \overline{L}$ , which is equivalent to  $0^m1^n \in L_{\#}$ .

It follows that the DCFL' language  $L_{\#}$  is DCFL'-simple under the truth-table reduction by Mealy machines. Since this reduction can be implemented by 1ANNs, we achieve the desired stronger separation DCFL'  $\subseteq$  (2ANN \ 1ANN) in the analog neuron hierarchy [10]. This result constitutes a non-trivial application of the proposed concept of DCFL'-simple problem. Moreover, if we could generalize the result to (nondeterministic) context-free languages (CFL), e.g. by proving that some DCFL' language is CFL'-simple (where CFL' = CFL \ REG), which would imply that  $L_{\#}$  is CFL'-simple by the transitivity of reduction, then we would achieve even stronger separation CFL'  $\subseteq$  (2ANN \ 1ANN). We note the interesting fact that  $L_{\#}$  cannot be CSL'-simple (under our reduction), since 1ANN accepts some context-sensitive languages outside CFL [9].

In general, if we show that some C-simple problem under a given reduction cannot be computed by a computational model  $\mathcal{M}$  that implements this reduction, then all problems in the class C are not solvable by  $\mathcal{M}$  either. The notion of C-simple problems can thus be useful for expanding known (e.g. technical) lower-bound results for individual problems to the whole classes of problems at once, as it was the case of the DCFL'-simple problem  $L_{\#} \notin 1$ ANN, expanding to DCFL'  $\cap 1$ ANN =  $\emptyset$ . It seems worthwhile to explore if looking for C-simple problems in other complexity classes C could provide effective tools for strengthening known lower bounds.

We remark that the hardest context-free language by Greibach [2] can be viewed as CFL-hard under a special type of our reduction  $\leq_{tt}^{A}$ . Related line of study concerns the types of reductions used in finite or pushdown automata with oracle. For example, nondeterministic finite automata with oracle complying with many-one restriction have been applied to establishing oracle hierarchies over the context-free languages [7]. For the same purpose, oracle pushdown automata have been used for many-one, truth-table, and Turing reducibil-

ities, respectively, inducing the underlying definitions also to oracle nondeterministic finite automata [13]. In addition, nondeterministic finite automata whose oracle queries are completed by the prefix of an input word that has been read so far and the remaining suffix, have been employed in defining a polynomial-size oracle hierarchy [1].

In the preliminary study [12], some considerations about the simplest DCFL' language have appeared, yet without formal definitions of DCFL'-simple problems, that included only sketches of incomplete proofs of weaker results based on the representation of DCFL by so-called deterministic monotonic restarting automata [5], which have initiated investigations of non-regularity degrees in DCFL [6].

In this paper we achieve a complete argument for  $L_{\#}$  to be a DCFL'-simple problem, within the framework of deterministic pushdown automata (DPDA) by using some ideas on regularity of pushdown processes from [4]. We now give an informal overview of the proof. Given a DPDA  $\mathcal{M}$  recognizing a non-regular language  $L \subseteq \Delta^*$ , it is easy to realize that some computations of  $\mathcal{M}$  (from the initial configuration) must be reaching configurations where the stack is arbitrarily large while it can be (almost) erased afterwards. Hence the existence of words  $v, x, w, y, z \in \Delta^+$  such that  $vx^m wy^m z \in L$  for all  $m \geq 0$  is obvious. However, we aim to guarantee that for all m, n the equality m=n holds if, and only if,  $vx^m wy^{n-1}z \notin L'$ and  $vx^m wy^n z \in L'$ , where L' is either the language L or its complement. This is not so straightforward but it is confirmed by our detailed analysis (in section 3). We study the computation of  $\mathcal{M}$  on an infinite word  $a_1 a_2 a_3 \cdots$  that visits infinitely many pairwise nonequivalent configurations. We use a natural congruence property of language equivalence on the set of configurations, and avoid some tedious technical details by a particular use of Ramsey's theorem. This allows us to extract the required tuple  $v, x, w, y, z \in \Delta^+$  from the mentioned infinite computation. We note that determinism of  $\mathcal{M}$  is essential in the presented proof; we leave open if it can be relaxed to show that  $L_{\#}$  is even CFL'-simple.

The rest of the paper is organized as follows. In section 2 we recall basic definitions and notation regarding DPDA and Mealy machines, introduce the novel concept of DCFL'-simple problems under truth-table reduction by Mealy machines and show some simple properties of the class DCFLS of DCFL'-simple problems. In section 3 we present the proof of the main technical result which shows that  $L_{\#}$  is DCFL'-simple. Finally, we summarize the results and list some open problems in section 4.

# 2 DCFL'-Simple Problem Under Truth-Table Mealy Reduction

In this section we define the truth-table reduction by Mealy machines, introduce the notion of DCFL'-simple problems, show their basic properties, and formulate the main technical result (theorem 1). But first we recall standard definitions of pushdown automata.

A pushdown automaton (PDA) is a tuple  $\mathcal{M} = (Q, \Sigma, \Gamma, R, q_0, X_0, F)$  where Q is a finite set of states including the start state  $q_0 \in Q$  and the set  $F \subseteq Q$  of accepting states, while the finite sets  $\Sigma \neq \emptyset$  and  $\Gamma \neq \emptyset$  represent the input and stack alphabets, respectively, with the initial stack symbol  $X_0 \in \Gamma$ . In addition, the set R contains finitely many transition rules  $pX \xrightarrow{a} q\gamma$  with the meaning that  $\mathcal{M}$  in state  $p \in Q$ , on the input  $a \in \Sigma_{\varepsilon} = \Sigma \cup \{\varepsilon\}$  (recall  $\varepsilon$  denotes the empty string), and with  $X \in \Gamma$  as the topmost stack symbol may read a, change the state to  $q \in Q$ , and pop X, replacing it by pushing  $\gamma \in \Gamma^*$ .

By a configuration of  $\mathcal{M}$  we mean  $p\alpha \in Q \times \Gamma^*$ , and we define relations  $\xrightarrow{a}$  for  $a \in \Sigma_{\varepsilon}$ 

on  $Q \times \Gamma^*$ : each rule  $pX \xrightarrow{a} q\gamma$  in R induces  $pX\alpha \xrightarrow{a} q\gamma\alpha$  for all  $\alpha \in \Gamma^*$ ; these relations are naturally extended to  $\xrightarrow{w}$  for  $w \in \Sigma^*$ . For a configuration  $p\alpha$  we define  $\mathcal{L}(p\alpha) = \{w \in \Sigma^* \mid p\alpha \xrightarrow{w} q\beta \text{ for some } q \in F \text{ and } \beta \in \Gamma^*\}$ , and  $\mathcal{L}(\mathcal{M}) = \mathcal{L}(q_0X_0)$  is the language accepted by  $\mathcal{M}$ . A PDA  $\mathcal{M}$  is deterministic (a DPDA) if there is at most one rule  $pX \xrightarrow{a} ...$  for each tuple  $p \in Q$ ,  $X \in \Gamma$ ,  $a \in \Sigma_{\varepsilon}$ ; moreover, if there is a rule  $pX \xrightarrow{\varepsilon} ...$ , then there is no rule  $pX \xrightarrow{a} ...$  for  $a \in \Sigma$ . We also use the standard assumption that all  $\varepsilon$ -steps are popping, that is, in each rule  $pX \xrightarrow{\varepsilon} q\gamma$  in R we have  $\gamma = \varepsilon$ .

The languages accepted by (deterministic) pushdown automata constitute the class of (deterministic) context-free languages; the classes are denoted by DCFL and CFL, respectively, whereas DCFL' = DCFL  $\setminus$  REG.

In the following theorem we formulate the main technical result: any language in DCFL' includes a certain "projection" of the language  $L_{\#} = \{0^n1^n \mid n \geq 1\}$ , which means that  $L_{\#}$  is in some sense the simplest language in the class DCFL'. The theorem, whose proof will be presented in section 3, thus provides an interesting property of DCFL'.

**Theorem 1.** Let  $L \subseteq \Delta^*$  be a non-regular deterministic context-free language over an alphabet  $\Delta$ . There exist non-empty words  $v, x, w, y, z \in \Delta^+$  and a language  $L' \in \{L, \overline{L}\}$  (where  $\overline{L} = \Delta^* \setminus L$  is the complement of L) such that for all  $m \ge 0$  and n > 0 we have

$$(vx^m wy^{n-1}z \notin L' \text{ and } vx^m wy^n z \in L') \text{ iff } m = n.$$
 (1)

In order to formalize the DCFL'-simple problems, we now define a Mealy machine  $\mathcal{A}$  with an oracle: it is a tuple  $\mathcal{A}=(Q,\Sigma,\Delta,\delta,\lambda,q_0,\{(\sigma_q,f_q)\mid q\in Q\})$  where Q is a finite set of states including the start state  $q_0\in Q$ , and the finite sets  $\Sigma\neq\emptyset$  and  $\Delta\neq\emptyset$  represent the input and output (oracle) alphabets, respectively. Moreover,  $\delta:Q\times\Sigma\to Q$  is a (partial) state-transition function which extends to input strings as  $\delta:Q\times\Sigma^*\to Q$  where  $\delta(q,\varepsilon)=q$  for every  $q\in Q$ , while  $\delta(q,wa)=\delta(\delta(q,w),a)$  for all  $q\in Q,w\in\Sigma^*,a\in\Sigma$ . Similarly,  $\lambda:Q\times\Sigma\to\Delta^*$  is an output function which extends to input strings as  $\lambda:Q\times\Sigma^*\to\Delta^*$  where  $\lambda(q,\varepsilon)=\varepsilon$  for all  $q\in Q$ , and  $\lambda(q,wa)=\lambda(q,w)\cdot\lambda(\delta(q,w),a)$  for all  $q\in Q,w\in\Sigma^*,a\in\Sigma$ . In addition, for each  $q\in Q$ , the tuple  $\sigma_q=(s_{q1},\ldots,s_{qr_q})$  of strings in  $\Delta^*$  contains  $r_q$  query suffixes, while  $f_q:\{0,1\}^{r_q}\to\{0,1\}$  is a truth table that aggregates the answers to the  $r_q$  oracle queries.

The above Mealy machine  $\mathcal{A}$  starts in the start state  $q_0$  and operates as a deterministic finite-state transducer that transforms an input word  $w \in \Sigma^*$  to the output string  $\mathcal{A}(w) = \lambda(q_0, w) \in \Delta^*$  written to a so-called oracle tape. The oracle tape is a semi-infinite, write-only tape which is empty at the beginning and its contents are only extended in the course of computation by appending the strings to the right. Namely, given a current state  $q \in Q$  and an input symbol  $a \in \Sigma$ , the machine  $\mathcal{A}$  moves to the next state  $\delta(q, a) \in Q$  and writes the string  $\lambda(q, a) \in \Delta^*$  to the oracle tape, if  $\delta(q, a)$  is defined; otherwise  $\mathcal{A}$  rejects the input. After reading the whole input word  $w \in \Sigma^*$ , the machine  $\mathcal{A}$  is in the state  $p = \delta(q_0, w) \in Q$ , while the oracle tape contains the output  $\mathcal{A}(w) = \lambda(q_0, w) \in \Delta^*$ .

Finally, the Mealy machine  $\mathcal{A}$ , equipped with an oracle  $L \subseteq \Delta^*$ , in this case denoted  $\mathcal{A}^L$ , queries the oracle whether  $\mathcal{A}(w)$  belongs to the (right) quotient  $L/s_{pi} = \{u \in \Delta^* \mid u \cdot s_{pi} \in L\}$ , for each suffix  $s_{pi}$  in  $\sigma_p$ , and the answers are aggregated by the truth table  $f_p$ . Thus, the oracle Mealy machine  $\mathcal{A}^L$  accepts the input word  $w \in \Sigma^*$  iff

$$f_p\left(\chi_{L/s_{p1}}(\mathcal{A}(w)), \chi_{L/s_{p2}}(\mathcal{A}(w)), \dots, \chi_{L/s_{pr_p}}(\mathcal{A}(w))\right) = 1$$

where  $p = \delta(q_0, w)$  and  $\chi_{L/s_{pi}} : \Delta^* \to \{0, 1\}$  is the characteristic function of  $L/s_{pi}$ , that is,  $\chi_{L/s_{pi}}(u) = 1$  if  $u \cdot s_{pi} \in L$ , and  $\chi_{L/s_{pi}}(u) = 0$  if  $u \cdot s_{pi} \notin L$ . The language accepted by the machine  $\mathcal{A}^L$  is defined as  $\mathcal{L}(\mathcal{A}^L) = \{w \in \Sigma^* \mid w \text{ is accepted by } \mathcal{A}^L\}$ .

We say that  $L_1 \subseteq \Sigma^*$  is truth-table reducible to  $L_2 \subseteq \Delta^*$  by a Mealy machine, which is denoted as  $L_1 \leq_{tt}^{\Lambda} L_2$ , if  $L_1 = \mathcal{L}(\mathcal{A}^{L_2})$  for some Mealy machine  $\mathcal{A}$  running with the oracle  $L_2$ . The following lemma shows that we can chain these reductions together since the relation  $\leq_{tt}^{\Lambda}$  is a preorder.

**Lemma 2.** The relation  $\leq_{tt}^{A}$  is reflexive and transitive.

**Proof:** The relation  $\leq_{tt}^{\Lambda}$  is reflexive since  $L = \mathcal{L}(\mathcal{A}^L) \subseteq \Sigma^*$  for the oracle Mealy machine  $\mathcal{A}^L = (\{q\}, \Sigma, \Sigma, \delta, \lambda, q, \{(\sigma_q, f_q)\})$  where  $\delta(q, a) = q$  and  $\lambda(q, a) = a$  for every  $a \in \Sigma$ ,  $\sigma_q = (\varepsilon)$ , and  $f_q$  is the identity.

Now we show that the relation  $\leq_{tt}^{A}$  is transitive. Let  $L_1 \leq_{tt}^{A} L_2$  and  $L_2 \leq_{tt}^{A} L_3$  which means  $L_1 = \mathcal{L}(\mathcal{A}_1^{L_2}) \subseteq \Sigma^*$  and  $L_2 = \mathcal{L}(\mathcal{A}_2^{L_3}) \subseteq \Delta^*$  for some oracle Mealy machines  $\mathcal{A}_1^{L_2} = (Q_1, \Sigma, \Delta, \delta_1, \lambda_1, q_0^1, \{(\pi_q, g_q) \mid q \in Q_1\})$  and  $\mathcal{A}_2^{L_3} = (Q_2, \Delta, \Theta, \delta_2, \lambda_2, q_0^2, \{(\varrho_q, h_q) \mid q \in Q_2\})$ , respectively. We will construct the oracle Mealy machine  $\mathcal{A}^{L_3} = (Q, \Sigma, \Theta, \delta, \lambda, q_0, \{(\sigma_q, f_q) \mid q \in Q\})$  such that  $L_1 = \mathcal{L}(\mathcal{A}^{L_3}) \subseteq \Sigma^*$  which implies the transitivity  $L_1 \leq^{\mathcal{A}} L_3$ . We define  $Q = Q_1 \times Q_2$  with  $q_0 = (q_0^1, q_0^2)$ ,  $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, \lambda_1(q_1, a)))$  and  $\lambda((q_1, q_2), a) = \lambda_2(q_2, \lambda_1(q_1, a))$  for every  $(q_1, q_2) \in Q$  and  $a \in \Sigma$ , which ensures  $\mathcal{A}(w) = \lambda(q_0, w) = \lambda_2(q_0^2, \lambda_1(q_0^1, w)) = \mathcal{A}_2(\mathcal{A}_1(w)) \in \Theta^*$  for every  $w \in \Sigma^*$ . For each state  $p = (p_1, p_2) \in Q$  in  $\mathcal{A}$ , we define the tuple of query suffixes from  $\Theta^*$ ,

$$\sigma_p = (\lambda_2(p_2, s_{p_1,i}) \cdot s_{p_2(i),j} \mid i = 1, \dots, r_{p_1}, j = 1, \dots, r_{p_2(i)})$$

where  $\pi_{p_1} = (s_{p_1,1}, s_{p_1,2} \dots, s_{p_1,r_{p_1}}) \in \Delta^{r_{p_1}}$  and  $\varrho_{p_2(i)} = (s_{p_2(i),1}, s_{p_2(i),2} \dots, s_{p_2(i),r_{p_2(i)}}) \in \Theta^{r_{p_2(i)}}$  are the query suffixes associated with  $p_1 \in Q_1$  and  $p_2(i) = \delta_2(p_2, s_{p_1,i}) \in Q_2$  for  $i \in \{1, \dots, r_{p_1}\}$ , respectively, and the truth table  $f_p = g_{p_1}(h_{p_2(1)}, \dots, h_{p_2(r_{p_1})})$  aggregates the answers to the corresponding oracle queries, which ensures  $L_1 = \mathcal{L}(\mathcal{A}^{L_3}) \subseteq \Sigma^*$ .

We say that a (decision) problem  $L_0 \subseteq \Sigma^*$  is DCFL'-simple if  $L_0 \leq_{tt}^A L$  for every non-regular deterministic context-free language  $L \subseteq \Delta^*$ . It follows from theorem 1 that the DCFL' language  $L_\#$  is an example of a DCFL'-simple problem. In addition, we denote by DCFLS the class of DCFL'-simple problems and formulate its basic properties.

**Corollary 3** (of theorem 1). The non-regular deterministic context-free language  $L_{\#} = \{0^n1^n \mid n \geq 1\}$  is DCFL'-simple.

**Proof:** Let  $L \subseteq \Delta^*$  be any DCFL' language. According to theorem 1, there are  $v, x, w, y, z \in \Delta^+$  and  $L' \in \{L, \overline{L}\}$  such that condition (1) holds for L'. We define the Mealy machine  $\mathcal{A}^L = (\{q_0, q_1, q_2\}, \{0, 1\}, \Delta, \delta, \lambda, q_0, \{(\sigma_q, f_q) \mid q \in Q\})$  with the oracle L, as  $\delta(q_0, 0) = \delta(q_1, 0) = q_1, \ \delta(q_1, 1) = \delta(q_2, 1) = q_2, \ \lambda(q_0, 0) = vx, \ \lambda(q_1, 0) = x, \ \lambda(q_1, 1) = w, \ \lambda(q_2, 1) = y, \ \sigma_{q_2} = (z, yz), \ f_{q_0} = f_{q_1} = 0, \ f_{q_2}(0, 0) = f_{q_2}(1, 1) = 0, \ \text{and} \ f_{q_2}(1, 0) = 1 - f_{q_2}(0, 1)$  where  $f_{q_2}(0, 1) = 1$  iff L' = L. It is easy to verify that  $L_\# = \mathcal{L}(\mathcal{A}^L)$ , which implies  $L_\# \leq_{tt}^A L$ .

<sup>&</sup>lt;sup>1</sup>Note that the described protocol works also for non-prefix-free languages since for any input prefix that has been read so far, the output value from the truth table determines whether the oracle Mealy machine is in an "accepting" state, deciding about this prefix analogously as a deterministic finite automaton. The truth-table reduction only requires that the given oracle answers do not influence further computation when subsequent input symbols are read.

#### Proposition 4.

- 1.  $REG \subseteq DCFLS$ .
- 2.  $DCFLS \subseteq DCFL$ , and  $L_R = \{wcw^R \mid w \in \{a,b\}^*\} \in DCFL \setminus DCFLS$ .
- 3. The class DCFLS is closed under complement and intersection with regular languages.

4. The class DCFLS is not closed under concatenation, intersection and union.

#### **Proof:** [Sketch.]

- 1. For any regular language L, consider a Mealy machine  $\mathcal{A}^{L_{\#}}$  with the DCFL'-simple oracle  $L_{\#}$ , that simulates a deterministic finite automaton recognizing L, while its constant truth tables produce 1 iff associated with the accept states. Hence,  $L \leq_{tt}^{\Lambda} L_{\#}$  which means L is DCFL'-simple according to lemma 2 and corollary 3 which also implies REG  $\neq$  DCFLS.
- 2. We first observe that DCFLS  $\subseteq$  DCFL. Let  $L \in$  DCFLS be any DCFL'-simple language which ensures  $L \leq_{tt}^{\Lambda} L_{\#}$  by an oracle Mealy machine  $\mathcal{A}^{L}_{\#}$ . The machine  $\mathcal{A}^{L}_{\#}$  can be simulated by a DPDA  $\mathcal{M}$  which extends a suitable DPDA  $\mathcal{M}_{\#}$  (e.g. with no  $\varepsilon$ -transitions) accepting  $L_{\#} = \mathcal{L}(\mathcal{M}_{\#})$ , so that the finite control of  $\mathcal{M}$  implements the finite-state transducer  $\mathcal{A}$  whose output is presented online as an input to  $\mathcal{M}_{\#}$ . Moreover, for each state q of  $\mathcal{A}$ , the finite control of  $\mathcal{M}$  evaluates the truth table  $f_q$  which aggregates the answers to the queries with  $r_q$  suffixes associated with q, by inspecting at most constant number of topmost stack symbols. Hence  $L = \mathcal{L}(\mathcal{M}) \in \text{DCFL}$ .

In order to show that DCFLS  $\neq$  DCFL, we prove that the DCFL  $L_R = \{wcw^R \mid w \in \{a,b\}^*\}$  over the alphabet  $\{a,b,c\}^*$  is not DCFL'-simple. For the sake of contradiction, suppose that  $L_R \leq_{tt}^A L_\#$  by a Mealy machine  $\mathcal{A}^{L\#} = (Q,\{a,b,c\}^*,\{0,1\}^*,\delta,\lambda,q_0,\{(\sigma_q,f_q)\mid q\in Q\})$  with the oracle  $L_\# = \{0^n1^n\mid n\geq 1\}$ , which means  $L_R = \mathcal{L}(\mathcal{A}^{L\#})$ . Consider all the  $2^k$  possible prefixes  $w\in\{a,b\}^k$  of inputs presented to  $\mathcal{A}^L$  that have the length |w|=k. These strings can bring  $\mathcal{A}^L$  into a finite number  $|\{\delta(q_0,w)\mid w\in\{a,b\}^k\}|\leq |Q|$  of distinct states while the length  $|\lambda(q_0,w)|$  of outputs written to the oracle tape is bounded by O(k). For  $\lambda(q_0,w)$  outside  $0^*1^*$ , the acceptance of words wu where  $u\in\{a,b,c\}^*$ , depends only on the truth values  $f_q(0,\ldots,0)$  associated with the states q from the finite set Q, due to  $\lambda(q_0,wu)\notin L_\#/s$  for any  $s\in\{0,1\}^*$ . On the other hand, the number of distinct outputs  $\lambda(q_0,w)$  in  $0^*1^*$  is bounded by O(k). This means that for a sufficiently large  $k\geq 1$ , there must be two distinct prefixes  $w_1,w_2\in\{a,b\}^k$  such that  $\delta(q_0,w_1)=\delta(q_0,w_2)$  and  $\lambda(q_0,w_1)=\lambda(q_0,w_2)$  in  $0^*1^*$ , which results in the contradiction  $w_1cw_2^R\in\mathcal{L}(\mathcal{A}^{L\#})\smallsetminus L_R$ .

- **3.** The class DCFLS is closed under complement since the truth tables can be negated. Furthermore, any oracle Mealy machine be can modified so that it simulates another given finite automaton in parallel and is forced to reject if this automaton rejects, which shows DCFLS to be closed under intersection with regular languages.
- **4.** Observe that  $(L_{\#})^2$  is not DCFL'-simple under truth-table reduction. In addition,  $L_1 = \{0^m 1^m 0^n \mid m, n \geq 1\}$  and  $L_2 = \{0^m 1^n 0^n \mid m, n \geq 1\}$  are DCFL'-simple while  $L_1 \cap L_2$  is not context-free. The proof for union follows from 3 and De Morgan's law.

## 3 Proof of the Main Result (Theorem 1)

Theorem 1 follows from the (more specific) next lemma that we prove in this section. By  $\mathbb{N}$  we denote the set  $\{0, 1, 2, \dots\}$ , and by [i, j] the set  $\{i, i+1, \dots, j\}$  (for  $i, j \in \mathbb{N}$ ).

**Lemma 5.** Let  $\mathcal{M} = (Q, \Sigma, \Gamma, R, p_0, X_0, F)$  be a DPDA where  $L = \mathcal{L}(p_0X_0)$  is non-regular (hence L belongs to DCFL'). There are  $v \in \Sigma^*$ ,  $x, w, y, z \in \Sigma^+$ ,  $p, q \in Q$ ,  $X \in \Gamma$ ,  $\gamma \in \Gamma^+$ ,  $\delta \in \Gamma^*$  such that the following four conditions hold:

1.  $p_0X_0 \xrightarrow{v} pX\delta$  and  $pX \xrightarrow{x} pX\gamma$ , which entails the infinite (stack increasing) computation

$$p_0 X_0 \xrightarrow{v} pX\delta \xrightarrow{x} pX\gamma\delta \xrightarrow{x} pX\gamma\gamma\delta \xrightarrow{x} pX\gamma\gamma\gamma\delta \xrightarrow{x} pX\gamma\gamma\gamma\delta \xrightarrow{x} \cdots;$$
 (2)

- 2.  $pX \xrightarrow{w} q$ ;
- 3.  $q\gamma \xrightarrow{y} q$ ,

hence  $q\gamma^{\ell}\delta' \xrightarrow{y^{\ell}} q\delta'$  for all  $\ell \in \mathbb{N}$  and  $\delta' \in \Gamma^*$ ;

- 4. one of the following cases is valid (depending on whether  $z \in \mathcal{L}(q\delta)$  or  $z \notin \mathcal{L}(q\delta)$ ):
  - (a)  $\mathcal{L}(q\gamma^k\delta) \ni y^\ell z$  iff  $k = \ell$  (for all  $k, \ell \in \mathbb{N}$ ), or  $\mathcal{L}(q\gamma^k\delta) \ni y^\ell z$  iff  $k \leq \ell$  (for all  $k, \ell \in \mathbb{N}$ );
  - (b)  $\mathcal{L}(q\gamma^k\delta) \ni y^\ell z$  iff  $k \neq \ell$  (for all  $k, \ell \in \mathbb{N}$ ), or  $\mathcal{L}(q\gamma^k\delta) \ni y^\ell z$  iff  $k > \ell$  (for all  $k, \ell \in \mathbb{N}$ ).

We note that  $p_0X_0 \xrightarrow{v} pX\delta \xrightarrow{x^m} pX\gamma^m\delta \xrightarrow{w} q\gamma^m\delta \xrightarrow{y^m} q\delta$  (for each  $m \in \mathbb{N}$ ); hence  $vx^mwy^mz \in L$  iff  $z \in \mathcal{L}(q\delta)$  (since z is nonempty). Theorem 1 indeed follows from the lemma: there is  $L' \in \{L, \overline{L}\}$  such that either  $vx^mwy^nz \in L'$  iff m = n (for all  $m, n \in \mathbb{N}$ ), or  $vx^mwy^nz \in L'$  iff  $m \leq n$  (for all  $m, n \in \mathbb{N}$ ). (In theorem 1 we also stated that v is nonempty. If  $v = \varepsilon$  here, then we simply take vx and yz as the new v, z, respectively.)

**Proof of Lemma 5** In the rest of this section we provide a proof of lemma 5, assuming a fixed DPDA  $\mathcal{M} = (Q, \Sigma, \Gamma, R, p_0, X_0, F)$  where  $L = \mathcal{L}(p_0 X_0)$  is non-regular. The proof structure is visible from the auxiliary claims that we state and prove on the way.

Convention. W.l.o.g. we assume that  $\mathcal{M}$  always reads the whole input  $w \in \Sigma^*$  from  $p_0X_0$ . This can be accomplished in the standard way, by adding a special bottom-of-stack symbol  $\bot$  and a (non-accepting) fail-state. (Each empty-stack configuration  $q\varepsilon$  becomes  $q\bot$ , and each originally stuck computation enters the fail-state where it loops. We also recall that all  $\varepsilon$ -steps are popping, and thus infinite  $\varepsilon$ -sequences are impossible.) Hence for any infinite word  $a_1a_2a_3\cdots$  in  $\Sigma^\omega$  there is the unique infinite computation of  $\mathcal{M}$  starting in  $p_0X_0$ ; it stepwise reads the whole infinite word  $a_1a_2a_3\cdots$ .

The left quotient of L by  $u \in \Sigma^*$  is the set  $u \setminus L = \{v \in \Sigma^* \mid uv \in L\}$ ; concatenation has priority over  $\setminus$ , hence  $u_1u_2 \setminus L = (u_1u_2) \setminus L$ . (The next claim is valid for any non-regular L.)

Claim 6. We can fix an infinite word  $a_1a_2a_3\cdots$  in  $\Sigma^{\omega}$   $(a_i \in \Sigma)$  such that  $a_1a_2\cdots a_i \setminus L \neq a_1a_2\cdots a_j \setminus L$  for all  $i \neq j$ .

**Proof:** Let us consider the labelled transition system  $\mathcal{T} = (LQ(L), \Sigma, (\stackrel{a}{\rightarrow})_{a \in \Sigma})$  where  $LQ(L) = \{u \setminus L \mid u \in \Sigma^*\}$  and  $\stackrel{a}{\rightarrow} = \{(L', a \setminus L') \mid L' \in LQ(L)\}$ . (We recall that  $L' = u \setminus L$  entails  $a \setminus L' = u a \setminus L$ .) Since L is non-regular, the set of states reachable from  $L = \varepsilon \setminus L$  in  $\mathcal{T}$  is infinite. The out-degree of states in  $\mathcal{T}$  is finite (in fact, bounded by  $|\Sigma|$ ), hence an application of König's lemma yields an infinite acyclic path  $L \xrightarrow{a_1} L_1 \xrightarrow{a_2} L_2 \xrightarrow{a_3} \cdots$ .

We call a configuration  $p\alpha$  of  $\mathcal{M}$  unstable if  $\alpha = Y\beta$  and R contains a rule  $pY \xrightarrow{\varepsilon} q$  (we recall that  $\varepsilon$ -steps are only popping); otherwise  $p\alpha$  is stable. Since  $\mathcal{M}$  is a deterministic PDA, for each unstable  $p\alpha$  we can soundly define the stable successor of  $p\alpha$  as the unique stable configuration  $p'\alpha'$  where  $p\alpha \xrightarrow{\varepsilon} p'\alpha'$  ( $\alpha'$  being a suffix of  $\alpha$ ). The path  $p\alpha \xrightarrow{\varepsilon} p'\alpha'$  might (not) go via an accepting state (in F), hence  $\mathcal{L}(p\alpha) = \mathcal{L}(p'\alpha')$  or  $\mathcal{L}(p\alpha) = \{\varepsilon\} \cup \mathcal{L}(p'\alpha')$ . (We note that the configurations in the computation (2) that start with pX are necessarily stable.)

Claim 7. Each configuration is visited at most twice by

the computation of 
$$\mathcal{M}$$
 from  $p_0 X_0$  on  $a_1 a_2 a_3 \cdots$  that is fixed by claim 6. (3)

**Proof:** The computation (3) is infinite, stepwise reading the whole word  $a_1a_2a_3\cdots$ , and it can be presented as

$$r_0\gamma_0 \xrightarrow{a_1} r_1\gamma_1 \xrightarrow{a_2} r_2\gamma_2 \xrightarrow{a_3} \cdots \text{ (for } r_0\gamma_0 = p_0X_0)$$

where each  $r_i\gamma_i$  is stable; each segment  $r_i\gamma_i \xrightarrow{a_{i+1}} r_{i+1}\gamma_{i+1}$  starts with a (visible)  $a_{i+1}$ -step that is followed by a (maybe empty) sequence of (popping)  $\varepsilon$ -steps via unstable configurations. Since such an  $\varepsilon$ -sequence might go through an accepting state, we can have  $r_i\gamma_i = r_j\gamma_j$  for  $i \neq j$  though  $a_1a_2\cdots a_i\backslash L \neq a_1a_2\cdots a_j\backslash L$ ; in this case L contains precisely one of the words  $a_1a_2\cdots a_i$  and  $a_1a_2\cdots a_j$ , and the languages  $a_1a_2\cdots a_i\backslash L$  and  $a_1a_2\cdots a_j\backslash L$  differ just on  $\varepsilon$ . Nevertheless, this reasoning entails that we cannot have  $r_i\gamma_i = r_j\gamma_j = r_\ell\gamma_\ell$  for pairwise different  $i,j,\ell$ .

Since each segment  $r_i \gamma_i \xrightarrow{a_{i+1}} r_{i+1} \gamma_{i+1}$  visits any unstable configuration at most once and  $r_{i+1} \gamma_{i+1}$  is the stable successor for all unstable configurations in the segment, we deduce that also each unstable configuration can be visited at most twice in the computation (3).

Claim 8. The computation (3) on  $a_1a_2a_3\cdots$  can be "stair-factorized", that is, written

$$p_0 X_0 \xrightarrow{v_0} p_1 X_1 \alpha_1 \xrightarrow{v_1} p_2 X_2 \alpha_2 \alpha_1 \xrightarrow{v_2} p_3 X_3 \alpha_3 \alpha_2 \alpha_1 \xrightarrow{v_3} \cdots$$
 (4)

so that for each  $i \in \mathbb{N}$  we have  $v_i \in \Sigma^+$  and  $p_i X_i \xrightarrow{v_i} p_{i+1} X_{i+1} \alpha_{i+1}$  where  $\alpha_{i+1}$  is a nonempty suffix of the right-hand side of a rule in R (i.e., a nonempty suffix of  $\gamma$  in a rule  $pX \xrightarrow{a} q\gamma$ ).

**Proof:** We consider the computation (3), and call a stable configuration  $pX\beta$  a level, with position  $i \in \mathbb{N}$ , if  $p_0X_0 \xrightarrow{a_1 \cdots a_i} pX\beta$  and all configurations visited by the computation  $pX\beta \xrightarrow{a_{i+1}a_{i+2}\cdots}$  after  $pX\beta$  have the stack longer than  $|X\beta|$ ; we note that each level  $pX\beta$  has a unique position  $POS(pX\beta)$ . Since each configuration is visited at most twice in (3), the set of levels is infinite, with elements  $p'_0X'_0$ ,  $p_1X_1\beta_1$ ,  $p_2X_2\beta_2$ , ... where  $0 \leq POS(p'_0X'_0) < POS(p_1X_1\beta_1) < POS(p_2X_2\beta_2) < \cdots$ . The computation (3) can thus be presented as

$$p_0X_0 \xrightarrow{v_0'} p_0'X_0' \xrightarrow{v_0''} p_1X_1\beta_1 \xrightarrow{v_1} p_2X_2\beta_2 \xrightarrow{v_2} p_3X_3\beta_3 \xrightarrow{v_3} \cdots$$

where  $|v_0'| = POS(p_0'X_0')$ , and  $|v_0v_1 \cdots v_{j-1}| = POS(p_jX_j\beta_j)$  for  $j \ge 1$ , putting  $v_0 = v_0'v_0''$ .

Each segment  $pX\beta \xrightarrow{v} p'X'\beta'$  between two neighbouring levels can be obviously written as  $pX\beta \xrightarrow{a} q\gamma_1\gamma_2\beta \xrightarrow{v'} p'X'\gamma_2\beta$  where  $pX \xrightarrow{a} q\gamma_1\gamma_2$  is a rule in R, both  $\gamma_1$  and  $\gamma_2$  are nonempty, v = av', and  $q\gamma_1 \xrightarrow{v'} p'X'$ . Hence the validity of the claim is clear.

We define the natural equivalence relation  $\sim$  on the set of configurations of  $\mathcal{M}$ : we put  $p\alpha \sim q\beta$  if  $\mathcal{L}(p\alpha) = \mathcal{L}(q\beta)$ .

We fix the presentation (4), calling  $p_i X_i \alpha_i \alpha_{i-1} \cdots \alpha_1$  the level-configurations (for all  $i \in \mathbb{N}$ ). Since we have  $\mathcal{L}(p_i X_i \alpha_i \alpha_{i-1} \cdots \alpha_1) \setminus \{\varepsilon\} = (v_0 v_1 \cdots v_{i-1} \setminus L) \setminus \{\varepsilon\}$ , there cannot be three level-configurations in the same  $\sim$ -class (i.e., in the same equivalence class w.r.t.  $\sim$ ). Hence any infinite set of level-configurations represents infinitely many  $\sim$ -classes. Now we show a congruence-property that might enable to shorten a level-configuration while keeping its  $\sim$ -class. We use the notation  $\mathrm{DS}(p\alpha)$  (the "down-states" of  $p\alpha$ ), putting

$$DS(p\alpha) = \{ q \mid p\alpha \xrightarrow{w} q \text{ for some } w \in \Sigma^* \}.$$

Claim 9. If  $q\gamma \sim q\gamma'$  for each  $q \in DS(p\beta)$ , then  $p\beta\gamma \sim p\beta\gamma'$ .

**Proof:** Let us consider  $w \in \Sigma^*$ . If  $w \in \mathcal{L}(p\beta)$ , then  $w \in \mathcal{L}(p\beta\mu)$  for all  $\mu \in \Gamma^*$ . If  $w \notin \mathcal{L}(p\beta)$  and there is no prefix v of w such that  $p\beta \stackrel{v}{\to} q$ , then  $w \notin \mathcal{L}(p\beta\mu)$  for all  $\mu \in \Gamma^*$ . If  $w \notin \mathcal{L}(p\beta)$  and w = vv' where  $pX\beta \stackrel{v}{\to} q$  (necessarily for some  $q \in DS(pX\beta)$ ), then  $w \in \mathcal{L}(p\beta\mu)$  iff  $v' \in \mathcal{L}(q\mu)$ . Hence the claim is clear.

The next claim is an immediate corollary.

Claim 10. Any computation  $p_0X_0 \xrightarrow{w_1} pX\beta_1 \xrightarrow{w_2} pX\beta_2\beta_1 \xrightarrow{w_3} p'X'\beta_3\beta_2\beta_1$  where  $pX \xrightarrow{w_2} pX\beta_2$  ( $w_2 \in \Sigma^+$ ),  $pX \xrightarrow{w_3} p'X'\beta_3$ , and  $q\beta_2\beta_1 \sim q\beta_1$  for each  $q \in DS(p'X'\beta_3)$  can be shortened to  $p_0X_0 \xrightarrow{w_1} pX\beta_1 \xrightarrow{w_3} p'X'\beta_3\beta_1$  where  $p'X'\beta_3\beta_1 \sim p'X'\beta_3\beta_2\beta_1$ .

The *i*-th level-configuration in (4) is reached by the computation  $p_0X_0 \xrightarrow{v_0v_1\cdots v_{i-1}} p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1$ . It can happen that there are  $j_1,j_2,\ 0\leq j_1< j_2\leq i$  such that  $p_{j_1}X_{j_1}=p_{j_2}X_{j_2}$  and  $q\alpha_{j_2}\alpha_{j_2-1}\cdots\alpha_1\sim q\alpha_{j_1}\alpha_{j_1-1}\cdots\alpha_1$  for all  $q\in \mathrm{DS}(p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_{j_2+1})$ . In this case we can shorten the computation as in claim 10, where  $v_{j_1}v_{j_1+1}\cdots v_{j_2-1}$  corresponds to the omitted  $w_2$ . The resulting shorter computation might be possible to be repeatedly shortened further (if it can be presented so that the conditions of claim 10 are satisfied). Now for each  $i\geq 1$  we fix a (stair-factorized) computation

$$p_{i,0}X_{i,0} \xrightarrow{v_{i,0}} p_{i,1}X_{i,1}\alpha_{i,1} \xrightarrow{v_{i,1}} p_{i,2}X_{i,2}\alpha_{i,2}\alpha_{i,1} \cdots \xrightarrow{v_{i,n_i-1}} p_{i,n_i}X_{i,n_i}\alpha_{i,n_i}\alpha_{i,n_i-1} \cdots \alpha_{i,1}$$
 (5)

that has arisen by a maximal sequence of the above shortenings of the prefix

$$p_0 X_0 \xrightarrow{v_0 v_1 \cdots v_{i-1}} p_i X_i \alpha_i \alpha_{i-1} \cdots \alpha_1 \text{ of } (4).$$

Hence  $p_{i,0}X_{i,0} = p_0X_0$ ,  $p_{i,n_i}X_{i,n_i} = p_iX_i$ ,  $\alpha_{i,n_i}, \alpha_{i,n_i-1}, \ldots, \alpha_{i,1}$  is a subsequence of  $\alpha_i, \alpha_{i-1}, \ldots, \alpha_1$ , and  $p_{i,n_i}X_{i,n_i}\alpha_{i,n_i}\alpha_{i,n_i-1}\cdots\alpha_{i,1} \sim p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1$ .

**Claim 11.** For each  $\ell \in \mathbb{N}$  there is i such that  $n_i > \ell$  (where  $n_i$  is from (5)).

**Proof:** As already discussed, the set of level-configurations represents infinitely many  $\sim$ -classes. The last configurations of computations (5) represent the same infinite set of  $\sim$ -classes, and their lengths thus cannot be bounded; since the lengths of all  $\alpha_{i,j}$  are bounded (they are shorter than the longest right-hand sides of the rules in R), the claim is clear.  $\square$ 

Now we come to a crucial claim in our proof of lemma 5. Besides the notation  $DS(p\alpha)$  we also introduce  $ES(p\alpha)$  (the by- $\varepsilon$ -reached down-states of  $p\alpha$ ), by putting

$$\mathrm{ES}(p\alpha) = \{ q \mid p\alpha \xrightarrow{\varepsilon} q \}.$$

Hence  $\mathrm{ES}(p\alpha) \subseteq \mathrm{DS}(p\alpha)$ , and  $|\mathrm{ES}(p\alpha)| \leq 1$  (due to the determinism of the DPDA  $\mathcal{M}$ ).

We recall that  $p\alpha \sim q\beta$  means  $\mathcal{L}(p\alpha) = \mathcal{L}(q\beta)$ . To handle the special case of the empty word  $\varepsilon$ , we also define a (much) coarser equivalence  $\sim_0$ : we put  $p\alpha \sim_0 q\beta$  if  $\varepsilon$  either belongs to both  $\mathcal{L}(p\alpha)$  and  $\mathcal{L}(q\beta)$ , or belongs to none of them.

**Claim 12.** There is a constant  $B \in \mathbb{N}$  determined by the DPDA  $\mathcal{M}$  such that for all  $i \in \mathbb{N}$  where  $n_i > B$  the final configuration in (5) can be written as

$$p_{i,n_i}X_{i,n_i}\alpha_{i,n_i}\alpha_{i,n_i-1}\cdots\alpha_{i,1}=\bar{p}\bar{X}\beta\gamma\delta$$

where the following conditions hold:

- 1.  $\gamma = \alpha_{i,j}\alpha_{i,j-1}\cdots\alpha_{i,j'+1}$  where  $n_i \geq j > j' \geq n_i B$  and  $p_{i,j}X_{i,j} = p_{i,j'}X_{i,j'}$  (and  $\beta = \alpha_{i,n_i}\alpha_{i,n_{i-1}}\cdots\alpha_{i,j+1}$ ,  $\delta = \alpha_{i,j'}\alpha_{i,j'-1}\cdots\alpha_{i,1}$ );
- 2. the sets  $DS(\bar{p}\bar{X}\beta)$  and  $DS(\bar{p}\bar{X}\beta\gamma)$  are equal, further being denoted by  $\bar{Q}$ ;
- 3. for each  $q \in \bar{Q}$ , if  $\mathrm{ES}(q\gamma) = \{q'\}$ , then  $\mathrm{ES}(q'\gamma) = \{q'\}$  (and  $q' \in \bar{Q}$ );
- 4. each  $q' \in \bar{Q}$  belongs to  $DS(q\gamma)$  for some self-containing  $q \in \bar{Q}$ , where  $q \in \bar{Q}$  is self-containing if  $q \in DS(q\gamma)$ ;
- 5. there is a state  $q' \in \bar{Q}$  for which  $q'\gamma\delta \not\sim q'\delta$  and  $q'\gamma\delta \sim_0 q'\delta$ .

**Proof:** We fix some i with  $n_i$  larger than a constant B determined by  $\mathcal{M}$  as described below (there are such i by claim 11). For convenience we put  $p_{i,n_i}X_{i,n_i}=\bar{p}\bar{X}$ ,  $n_i=n$ , and  $\alpha_{i,j}=\bar{\alpha}_j$ , hence the final configuration in (5) is  $p_{i,n_i}X_{i,n_i}\alpha_{i,n_i}\alpha_{i,n_i-1}\cdots\alpha_{i,1}=\bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}\cdots\bar{\alpha}_1$ . We view the n+1 prefixes

$$\bar{p}\bar{X}, \ \bar{p}\bar{X}\bar{\alpha}_n, \ \bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}, \ \bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}\bar{\alpha}_{n-2}, \dots, \ \bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}\cdots\bar{\alpha}_1$$

as the vertices of a complete graph with coloured edges.

For  $\bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}\cdots\bar{\alpha}_1=\bar{p}\bar{X}\mu\nu\rho$ , where  $\mu=\bar{\alpha}_n\bar{\alpha}_{n-1}\cdots\bar{\alpha}_{j+1},\ \nu=\bar{\alpha}_j\bar{\alpha}_{j-1}\cdots\bar{\alpha}_{j'+1}$ , and  $\rho=\bar{\alpha}_{j'}\bar{\alpha}_{j'-1}\cdots\bar{\alpha}_1,\ n\geq j>j'\geq 0$ , the edge between the vertices  $\bar{p}\bar{X}\mu$  and  $\bar{p}\bar{X}\mu\nu$  has the following tuple as its colour:

$$\left(p_{i,j}X_{i,j},\ p_{i,j'}X_{i,j'},\ \mathrm{DS}(\bar{p}\bar{X}\mu),\ \mathrm{DS}(\bar{p}\bar{X}\mu\nu),\ (\mathrm{DS}(q\nu),\mathrm{ES}(q\nu))_{q\in\mathrm{DS}(\bar{p}\bar{X}\mu)},\ \mathrm{Q}_{\not\sim},\ \mathrm{Q}_{0}\right)$$

where  $Q_{\not\sim} = \{q' \in DS(\bar{p}\bar{X}\mu) \mid q'\nu\rho \not\sim q'\rho\}$  and  $Q_0 = \{q' \in Q_{\not\sim} \mid q'\nu\rho \sim_0 q'\rho\}$  (and  $p_{i,j}X_{i,j}, p_{i,j'}X_{i,j'}$  are taken from (5)).

Since the set of colours is bounded (by a constant determined by  $\mathcal{M}$ ), Ramsey's theorem yields a bound B guaranteeing that there is a monochromatic clique of size 3 among the vertices  $\bar{p}\bar{X}$ ,  $\bar{p}\bar{X}\bar{\alpha}_n$ ,  $\bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}$ , ...,  $\bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}\cdots\bar{\alpha}_{n-B}$ . (We have soundly chosen i so that  $n=n_i$  is bigger than B.) We fix such a monochromatic clique MC, denoting its 3 vertices as

$$\bar{p}\bar{X}\beta, \bar{p}\bar{X}\beta\gamma, \bar{p}\bar{X}\beta\gamma\bar{\gamma}$$
, and its colour as  $C = (p'X', p'X', \bar{Q}, \bar{Q}, (\mathcal{D}_q, \mathcal{E}_q)_{q \in \bar{Q}}, Q', Q'_0)$ .

This is sound, since the fact that both edges  $\{\bar{p}\bar{X}\beta,\bar{p}\bar{X}\beta\gamma\}$  and  $\{\bar{p}\bar{X}\beta\gamma,\bar{p}\bar{X}\beta\gamma\bar{\gamma}\}$  have the same colour entails that the first component in this colour is the same as the second component, and the third component is the same as the fourth component.

We now show that the conditions 1–5 are satisfied for the presentation of  $\bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}\cdots\bar{\alpha}_1$  as  $\bar{p}\bar{X}\beta\gamma\delta$ , where  $\delta=\bar{\gamma}\bar{\alpha}_k\bar{\alpha}_{k-1}\cdots\bar{\alpha}_1$  for the respective k.

Conditions 1 and 2 are trivial (due to the colour C).

Condition 3: Let  $q \in \bar{Q}$  and  $\mathrm{ES}(q\gamma) = \{q'\}$  (hence also  $q' \in \bar{Q}$ ). Then  $\mathcal{E}_q = \mathrm{ES}(q\gamma) = \mathrm{ES}(q\gamma\bar{\gamma}) = \{q'\}$  (since MC is monochromatic). This entails  $\mathrm{ES}(q'\bar{\gamma}) = \{q'\}$ , hence  $\mathcal{E}_{q'} = \{q'\}$ , which in turn entails  $\mathrm{ES}(q'\gamma) = \{q'\}$ .

Condition 4: We first note a general fact:  $DS(p\mu\nu) = \bigcup_{q \in DS(p\mu)} DS(q\nu)$ . Since  $\bar{Q} = DS(\bar{p}\bar{X}\beta) = DS(\bar{p}\bar{X}\beta\gamma) = DS(\bar{p}\bar{X}\beta\gamma\bar{\gamma})$ , for each  $q' \in \bar{Q}$  there is thus  $q \in \bar{Q}$  such that  $q' \in \mathcal{D}_q$ . We also have the following "transitivity": if  $q_1, q_2, q_3 \in \bar{Q}$ ,  $q_1 \in \mathcal{D}_{q_2}$ , and  $q_2 \in \mathcal{D}_{q_3}$ , then  $q_1 \in \mathcal{D}_{q_3}$  (since MC is monochromatic). For any  $q' \in \bar{Q}$  there is clearly a "chain"  $q' = q_1, q_2, q_3, \ldots, q_\ell$  where  $\ell > 1$ ,  $q_j \in \mathcal{D}_{q_{j+1}}$  for all  $j \in [1, \ell-1]$ , and  $q_j = q_\ell$  for some  $j < \ell$ . By the above transitivity,  $q_\ell$  is self-containing  $(q_\ell \in \mathcal{D}_{q_\ell})$  and thus  $q_\ell \in DS(q_\ell\gamma)$  and  $q' \in \mathcal{D}_{q_\ell}$  (hence  $q' \in DS(q_\ell\gamma)$ ).

Condition 5: For any three configurations at least two belong to the same  $\sim_0$ -class. Since the edges among the vertices  $\bar{p}\bar{X}\beta$ ,  $\bar{p}\bar{X}\beta\gamma$ ,  $\bar{p}\bar{X}\beta\gamma\bar{\gamma}$  have the same  $Q_0'$  in their colour C, we get that  $Q_0' = Q'$ , and thus also  $q'\gamma\delta \sim_0 q'\delta$  for all  $q' \in \bar{Q}$  such that  $q'\gamma\delta \not\sim q'\delta$ . Now if for all  $q' \in \bar{Q}$  we had  $q'\gamma\delta \sim q'\delta$  (which includes the case  $\bar{Q} = \emptyset$ ), then we would get a contradiction with our choice of (5) since it could have been shortened as in claim 10.  $\square$ 

Now we are already close to lemma 5:

Claim 13. There are  $v \in \Sigma^*$ ,  $x, w, y, z \in \Sigma^+$ ,  $p, q \in Q$ ,  $X \in \Gamma$ ,  $\gamma \in \Gamma^+$ ,  $\delta \in \Gamma^*$  such that  $p_0 X_0 \xrightarrow{v} pX \delta$ ,  $pX \xrightarrow{x} pX \gamma$ ,  $pX \xrightarrow{w} q$ ,  $q\gamma \xrightarrow{y} q$ , and

- either  $z \in \mathcal{L}(q\delta)$  and  $z \notin \mathcal{L}(q\gamma^{\ell}\delta)$  for all  $\ell > 0$ ,
- or  $z \notin \mathcal{L}(q\delta)$  and  $z \in \mathcal{L}(q\gamma^{\ell}\delta)$  for all  $\ell > 0$ .

**Proof:** We fix one  $\bar{p}\bar{X}\beta\gamma\delta$  guaranteed by claim 12 (satisfying the respective conditions 1–5). There are  $v \in \Sigma^*$ ,  $x, w, y, \bar{z} \in \Sigma^+$ ,  $p, q \in Q$ ,  $X \in \Gamma$ ,  $\gamma \in \Gamma^+$ ,  $\delta \in \Gamma^*$ ,  $q' \in \mathrm{DS}(q\gamma)$  such that

$$p_0 X_0 \xrightarrow{v} p X \delta$$
,  $p X \xrightarrow{x} p X \gamma$ ,  $p X \xrightarrow{w} q$ ,  $q \gamma \xrightarrow{y} q$ , and  $\mathcal{L}(q' \gamma \delta)$  and  $\mathcal{L}(q' \delta)$  differ on  $\bar{z}$  (i.e.,  $\bar{z} \in (\mathcal{L}(q' \gamma \delta) \setminus \mathcal{L}(q' \delta)) \cup (\mathcal{L}(q' \delta) \setminus \mathcal{L}(q' \gamma \delta))$ .

(Indeed: The respective computation (5) can be written  $p_0X_0 \xrightarrow{v} pX\delta \xrightarrow{x} pX\gamma\delta \xrightarrow{w'} \bar{p}\bar{X}\beta\gamma\delta$  where x and  $\gamma$  are nonempty. The claimed q' and [nonempty]  $\bar{z}$  are guaranteed by 5 in claim 12, and q is a respective self-containing state from 4. Since  $q \in DS(\bar{p}\bar{X}\beta)$  and  $q \in DS(q\gamma)$ , we get  $pX\gamma\delta \xrightarrow{w'w''} q\gamma\delta \xrightarrow{y} q\delta$ , where  $w'' \neq \varepsilon$ . We also have  $y \neq \varepsilon$ , since otherwise  $DS(q\gamma) = ES(q\gamma) = \{q\}, q' = q$ , and we could not have  $q\gamma\delta \nsim q\delta$  and  $q\gamma\delta \sim_0 q\delta$ .)

Since  $q' \in DS(q\gamma)$ , we can fix z' such that  $q\gamma \xrightarrow{z'} q'$ . Hence the languages  $\mathcal{L}(q\gamma\delta)$  and  $\mathcal{L}(q\gamma\delta)$  differ on  $z = z'\bar{z}$ ; more generally,  $\mathcal{L}(q\gamma^{\ell+1}\gamma\delta)$  and  $\mathcal{L}(q\gamma^{\ell}\gamma\delta)$  differ on  $y^{\ell}z$  for all  $\ell \geq 0$ . Now we aim to find out for which  $\ell$  we have  $z \in \mathcal{L}(q\gamma^{\ell}\delta)$ .

We recall that  $\bar{Q} = \mathrm{DS}(\bar{p}\bar{X}\beta) = \mathrm{DS}(\bar{p}\bar{X}\beta\gamma)$ ; hence  $\bigcup_{\bar{q}\in\bar{Q}}\mathrm{DS}(\bar{q}\gamma) = \bar{Q}$ . Since  $q\in\bar{Q}$ , we get that  $\mathrm{DS}(q\gamma^d)\subseteq\bar{Q}$  for all  $d\in\mathbb{N}$  (by induction). We now distinguish two cases:

- 1. For each prefix  $z_1$  of z and each  $d \leq |z|$  we have: if  $q\gamma^d \xrightarrow{z_1} \bar{q}$ , then  $\mathrm{ES}(\bar{q}\gamma) = \emptyset$ .
- 2. There are a prefix  $z_1$  of z,  $d \le |z|$ , and  $\bar{q}, q'' \in \bar{Q}$  such that  $q\gamma^d \xrightarrow{z_1} \bar{q}$  and  $\mathrm{ES}(\bar{q}\gamma) = \{q''\}$ .

In the case 1 we clearly have either  $\forall \ell > |z| : z \in \mathcal{L}(q\gamma^{\ell}\delta)$  or  $\forall \ell > |z| : z \notin \mathcal{L}(q\gamma^{\ell}\delta)$  (here  $\delta$  plays no role). In the case 2 we recall that  $\bar{q}\gamma \xrightarrow{\varepsilon} q''$  entails that  $\bar{q}\gamma^k\delta \xrightarrow{\varepsilon} q''\delta$  for all  $k \geq 1$  (since  $\mathrm{ES}(q''\gamma) = \{q''\}$  by 3 in claim 12). Hence we have either  $\forall \ell > |z| + 1 : z \in \mathcal{L}(q\gamma^{\ell}\delta)$  or  $\forall \ell > |z| + 1 : z \notin \mathcal{L}(q\gamma^{\ell}\delta)$ .

Since  $\mathcal{L}(q\gamma^2\delta)$  and  $\mathcal{L}(q\gamma^1\delta)$  differ on z, we deduce that there is  $\ell_0 \geq 1$  such that either  $z \in \mathcal{L}(q\gamma^{\ell_0}\delta)$  and  $z \notin \mathcal{L}(q\gamma^{\ell}\delta)$  for all  $\ell > \ell_0$ , or  $z \notin \mathcal{L}(q\gamma^{\ell_0}\delta)$  and  $z \in \mathcal{L}(q\gamma^{\ell}\delta)$  for all  $\ell > \ell_0$ . Hence for  $\bar{\delta} = \gamma^{\ell_0}\delta$  we have either  $z \in \mathcal{L}(q\bar{\delta})$  and  $z \notin \mathcal{L}(q\gamma^{\ell}\bar{\delta})$  for all  $\ell > 0$ , or  $z \notin \mathcal{L}(q\bar{\delta})$  and  $z \in \mathcal{L}(q\gamma^{\ell}\bar{\delta})$  for all  $\ell > 0$ . Since for  $\bar{v} = vx^{\ell_0}$  we have  $p_0X_0 \xrightarrow{\bar{v}} pX\bar{\delta}$ , the claim is proven.  $\square$ 

Claim 13 is a weaker version of lemma 5; it shows that there is  $L' \in \{L, \overline{L}\}$  such that  $vx^mwy^mz \in L'$  and  $vx^mwy^nz \notin L'$  for m > n. To handle the case m < n, we have to find out for which  $\ell$  we have  $y^{\ell}z \in \mathcal{L}(q\delta)$ . We thus look at the computation from  $q\delta$  on the infinite word  $y^{\omega}$  (recalling our convention that this computation is infinite, stepwise reading the word  $yyy\cdots$ ), and use the obvious fact that after a prefix this computation becomes "periodic" (either cycling among finitely many configurations, or increasing the stack forever).

Claim 14. For any configuration  $q\delta$  and words y, z there are numbers  $k \geq 0$  and P > 0 ("period") such that for all  $\ell \geq k$  the remainder ( $\ell \mod P$ ) determines whether or not  $\mathcal{L}(q\delta) \ni y^{\ell}z$ .

**Proof:** We assume  $y \neq \varepsilon$  (otherwise the claim is trivial). For the infinite computation from  $q\delta$  on  $yyy\cdots$  there are obviously  $k_1 \geq 0$ ,  $k_2 > 0$ ,  $\bar{q} \in Q$ , and  $\rho, \mu, \nu \in \Gamma^*$  such that the computation can be written  $q\delta \xrightarrow{y^{k_1}} \bar{q}\rho\nu \xrightarrow{y^{k_2}} \bar{q}\rho\mu\nu \xrightarrow{y^{k_2}} \bar{q}\rho\mu\mu\nu \xrightarrow{y^{k_2}} \bar{q}\rho\mu\mu\nu \xrightarrow{y^{k_2}} \bar{q}\rho\mu\mu\nu \xrightarrow{y^{k_2}} \cdots$  where  $\bar{q}\rho \xrightarrow{y^{k_2}} \bar{q}\rho\mu$ . (We have  $\mu = \varepsilon$  if the computation visits only finitely many configurations, and otherwise we consider the stair-factorization of the computation.)

For each  $j \in [0, k_2-1]$  we put  $\bar{q}\rho \xrightarrow{y^j} \bar{q}\rho_j$ , and we have two possible cases:

- 1. There is  $d_0 \geq 0$  such that for all  $d \geq d_0$  performing z from  $\bar{q}\rho_j\mu^d\nu$  does not reach  $\nu$  at the bottom.
- 2. There are  $d_0 \geq 0$ , a prefix z' of z,  $q' \in Q$ , and  $\bar{d} \in [1, |Q|]$  such that  $\bar{q}\rho_j \mu^{d_0} \xrightarrow{z'} q'$  and  $q'\mu^{\bar{d}} \xrightarrow{\varepsilon} q'$ .

In the case 1 either  $\mathcal{L}(q\delta) \ni y^{d \cdot k_2 + j} z$  for all  $d \ge d_0$ , or  $\mathcal{L}(q\delta) \not\ni y^{d \cdot k_2 + j} z$  for all  $d \ge d_0$ . In the case 2, for each  $d \ge 0$  we have  $q' \mu^d \stackrel{\varepsilon}{\to} q_d$  where  $q_{d_1} = q_{d_2}$  if  $d_1 \equiv d_2 \pmod{\bar{d}}$ . Hence for each  $d \ge d_0$ , the (non)membership of  $y^{d \cdot k_2 + j} z$  in  $\mathcal{L}(q\delta)$  is determined by  $(d \mod \bar{d})$ . The claim is thus clear.

Now we finish the proof of lemma 5. We take the notation from claim 13; for the respective  $q\delta, y, z$  we add k, P from claim 14. Let  $k_0$  be a multiple of P that is bigger than k. We now view  $x^{k_0}$ ,  $y^{k_0}$ ,  $\gamma^{k_0}$  as new  $x, y, \gamma$ , respectively. Claims 13 and 14 now yield the statement of lemma 5.

## 4 Conclusion and Open Problems

In this paper, we have introduced a new notion of the C-simple problem that reduces to each problem in C, being thus a conceptual counterpart to the C-hard problem to which each problem in C reduces. We have illustrated this concept on the definition of the DCFL'-simple problem that reduces to each DCFL' language under the truth-table reduction by Mealy machines. We have proven that the DCFL' language  $L_{\#} = \{0^n1^n \mid n \geq 1\}$  is DCFL'-simple, and thus represents the simplest languages in the class DCFL'. This result finds its application in expanding the known lower bound for  $L_{\#}$ , namely that  $L_{\#}$  cannot be recognized by the neural network model 1ANN, to all DCFL' languages. Moreover, the class DCFLS of DCFL'-simple problems containing the regular languages is a strict subclass of DCFL and has similar closure properties as DCFL.

We note that the hardest context-free language  $L_0$  by Greibach [2], where each L in CFL is an inverse homomorphic image of  $L_0$  or  $L_0 \setminus \{\varepsilon\}$ , can be viewed as CFL-hard w.r.t. a many-one reduction based on Mealy machines realizing the respective homomorphisms. Our aims in the definition of DCFL'-simple problems cannot be achieved by such a many-one reduction, hence we have generalized it to a truth-table reduction. We can alternatively consider a general Turing reduction that is implemented by a Mealy machine which queries the oracle at special query states, each associated with a corresponding query suffix, while its next transition from the query state depends on the given oracle answer. The oracle Mealy machine then accepts an input word if it reaches an accept state after reading the input. The language  $L_{\#}$  proves to be DCFL'-simple under this Turing reduction allowing for an unbounded number of online oracle queries; this can be shown by claim 13 (a weaker version of lemma 5).

It is natural to try extending our result to non-regular nondeterministic (or at least unambiguous) context-free languages, by possibly showing that  $L_{\#}$  is CFL'-simple. Another important challenge for further research is looking for C-simple problems for other complexity classes C and suitable reductions. This could provide an effective tool for strengthening lower-bounds results known for single problems to the whole classes of problems, which deserves a deeper study.

#### Acknowledgements

Presented research has been partially supported by the Czech Science Foundation, grant GA19-05704S, and by the institutional support RVO: 67985807 (J. Šíma). J. Šíma also thanks Martin Plátek for his intensive collaboration at the first stages of this research.

### References

- [1] Anabtawi, M., Hassan, S., Kapoutsis, C.A., Zakzok, M.: An oracle hierarchy for small one-way finite automata. In: Proceedings of LATA 2019. pp. 57–69. LNCS 11417, Springer (2019). https://doi.org/10.1007/978-3-030-13435-8\_4
- [2] Greibach, S.A.: The hardest context-free language. SIAM J. Comput. **2**(4), 304–310 (1973). https://doi.org/10.1137/0202025
- [3] Hopcroft, J.E., Ullman, J.D.: Formal languages and their relation to automata. Addison-Wesley (1969), https://www.worldcat.org/oclc/00005012

- [4] Jančar, P.: Deciding semantic finiteness of pushdown processes and first-order grammars w.r.t. bisimulation equivalence. J. Comput. Syst. Sci. **109**, 22–44 (2020). https://doi.org/10.1016/j.jcss.2019.10.002
- [5] Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Restarting automata. In: Proceedings of FCT 1995. pp. 283–292. LNCS 965, Springer (1995). https://doi.org/10.1007/3-540-60249-6\_60
- [6] Mráz, F., Pardubská, D., Plátek, M., Šíma, J.: Pumping deterministic monotone restarting automata and DCFL. In: Proceedings of ITAT 2020. pp. 51–58. CEUR Workshop Proceedings 2718 (2020), http://ceur-ws.org/Vol-2718/paper13.pdf
- [7] Reinhardt, K.: Hierarchies over the context-free languages. In: Proceedings of IMYCS 1990. pp. 214–224. LNCS 464, Springer (1990). https://doi.org/10.1007/3-540-53414-8\_44
- [8] Siegelmann, H.T.: Neural networks and analog computation Beyond the Turing limit. Birkhäuser (1999)
- [9] Šíma, J.: Analog neuron hierarchy. Neural Netw. **128**, 199–215 (2020). https://doi.org/10.1016/j.neunet.2020.05.006
- [10] Šíma, J.: Stronger separation of analog neuron hierarchy by deterministic context-free languages (2021), arXiv:2102.01633 (submitted to a journal)
- [11] Šíma, J., Orponen, P.: General-purpose computation with neural networks: A survey of complexity theoretic results. Neural Comput. **15**(12), 2727–2778 (2003). https://doi.org/10.1162/089976603322518731
- [12] Šíma, J., Plátek, M.: One analog neuron cannot recognize deterministic context-free languages. In: Proceedings of ICONIP 2019, Part III. pp. 77–89. LNCS 11955, Springer (2019). https://doi.org/10.1007/978-3-030-36718-3\_7
- [13] Yamakami, T.: Oracle pushdown automata, nondeterministic reducibilities, and the hierarchy over the family of context-free languages. In: Proceedings of SOFSEM 2014. pp. 514–525. LNCS 8327, Springer (2014). https://doi.org/10.1007/978-3-319-04298-5\_45, (full version arXiv:1303.1717)