The Simplest Non-Regular Deterministic Context-Free Language

Petr Jančar
Dept. of Computer Science, Faculty of Science, Palacký University Olomouc, Czech Republic

Jiří Šíma
Institute of Computer Science of the Czech Academy of Sciences, Prague, Czech Republic

Abstract

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

By describing an application in the area of neural networks (elaborated in another paper), we demonstrate that $\mathcal{C}$-simple problems under suitable reductions can provide a tool for expanding the lower-bound results known for single problems to the whole classes of problems.

2012 ACM Subject Classification Theory of computation → Grammars and context-free languages; Theory of computation → Problems, reductions and completeness; Theory of computation → Transducers

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

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.63

Related Version *Full Version*: `https://arxiv.org/abs/2102.10416`

Funding Presented research has been partially supported by the Czech Science Foundation, grant GA19-05704S, and by the institutional support RVO: 67985807 (J. Šíma).

Acknowledgements J. Šíma also thanks Martin Plátek for his intensive collaboration at the first stages of this research.

1 Introduction

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

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 Σ) 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, we have a finite-state transducer $\mathcal{A}$ that transforms a given input word $w \in \Sigma^*$ to a word $\mathcal{A}(w) \in \Delta^*$ (a query prefix), and each state q of $\mathcal{A}$ is associated with a finite tuple $\sigma_q = (s_{q,1}, s_{q,2}, \ldots, s_{q,r_q})$ of r_q words from Δ^* (query suffixes), and with a truth table $f_q : \{0,1\}^{r_q} \to \{0,1\}$. The oracle-machine $\mathcal{A}^L$ behaves like $\mathcal{A}$, hence it reads an input word w (translating it to $\mathcal{A}(w)$) by which it enters a state q, and then submits r_q queries, i.e. the words $\mathcal{A}(w) \cdot s_{q,i}$ for all $i \in \{1, 2, \ldots, r_q\}$, to the oracle that for each $i \in \{1, 2, \ldots, r_q\}$ decides whether or not $\mathcal{A}(w) \cdot s_{q,i}$ is in L (or, equivalently, whether or not $\mathcal{A}(w)$ belongs to the quotient $L/s_{q,i} = \{v \in \Delta^* \mid v \cdot s_{q,i} \in L\}$); the oracle-answers are then aggregated by the truth table f_q, which decides whether or not $w \in L_0$.

This truth-table reduction by Mealy machines induces a preorder on the class of languages; we denote this preorder by $\leq^{\mathrm{A}}_{tt}$, using the superscript "A" to stress that our truth-table reduction is realized by simple *automata*, not by general Turing machines. The main technical result of this paper is that the DCFL′ language $L_\# = \{0^n 1^n \mid n \geq 1\}$ (over the binary alphabet $\{0,1\}$) is DCFL′-simple, since $L_\# \leq^{\mathrm{A}}_{tt} L$ 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 [9, 12]. The so-called analog neuron hierarchy [10] of binary-state NNs with increasing number of α extra analog-state neurons, denoted as αANN 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 αANN also for the class of languages accepted by αANNs, which can clearly be distinguished by the context. The separation 1ANN $\subsetneq$ 2ANN has been witnessed by the DCFL′ language $L_\# \in$ 2ANN $\setminus$ 1ANN. The proof of $L_\# \notin$ 1ANN is rather technical (based on the Bolzano-Weierstrass theorem) which could hardly be generalized to other DCFL′ languages, while it was conjectured that $L \notin$ 1ANN 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 $L_\#$ 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. If the composition of a 1ANN that carries out the reduction of $L_\#$ to L with a hypothetical 1ANN accepting L can be realized by another 1ANN, which would thus accept $L_\#$, we get that no 1ANN accepting L can exist, since $L_\#$ has been proven not to be accepted 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* [4, 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 $S \Rightarrow^* vx^m wy^m z \in L$ 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 $L_1 = \{0^m1^n \mid 1 \leq m \leq n\}$ 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 $v = w = z = \varepsilon$ (ε 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 each DCFL′ language $L \subseteq \Delta^*$, over an alphabet Δ, there are 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 $vx^mwy^nz \in L'$. In fact, we even show that either for all $m,n \geq 0$ we have $vx^mwy^nz \in L'$ iff $m = n$, or for all $m,n \geq 0$ we have $vx^mwy^nz \in L'$ iff $m \leq n$. We note that this technical result seems interesting on its own since in the class DCFL it substantially strengthens the known result for context-free languages (CFL) [2, Theorem 2.10] that for any CFL′ language $L \subseteq \Delta^*$ (where CFL′ = CFL $\setminus$ REG) there is a so-called non-degenerated iterative pair $(v,x,w,y,z) \in (\Delta^*)^5$ with non-empty xy, satisfying $vx^mwy^mz \in L$ for all $m \geq 0$ and $vx^mwy^nz \notin L$ for some $m \neq n$.

Hence the inconsistent 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 $vx^mwy^{n-1}z \stackrel{?}{\in} L$ and $vx^mwy^nz \stackrel{?}{\in} L$. 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 $\setminus$ 1ANN) in the analog neuron hierarchy [10, 11]. This result constitutes a non-trivial application of the proposed concept of DCFL′-simple problem. Moreover, if we could generalize the result to (nondeterministic) CFL, e.g. by proving that some DCFL′ language is CFL′-simple, which would imply that $L_{\#}$ is CFL′-simple by the transitivity of reduction, then we would achieve that even the intersection of CFL′ and 1ANN is empty. We note the interesting fact that $L_{\#}$ cannot be CSL′-simple (under our reduction), since 1ANN accepts some context-sensitive languages outside CFL [10].

In general, if we show that some $\mathcal{C}$-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 $\mathcal{C}$ are not solvable by $\mathcal{M}$ either. The notion of $\mathcal{C}$-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 $L_{\#} \notin$ 1ANN, expanding to DCFL′ $\cap$ 1ANN $= \emptyset$. It seems worthwhile to explore if looking for $\mathcal{C}$-simple problems in other complexity classes $\mathcal{C}$ could provide effective tools for strengthening known lower bounds.

We remark that the hardest context-free language by Greibach [3] can be viewed as CFL-hard under a special type of our reduction $\leq^{\mathrm{A}}_{tt}$. 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 [8]. For the same purpose, oracle pushdown automata have been used for many-one, truth-table, and Turing reducibilities, respectively, inducing the underlying definitions also to oracle nondeterministic finite automata [14]. 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 [13], 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 [6], which have initiated investigations of non-regularity degrees in DCFL [7].

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 [5]. 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 non-equivalent 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 that ε 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 ε-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 $\mathrm{DCFL}' = \mathrm{DCFL} \setminus \mathrm{REG}$.

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 Δ. 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*

- *either for all $m, n \geq 0$ we have $vx^mwy^nz \in L'$ iff $m = n$,*
- *or for all $m, n \geq 0$ we have $vx^mwy^nz \in L'$ iff $m \leq n$;*

this entails that for all $m \geq 0$ and $n > 0$ we have

$$\left(vx^mwy^{n-1}z \notin L' \text{ and } vx^mwy^nz \in L'\right) \quad \text{iff} \quad m = n\,. \tag{1}$$

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 $\sigma_q = (s_{q,1}, s_{q,2}, \ldots, s_{q,r_q})$ of strings in Δ^* 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 $L/s_{p,i} = \{u \in \Delta^* \mid u \cdot s_{p,i} \in L\}$, for each suffix $s_{p,i}$ in σ_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

$$f_p\left(\chi_{L/s_{p,1}}(\mathcal{A}(w)), \chi_{L/s_{p,2}}(\mathcal{A}(w)), \ldots, \chi_{L/s_{p,r_p}}(\mathcal{A}(w))\right) = 1$$

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

We say that $L_1 \subseteq \Sigma^*$ is *truth-table reducible* to $L_2 \subseteq \Delta^*$ *by a Mealy machine*, which is denoted as $L_1 \leq_{tt}^{\mathrm{A}} L_2$, 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 $\leq_{tt}^{\mathrm{A}}$ is a preorder.

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

Proof. The relation $\leq_{tt}^{\mathrm{A}}$ 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 $\leq_{tt}^{\mathrm{A}}$ is transitive. Let $L_1 \leq_{tt}^{\mathrm{A}} L_2$ and $L_2 \leq_{tt}^{\mathrm{A}} L_3$ 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 $L_1 \leq_{tt}^{\mathrm{A}} L_3$. 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 Θ^*,

$$\sigma_p = \left(\lambda_2(p_2, s_{p_1,i}) \cdot s_{p_2(i),j} \,\middle|\, i = 1, \ldots, r_{p_1}\,,\ j = 1, \ldots, r_{p_2(i)}\right)$$

where $\pi_{p_1} = (s_{p_1,1}, s_{p_1,2} \ldots, s_{p_1,r_{p_1}}) \in \Delta^{r_{p_1}}$ and $\varrho_{p_2(i)} = (s_{p_2(i),1}, s_{p_2(i),2} \ldots, s_{p_2(i),r_{p_2(i)}}) \in \Theta^{r_{p_2(i)}}$ 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 $i \in \{1, \ldots, r_{p_1}\}$, respectively, and the truth table $f_p = g_{p_1}(h_{p_2(1)}, \ldots, h_{p_2(r_{p_1})})$ aggregates the answers to the corresponding oracle queries, which ensures $L_1 = \mathcal{L}(\mathcal{A}^{L_3}) \subseteq \Sigma^*$. ◀

We say that $L_0 \subseteq \Sigma^*$ is *DCFL′-simple* if $L_0 \leq_{tt}^{\mathrm{A}} L$ for every non-regular deterministic context-free language $L \subseteq \Delta^*$. We show that Theorem 1 entails that the DCFL′ language $L_{\#}$ is DCFL′-simple. 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 $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ 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 $\delta(q_0, 0) = \delta(q_1, 0) = q_1$, $\delta(q_1, 1) = \delta(q_2, 1) = q_2$, $\lambda(q_0, 0) = vx$, $\lambda(q_1, 0) = x$, $\lambda(q_1, 1) = w$, $\lambda(q_2, 1) = y$, $\sigma_{q_0} = \sigma_{q_1} = ()$

[1] 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.

$(r_{q_0} = r_{q_1} = 0)$, $\sigma_{q_2} = (z, yz)$ $(r_{q_2} = 2)$, $f_{q_0} = f_{q_1} = 0$, $f_{q_2}(0,0) = f_{q_2}(1,1) = 0$, 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 $L_\# \leq^{\mathrm{A}}_{tt} L$. Hence, $L_\#$ is DCFL′-simple. ◀

▶ **Proposition 4.**

1. *REG* $\subsetneq$ *DCFLS.*
2. *DCFLS* $\subsetneq$ *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, $L \leq^{\mathrm{A}}_{tt} L_\#$ 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 $L \leq^{\mathrm{A}}_{tt} L_\#$ 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 ε-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 $L = \mathcal{L}(\mathcal{M}) \in$ 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 $L_R \leq^{\mathrm{A}}_{tt} L_\#$ 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 $w_1cw_2^R \in \mathcal{L}(\mathcal{A}^{L_\#}) \setminus L_R$.
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 $R = \{1\}^*$, $L_1 = \{0^m1^m0^n \mid m,n \geq 1\}$, $L_2 = \{0^m1^n0^n \mid m,n \geq 1\}$, and $L_3 = L_1 \cup (\{1\} \cdot L_2)$ are DCFL′-simple while $R \cdot L_3 \notin$ DCFL, $L_1 \cap L_2 \notin$ CFL, and $L_1 \cup L_2 \notin$ DCFL are not DCFL′-simple according to 2. ◀

3 Proof of the Main Result (Theorem 1)

Theorem 1 follows from the (more specific) next lemma that we prove in this section. (**See Appendix for an informal overview with figures.**)

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 $L=\mathcal{L}(\mathcal{M})=\mathcal{L}(p_0X_0)$ 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*

$$p_0X_0\xrightarrow{v}pX\delta\xrightarrow{x}pX\gamma\delta\xrightarrow{x}pX\gamma\gamma\delta\xrightarrow{x}pX\gamma\gamma\gamma\delta\xrightarrow{x}\cdots; \tag{2}$$

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 $k\le\ell$ (for all $k,\ell\in\mathbb{N}$);*
   b. *$\mathcal{L}(q\gamma^k\delta)\ni y^\ell z$ iff $k\ne\ell$ (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 $m\le n$ (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 $L=\mathcal{L}(p_0X_0)$ 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 $u\in\Sigma^*$ from p_0X_0. This can be accomplished in the standard way, by adding a special bottom-of-stack symbol $\perp$ and a (non-accepting) fail-state. (Each empty-stack configuration $q\varepsilon$ becomes $q\perp$, and each originally stuck computation enters the fail-state where it loops. We also recall that all ε-steps are popping, and thus infinite ε-sequences are impossible.) Hence for any infinite word $a_1a_2a_3\cdots$ in Σ^ω there is a 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 $u\backslash L=\{v\in\Sigma^*\mid uv\in L\}$; concatenation has priority over $\backslash$, hence $u_1u_2\backslash L=(u_1u_2)\backslash L$. (The next claim is valid for any non-regular L.)

▷ Claim 6. We can fix an infinite word $a_1a_2a_3\cdots$ in Σ^ω ($a_i\in\Sigma$) such that $a_1a_2\cdots a_i\backslash L\ne a_1a_2\cdots a_j\backslash L$ for all $i\ne j$.

Proof. Let us consider the labelled transition system $\mathcal{T}=(\mathrm{LQ}(L),\Sigma,(\xrightarrow{a})_{a\in\Sigma})$ where $\mathrm{LQ}(L)=\{u\backslash L\mid u\in\Sigma^*\}$ and $\xrightarrow{a}=\{(L',a\backslash L')\mid L'\in\mathrm{LQ}(L)\}$. (We recall that $L'=u\backslash L$ entails $a\backslash L'=ua\backslash L$.) Since L is non-regular, the set of states reachable from $L=\varepsilon\backslash L$ 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 ε-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 $p\alpha \xrightarrow{\varepsilon} p'\alpha'$ (α' being a suffix of α). If the path $p\alpha \xrightarrow{\varepsilon} p'\alpha'$ does not go via an accepting state (in F), then $\mathcal{L}(p\alpha) = \mathcal{L}(p'\alpha')$; otherwise $\mathcal{L}(p\alpha) = \{\varepsilon\} \cup \mathcal{L}(p'\alpha')$. (We note that the configurations in the computation (2) that start with pX are necessarily stable: since we have $pX \xrightarrow{x} pX\gamma$ for $x \in \Sigma^+$, we cannot have $pX \xrightarrow{\varepsilon} p'$.)

▷ Claim 7. Each configuration is visited at most twice by

$$\text{the computation of } \mathcal{M} \text{ from } p_0X_0 \text{ on } a_1a_2a_3\cdots \text{ that is fixed by Claim 6.} \tag{3}$$

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) ε-steps via unstable configurations. Since such an ε-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 ε. 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, ℓ.

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

$$p_0X_0 \xrightarrow{v_0} p_1X_1\alpha_1 \xrightarrow{v_1} p_2X_2\alpha_2\alpha_1 \xrightarrow{v_2} p_3X_3\alpha_3\alpha_2\alpha_1 \xrightarrow{v_3} \cdots \tag{4}$$

so that for each $i \in \mathbb{N}$ we have $v_i \in \Sigma^+$ and $p_iX_i \xrightarrow{v_i} p_{i+1}X_{i+1}\alpha_{i+1}$ where α_{i+1} is a nonempty suffix of the right-hand side of a rule in R (i.e., a nonempty suffix of γ 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 $|X\beta|$; each level $pX\beta$ has thus a unique position which we denote $\text{POS}(pX\beta)$. Since each configuration is visited at most twice in (3), and the set of configurations with a fixed length is finite, we get that the set of levels is infinite, with elements $p'_0X'_0$, $p_1X_1\beta_1$, $p_2X_2\beta_2$, ... where $0 \leq \text{POS}(p'_0X'_0) < \text{POS}(p_1X_1\beta_1) < \text{POS}(p_2X_2\beta_2) < \cdots$. The computation (3) can be thus presented as

$$p_0X_0 \xrightarrow{v'_0} p'_0X'_0 \xrightarrow{v''_0} p_1X_1\beta_1 \xrightarrow{v_1} p_2X_2\beta_2 \xrightarrow{v_2} p_3X_3\beta_3 \xrightarrow{v_3} \cdots$$

where $|v'_0| = \text{POS}(p'_0X'_0)$, and $|v_0v_1\cdots v_{j-1}| = \text{POS}(p_jX_j\beta_j)$ for $j \geq 1$, putting $v_0 = v'_0v''_0$.

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 γ_1 and γ_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 $p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1$ the *level-configurations* (for all $i \in \mathbb{N}$). Since we have $\mathcal{L}(p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1) \setminus \{\varepsilon\} = (v_0v_1\cdots v_{i-1}\backslash L) \setminus \{\varepsilon\}$, 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 its $\sim$-class is preserved. We use the notation $\mathrm{DS}(p\alpha)$ (the "down-states" of $p\alpha$), putting

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

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

Proof. Let us consider $u \in \Sigma^*$. If $u \in \mathcal{L}(p\beta)$, then $u \in \mathcal{L}(p\beta\mu)$ for all $\mu \in \Gamma^*$. If $u \notin \mathcal{L}(p\beta)$ and there is no prefix u' of u such that $p\beta \xrightarrow{u'} q$, then $u \notin \mathcal{L}(p\beta\mu)$ for all $\mu \in \Gamma^*$. If $u \notin \mathcal{L}(p\beta)$ and $u = u'u''$ where $pX\beta \xrightarrow{u'} q$ (necessarily for some $q \in \mathrm{DS}(pX\beta)$), then $u \in \mathcal{L}(p\beta\mu)$ iff $u'' \in \mathcal{L}(q\mu)$. 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 $q \in \mathrm{DS}(p'X'\beta_3)$ 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, $0 \le j_1 < j_2 \le i$ 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 $i \ge 1$ we fix a (stair-factorized) computation

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

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

$$p_0X_0 \xrightarrow{v_0v_1\cdots v_{i-1}} p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1 \text{ of (4)}.$$

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. ◁

Now we come to a crucial claim in our proof of Lemma 5. Besides the notation $\mathrm{DS}(p\alpha)$ we also introduce $\mathrm{ES}(p\alpha)$ (the by-ε-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 $|\mathrm{ES}(p\alpha)| \le 1$ (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 ε, we also define a (much) coarser equivalence $\sim_0$: we put $p\alpha \sim_0 q\beta$ if ε either belongs to both $\mathcal{L}(p\alpha)$ and $\mathcal{L}(q\beta)$, or belongs to none of them.

The next claim is rather technical but it captures some straightforward combinatorial observations that are handled by a simple use of Ramsey's theorem. Informally speaking, if n_i in the final configuration in (5) is sufficiently large, then we can find a convenient pumping segment in this configuration. (All this should be easily understandable after reading the informal overview with figures in Appendix.)

▷ Claim 12. There is a constant $\mathrm{B} \in \mathbb{N}$ determined by the DPDA $\mathcal{M}$ such that for all $i \in \mathbb{N}$ where $n_i > \mathrm{B}$ 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 $n_i \geq j > j' \geq n_i - \mathrm{B}$ 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 $\mathrm{DS}(\bar{p}\bar{X}\beta)$ and $\mathrm{DS}(\bar{p}\bar{X}\beta\gamma)$ 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 $\mathrm{DS}(q\gamma)$ for some self-containing $q \in \bar{Q}$, where $q \in \bar{Q}$ is *self-containing* if $q \in \mathrm{DS}(q\gamma)$;
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

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

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*:

$$\Big(p_{i,j}X_{i,j},\ p_{i,j'}X_{i,j'},\ \mathrm{DS}(\bar{p}\bar{X}\mu),\ \mathrm{DS}(\bar{p}\bar{X}\mu\nu),\ (\mathrm{DS}(q\nu),\mathrm{ES}(q\nu))_{q\in\mathrm{DS}(\bar{p}\bar{X}\mu)},\ \mathrm{Q}_{\not\sim},\ \mathrm{Q}_0\Big)$$

where $\mathrm{Q}_{\not\sim} = \{q' \in \mathrm{DS}(\bar{p}\bar{X}\mu) \mid q'\nu\rho \not\sim q'\rho\}$ and $\mathrm{Q}_0 = \{q' \in \mathrm{Q}_{\not\sim} \mid q'\nu\rho \sim_0 q'\rho\}$ (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 $\bar{p}\bar{X}$, $\bar{p}\bar{X}\bar{\alpha}_n$, $\bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}$, $\ldots$, $\bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}\cdots\bar{\alpha}_{n-\mathrm{B}}$. (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

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

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: $\mathrm{DS}(p\mu\nu) = \bigcup_{q\in\mathrm{DS}(p\mu)} \mathrm{DS}(q\nu)$. Since $\bar{Q} = \mathrm{DS}(\bar{p}\bar{X}\beta) = \mathrm{DS}(\bar{p}\bar{X}\beta\gamma) = \mathrm{DS}(\bar{p}\bar{X}\beta\gamma\bar{\gamma})$, 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_ℓ is self-containing ($q_\ell \in \mathcal{D}_{q_\ell}$ and thus $q_\ell \in \mathrm{DS}(q_\ell\gamma)$) and $q' \in \mathcal{D}_{q_\ell}$ (hence $q' \in \mathrm{DS}(q_\ell\gamma)$).

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. ◁

Now we state a weaker version of 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 $p_0X_0 \xrightarrow{v} pX\delta$, $pX \xrightarrow{x} pX\gamma$, $pX \xrightarrow{w} q$, $q\gamma \xrightarrow{y} q$, 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

$$p_0X_0 \xrightarrow{v} pX\delta,\ pX \xrightarrow{x} pX\gamma,\ pX \xrightarrow{w} q,\ q\gamma \xrightarrow{y} q, \text{ and } \mathcal{L}(q'\gamma\delta) \text{ and } \mathcal{L}(q'\delta) \text{ differ on } \bar{z}$$

(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 γ 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 $q \in \mathrm{DS}(\bar{p}\bar{X}\beta)$ and $q \in \mathrm{DS}(q\gamma)$, 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 $\mathrm{DS}(q\gamma) = \mathrm{ES}(q\gamma) = \{q\}$, $q' = q$, and we could not have $q\gamma\delta \not\sim q\delta$ and $q\gamma\delta \sim_0 q\delta$.

Since $q' \in \mathrm{DS}(q\gamma)$, we can fix z' such that $q\gamma \xrightarrow{z'} q'$. Hence the languages $\mathcal{L}(q\gamma\gamma\delta)$ 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 ℓ 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, $d \leq |z|$, 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 δ 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. ◁

Claim 13 shows that there is $L' \in \{L, \overline{L}\}$ such that $vx^mwy^mz \in L'$ and $vx^mwy^nz \notin L'$ for $m > n$, which is a weaker version of Lemma 5. To handle the case $m < n$, we have to find out for which ℓ we have $y^\ell z \in \mathcal{L}(q\delta)$. We thus look at the computation from $q\delta$ on the infinite word y^ω (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 $\mathrm{S} \geq 0$ ("shift") and $\mathrm{P} > 0$ ("period") such that for all $\ell \geq \mathrm{S}$ the remainder (ℓ 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 $q\delta \xrightarrow{y^{k_1}} \bar{q}\rho\nu \xrightarrow{y^{k_2}} \bar{q}\rho\mu\nu \xrightarrow{y^{k_2}} \bar{q}\rho\mu\mu\nu \xrightarrow{y^{k_2}} \bar{q}\rho\mu\mu\mu\nu \xrightarrow{y^{k_2}} \cdots$ 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 $\bar{q}\rho \xrightarrow{y^j} \bar{q}_j\rho_j$, and we have two possible cases:

1. There is $d_0 \geq 0$ such that for all $d \geq d_0$ performing z from $\bar{q}_j\rho_j\mu^d\nu$ does not reach ν 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 $\bar{q}_j\rho_j\mu^{d_0} \xrightarrow{z'} q'$ 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 $d \geq d_0$, or $\mathcal{L}(q\delta) \not\ni y^{d\cdot k_2+j}z$ for all $d \geq d_0$.
In the case 2, for each $d \geq 0$ we have $q'\mu^d \xrightarrow{\varepsilon} q_d$ where $q_{d_1} = q_{d_2}$ if $d_1 \equiv d_2 \pmod{\bar{d}}$. Hence for each $d \geq d_0$, 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 S, P from Claim 14. Let k_0 be a multiple of P that is bigger than S. We now view x^{k_0}, y^{k_0}, γ^{k_0} as new x, y, γ, 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 $\mathcal{C}$-simple problem that reduces to each problem in $\mathcal{C}$, being thus a conceptual counterpart to the $\mathcal{C}$-hard problem to which each problem in $\mathcal{C}$ 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 [3], where each L in CFL is an inverse homomorphic image of L_0 or $L_0 \smallsetminus \{\varepsilon\}$, 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 $\mathcal{C}$-simple problems for other complexity classes $\mathcal{C}$ 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.

References

1 M. Anabtawi, S. Hassan, Christos A. Kapoutsis, and M. Zakzok. An oracle hierarchy for small one-way finite automata. In *Proceedings of LATA 2019*, LNCS 11417, pages 57–69. Springer, 2019. doi:10.1007/978-3-030-13435-8_4.

2 Jean Berstel and Luc Boasson. Context-free languages. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 59–102. Elsevier and MIT Press, 1990. doi:10.1016/b978-0-444-88074-1.50007-x.

3 Sheila A. Greibach. The hardest context-free language. *SIAM J. Comput.*, 2(4):304–310, 1973. doi:10.1137/0202025.

4 John E. Hopcroft and Jeffrey D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley, 1969. URL: https://www.worldcat.org/oclc/00005012.

5 Petr Jančar. Deciding semantic finiteness of pushdown processes and first-order grammars w.r.t. bisimulation equivalence. *J. Comput. Syst. Sci.*, 109:22–44, 2020. doi:10.1016/j.jcss.2019.10.002.

6 Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel. On monotonic automata with a restart operation. *J. Autom. Lang. Comb.*, 4(4):287–311, 1999. doi:10.25596/jalc-1999-287.

7 František Mráz, Dana Pardubská, Martin Plátek, and Jiří Šíma. Pumping deterministic monotone restarting automata and DCFL. In *Proceedings of ITAT 2020*, CEUR Workshop Proceedings 2718, pages 51–58, 2020. URL: http://ceur-ws.org/Vol-2718/paper13.pdf.

8 Klaus Reinhardt. Hierarchies over the context-free languages. In *Proceedings of IMYCS 1990*, LNCS 464, pages 214–224. Springer, 1990. doi:10.1007/3-540-53414-8_44.

9 Hava T. Siegelmann. *Neural networks and analog computation – Beyond the Turing limit*. Birkhäuser, 1999.

10 Jiří Šíma. Analog neuron hierarchy. *Neural Netw.*, 128:199–215, 2020. doi:10.1016/j.neunet.2020.05.006.

11 Jiří Šíma. Stronger separation of analog neuron hierarchy by deterministic context-free languages, 2021. (submitted to a journal). arXiv:2102.01633.

12 Jiří Šíma and Pekka Orponen. General-purpose computation with neural networks: A survey of complexity theoretic results. *Neural Comput.*, 15(12):2727–2778, 2003. doi:10.1162/089976603322518731.

13 Jiří Šíma and Martin Plátek. One analog neuron cannot recognize deterministic context-free languages. In *Proceedings of ICONIP 2019, Part III*, LNCS 11955, pages 77–89. Springer, 2019. doi:10.1007/978-3-030-36718-3_7.

14 Tomoyuki Yamakami. Oracle pushdown automata, nondeterministic reducibilities, and the hierarchy over the family of context-free languages. In *Proceedings of SOFSEM 2014*, LNCS 8327, pages 514–525. Springer, 2014. (full version arXiv:1303.1717). doi:10.1007/978-3-319-04298-5_45.

A Informal overview of Lemma 5 and of its proof

Given a DPDA $\mathcal{M}$ accepting a non-regular language $L = \mathcal{L}(\mathcal{M}) = \mathcal{L}(p_0X_0) \subseteq \Sigma^*$, Lemma 5 claims that there is a word $v \in \Sigma^*$ and nonempty words $x, w, y, z \in \Sigma^+$ with the properties depicted in Figure 1, which entail the following conditions:

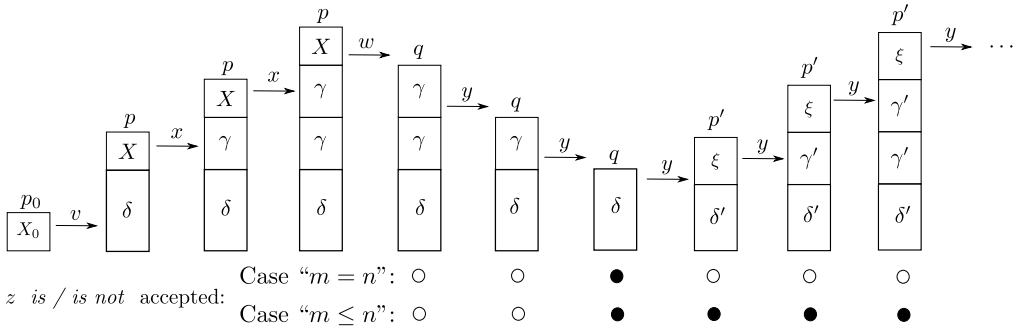

Figure 1 DPDA computation scheme where either $vx^m wy^n z$ is (or is not) accepted iff $m = n$, or $vx^m wy^n z$ is (or is not) accepted iff $m \leq n$.

1. (*the pumping condition*)
   $p_0X_0 \xrightarrow{v} pX\delta \xrightarrow{x^m} pX\gamma^m\delta \xrightarrow{w} q\gamma^m\delta \xrightarrow{y^m} q\delta$ for all $m \geq 0$ (since $pX \xrightarrow{x} pX\gamma$ and $q\gamma \xrightarrow{y} q$); hence $z \in \mathcal{L}(q\delta)$ entails $vx^m wy^m z \in L$ for all $m \geq 0$, and $z \notin \mathcal{L}(q\delta)$ entails $vx^m wy^m z \notin L$ for all $m \geq 0$;
2. (*the prefix condition*)
   the prefix differs from $q\delta$ *on* z in the sense that the languages of all configurations reachable by $vx^m wy^n$ where $m > n$ differ from $\mathcal{L}(q\delta)$ on z; referring to Figure 1, $z \in \mathcal{L}(q\gamma^k\delta) \triangle \mathcal{L}(q\delta)$ for all $k > 0$, where $A \triangle B$ denotes $(A \setminus B) \cup (B \setminus A)$;
3. (*the suffix condition*)
   the suffix (all configurations reachable by $vx^m wy^n$ where $m < n$) *either differs from, or coincides with,* $q\delta$ *on* z; referring to Figure 1, either $z \in \mathcal{L}(q\delta) \triangle \mathcal{L}(p'\xi(\gamma')^k\delta')$ for all $k \geq 0$, or $z \in \mathcal{L}(q\delta) \cap \mathcal{L}(p'\xi(\gamma')^k\delta')$ for all $k \geq 0$.

The prefix condition 2 implies that the stack segment γ is nonempty (while γ' might be empty). The conditions also imply that $q \in \mathrm{DS}(q\gamma)$ and $\mathrm{ES}(q\gamma) = \emptyset$, when we use the following definitions: $\mathrm{DS}(r\alpha) = \{r' \mid r\alpha \xrightarrow{u} r' \text{ for some } u \in \Sigma^*\}$ (the "down-states" of $r\alpha$) and $\mathrm{ES}(r\alpha) = \{r' \mid r\alpha \xrightarrow{\varepsilon} r'\}$ (which is either the empty set or a singleton containing the down-state reached by a sequence of ε-poppings from $r\alpha$). Hence $\mathrm{ES}(r\alpha) \subseteq \mathrm{DS}(r\alpha)$, and $\mathrm{ES}(r\alpha) \neq \emptyset$ entails $\mathrm{ES}(r\alpha) = \mathrm{DS}(r\alpha) = \{r'\}$ for some r'. Figure 2 depicts an example of $\mathrm{DS}(pX\gamma^5)$, using the obvious compositional approach based on $\mathrm{DS}(pX)$ and $\mathrm{DS}(q_i\gamma)$ and $\mathrm{ES}(q_i\gamma)$ where $i \in \{1, 2, 3, 4, 5\}$, assuming that the state set of $\mathcal{M}$ is $Q = \{q_1, q_2, q_3, q_4, q_5\}$ and $p = q_2$. (Here the stack is presented horizontally.)

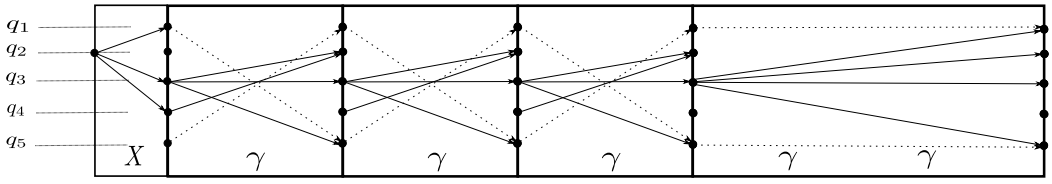

Figure 2 Each directed path from the leftmost black point to the rightmost upper point shows that $q_1 \in \mathrm{DS}(q_2X\gamma^5)$. The completely-dashed paths correspond to ε-sequences; e.g. $\mathrm{ES}(q_5\gamma\gamma) = \{q_5\}$.

Getting tuples (v, x, w, y, z) satisfying the pumping condition 1

Since $L = \mathcal{L}(p_0X_0)$ is not regular, it is clear that from p_0X_0 the computations of $\mathcal{M}$ can reach configurations with arbitrary stack-heights, more precisely configurations with arbitrarily long erasable stack-tops. (The stack-top α in a configuration $p\alpha\beta$ is erasable if $\mathrm{DS}(p\alpha) \neq \emptyset$.)

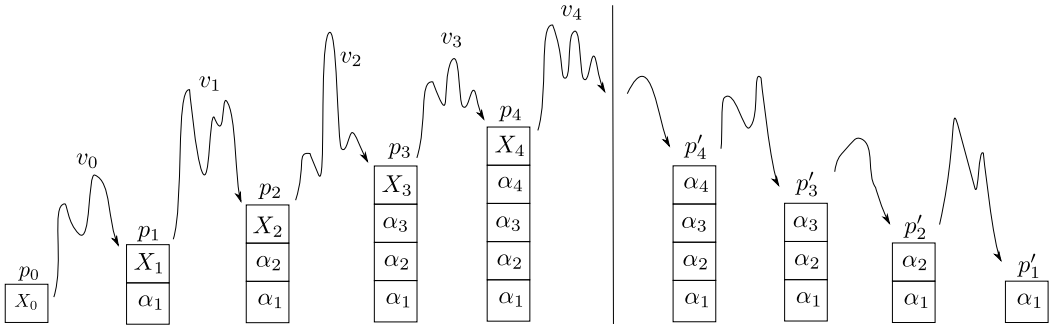

Figure 3 Stair-factorization.

Moreover, such long stack-tops must be also erasable by using many "solid-line" segments that use visible (i.e. non-ε) steps. Indeed: if all possible stack-erasings would in principle go along the dashed lines, i.e. by ε-popping, like from $q_1\gamma^k$ in Figure 2, then this would also entail regularity of L, since even long (erasable) stacks could be replaced with their equivalents of bounded lengths in such cases.

Using the above observations, it is standard to derive the existence of various tuples (v, x, w, y, z) satisfying the pumping condition 1. A crucial fact is that any computation $p_0X_0 \xrightarrow{u} r\alpha$ where the stack-content α is long can be *stair-factorized* into a long sequence of "stairs", as depicted on the left in Figure 3: here $p_iX_i \xrightarrow{v_i} p_{i+1}X_{i+1}\alpha_{i+1}$ and α_{i+1} is a nonempty suffix of the right-hand side of a rewriting rule of $\mathcal{M}$ (for $i = 0, 1, 2, \dots$). If a (long) stack $\alpha_k\alpha_{k-1}\cdots\alpha_1$ is first built and then its (long) top $\alpha_k\alpha_{k-1}\cdots\alpha_{j+1}$ gets erased, we let p'_i denote the state in which $\alpha_i\alpha_{i-1}\cdots\alpha_j\alpha_{j-1}\cdots\alpha_1$ is exposed during this erasing (for $i = k, k-1, \dots, j$); such p'_i are depicted on the right in Figure 3, assuming $j = 1$. By the pigeonhole principle, a triple (p, X, p') repeats in a sufficiently long sequence (p_j, X_j, p'_j), $(p_{j+1}, X_{j+1}, p'_{j+1})$, $\dots$, (p_k, X_k, p'_k), which naturally yields a "pumping tuple" (v, x, w, y, z).

Pumping-operation on (v, x, w, y, z) (preserving the conditions 1, 2, 3 that hold)

Looking at Figure 1, we observe that if the pumping condition 1 holds for a tuple (v, x, w, y, z), then it is preserved by the *pumping-operation* on (v, x, w, y, z) that consists in replacing x and y with their "multiples" x^{k_0} and y^{k_0}, for any $k_0 \geq 1$. Moreover, if (v, x, w, y, z) also happens to satisfy the prefix condition 2, then also this condition is preserved by the pumping-operation (for any $k_0 \geq 1$). The same is true for the suffix condition 3.

Establishing the suffix condition 3 (by the pumping-operation for suitable k_0)

Given (v, x, w, y, z) that satisfies the pumping condition 1, we now show that the pumping-operation (for an appropriate number k_0) establishes the suffix condition 3. First we observe that if $\mathcal{M}$ starts in $q\delta$ and processes $y^\omega = yyy\cdots$, then the respective infinite computation necessarily enters a "cycle" after a "prelude". This is depicted in Figure 1, but there both the prelude and the cycle process the word y. Generally, we would get a prelude $q\delta \xrightarrow{y^{k_1}} p'\xi\delta'$ and a cycle $p'\xi\delta' \xrightarrow{y^{k_2}} p'\xi\gamma'\delta'$ (where $p'\xi \xrightarrow{y^{k_2}} p'\xi\gamma'$ and γ' might be empty) for some numbers k_1, k_2 (where $k_2 > 0$). We now show that the set

$$A = \{\ell \in \mathbb{N} \mid z \in \mathcal{L}(\mathrm{C}_\ell)\} \text{ where } \mathrm{C}_\ell \text{ are the configurations satisfying } q\delta \xrightarrow{y^\ell} \mathrm{C}_\ell \tag{6}$$

is ultimately periodic; i.e., for a *shift* $\mathrm{S} \geq 0$ and a *period* $\mathrm{P} > 0$ we have that for all $\ell \geq \mathrm{S}$ the remainder (ℓ mod P) determines whether or not $\ell \in A$. Generally we cannot simply take $\mathrm{P} = k_2$ as a suitable period, since z might "embark" on popping the γ'-segments along "dashed paths": processing a prefix z_1 of z from $p'\xi\gamma'\gamma'\cdots\gamma'\delta'$ might reach a configuration $q'\gamma'\gamma'\cdots\gamma'\delta'$ like $q_1\gamma\gamma\gamma\cdots$ in Figure 2, in which case we have $q'\gamma'\gamma'\cdots\gamma'\delta' \xrightarrow{\varepsilon} r\delta'$, and it is the state r in which the bottom δ' is reached that determines whether z is accepted or not (i.e., whether $z_2 \in \mathcal{L}(r\delta')$ when $z = z_1z_2$). We thus might need to choose P as a multiple of k_2, guaranteeing that the above mentioned state r (in which δ' is reached) is also repeating with the period P.

Having a shift S and a period P characterizing the ultimate periodicity of the set A defined by (6), we choose $k_0 \geq \mathrm{S}$ that is a multiple of P. Then replacing x and y with x^{k_0} and y^{k_0} indeed guarantees the suffix condition 3; an important point is that the "suffix" might *differ from, or coincide with*, $q\delta$ on z.

Establishing the prefix condition 2

Given a tuple (v, x, w, y, z) satisfying the pumping condition 1, if we aim to establish the prefix condition 2 by the pumping-operation, then it is natural to consider the "prefix-counterpart" of (6), namely the set

$$A' = \{\ell \in \mathbb{N} \mid z \in \mathcal{L}(q\gamma^\ell\delta)\} \tag{7}$$

(recall Figure 1). It is again clear that A' is ultimately periodic, but a problem is that we have to guarantee that the "prefix" has to *differ from* $q\delta$ on z (unlike the "suffix" that can also coincide).

We now show that if A' is nontrivial ($\emptyset \subsetneq A' \subsetneq \mathbb{N}$), then we can establish the prefix condition 2 easily. Let S be the shift and P the period of a presentation of A' as an ultimately periodic set, and let $i_0 \in A'$ and $i_1 \notin A'$. Let $k_0 \geq \max\{i_0, i_1, \mathrm{S}\}$ be a multiple of P, and let $j \in \{0, 1\}$ be such that i_j differs from k_0 on the membership in A'. Instead of (v, x, w, y, z) we now take (v', x', w, y', z) where $v' = vx^{i_j}$, $x' = x^{k_0}$, $y' = y^{k_0}$. Referring to Figure 1, by this change δ is replaced with $\overline{\delta} = \gamma^{i_j}\delta$, and γ is replaced with $\overline{\gamma} = \gamma^{k_0}$. We have $z \in \mathcal{L}(q\overline{\delta}) \triangle \mathcal{L}(q(\overline{\gamma})^k\overline{\delta})$ for all $k > 0$; hence the prefix condition 2 is indeed established.

It remains to explore if we can have the case that for each tuple (v, x, w, y, z) satisfying the pumping condition 1 the "prefix" set A' defined by (7) (when referring to the notation of Figure 1) is trivial. Since we can choose z freely, this case would, in fact, entail that $q\delta \sim q\gamma\delta \sim q\gamma\gamma\delta \sim \cdots$, where $\mathrm{C} \sim \mathrm{C}'$ stands for $\mathcal{L}(\mathrm{C}) = \mathcal{L}(\mathrm{C}')$ for any configurations C, C'. We now show that this case cannot happen since the language $L = \mathcal{L}(q_0X_0)$ is non-regular.

First, it is straightforward to derive that we can fix a *crucial infinite computation of* $\mathcal{M}$ *from* p_0X_0, processing some word $a_1a_2a_3\cdots$, whose stair-factorization has infinitely many stairs and each stair represents its own equivalence class of $\sim$. We can view the left part of Figure 3 as a prefix of this crucial computation; we thus have $p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1 \not\sim p_jX_j\alpha_j\alpha_{j-1}\cdots\alpha_1$ for all $i \neq j$. (The existence of such an infinite computation follows by the fact that the set of left quotients $\{u\backslash L \mid u \in \Sigma^*\}$, where $u\backslash L = \{u' \in \Sigma^* \mid uu' \in L\}$, is infinite since L is non-regular, and by König's lemma since the tree of all computations of $\mathcal{M}$ from p_0X_0 is finitely branching.)

We are not done, since even in this crucial infinite computation (with pairwise non-equivalent stairs) a "pumping" tuple (v, x, w, y, z) derived by the above-mentioned pigeonhole principle might not guarantee that $q\delta \not\sim q\gamma\delta$ (and we might have $q\delta \sim q\gamma\delta \sim q\gamma\gamma\delta \sim \cdots$). For instance, let us assume that in Figure 3 we have $p_1X_1 = p_3X_3$, and that $Q = \{q_1, q_2, q_3\}$ as depicted in Figure 4. We can have $q_2\alpha_1 \sim q_2\alpha_3\alpha_2\alpha_1$ (as denoted by the rectangles in

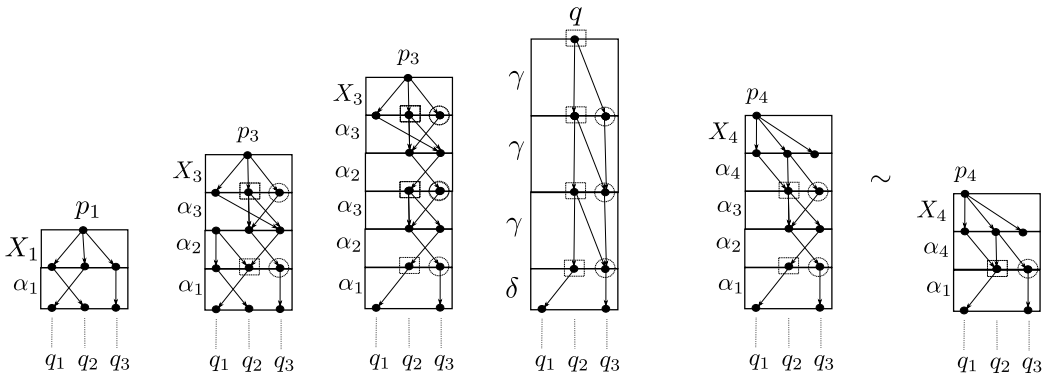

Figure 4 Shortening of configurations (here $p_1X_1 = p_3X_3$).

Figure 4) and $q_3\alpha_1 \sim q_3\alpha_3\alpha_2\alpha_1$ (as denoted by the circles), but $q_1\alpha_1 \not\sim q_1\alpha_3\alpha_2\alpha_1$ (which causes that $p_1X_1\alpha_1 \not\sim p_3X_3\alpha_3\alpha_2\alpha_1$). By putting $pX = p_1X_1 = p_3X_3$, $\delta = \alpha_1$, $x = v_1v_2$ (referring to Figure 3), and $\gamma = \alpha_3\alpha_2$, we get a "pumping" $p_0X_0 \xrightarrow{v_0} pX\delta \xrightarrow{x} pX\gamma\delta \xrightarrow{x} pX\gamma\gamma\delta$ where $pX\gamma\gamma\delta$ is depicted as the third configuration in Figure 4. (We have omitted unreachable "black points".) Here we indeed have $q\delta \sim q\gamma\delta \sim q\gamma\gamma\delta \sim \cdots$, as is highlighted by the fourth configuration in Figure 4.

In our example we can also note that some configurations in the crucial infinite computation might be safely shortened while their equivalence classes are preserved. This is depicted on the right in Figure 4: we have $p_0X_0 \xrightarrow{v_0v_1v_2v_3} p_4X_4\alpha_4\alpha_3\alpha_2\alpha_1$, but we obviously have $p_4X_4\alpha_4\alpha_3\alpha_2\alpha_1 \sim p_4X_4\alpha_4\alpha_1$; this shorter representant of the equivalence class of $p_4X_4\alpha_4\alpha_3\alpha_2\alpha_1$ is reachable by omitting v_1v_2, i.e., $p_0X_0 \xrightarrow{v_0v_3} p_4X_4\alpha_4\alpha_1$.

Nevertheless, the crucial computation visits infinitely many equivalence classes, so the sizes of the stair-configurations $p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1$ must grow above any bound even when we first shorten them maximally in the (repeated) described way. Let us now fix a stair-configuration that has been maximally shortened in the above way and is still sufficiently long. By straightforward combinatorial arguments (that can presented as an application of Ramsey's theorem to avoid tedious technicalities) we can derive that this (shortened) configuration can be written as $\bar{p}\bar{X}\beta\gamma\delta$ where γ is nonempty, and

- γ can be pumped (having the same "lower" p_jX_j and "upper" $p_{j'}X_{j'}$, like $\gamma = \alpha_3\alpha_2$ in Figure 4);
- the sets $\mathrm{DS}(\bar{p}\bar{X}\beta)$ and $\mathrm{DS}(\bar{p}\bar{X}\beta\gamma)$ are equal, further being denoted by $\bar{Q}$ (e.g., in Figure 4 $p_4X_4\alpha_4\alpha_3\alpha_2\alpha_1 = \bar{p}\bar{X}\beta\gamma\delta$ where $\bar{Q} = \{q_2, q_3\}$);
- there is $q' \in \bar{Q}$ for which $\mathcal{L}(q'\gamma\delta)$ and $\mathcal{L}(q'\delta)$ differ on a nonempty word $\bar{z}$ (e.g., now let the circled $q_3\gamma\delta$ and $q_3\delta$ in Figure 4 differ in this way, hence we can choose $q' = q_3$);
- moreover, this $q' \in \bar{Q}$ belongs to $\mathrm{DS}(q\gamma)$ for some self-containing $q \in \bar{Q}$, where $q \in \bar{Q}$ is *self-containing* if $q \in \mathrm{DS}(q\gamma)$ (let $q = q_2$ in our example, though here also q_3 is possible).

It is then clear that $q\gamma \xrightarrow{y} q$ and $q\gamma \xrightarrow{z'} q'$ for some nonempty y, z'; this entails $q\gamma\gamma\delta \not\sim q\gamma\delta$ since they differ on $z = z'\bar{z}$. (This is sufficient for us even if we cannot deduce that $q\gamma\delta \not\sim q\delta$.)

A formal proof is given in the main part of the paper.