Available online at www.sciencedirect.com
ScienceDirect
Information and Computation 205 (2007) 1387–1412
Information and Computation
www.elsevier.com/locate/ic

Lower bound technique for length-reducing automata☆

Tomasz Jurdziński *, Krzysztof Loryś

Institute of Computer Science, Wrocław University, Joliot-Curie 15, PL-50-383 Wrocław, Poland

Received 4 March 2006; revised 12 January 2007
Available online 27 February 2007

Abstract

The class of growing context-sensitive languages (GCSL) was proposed as a naturally defined subclass of context-sensitive languages whose membership problem is solvable in polynomial time. Growing context-sensitive languages and their deterministic counterpart called Church–Rosser languages (CRL) complement the Chomsky hierarchy in a natural way, as the classes filling the gap between context-free languages and context-sensitive languages. They possess characterizations by a natural machine model, length-reducing two-pushdown automata (lrTPDA). We introduce a lower bound technique for lrTPDAs. Using this technique, we prove the conjecture of McNaughton, Narendran and Otto that the set of palindromes is not in CRL. As a consequence we obtain that CFL∩coCFL as well as UCFL∩coUCFL are not included in CRL, where UCFL denotes the class of unambiguous context-free languages; this solves an open problem posed by Beaudry, Holzer, Niemann and Otto. Another corollary is that CRL is a strict subset of GCSL∩coGCSL.

Keywords: Automata and formal languages; Kolmogorov complexity; String rewriting

1. Introduction and related work

Formalisms describing language classes located between context-free languages (CFL) and context-sensitive languages (CSL) have been intensively studied for many years. One of the motivations was to find families with acceptable computational complexity and sufficient expressibility, as well as having natural characterizations by grammars and machine models. Neither CSL nor CFL fulfil these demands. The membership problem of the former is PSPACE-complete which makes it too powerful for practical applications. The latter are not powerful enough to express important syntactical aspects of programming languages.

One of the approaches was to restrict the lengths of context-sensitive derivations linearly [2]. Another direction exploited extensions of machine model characterizations of context-free languages (e.g., LOGCFL; flip-stack

☆ Partially supported by DFG Grant GO 493/1-1 and Komitet Badań Naukowych, Grant 8 T11C 04419. Part of this research has been done while the first author was at Institute of Computer Science, Chemnitz University of Technology, Germany. An extended abstract of this paper appeared in the ICALP02 Proceedings [17].

* Corresponding author.
E-mail addresses: tju@ii.uni.wroc.pl (T. Jurdziński), lorys@ii.uni.wroc.pl (K. Loryś)

0890-5401/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2007.02.003

pushdown automata [13,28]), or extensions of context-free grammars (e.g., parallel communicating grammars [8], conjunctive and boolean grammars [26,27]). Moreover, restrictions of the machine characterizations of CSL were also considered [12]. Very few of these models enjoy natural characterizations by machine models and grammar like structures, while having acceptable complexity.

One of the most interesting proposals was presented by Dahlhaus and Warmuth [9]. They considered grammars with strictly growing rules, i.e., such that the right-hand side of the production is longer than the left-hand one. The resulting class of growing context-sensitive languages (GCSL) complements the Chomsky hierarchy in a natural way [20] and it enjoys several good properties, justifying exploration of this class [30]. It is easy to see that GCSL is contained in NSPACE(n), because the length of every derivation in a growing context-sensitive grammar is linearly bounded. Dahlhaus and Warmuth showed a surprising result that each language generated by a growing context-sensitive grammar is in LOGCFL and hence can be recognized in deterministic polynomial time. Buntrock and Loryś [4,5] showed that this class has many closure properties and it is an abstract family of languages. They also proved that a larger class of grammars defines GCSL. Niemann and Woinowski showed another characterization of GCSL, by acyclic context-sensitive grammars [25].

Buntrock and Otto [6,3] gave a characterization of GCSL by a nondeterministic machine model, *shrinking two-pushdown automata* (sTPDA). Unexpectedly, it turned out that the class of languages recognized by deterministic sTPDAs is equal to the class of Church–Rosser languages (CRL), which were introduced by McNaughton et al. [21] as languages defined by finite, length-reducing and confluent string-rewriting systems [6,24,22]. Moreover, Niemann and Otto [24,22] showed the equivalence of the language classes defined by sTPDAs and by length-reducing two-pushdown automata. Recently, Holzer and Otto considered generalizations of these models and they related them to other complexity and formal language classes [14].

The class of Church–Rosser Languages contains many important languages that are not context-free, while its membership problem has linear time complexity. Moreover, CRL is a strict superset of DCFL, but its definition is more intuitive than that of DCFL [30]. Recently, some new properties of CRL were shown, which justified the applications of this class in parser construction [29]. Moreover, each language in CRL can be defined by a rewriting system in an elegant "normal form" [30]. However, Church–Rosser Languages seem to have a weakness. Already McNaughton et al. [21] conjectured that CRL does not include all context-free languages. They also stated the conjecture that a very simple context-free language of palindromes is not in CRL. Now, it is known that Church–Rosser languages do not contain context-free languages, because CRL is closed under complement, while CFL is not and there are context-free languages whose complements are not even growing context-sensitive [6]. However, the question about palindromes was restated in a more general form by Beaudry et al. [1] who posed the question whether the set of unambiguous context-free languages (UCFL) is included in CRL. Observe that the complement of the language of palindromes is in GCSL too, so the techniques used hitherto seem to be insufficient for proving the conjecture that the language of palindromes is not in CRL.

2. Our result

Most of the known lower bounds for Church–Rosser languages were obtained by the fact that the complements of some fixed Church–Rosser languages are not included in a larger class. We introduce a lower bound technique tailored to length-reducing two-pushdown automata. It exploits the notion of computation graphs, cut and paste technique and the incompressibility method (Kolmogorov complexity arguments) [18].

Specifically, we show that the set of palindromes of even lengths is not a Church–Rosser language, proving the conjecture stated in [21]. By w^R we denote the reversal of the word w, i.e., if $w = w_1 w_2 \dots w_m$, then $w^R = w_m w_{m-1} \dots w_2 w_1$, where $w_i \in \Sigma$, for $i \in \{1, \dots, m\}$, and Σ is an alphabet of constant size.

Theorem 1. *The language* $PAL = \{ww^R : w \in \{0,1\}^*\}$ *does not belong to CRL.*

We show that a length-reducing deterministic two-pushdown automaton (lrDTPDA) working on words from a certain set of palindromes has to satisfy two contradictory conditions. On one hand it has to move the contents

of one pushdown to the other very often. On the other hand it must not lose too much information about any part of the input. Satisfaction of those two requirements simultaneously is impossible for a length-reducing automaton.

As a technical tool, we use a kind of pumping lemma. Note that even Church–Rosser languages over one-letter alphabet need not to be semilinear (e.g., $\{a^{2^n} | n \in \mathbb{N}\} \in$ CRL, [21]), so a pumping technique has to be used in a rather sophisticated way.

Observe that PAL, as well as coPAL, are unambiguous context-free languages, that is, they belong to UCFL (see [11]). Therefore, as a corollary from Theorem 1, we obtain the following result answering the open question in [1].

Corollary 2. *The classes CFL∩coCFL and UCFL∩coUCFL are not included in CRL.*

Note that this is quite a tight separation. Indeed, there exists a strict hierarchy of context-free languages with respect to the degree of ambiguity [11] and the unambiguous languages define its last but one level (see Theorem 7.3.1 in [11]). The lowest level of this hierarchy is equal to the class of deterministic context-free languages which is strictly included in CRL.

The remaining part of the paper describes the proof of Theorem 1 and techniques developed to this aim. In Section 3 we introduce some basic notions and definitions. Next, in Section 4, we present a high level description of the strategy of the proof. Sections 5, 6, 7 and 8 introduce formal methodology used in our proof. Section 9 describes the proof of Theorem 1. Finally, in Section 10, we summarize our results.

3. Preliminaries

Throughout the paper ε denotes the empty word, $\mathbb{N}$, $\mathbb{N}_+$ denote the set of non-negative and positive integers. For a word x, let $|x|$, $x[i]$ and $x[i,j]$ denote the length of x, the ith symbol of x and the factor $x[i]\dots x[j]$ respectively, where $0 < i \leqslant j \leqslant |x|$. Let $[i,j] = \{l \in \mathbb{N} \mid i \leqslant l \leqslant j\}$. Let x^R denote the reverse of the word x, that is $x^R = x[n]x[n-1]\dots x[2]x[1]$ for $|x| = n$. Moreover, let $|x|$ be the length of x if x is a path in a graph (i.e., the number of edges in the path), and the number of its elements if x is a set. For a function $f : X \to Y$ and a sequence $x = x_1 \dots x_n$ of elements of X, by $f(x)$ we denote the sequence $f(x_1) f(x_2) \dots f(x_n)$.

In the following, the lower case letters $a, b, \dots, p$ and r, s denote natural numbers. The letter q usually denotes states of automata. The lower case letters u, v, w, x, y, z denote words over finite alphabets. The upper case letter C is used to denote configurations, the letters G, H, J denote so called computation graphs. Vertices of graphs are denoted by π, ρ, μ, paths by σ.

Growing context-sensitive languages are basically defined by growing grammars [9] and Church–Rosser languages are defined by finite, length-reducing and confluent string-rewriting systems [21]. We do not make use of these characterizations, so we omit the formal definitions. We describe characterizations of these classes by length-reducing two-pushdown automata.

Definition 3. A **two-pushdown automaton** (TPDA) $M = (Q, \Sigma, \Gamma, q_0, \bot, F, \delta)$ with a window of length $k = 2j$ is a nondeterministic automaton with two pushdown stores (and no input tape). It is defined by the set of states Q, the input alphabet Σ, the tape alphabet Γ ($\Sigma \subseteq \Gamma$), the initial state $q_0 \in Q$, the bottom marker of the pushdown stores $\bot \in \Gamma \setminus \Sigma$, the set of accepting states $F \subseteq Q$ and the transition relation

$$\delta : Q \times \Gamma_{\bot,j} \times \Gamma_{j,\bot} \to \mathcal{P}(Q \times \Gamma^* \times \Gamma^*),$$

where $\Gamma_{\bot,j} = \Gamma^j \cup \{\bot\, v : |v| \leqslant j-1, v \in \Gamma^*\}$, $\Gamma_{j,\bot} = \Gamma^j \cup \{v\, \bot : |v| \leqslant j-1, v \in \Gamma^*\}$, and $\mathcal{P}(Q \times \Gamma^* \times \Gamma^*)$ denotes the set of finite subsets of $Q \times \Gamma^* \times \Gamma^*$. The automaton M is a *deterministic* two-pushdown automaton (DTPDA) if δ is a (partial) function from $Q \times \Gamma_{\bot,j} \times \Gamma_{j,\bot}$ into $Q \times \Gamma^* \times \Gamma^*$. A (D)TPDA is called *length-reducing* (lr(D)TPDA) if $(p, u', v') \in \delta(q, u, v)$ implies $|u'v'| < |uv|$, for all $q \in Q$, $u \in \Gamma_{\bot,j}$, and $v \in \Gamma_{j,\bot}$.

A *configuration* of a (deterministic) two-pushdown automaton M is described by a word uq_iv^R, where q_i is the current state, $u \in \Gamma^*$ is the contents of the first pushdown store and $v \in \Gamma^*$ is the contents of the second pushdown store. Both u and v have the bottom of the pusdown store at the first position and the top at the end,

so the bottom marker $\perp$ occurs on both ends of the word uq_iv^R. As we denote configurations by strings, we can also describe transitions defined by the function δ in this way. A transition $\delta(q,u,v) = (q',u',v')$ is described equivalently as $uqv^R \to u'q'(v')^R$. We define a single step computation relation $\vdash_M$ on configurations in a natural way, i.e.,

$$uzqxv \vdash_M uz'q'x'v \quad \text{if} \quad zqx \to z'q'x'.$$

Observe that the window of M in a configuration uqv ($q \in Q$) contains at most k symbols from the neighborhood of q. Furthermore, $\vdash_M^*$ is a computation relation, defined as a transitive closure of $\vdash_M$.

For an input word $x \in \Sigma^*$, the corresponding *initial configuration* is $\perp\ q_0x\ \perp$, i.e., the input word is given as the contents of the second pushdown store. Recall that the automaton does not contain a read only input tape. The automaton M *finishes* its computation by empty pushdown stores. In particular, $L(M) = \{x \in \Sigma^* : \exists_{q\in F}\ \perp\ q_0x\ \perp \vdash_M^*\ q\}$, where $L(M)$ is the language accepted by M.

We also require that the special symbol $\perp$ can only occur on bottoms of the pushdowns and no other symbol can occur on the bottom. In particular, the symbol $\perp$ may be removed from the pushdown only in the last step of the computation. Further, we assume that each transition $uqv \to u'q'v'$ reduces the length by exactly one, i.e., $|uv| = |u'v'| + 1$. One can show by standard techniques that this assumption does not make the model weaker.

The classes GCSL and CRL can be characterized by two-pushdown length-reducing automata.

Theorem 4 ([22,24]). *A language is accepted by a length-reducing TPDA if and only if it is a growing context-sensitive language*.

Theorem 5 ([22,24]). *A language is accepted by a length-reducing DTPDA if and only if it is a Church–Rosser language*.

If not stated otherwise, in the following n will denote the length of an input word and k will denote the size of the window of the analyzed TPDA.

4. High level description of the proof strategy

A main intuition justifying the conjecture that palindromes are not in CRL is based on the following observation. Assume that we check if w is a palindrome by comparing consecutive "symmetric" positions starting from $w[1]$ and $w[n]$, $w[2]$ and $w[n-1]$, and so on. Then each comparison forces a length-reducing DTPDA to move its window through the whole word. Each step reduces the length of configuration, so when moving through the whole word, we shorten the configuration linearly. Thus, after a logarithmic number of comparisons, we lose almost all information about the input word. Another strategy would be to check consecutive symmetric positions starting from the "center" of the input word. But, as the automaton is deterministic, it is not able to detect the "center" of the input word without destroying its content. A position is a good candidate for being the center (when the input is in fact a palindrome) if the symmetric positions of the input around this "candidate" are equal. However, in order to verify such a candidate, one has to remove consecutive symbols (as each step reduces the length), losing information about the neighborhood of that candidate. Then, if the candidate is wrong (i.e., it is not the center), we will be unable to check other candidates.

With these observations we analyze computations of a lrDTPDA M on inputs from the family $(ww^R)^*$, where the shortest description of w (i.e., its Kolmogorov complexity) is much larger than the size of the automaton M. On the basis of properties of computations on these inputs, we construct computations on other inputs using cut and paste technique and pumping. Note that each input from the family $(ww^R)^*$ is a palindrome. However, the size of w makes it impossible to detect a periodic structure. On the other hand, the input word contains many positions that are candidates for the center (positions on "borders" between w and w^R or w^R and w). When we compare symmetric positions starting from such a "center candidate", we do not detect quickly that our candidate is wrong, losing information about the contents of the subword checked during this process.

Our formal analysis proceeds by partitioning computations into stages. One stage starts when one of the pushdowns is (almost) empty and it finishes when the opposite pushdown is (almost) empty. Note that the configuration at the end of such a stage is linearly shorter than the configuration at the beginning of the stage. The automaton is not allowed to "remove" information about the input during the first stage, because it does not even "know" the length of the input word. Recall that incompressibility of w and the choice of its length unable M to detect the periodic structure of the input. Therefore, these conditions force the automaton to leave the structure of the pushdowns almost as it was in the initial configuration. In this way, the configuration at the end of the first stage is also (almost) periodic. This periodicity makes it impossible to "detect" the center of the input word in the second stage and it forces the automaton to work "similarly" as in the first stage. Applying this strategy to consecutive stages (that shorten the configuration linearly) we finally get a short configuration which does not store information about the whole input. Simultaneously, no progress in the process of checking if the input word is a palindrome is done during these stages.

A formal proof following this strategy is presented in Section 9. It uses the methodology introduced in Sections 5, 6, 7 and 8. In these sections we formulate conditions which allow us to use a cut and paste technique and pumping for computations of lrTPDAs. We expect that these techniques, combined with the incompressibility method [18], are applicable to a broader class of problems concerning language classes related to lrTPDAs or similar machine models.

5. Computation graphs and their properties

We introduce the notion of computation graph, similar to derivation graphs from [9] and [19]. Each computation of a length-reducing two-pushdown automaton $M = (Q, \Sigma, \Gamma, q_0, \bot, F, \delta)$ corresponds to a planar directed acyclic graph. Vertices in such a graph are labeled with symbols, transitions and states used during the computation. Let $\omega(\pi)$ denote the label of the vertex π, where ω is a function from the set of vertices of the graph to $\Gamma \cup Q \cup \delta$. Vertices labeled with symbols, states and transitions are called *symbol vertices*, *state vertices* and *transition vertices*, respectively.

Let C_0 be an initial configuration of an lrTPDA. A *computation graph* $G_{(j)} = (V_j, E_j)$ corresponding to the computation $C_0 \vdash C_1 \vdash \cdots \vdash C_j$ is defined inductively (see Fig. 1 and Example 1):

$j = 0$: Let $C_0 = \bot\, q_0 x_1 x_2 \ldots x_n\, \bot$, where $x_i \in \Sigma$ for $i \in [1, n]$. Then, the set of vertices of $G_{(0)} = (V_0, E_0)$ is equal to $\{\rho_{-2}, \ldots, \rho_{n+2}\}$ such that $\omega(\rho_i) = x_i$ for $1 \leqslant i \leqslant n$, $\omega(\rho_i) = \bot$ for $i \in \{-2, -1, n+1, n+2\}$ and $\omega(\rho_0) = q_0$. The graph $G_{(0)}$ has no edges, i.e., $E_0 = \emptyset$.

$j > 0$: Assume that the computation $C_0 \vdash C_1 \vdash_M \cdots \vdash_M C_{j-1}$ corresponds to the graph $G_{(j-1)}$, and the transition $z \to z'$ is executed in C_{j-1} for $z, z' \in \Gamma^* Q \Gamma^*$, i.e.,

$$C_{j-1} = y_1 z y_2 \vdash_M y_1 z' y_2 = C_j.$$

Let $z = z_1 \ldots z_p$ and $z' = z'_1 \ldots z'_{p'}$ where $z_i, z'_i \in Q \cup \Gamma$ for each i. The graph $G_{(j)}$ is constructed from $G_{(j-1)}$ by adding:

- the vertices $\pi'_1, \ldots, \pi'_{p'}$ labeled with $z'[1], \ldots, z'[p']$; they correspond to the word z';
- the vertex D_j which corresponds to the transition $z \to z'$; D_j is labeled with the transition $z \to z'$;
- the edges $(\pi_1, D_j), \ldots, (\pi_p, D_j)$, where the vertices $\pi_1, \ldots, \pi_p$ are labeled with $z[1], \ldots, z[p]$ and they correspond to the rewritten word z;
- the edges $(D_j, \pi'_1), \ldots, (D_j, \pi'_{p'})$.

Computation graphs are planar, what follows from the fact that only sinks are connected to the new transition vertex in each step. There is a natural left to right order among the sources of the computation graph, induced by left to right order of positions of symbols into the initial configuration. For two vertices π_1 and π_2, $\pi_1 \prec \pi_2$ denotes that π_1 precedes π_2 according to this order. Similarly, there is a natural left to right order among the parents and the children of each transition vertex. Moreover, the children of a transition vertex D inherit the

position among other sinks from the parents of D. That is, if the sinks $\pi_1 \prec \ldots \prec \pi_p$ of $G_{(j-1)}$ become internal vertices of $G_{(j)}$ such that $G_{(j-1)} \vdash G_{(j)}$ and they are "replaced" with $\pi'_1 \prec \ldots \prec \pi'_{p'}$ in $G_{(j)}$, then $\pi \prec \pi'_1$ for each sink π of $G_{(j-1)}$ such that $\pi \prec \pi_1$ and $\pi'_{p'} \prec \pi$ for each sink π of $G_{(j-1)}$ such that $\pi_p \prec \pi$. Observe that this ordering induces a natural left to right order among the sinks of the computation graph, adequate to the order of symbols in the associated configuration.

Note that *sources* (vertices with no incoming edges) of $G_{(j)}$ correspond to the initial configuration C_0, the sequence of their labels will be denoted as $Src(G_{(j)})$. Similarly, *sinks* (vertices with no outcoming edges) of $G_{(j)}$ correspond to the last configuration described by $G_{(j)}$ (i.e., C_j) and the sequence of their labels is denoted as $Snk(G)$. However, ρ_{-2} and ρ_{n+2} are artificial vertices introduced for technical reasons. They do not correspond to any symbol nor state of configurations and hence they remain as sources and sinks in all graphs $G_{(j)}$. Thus $\perp C_0 \perp = Src(G_{(j)})$ and $\perp C_j \perp = Snk(G_{(j)})$. For a configuration C_j described by the sinks of $G_{(j)}$ we talk that it is the configuration *associated* with $G_{(j)}$. (Note that a particular configuration may be associated with many graphs, describing different computations with the same last configuration.)

Let π_1, π_2 be vertices such that (π_1, π_2) is an edge in a graph. Then π_1 is called the *parent* of π_2 and π_2 is called the *child* of π_1.

We extend the single step transition relation $\vdash_M$ to computation graphs: $G \vdash_M G'$ if there exist the configurations C, C' and C_0 such that G corresponds to the computation $C_0 \vdash_M^* C$, and G' corresponds to the computation $C_0 \vdash^* C \vdash C'$.

Let $init(u)$ for $u \in \Sigma^*$ denote the computation graph corresponding to the initial configuration on the input word u.

In this paper we apply the term *path* exclusively to paths that start in a source vertex and finish in a sink of the graph. In case that the first vertex of a "path" is not a source or the last vertex is not a sink, we talk about a *subpath*. Let σ be a (sub)path in G which starts with a source vertex π' and ends with a vertex π. We say that σ is ***short*** if there is no (sub)path in G which starts with a source vertex, ends with π and its lengths is smaller than the length of σ.

The relation $\prec$ among vertices of the graph induces a left-to-right partial order of paths. A path σ_1 is to the left of a path σ_2 iff none of vertices of σ_1 is to the right of any vertex of σ_2. Note that σ_1 and σ_2 may have common vertices, however it is not allowed that σ_1 contains a child of a common vertex π that is to the right of a child of π that belongs to σ_2. A path σ is the *leftmost* (*rightmost*) path in a set of paths S if it is to the left (right) of every path $\sigma' \in S$. (The uniqueness of a leftmost/rightmost path follows from the fact that $\prec$ is antisymmetric.)

Let us fix a length-reducing (deterministic) two-pushdown automaton $M = (Q, \Sigma, \Gamma, q_0, \perp, F, \delta)$ with the window of length k. The remaining part of this section concerns properties of computation graphs corresponding to computations of the automaton M.

Definition 6. The ***height*** of a vertex π in a computation graph G is equal to the number of transition vertices in a short (sub)path σ which ends with π.

We say that a vertex π is an *i*-***successor*** for $i \in [-2, n+2]$ if one of short (sub)paths which end in π starts with ρ_i, where $\rho_{-2}, \ldots, \rho_{n+2}$ are the consecutive source vertices of G.

Let us note here that a sink of a graph may be an i-successor for many is (and at least one). On the other hand, it is possible that no sink is a j-successor for some j (e.g., there is no 2-successor in the graph $G_{(4)}$ in Example 1).

Example 1. The graph $G_{(4)}$ presented at Fig. 1 describes the following computation of an automaton $M = (Q, \Sigma, \Gamma, q_0, \perp, F, \delta)$ on the input word *abcdafahia*:

$$
\begin{array}{lll}
(C_0) & \underline{\perp\, q_0 ab}\, cda\, fahia\, \perp & \vdash_M \\
(C_1) & \underline{\perp\, A q_1 cd}\, a\, fahia\, \perp & \vdash_M \\
(C_2) & \underline{\perp\, A c q_2 a f}\, ahia\, \perp & \vdash_M \\
(C_3) & \perp\, \underline{C A F q_2 a h}\, ia\, \perp & \vdash_M \\
(C_4) & \perp\, \underline{C I q_3 h H}\, ia\, \perp &
\end{array}
$$

As follows from the figure, $k = 4$, where k denotes the size of the window of M. Furthermore, $\{a, b, c, d, f, h, i\} \subseteq \Sigma$, $\{A, C, F, H\} \subseteq \Gamma$. Underlined symbols describe here the contents of the window of M in consecutive

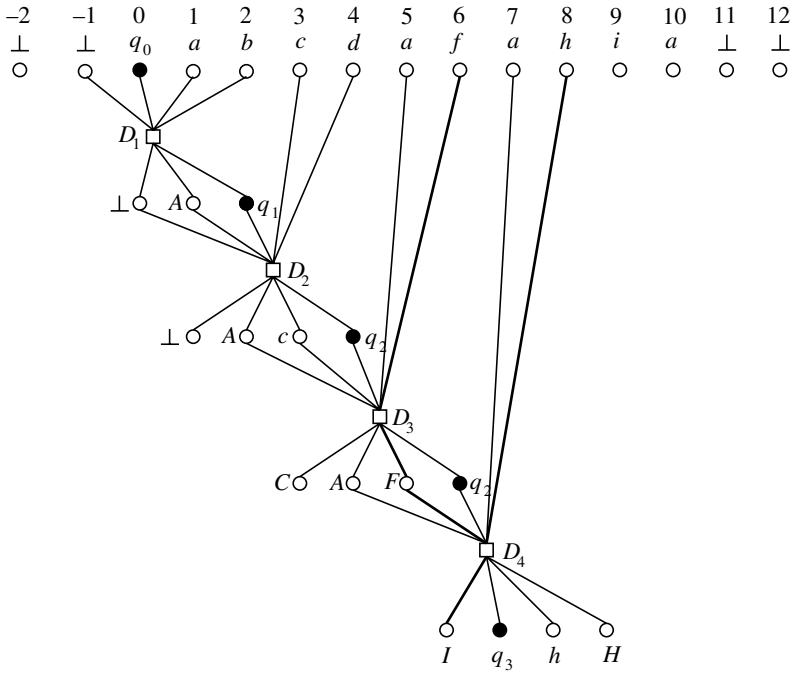


Fig. 1. A computation graph corresponding to the computation $C_0 \vdash_M^* C_4$ (the edges are oriented downwards).

configurations. Note that $Src(G_{(4)}) = \perp C_0 \perp$ and $Snk(G_{(4)}) = \perp C_4 \perp$, that is $Src(G_{(4)})$ is equal to the initial configuration and $Snk(G_{(4)})$ is equal to the final configuration of the computation described by $G_{(4)}$ (with two extra occurences of $\perp$). Transition vertices are denoted at the figure by squares and they have the following labels describing consecutive transitions of the automaton M:

$$\omega(D_1) = \langle \perp\, q_0ab \to \perp\, Aq_1\rangle,$$
$$\omega(D_2) = \langle \perp\, Aq_1cd \to \perp\, Acq_2\rangle,$$
$$\omega(D_3) = \langle Acq_2af \to CAFq_2\rangle,$$
$$\omega(D_4) = \langle AFq_2ah \to Iq_3hH\rangle.$$

State vertices and symbol vertices are denoted by filled and not filled circles respectively, and their labels are given in the figure.

Let σ be the path which starts in the eighth symbol of the input word (ρ_8) and consists of the vertices with labels $(h, \omega(D_4), I)$. Then, σ is short in $G_{(4)}$. Let σ' be the path which starts in the 6th symbol of the input word and consists of the vertices with labels $(f, \omega(D_3), F, \omega(D_4), I)$. The path σ' is not short because it is longer than σ, while the sink of σ' and the sink of σ are equal.

Below, we enumerate some basic properties of computation graphs. Although their proofs are quite simple, we present them for completeness.

Proposition 7. *Let G be a computation graph, let $\rho_{-2}, \ldots, \rho_{n+2}$ be the sources of G, where $\rho_1, \ldots, \rho_n$ correspond to the input word.*

(*a*) *Let π be a vertex in G. Then, there exists $-2 \leqslant i \leqslant n+2$ such that π is an i-successor.*
(*b*) *Let D be a transition vertex in G and let π be a child of D. Then, there exists π', a parent of D such that if π' is an i-successor for $i \in [-2, n+2]$ then π is an i-successor as well.*
(*c*) *Let σ_1, σ_2 be short paths in G and let $\pi_1, \ldots, \pi_p$ be all common vertices of σ_1 and σ_2 written in top-down order. Let σ_i^j be a subpath of σ_i that starts with π_{j-1} and finishes at π_j for $i = 1, 2$ and $j \in [1, p+1]$ (with*

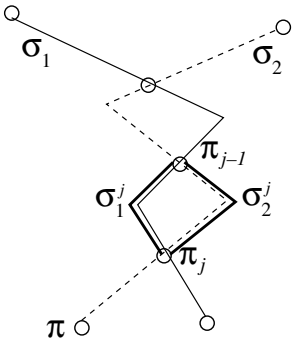


Fig. 2. Intersecting short paths.

two exceptions: σ_i^1 starts in a source of σ_i and σ_i^{p+1} finishes in the sink of σ_i). Then, for every sequence $a_1, \ldots, a_{p+1} \in \{1,2\}^{p+1}$, the path $\sigma_{a_1}^1 \sigma_{a_2}^2 \ldots \sigma_{a_{p+1}}^{p+1}$ is also a short path in G.

(*d*) *Let π, π' be vertices of G such that $\pi \prec \pi'$. Then, there exist $j_1 \leqslant j_2$ such that π is a j_1-successor and π' is a j_2-successor.*

(*e*) *Let σ be a short path in G, $G \vdash_M G'$ and let the sink of σ be also a sink of G'. Then σ is a short path in G' as well.*

Proof.

(a) There exists at least one (sub)path from a source vertex to π, for every vertex π. Let σ be a (sub)path with the minimum length among them, $\rho_i \in \sigma$ for some $-2 \leqslant i \leqslant n+2$. Thus, π is an i-successor.

(b) Observe that D is the only parent of π, by the definition of the computation graph. Let σ be a short path which starts with a source vertex and ends with π. Let π' be the parent of D that belongs to σ and let σ' be the subpath of σ that starts with the source of σ and ends with π'. Then, σ' is a short subpath. (Indeed, otherwise σ would not be a short path.) Thus, the length of each short subpath ending with π' is equal to the length of σ'. And, such a subpath can be extended by D and π, giving a short path. Therefore, if π' is an i-successor for $i \in [-2, n+2]$ then π is an i-successor as well.

(c) It is enough to show that $|\sigma_1^j| = |\sigma_2^j|$ for every $1 \leqslant j \leqslant p$. For the sake of contradiction assume that it is not the case. W.l.o.g. assume $|\sigma_1^j| < |\sigma_2^j|$ for some $1 \leqslant j \leqslant p$. Let π be a sink of σ_2. If we replace σ_2^j with σ_1^j in σ_2 then we obtain a path that ends with π and it is shorter than σ_2 (see Fig. 2). Thus, σ_2 is not a short path with the endpoint in π. Contradiction.

(d) Let j_1 be the minimal value such that π is a j_1-successor. For the sake of contradiction assume that the statement is false. Then, for each j_2, if π' is a j_2-successor then $j_2 < j_1$. Let σ be a short path which starts with ρ_{j_1} and ends with π and let σ' be a short path which starts with ρ_{j_2} and ends with π'. The paths σ and σ' cross, since $\pi \prec \pi'$ and $\rho_{j_2} \prec \rho_{j_1}$. Because of planarity, crossing points are vertices of the graph. Let μ be the first common vertex of σ and σ'. If we replace the subpath of σ' that starts with ρ_{j_2} and finishes at μ with the appropriate subpath of σ, then we obtain a short path with the source in ρ_{j_1} and the sink in π', by the above item (c). Thus, π' is a j_1-successor—contradiction.

(e) Obvious. □

Proposition 7(c) has an important consequence which will be implicitly used in the paper.

Corollary 8. *Let G be a computation graph, let V_{src} and V_{snk} be subsets of the set of sources of G and the set of sinks of G, respectively. Let P be a set of short paths with sources in V_{src} and sinks in V_{snk}. Then, either P is empty or it contains a path which is located to the right/left of all other paths in P.*

The following lemma shows the dependency between the maximal number of sinks of some height and the number of sources (i.e., the length of the input word) in a computation graph.

Lemma 9 (Heights Lemma)**.** *Let $G = (V, E)$ be a computation graph corresponding to a computation on an input word of length $n > 0$. Let p_h be the number of sinks of G with height greater or equal to h. Then $p_h \leqslant 6n(k/(k+1))^h$, where k is the window length of M.*

Proof. Let us assign a weight to each non-transition vertex π of the graph G, denoted $weight(\pi)$. Weights of *source* vertices are equal to 1. The weight of any other non-transition vertex π is equal to $\mathrm{sum}(D_\pi)/\mathrm{out}(D_\pi)$ where D_π is a parent of π and

$$\mathrm{sum}(D_\pi) = \Sigma_{\{\mu:(\mu,D_\pi)\in E\}} weight(\mu), \quad \mathrm{out}(D_\pi) = |\{\mu : (D_\pi, \mu) \in E\}|.$$

In other words, the sum of the weights of the parents of D_π is uniformly distributed among the children of D_π. The following fact can be shown by induction: if $height(\pi) \geqslant i$ then $weight(\pi) \geqslant ((k+1)/k)^i$. One can easily check that this is true for $i = 0$. Assume that the inequality is satisfied for all $j < i$. Let $height(\pi) = i$. Then, for every μ such that $(\mu, D_\pi) \in E$, $height(\mu) \geqslant i - 1$. So, $weight(\mu) \geqslant ((k+1)/k)^{i-1}$ by the induction hypothesis. Let $\mathrm{in}(D_\pi)$ be equal to the in-degree of D_π, i.e., $\mathrm{in}(D_\pi) = |\{\mu : (\mu, D_\pi) \in G\}|$. Then

$$weight(\pi) \geqslant \frac{\mathrm{in}(D_\pi)((k+1)/k)^{i-1}}{\mathrm{out}(D_\pi)} \geqslant \left(\frac{k+1}{k}\right)^i,$$

since $\mathrm{in}(D_\pi)/\mathrm{out}(D_\pi) \geqslant (k+1)/k$ (what follows from the inequality $1 \leqslant \mathrm{out}(D_\pi) < \mathrm{in}(D_\pi) \leqslant k+1$, a consequence of the fact that M is length-reducing and its window length is equal to k). Moreover, the sum of the weights of all sinks is equal to $n + 5$, because this sum does not change after any step of the computation and there are $n + 5$ sources in G . Let P_h be the set of sinks with height greater or equal to h and let $|P_h| = p_h$. Then,

$$p_h \left(\frac{k+1}{k}\right)^h \leqslant \sum_{v\in P_h} weight(v) \leqslant n + 5,$$

so $p_h \leqslant (k/(k+1))^h(n+5) \leqslant 6n(k/(k+1))^h$ for $n \geqslant 1$. □

As a simple consequence of the above lemma, we can bound the maximal height of each vertex of a computation graph.

Corollary 10. *Let G be a computation graph corresponding to the computation of M on an input word of length n. Then, there are no vertices in G with height greater than*

$$\frac{\log(6n)}{\log((k+1)/k)} = O(\log n).$$

By Corollary 10, the length of each short path is $O(\log n)$ where n is the length of the input word.

6. Cut and paste technique and pumping

One of key properties of context-free derivations is the independence of a subderivation starting in a particular nonterminal from the remaining part of the derivation. It allows one to replace such subderivation with another subderivation starting from the same nonterminal, which gives another correct derivation of (probably) another terminal word. Using this fact, one can cut and paste subderivations, obtaining various correct derivations. This property makes it also possible to formulate very powerful pumping lemmas.

These conditions do not hold for growing grammars and for TPDAs. However, dependencies between subcomputations can be described by paths partitioning computation graphs into subgraphs. Therefore, cut and paste and pumping are possible under some conditions.

Definition 11 (*Description of a path*)**.** Let $\sigma = \pi_1, \pi_2, \ldots, \pi_{2l+1}$ be a path in a computation graph $G = (V, E)$. A *description* of the path σ, $\mathrm{desc}(\sigma)$, consists of the sequence $\omega(\pi_1), \ldots, \omega(\pi_{2l+1})$ of labels of the vertices on the

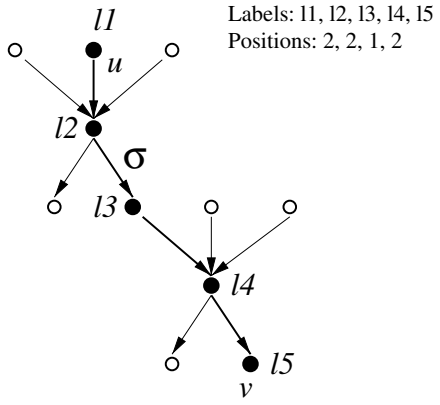


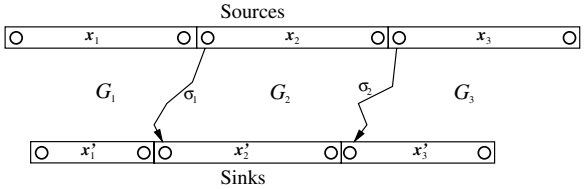
Fig. 3. Description of a path.

path and the sequence $p_1, p_2, \ldots, p_{2l}$ of numbers such that π_{j-1} is the (p_{j-1})st parent of π_j and π_{j+1} is the (p_j)th child of π_j for each even j, where the numbering is according to the left to right order $\prec$ (see Fig. 3 and Example 2). (Note that π_j is a transition vertex for each even j.)
We say that σ is a *γ-path* if $\mathrm{desc}(\sigma) = \gamma$.

Let us recall that each path starts with a source and finishes at a sink, so it induces a natural partition of a computation graph into two subgraphs.

Definition 12 (*Decomposition*)**.** A sequence of descriptions $\gamma_1, \ldots, \gamma_{l-1}$ (for $l > 1$) decomposes a computation graph G into subgraphs $G_1, \ldots, G_l$ if the following conditions are satisfied:

1. There exist paths $\sigma_1, \ldots, \sigma_{l-1}$ in G such that $\mathrm{desc}(\sigma_i) = \gamma_i$ for $i \in [1, l-1]$ and σ_i is located to the left of σ_{i+1} for $i \in [1, l-1]$
2. G_1 is the subgraph of G induced by all vertices located to the left of σ_1 or inside σ_1, G_l is the subgraph of G induced by all vertices located to the right of σ_l or inside σ_l, and G_i for $1 < i < l$ is the subgraph of G induced by all vertices which lay between σ_{i-1} and σ_i, including these paths (i.e., vertices from σ_i belong to G_{i-1} and to G_i).



If there exist descriptions $\gamma_1, \ldots, \gamma_{l-1}$ which decompose a computation graph G into subgraphs $G_1, \ldots, G_l$, G will be denoted by $G_1 \ldots G_l$.

We extend the notions *Src* and *Snk* to subgraphs of computation graphs. For a subgraph H, $Src(H)$ ($Snk(H)$, respectively) is the sequence of labels of all (but the rightmost one, when H is not the rightmost subgraph) sources (sinks, respectively) in H. Obviously, if $G = G_1 \ldots G_l$ then $Src(G) = Src(G_1) \ldots Src(G_l)$ and $Snk(G) = Snk(G_1) \ldots Snk(G_l)$.

Example 2. Let σ be the path in the graph $G_{(4)}$ from Fig. 1 which starts in the 6th symbol of the input word and consists of the vertices with labels $f, \omega(D_3), F, \omega(D_4), I$. The description of σ is equal to this sequence of labels and the sequence $(5, 3, 2, 1)$. The values 3 and 2 say that the vertex with the label F is the 3rd child of D_3

and the 2nd parent of D_4. This path decomposes $G_{(4)}$ into two subgraphs J, J' such that $Src(J) = \bot\bot\ q_0abcda$, $Src(J') = fahia\ \bot\bot$, $Snk(J) = \bot\bot\ C$, $Snk(J') = Iq_3hHia\ \bot\bot$.

Now we show that, under some conditions, one can cut and paste some subgraphs, obtaining in this way new computation graphs. Moreover, we provide conditions under which pumping is possible. In the following we shall write $\langle\tau\rangle_j$ as an abbreviation for $\tau, \ldots, \tau$, where τ is taken j times.

Lemma 13 (Cut and Paste Lemma). *Assume that the descriptions γ_1 and γ_2 decompose a computation graph G into G_1, G_2, G_3 and they decompose a computation graph H into H_1, H_2, H_3. Let $x_i = Src(G_i)$, $y_i = Src(H_i)$, $x'_i = Snk(G_i)$, $y'_i = Snk(H_i)$ for $i = 1, 2, 3$. Then, the graph $J = G_1H_2G_3$ is the computation graph corresponding to the computation $x_1y_2x_3 \vdash^*_M x'_1y'_2x'_3$. Moreover, γ_1, γ_2 decompose J into G_1, H_2, and G_3.*

Lemma 14 (Pumping Lemma). *Assume that descriptions $\langle\gamma\rangle_2$ decompose a computation graph G into G_1, G_2, G_3 and $x_j = Src(G_j), x'_j = Snk(G_j)$ for $j = 1, 2, 3$. Then $J = G_1G_2^iG_3$ for each $i > 0$ is a computation graph associated with the computation $x_1x_2^ix_3 \vdash^*_M x'_1(x'_2)^ix'_3$. Moreover, $\langle\gamma\rangle_{i+1}$ decompose J into $G_1, \langle G_2\rangle_i, G_3$.*

To avoid a tedious analysis while checking whether the resulting graphs still correspond to computations of M, we introduce the notion of correct graph. Let $M = (Q, \Sigma, \Gamma, q_0, \bot, F, \delta)$ be a (D)TPDA. Let $G = (V, E)$ be a graph with a partial order relation $\prec$ in the set V and let $\omega : V \to \Gamma \cup Q \cup \delta$ be a labeling of V. As in a computation graph, vertices of G are called symbol vertices, state vertices and transition vertices (according to the labeling). The graph G is correct with respect to M if it possesses the following properties:

(a) The set S of vertices with no incoming edges has at least five elements and it is linearly ordered: $\rho_{-2} \prec \ldots \prec \rho_{n+2}$, where $S = \{\rho_i\}_{i\in[-2,n+2]}$. Moreover, $\omega(\rho_i) \in \Sigma$ for $1 \leqslant i \leqslant n$, $\omega(\rho_i) = \bot$ for $i \in \{-2, -1, n+1, n+2\}$ and $\omega(\rho_0) = q_0$. In other words, the sources describe the initial configuration of M for the input word $\omega(\rho_1)\ldots\omega(\rho_n)$.

(b) Let D be a transition vertex in G, labeled with a transition $z \to z'$ of M, where $z = z_1\ldots z_p, z' = z'_1\ldots z'_p$ and $z_i, z'_i \in Q \cup \Gamma$ for each i. Then G contains the consecutive (with respect to the order $\prec$) vertices $\pi_1, \ldots, \pi_p$ labeled with $z_1, \ldots, z_p$ and the consecutive vertices $\pi'_1, \ldots, \pi'_{p'}$ labeled with $z'_1, \ldots, z'_{p'}$. The set of all edges incident to D is equal to

$$\{(D, \pi'_1), \ldots, (D, \pi'_{p'})\} \cup \{(\pi_1, D), \ldots, (\pi_p, D)\}.$$

Moreover, for every vertex $\pi \in V$ such that $\pi \prec \pi_1$ ($\pi_p \prec \pi$ respectively) it holds $\pi \prec \pi'_1$ ($\pi'_r \prec \pi$ respectively).

(c) The fan-in and the fan-out of every symbol vertex and every state vertex is at most one. No edges are incident to ρ_{-2} nor ρ_{n+2}.

(d) There are no other edges and vertices aside from the specified above.

(e) There are no cycles in G.

By a simple induction with respect to the number of transition vertices in a graph, we can show the following proposition.

Proposition 15. *Let G be a correct graph with respect to a (deterministic) two-pushdown automaton $M = (Q, \Sigma, \Gamma, q_0, \bot, F, \delta)$. Then, G is a computation graph.*

Proposition 15 substantially simplifies the proofs of Cut and Paste Lemma and Pumping Lemma.

Proof of Lemma 13. First, we show that if G and H are correct graphs then $J = G_1H_2G_3$ is a correct graph as well, where $G_1H_2G_3$ is formed by identifying the vertices on the borders of H_2 with the appropriate vertices of G_1 and G_3. The "local correctness" ((b)–(d) in the definition of correctness) is obviously satisfied for all internal vertices of G_1, H_2 and G_3. The equality of the descriptions of the paths on the borders of decompositions of G and H guarantees that it is also satisfied for all vertices on the borders of the subgraphs G_1, H_2, and G_3 of J. One can easily check that the set of sources of J satisfies the item (a) of the definition of correctness. Finally, there is no cycle in J, since such a cycle would imply that there is also a cycle in G_1 or H_2, or G_3 what cannot happen by correctness of G and H (see Fig. 4).

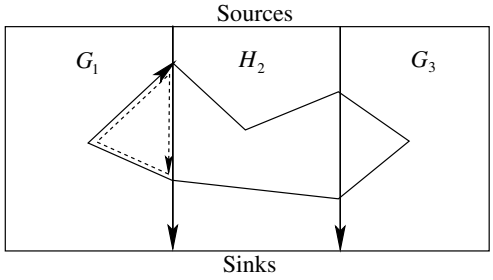


Fig. 4. The cycle in $G_1H_2G_3$ (solid lines) induces cycles in G_1 or H_2 or G_3 (dashed lines).

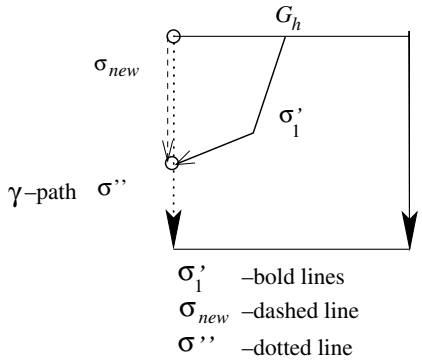


Fig. 5. The subgraph which contains the subpath σ'_1.

We showed that $J = G_1H_2G_3$ is a correct graph. So, by Proposition 15, J is the computation graph corresponding to the computation $x_1y_2x_3 \vdash^*_M x'_1y'_2x'_3$. □

Proof of Lemma 14. We prove the lemma by induction with respect to i. By the assumption, $\langle\gamma\rangle_2$ decompose the computation graph $G_1G_2G_3$ into G_1, G_2, G_3. Now, assume that $G_1G_2^{i-1}G_3$ is a computation graph such that $\langle\gamma\rangle_i$ decompose it into $G_1, \langle G_2\rangle_{i-1}, G_3$. So, in particular, γ decomposes $G_1G_2^{i-1}G_3$ into $G_1G_2^{i-1}$, G_3 and it decomposes $G_1G_2G_3$ into G_1, G_2G_3. Thus, $G_1G_2^iG_3$ is a computation graph (by Lemma 13), which satisfies the conditions stated in Lemma 14. □

Finally, we show a technical and intuitively clear result saying that if one applies Pumping Lemma to a decomposition of a graph based on short paths then the paths defining the decomposition into "pumped" subgraphs are short as well.

Proposition 16. *Assume that $\langle\gamma\rangle_2$ decompose a computation graph G into G_1, G_2, G_3 and γ-paths on the borders of G_2 are short in G. Then, for every $j > 0$ and $i \leqslant j$, the path on the right border of $G_1G_2^i$ is short in the computation graph $G_1G_2^jG_3$.*

Proof. Pumping Lemma implies that $\langle\gamma\rangle_{j+1}$ decompose $G_1G_2^jG_3$ into $G_1, \langle G_2\rangle_j, G_3$ for every $j \in \mathbb{N}$. For the sake of contradiction assume that there exist $j > 0$ and $i \leqslant j$ such that the γ-path σ on the right border of $G_1G_2^i$ is not short in the graph $G = G_1G_2^jG_3$, i.e., there exists a path σ' in G that has the endpoint in the leftmost sink of the suffix $G_2^{j-i}G_3$ of G and the length of σ' is smaller than the length of γ-paths. Let s be the length of a short path σ' in G which satisfies the above conditions. We split σ' into minimal number of subpaths $\sigma'_1, \ldots, \sigma'_p$ such that σ'_i is included in one subgraph of the partition $G_1, \langle G_2\rangle_j, G_3$, and the last vertex of σ'_i is equal to the first vertex of σ'_{i+1} (recall that paths on borders between two subgraphs belong to both incident subgraphs). Let σ' be a path of length s satisfying the above conditions, with the minimal number of such subpaths. Then, σ'_1 starts in a source vertex and it is included in G_h for $h \in \{1, 2, 3\}$. Let σ'' be a γ-path on the border of G_h at which σ'_1 has its last vertex (see Fig. 5). Let σ_{new} be a subpath of σ'' that starts with a source and finishes at the last common

vertex of σ_1' and σ''. Note that the lengths of σ_{new} and σ_1' are equal. Indeed, if σ_{new} is shorter than σ_1' then one may replace σ_1' with σ_{new} obtaining a path which is shorter than σ', so σ' is not short. On the other hand, if σ_1' is shorter than σ_{new} then σ'' is not short in $G_1G_2G_3$. Thus, after replacing the subpath σ_1' of σ' by σ_{new}, we obtain a path with the same sink and the same length as σ'. However, this new path has a partition into smaller number of subpaths (contained in the subgraphs G_l for $l = 1, 2, 3$) than the minimal partition of σ', because σ_{new} may be joined with the second subpath (σ_2') included in the subgraph that is a neighbor of the subgraph G_h containing σ_1'. Contradiction. □

7. Images

In this section we define the notion of image. It allows us to extract information about subwords of the input word that is accessible to the automaton in the configuration described by the sinks of a graph.

We will treat subwords of configurations as representatives of subwords of the input word. Such representatives, or "images", should satisfy some natural properties. In particular, images of disjoint subwords should be (almost) disjoint. The order of images in configurations should agree with the order of appropriate subwords in the input word. Moreover, we expect that one step of computation changes only images which are "involved" in this step.

Such a partition of a configuration into independent parts describing appropriate subwords of the input word is natural in the initial configuration. However, during the computation, distant parts of configurations become dependent on the same subword of the input word. As we have seen in the previous section, one can "extract" these dependencies using descriptions of paths. Moreover, as each sink is also the endpoint of a short path (i.e., a path of logarithmic length, see Corollary 10), such a path does not add large overhead to information stored in the configuration.

It might seem that a good candidate for the image of a given subword of the input word could be a word stored between sinks of short paths starting from the left end and from the right end of this subword, respectively, together with appropriate short paths. However, the following technical difficulties will complicate the formal definition. First, we would like to avoid a situation that images of two disjoint words overlap too much. Second, there might exist source vertices which are not starting points of any short path (for example, the vertex ρ_2 in Fig. 1) or they are starting points of many short paths. Thus, in order to obtain the unique images and to guarantee that the images of disjoint subwords are (almost) disjoint, we do the following. For a given subword of the input word and a computation graph G, we decompose G into subgraphs G_1, G_2, G_3 such that the paths on the borders of G_2 are short and G_2 contains the (vertices of) the considered subword. Moreover, we require that G_2 is the "smallest" subgraph of G determined in the above way. Then, the image is defined by the short paths on the borders of G_2 and the sequence of sinks between them (these sinks describe some subword of the configuration associated with G).

Definition 17. Let G be a computation graph corresponding to the computation of a (deterministic) two-pushdown automaton on an input word of length n. Then, $RL_G(i)$ for $i \in [-1, n+1]$ denotes the rightmost sink vertex in the set $\{\pi \mid \exists_{j \leqslant i} \ \pi \text{ is a } j-\text{successor}\}$. Similarly $LR_G(i)$ denotes the leftmost sink vertex in the set $\{\pi \mid \exists_{j \geqslant i} \ \pi \text{ is a } j-\text{successor}\}$.

Definition 18 (*Image*)**.** Let G be a computation graph corresponding to the computation on an input word of length n. Let $-1 \leqslant l < r \leqslant n+1$, $\pi_l = RL_G(l)$, and $\pi_r = LR_G(r)$. If there is no sink vertex π such that $\pi_l \prec \pi \prec \pi_r$ then the (l, r)-*image* is *undefined* in G. Otherwise, the (l, r)-*image* is *defined* in G and it is equal to $(\sigma_l, \widehat{\sigma_r}, \tau)$, where (see Fig. 6):

- σ_l is the rightmost short path with the sink in π_l and the source at the vertex corresponding to the lth symbol of the input word or to the left of it,
- $\widehat{\sigma_r}$ is the leftmost short path with the sink in π_r and the source at the vertex corresponding to the rth symbol of the input word or to the right of it,
- $\tau = \tau_1\tau_2 \ldots \tau_p$ is the sequence of all sink vertices between π_l and π_r (i.e., $\pi_l = \tau_1 \prec \tau_2 \prec \ldots \prec \tau_p = \pi_r$).

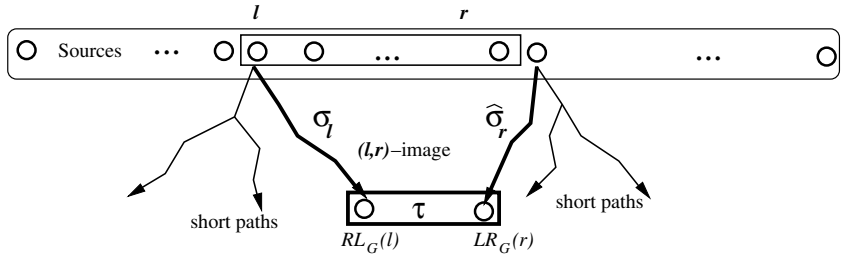


Fig. 6. Image.

Let the *length* (or the size) of the image $(\sigma_l, \widehat{\sigma_r}, \tau)$ be equal to the number of its sinks, i.e., $|\tau|$.
We say that the (l,r)-image in a graph G and the (l',r')-image in a graph G' are *equivalent* if both these images are undefined or they are equal to $(\sigma_l, \widehat{\sigma_r}, \tau)$ and $(\sigma'_l, \widehat{\sigma'_r}, \tau')$ respectively such that $\mathrm{desc}(\sigma_l) = \mathrm{desc}(\sigma'_l)$, $\mathrm{desc}(\widehat{\sigma_r}) = \mathrm{desc}(\widehat{\sigma'_r})$ and $\omega(\tau) = \omega(\tau')$.

Example 3. Let $\rho_{-2}, \ldots, \rho_{12}$ denote the sources of the graph $G_{(4)}$ presented in Fig. 1. The following table describes some images in $G_{(4)}$:

(l,r)	$desc(\sigma_l)$	$desc(\widehat{\sigma_r})$	$\omega(\tau)$	$source(\sigma_l)$	$source(\widehat{\sigma_r})$	def.?
$(3,8)$	$(c,\omega(D_2),\bot)$	$(h,\omega(D_4),I)$	$\bot\ CI$	ρ_3	ρ_8	Y
$(2,6)$	$(\bot)$	$(f,\omega(D_3),C)$	$\bot\bot\ C$	ρ_{-2}	ρ_6	Y
$(3,5)$	$(c,\omega(D_2),\bot)$	$(a,\omega(D_3),C)$	$\bot\ C$	ρ_3	ρ_5	N
$(7,8)$	$(a,\omega(D_4),H)$	$(h,\omega(D_4),I)$	$\emptyset$	ρ_7	ρ_8	N
$(1,9)$	$(\bot)$	(i)	$\bot\bot\ CIq_3hHi$	ρ_{-2}	ρ_9	Y
$(9,11)$	(i)	$(\bot)$	$ia\ \bot$	ρ_9	ρ_{11}	Y

Notice that the $(3,5)$-image is undefined (because there is no sink vertex between sinks of the paths σ_3 and $\widehat{\sigma_5}$). Furthermore, the $(7,8)$-image is undefined (because the sink of σ_7 is to the right of the sink of $\widehat{\sigma_8}$).

Now we can explain the technical reason for which two special artificial vertices ρ_{-2} and ρ_{n+2} are added to each computation graph (see the definition in Section 5). As these vertices have no incident edges in each computation graph, they form two short paths at the left end and the right end of the graph. This ensures that the values $RL_G(i)$ and $LR_G(i)$ are defined for each $i \in [-1, n+1]$, where n is the length of the input word.

Now, we have got the notion which satisfies natural properties stated below. Although these properties are intuitively clear, the formal proofs are presented for completeness.

We say that a vertex π of a computation graph G is (l,r)-*bounded* for $l, r \in \mathbb{N}$ if π is not a j-successor for each $j \leqslant l$ and each $j \geqslant r$.

Proposition 19. *Let G be a computation graph which corresponds to the computation on the input word of length n, let $-1 \leqslant l < r \leqslant n+1$. Then,*

(*a*) *Let π be a sink in G. Then, $RL_G(l) \prec \pi \prec LR_G(r)$ iff π is (l,r)-bounded.*
(*b*) *The (l,r)-image is defined in G iff at least one sink of G is (l,r)-bounded.*
(*c*) *If the (l,r)-image is defined in G then the size of the (l,r)-image is equal to the number of (l,r)-bounded sinks of G plus two.*

Proof. (a) Assume that $RL_G(l) \prec \pi \prec LR_G(r)$ for a sink π of G. Then, according to Definition 17, π is not j-successor for each $j \leqslant l$ and each $j \geqslant r$. That is, π is (l,r)-bounded.

Now, assuming that the relationship $RL_G(l) \prec \pi \prec LR_G(r)$ is not satisfied, we show that π is not (l,r)-bounded. As all sinks are ordered by the relation $\prec$, our assumption implies that $\pi \preceq RL_G(l)$ or $LR_G(r) \preceq \pi$. W.l.o.g., assume that $\pi \preceq RL_G(l)$. Then, Proposition 7(d) implies that π is j-successor for some $j \leqslant l$. So, π is not (l,r)-bounded.

The items (b) and (c) follow directly from (a) and from the definition of the image. □

Proposition 20. *Let G and G' be computation graphs corresponding to the computation on an input word of length n, such that $G \vdash_M G'$. Let $-1 \leqslant l < r \leqslant n+1$. Assume that the (l,r)-image is defined in G and it is equal to $(\sigma_l, \widehat{\sigma_r}, \tau)$. Then,*

(a) *If the window does not contain any vertex from τ in G then the (l,r)-image in G and the (l,r)-image in G' are equal.*
(b) *If $|\tau| > 2k$ then the (l,r)-image is defined in G'.*
(c) *If the (l,r)-image is defined in G' then its length is not smaller than $|\tau| - k$ and not larger than $|\tau| + k$.*
(d) *Assume that the (l',r')-image is defined in G for $r < l' < r' \leqslant n+1$ and it is equal to $(\sigma'_{l'}, \widehat{\sigma'}_{r'}, \tau')$. Then, $|\tau \cap \tau'| \leqslant 2$, i.e., the (l,r)-image and the (l',r')-image overlap by at most two sinks.*
(e) *The paths σ_l and $\widehat{\sigma_r}$ are disjoint.*

Proof.

(a) As the window does not contain any vertex of τ in G, all vertices of τ are the sinks in G'. Moreover, no new path with sinks in τ appears in G'. Finally, each new sink in G' appears either to the left of $\tau[1]$ or to the right of $\tau[|\tau|]$. Thus, each new sink π is a j-successor for $j \leqslant l$ (when $\pi \prec \tau[1]$) or $j \geqslant r$ (when $\tau[|\tau|] \prec \pi$). Therefore, $\tau[1] = RL_{G'}(l)$, $\tau[|\tau|] = LR_{G'}(r)$, the paths σ_l and $\widehat{\sigma_r}$ are the borders of the (l,r)-image in G' (see Proposition 7(e)) and the image remains unchanged.

(b) According to Proposition 19(a), the number of (l,r)-bounded sinks in G is $|\tau| - 2 > 2k - 2 > k$, what follows from the assumption that $k > 2$. As the window size is equal to k, at most k of (l,r)-bounded sinks in G become internal vertices of G'. So, the number of (l,r)-bounded sinks of G' is greater than 0. Thus, the (l,r)-image is defined in G' by Proposition 19(b).

(c) Observe that at most k of (l,r)-bounded sinks in G become internal vertices in G'. Moreover, at most k new sinks appear in G' what implies that there are at most k new (l,r)-bounded sinks in G'. Thus, the claim follows from Proposition 19(a,c).

(d) Assume that τ and τ' overlap by more than two sinks. Then, there exists a vertex μ which is internal (neither first nor last) for both τ and τ'. So, according to Proposition 19(a), μ is (l,r)-bounded and it is (l',r')-bounded. Thus, for each j such that μ is a j-successor, we have $l < j < r$ and $l' < j < r'$. Thus, $l' < r$— contradiction.

(e) For the sake of contradiction assume that σ_l and $\widehat{\sigma_r}$ are not disjoint. By the definition of the image they are short. Let π be a first common vertex of σ_l and $\widehat{\sigma_r}$. If we replace a subpath of σ_l which starts with the source of σ_l and finishes at π by the appropriate subpath of $\widehat{\sigma_r}$ then we obtain a short path (by Proposition 7(c)) with the source in ρ_j for some $j \geqslant r$ and the sink in $\tau[1]$. Thus $\tau[|\tau|]$ is not the leftmost vertex being a j-successor for $j \geqslant r$. □

Proposition 21. *Let G and G' be computation graphs corresponding to the computation on an input word of length n, $G \vdash G'$, and $-1 \leqslant l < r \leqslant n+1$. Assume that the (l,r)-image is undefined in G. Then, the (l,r)-image is undefined in G' as well.*

Proof. As the (l,r)-image is undefined in G, no sink of G is (l,r)-bounded. Let π' be a sink of G' which does not appear in G. According to Proposition 7(b), there is a sink π of G which satisfies the following condition: if π is a j-successor for $j \in [-2, n+2]$ then π' is a j-successor as well. As π is not (l,r)-bounded, π is a j-successor for some $j \leqslant l$ or $j \geqslant r$. Thus, π' is a j-successor for $j \leqslant l$ or $j \geqslant r$, that is π' is not (l,r)-bounded. Therefore, there is no (l,r)-bounded sink in G' and the (l,r)-image is undefined in G', by Proposition 19(b). □

The following lemma says that if we cut and paste subgraphs using short paths then the images of subwords of the input word that were contained in appropriate subgraphs remain unchanged.

Lemma 22. *Assume that descriptions γ_1, γ_2 decompose computation graphs G and H into G_1, G_2, G_3 and H_1, G_2, H_3, respectively. Moreover, assume that the paths on the borders of G_2 are short in G as well as in H. Let $g = |Src(G_1)|$, $h = |Src(H_1)|$. Then the $(g+l, g+r)$-image in G and the $(h+l, h+r)$-image in H are equivalent for every $0 \leqslant l < r \leqslant |Src(G_2)|$.*

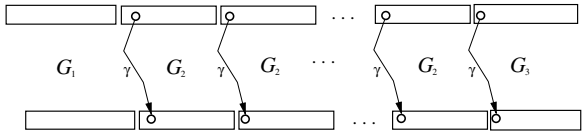


Fig. 7. Periodicity property.

In other words, for any subword of the input word that is included in G_2, its image is independent of the remaining part of the graph and it is included in G_2.

Proof. Let σ_1 and σ_2 be the paths on the borders of G_2 in G. Let $(\sigma_1', \sigma_2', \tau)$ be the $(g+l, g+r)$-image in $G = G_1G_2G_3$. We show that it is contained in the subgraph G_2 and it does not depend on G_1 and G_3, what finishes our proof. First, observe that σ_1' is to the right of σ_1. Indeed, the source of σ_1' is located to the right of the source of σ_1 (or they are equal), moreover σ_1' is the rightmost short path with the appropriate sink—see Proposition 7(c) and the definition of the image. (Note that σ_1 may coincide with σ_1'.) By the same arguments one can show that σ_2' is located to the left of σ_2. Thus, τ is contained in G_2 as well. So, the $(g+l, g+r)$-image is determined only by the subgraph G_2. Similar arguments can be applied when the $(g+l, g+r)$-image is undefined in G. □

8. Periodic computation graphs and computations on periodic inputs

Let $M = (Q, \Sigma, \Gamma, q_0, \bot, F, \delta)$ be a deterministic lrTPDA with the window of size k. All considerations in this section concern computations of this automaton.

As our proof strategy is based on analysis of computations on periodic inputs (see Section 4), we introduce some notions and analyze properties concerning such computations. More precisely, we analyze computations on inputs of the form w^* for $w \in \Sigma^*$, where consecutive (non-overlapping) occurrences of w are called *blocks*. The ith copy of w is called the ith block of the input word. Moreover, the $\lceil p/2 \rceil$-th block of w^p is called the *middle block* of w^p. Finally, let the image of the ith block of w^p in a graph G be the $(m(i-1)+1, mi)$-image in G, where $m = |w|$.

Let G be a computation graph associated with a computation on an input word of the form w^*. We say that paths σ_1, σ_2 are *parallel* in G iff descriptions of σ_1 and σ_2 are equal and the distance between their sources is divisible by $|w|$. In other words, the sources of σ_1 and σ_2 start at the same positions in appropriate blocks.

First, we define the technical notion of *periodicity property* that specifies conditions under which a computation graph is treated as periodic. Let $\mathrm{shift}(u, l)$ denote the left cyclic shift of the word u by l positions, i.e., $\mathrm{shift}(u, l) = u[l+1] \dots u[n]u[1] \dots u[l]$ for the word u of length n.

Definition 23 (*Periodicity property*)**.** Let $G_1G_2G_3$ be a computation graph, let $w \in \Sigma^m$, and $\alpha, r, j \in \mathbb{N}_+$. Moreover, let γ denotes the description of the path on the border between G_1 and G_2. Furthermore, assume that

$$\forall_{i>0} \quad init(w^{\alpha+ir}) \vdash_M^* G_1G_2^iG_3, \text{ and}$$

(a) $m\alpha \leqslant r \leqslant m^j$, α is odd, r is even.
(b) $Src(G_2) = (\mathrm{shift}(w, a))^r$ for some $a \in \mathbb{N}$.
(c) The sequence $\langle\gamma\rangle_{i+1}$ decomposes $G_1G_2^iG_3$ into G_1, $\langle G_2\rangle_i$, G_3 for every $i > 0$.
(d) The γ-path on the right border of $G_1G_2^l$ is short in $G_1G_2^iG_3$ for every $0 \leqslant l \leqslant i$ and $i > 0$.

Then, we say that the word w satisfies the *periodicity property* with respect to the tuple $(G_1, G_2, G_3, \alpha, r, j)$ and the description γ is ***associated*** with the parameter $(G_1, G_2, G_3, \alpha, r, j)$ and the word w.

Let us shortly discuss an intuitive meaning of the above technical definition. The parameter α bounds the size of the subgraphs G_1 and G_3. (Note that $\alpha \cdot m = |Src(G_1)| + |Src(G_3)| - 5$ according to the above definition.)

The item (a) ensures that these nonperiodic parts are much "smaller" than the pumped subgraph G_2. Further, r describes a "level" at which periodicity is still satisfied (r blocks of the input are mapped onto one current block of sinks of G_2). Finally, the parameters α and r should be bounded polynomially with respect to the size of the block, and j specifies the degree of this polynomial. Observe that if the number of blocks is polynomial with respect to m then short paths of a computation graph have logarithmic length with respect to the length of the block, because $O(\log n) = O(\log(poly(m)) = O(\log m)$ (where n is the length of the input word).

By adding the requirement that each block should have the long image, we define *strong periodicity property*.

Definition 24 (*Strong periodicity property*)**.** A word $w \in \Sigma^m$ satisfies the *strong periodicity property* with respect to $(G_1, G_2, G_3, \alpha, r, j)$ if it satisfies the periodicity property with respect to $(G_1, G_2, G_3, \alpha, r, j)$, the image of each block of $w^{\alpha+ir}$ is defined in $G_1G_2^iG_3$ for each $i > 0$, and the length of this image is greater than $c'm$ in $G_1G_2^iG_3$, where $c' = 1/(8\lceil\log(|\Gamma| + |Q|\rceil))$.

By combining the conditions stated in the definition of the periodicity property with properties of images, we show that periodicity of the graph implies periodicity of the images of blocks of the input word.

Proposition 25. *Assume that w satisfies the periodicity property with respect to* $(G_1, G_2, G_3, \alpha, r, j)$, $|w| = m > 4$. *Then*,

(*a*) *The number of sinks of* G_a, $|Snk(G_a)|$, *is not larger than* m^{j+2} *for* $a = 1, 3$.
(*b*) *The lth block of the input* $w^{\alpha+ir}$ *is contained in the infix subgraph* G_2^i *of the computation graph* $G_1G_2^iG_3$ *for every* $i > 0$ *and* $\lceil(\alpha + r)/2\rceil \leqslant l \leqslant \alpha + ir - \lfloor(\alpha + r)/2\rfloor$. *In particular, the middle block of* $w^{\alpha+ir}$ *is included in the infix* G_2^i *for each* $i > 0$.
(*c*) *Assume that the bth block and the* $(b + 2r)$*th block of* $w^{\alpha+ir}$ *are contained in the infix subgraph* G_2^i *of the graph* $G_1G_2^iG_3$ *for* $i > 1$. *Then the images of the blocks b and* $b + 2r$ *in* $G_1G_2^iG_3$ *are equivalent.*
(*d*) *If there exists* $a > 1$ *such that the image of every block of* $w^{\alpha+ar}$ *is (defined and) longer than* $c'm$ *in* $G_1G_2^aG_3$ *then w satisfies the strong periodicity property with respect to* $(G_1, G_2, G_3, \alpha, r, j)$. *The constant* c' *is taken from the definition of the strong periodicity property.*

Proof. Let γ be the description (of a path) associated with $(G_1, G_2, G_3, \alpha, r, j)$.
(a) Let $|w| = m$. Observe that

$$|Snk(G_1)| \leqslant |Snk(G_1G_2G_3)| \leqslant (\alpha + r)|w| + 5 \leqslant 2m^j \cdot m + 5 \leqslant m^{j+2}.$$

The second inequality follows from the fact that M is length-reducing and the number of sources of $G_1G_2G_3$, $|Src(G_1G_2G_3)|$, is equal to the length of the input word plus 5; in the third inequality we use the fact that $\alpha \leqslant r \leqslant m^j$ by the item (a) of Definition 23.
(b) We show that the lth block is located to the right of the prefix G_1 (and one can show in the analogous way that the lth block is located to the left of G_3). Observe that

$$|Src(G_1)| \leqslant |Src(G_1)| + |Src(G_3)| = m\alpha + 5 \leqslant \lfloor(\alpha + r)/2\rfloor \cdot m,$$

while the number of sources located to the left of the lth block is

$$(l - 1)m + 3 > \lfloor(\alpha + r)/2\rfloor \cdot m$$

for each $l \geqslant \lceil(\alpha + r)/2\rceil$ and $m > 4$.
(c) Note that $|Src(G_2)| = r|w|$, so each block that is contained in the infix G_2^i is also contained in at most two consecutive instances of G_2. Assume that the block b of $w^{\alpha+ir}$ is contained in b'th and $(b' + 1)$st instances of G_2. Then the block $b + 2r$ is contained in the $(b' + 2)$th and the $(b' + 3)$th instances of G_2. Moreover, the positions of the blocks b and $b + 2r$ in the appropriate subgraphs G_2G_2 are equal. Thus, as all paths partitioning the graph $G_1G_2^iG_3$ into G_1, $\langle G_2\rangle_i$, G_3 are short by the periodicity property, the image of the block b and the image of the block $b + 2r$ are equivalent, by Lemma 22.

(d) We have to show that the size of the image of each block w of $w^{\alpha+br}$ is greater than $c'm$ in $G_1G_2^bG_3$ for each $b \geqslant 0$. Note that each block w in the graph of the form $G_1G_2^bG_3$ is included in the appropriate subgraph G_1G_2, G_2G_2 or G_2G_3. Indeed, this follows from the fact that G_2 contains at least $2m$ sources by conditions (a) and (b) of the periodicity property. As all paths on the borders of each copy of G_2 are short (by the periodicity property), the image of each block w is determined only by G_1G_2, G_2G_2 or G_2G_3 (Lemma 22). As all such images occur in $G_1G_2^aG_3$ (because of the above item (c) and the fact that $a > 1$), they are longer than $c'm$ by the assumption. Thus, the image of each block is longer than $c'm$ in $G_1G_2^bG_3$. □

Now we concentrate on the analysis of (sub)computations that start in periodic graphs from the family $G_1G_2^*G_3$ and finish after the window moves through the whole periodic infix G_2^*.

Definition 26 (*Opposite border graph*)**.** Assume that a word w satisfies the periodicity property with respect to $(G_1, G_2, G_3, \alpha, r, j)$, the window in $G_1G_2^iG_3$ is located in (the sinks of) the subgraph G_1 (G_3 respectively) for each $i > 0$. The opposite border graph with respect to $G_1G_2^iG_3$ for $i > 0$ is equal to $H(i)$ such that

$$G_1G_2^iG_3 \vdash_M^* G'G_3 \vdash_M H(i)$$

and at least one sink of G_3 (G_1 respectively) becomes an internal vertex in $H(i)$. In other words, $G'G_3$ is the first graph following $G_1G_2^iG_3$ in which the window contains a sink of G_3 (G_1 respectively) and $H(i)$ is the first graph following $G_1G_2^iG_3$ in which some sink(s) of G_3 becomes an internal vertex.

The condition (a) of the definition of periodicity property guarantees that the periodic part of the graph is large with respect to "non-periodic" parts (G_1 and G_3). Especially, if the strong periodicity property is satisfied then the computation that "moves" the window through the whole periodic infix G_2^*, shortens the lengths of the configuration by at least a linear factor of the length of the input word.

Proposition 27. *Assume that a word w satisfies the strong periodicity property with respect to $(G_1, G_2, G_3, \alpha, r, j)$. Let $H(i)$ be the opposite border graph with respect to $G_1G_2^iG_3$ for $i > 0$. Then, the computation*

$$G_1G_2^iG_3 \vdash_M^* H(i)$$

shortens the length of the configuration by at least $n \cdot c'/(8k)$, where n is the length of the input word and c' is the constant which appears in the definition of the strong periodicity property. In other words, the configuration associated with $H(i)$ is shorter than the configuration associated with $G_1G_2^iG_3$ by at least $n \cdot c'/(8k)$ symbols.

Proof. Note that the subgraph G_2^i contains at least $ir - 2 > ir/2$ blocks of the input word (see the item (b) in Definition 23). Thus, by Lemma 22, it contains also the images of these blocks. So, the number of its sinks is not smaller than $(ir/2) \cdot (c'm - 2) > ir \cdot c'm/4$, because the images of two consecutive blocks overlap by at most two sinks (Proposition 20(d)) and the image of each block is not shorter than $c'm$ (by the strong periodicity property). Next, $ir > \alpha$ by the condition (a) of the definition of periodicity property. So, the number of sinks of G_2^i is not smaller than

$$ir \cdot c'm/4 > (\alpha + ir)/2 \cdot c'm/4 = n \cdot c'/8,$$

where $n = (\alpha + ir)m$ is the length of the input word. As the size of the window of M is equal to k, M has to make at least $n \cdot c'/8 \cdot 1/k$ steps in order to move its window through the whole subgraph G_2^i. Each step of the computation reduces the length by at least one, so the whole process shortens the configuration by at least $n \cdot c'/(8k)$ symbols. □

9. Proof of Theorem 1

Our proof exploits the notion of Kolmogorov complexity (cf. [18]). Recall that Kolmogorov complexity of a binary word x, $K(x)$, is equal to the length of the shortest program (written binary) that prints x. We use the

fact that there exist binary words of length n with Kolmogorov complexity greater than $n-1$ for every natural number n. Such words are called *incompressible*.

For the sake of contradiction assume that a length-reducing deterministic two-pushdown automaton $M = (Q, \Sigma, \Gamma, q_0, \bot, F, \delta)$ with the window of size k recognizes the language PAL. We will analyze computations of M on inputs which consist of many repetitions of a block that is a palindrome. However, in order to guarantee that short paths of computation graphs are also short with respect to the length of the block, we assume that the number of blocks is polynomially bounded with respect to the length of the block. Let

$$\mathcal{W}_d = \{(ww^R)^{2j-1} \mid |ww^R| = m,\ K(w) > m/2 - 1,\ 2j-1 \leqslant m^d\}.$$

Note that all the elements of $(ww^R)^*$ are palindromes for every $w \in \{0,1\}^*$. As in the previous section, consecutive copies of ww^R are called *blocks*.

As we mentioned in Section 4, the proof of Theorem 1 goes by partitioning computations into stages. Each stage starts in a periodic graph (satisfying the strong periodicity property) and finishes in the opposite border graph, i.e., after moving the window through the whole periodic part. A key step of the proof is that, under some conditions, the opposite border graph has to satisfy strong periodicity property as well (Periodicity Preserving Lemma). On one hand we shorten substantially the length of the configurations in each stage (as we make many length-reducing steps). On the other hand, the configuration should be "long" at the end of each stage, in order to store long images of all blocks. This gives the intended contradiction.

First, we provide a useful lemma saying that, for inputs from $\mathcal{W}_d$, M should store the "unique" description (i.e., the long image) of every block, as long as the image of the middle block is defined.

Lemma 28 (Middle Block Lemma)**.** *For every $d > 0$, there exists $s_d \in \mathbb{N}_+$ such that for each $x = (ww^R)^{2j-1}$ from $\mathcal{W}_d$ with $|ww^R| > s_d$, the following property holds. If G is a computation graph corresponding to a computation of M on x and the image of the middle block is defined in G then the images of all other blocks are also defined in G and the length of the image of each block (but the middle one, possibly) is greater than $c'm$, where $c' = 1/(8\lceil\log(|\Gamma| + |Q|)\rceil)$ is the constant from the definition of the strong periodicity property.*

The proof of Middle Block Lemma is presented in Section 9.1.

By combining properties of periodic computation graphs described in Section 8 (in particular "parallelity" of images, Proposition 25(c)) with Middle Block Lemma, we show that the strong periodicity property of a graph G implies the strong periodicity property in the opposite border graph with respect to G (i.e., in the graph obtained after moving the window through the whole periodic part of G). However, the "level" of periodicity is dropped down during such stage (see the growth of the last parameter, j, in Lemma 29).

Lemma 29 (Periodicity Preserving Lemma)**.** *For every $j \geqslant 1$, there exists $s'_j \in \mathbb{N}$ which satisfies the following conditions. Let $w \in \Sigma^*$ be an incompressible word such that $2|w| = m > s'_j$. Then, if the word ww^R satisfies the strong periodicity property with respect to $(G_1, G_2, G_3, \alpha, r, j)$ for some graphs G_1, G_2, G_3 and numbers α, r, j then ww^R satisfies the strong periodicity property with respect to $(H_1, H_2, H_3, \alpha', r', 20j)$ such that*:

- $\alpha' = \alpha + gr$, $r' = fr$ *for some* $g, f \in \mathbb{N}$,
- $init((ww^R)^{\alpha'+ir'}) \vdash_M^* G_1 G_2^{g+if} G_3 \vdash_M^* H_1 H_2^i H_3$ *for every* $i > 0$, *where* $H_1 H_2^i H_3$ *is the opposite border graph with respect to* $G_1 G_2^{g+if} G_3$.

The proof of the above lemma is presented in Section 9.2.

Now, we are ready to prove Theorem 1. If we choose a long incompressible word w and appropriate parameters α, r, j, then the word ww^R satisfies the strong periodicity property with respect to $(G_1, G_2, G_3, \alpha, r, j)$, where the graph $G_1G_2G_3$ corresponds to the initial configuration on the input word $(ww^R)^{\alpha+r}$. Then we apply Periodicity Preserving Lemma several times, obtaining consecutive "border" graphs (i.e., graphs describing configurations in which one pushdown is relatively short; interchangeably the left and the right one). These graphs preserve periodic structure, and the image of each block should be longer than $c'm$ in them. On the other hand, M reduces the length of configurations in each step, what in consequence makes it impossible to preserve "long" images of blocks (required by Periodicity Preserving Lemma) and gives contradiction. Below, we present details of this scenario.

Let $l \in \mathbb{N}$ be a constant, let $w \in \Sigma^*$ be an incompressible word such that $|ww^R| = m > s'_{20^{l+1}}$, where s'_j denotes the constant from Periodicity Preserving Lemma. We choose $m > s'_{20^{l+1}}$ in order to apply Periodicity Preserving Lemma l times. Let $\alpha_0 = 3$, $r_0 = 6m$, $j_0 = 20$. Let G be a computation graph corresponding to the initial configuration on the input word $(ww^R)^{r_0+3}$. Let $G_{1,0}, G_{2,0}, G_{3,0}$ be the subgraphs of G corresponding respectively to the first block of $(ww^R)^{r_0+3}$, the next r_0 blocks and the last two blocks. One can easily verify that ww^R satisfies the strong periodicity property with respect to $(G_{1,0}, G_{2,0}, G_{3,0}, \alpha_0, r_0, j_0)$. Now, we apply Periodicity Preserving Lemma l times obtaining $G_{1,i}, G_{2,i}, G_{3,i}, \alpha_i, r_i$ for $i \in [1, l]$ such that ww^R satisfies the strong periodicity property with respect to $(G_{1,i}, G_{2,i}, G_{3,i}, \alpha_i, r_i, 20^{i+1})$, i.e.:

(A) There exists a computation

$$G_{1,0}G_{2,0}^{p_0}G_{3,0} \vdash_M^* G_{1,1}G_{2,1}^{p_1}G_{3,1} \vdash_M^* \cdots \vdash_M^* G_{1,l}G_{2,l}^{p_l}G_{3,l}$$

for each natural number p_l, where $G_{1,0}G_{2,0}^{p_0}G_{3,0}$ is the computation graph associated with the initial configuration on the input word $(ww^R)^{\alpha_l + p_l r_l}$, $p_0 \geqslant \ldots \geqslant p_{l-1} \geqslant p_l$ are natural numbers, such that $p_i = p_{i+1} \cdot \frac{r_{i+1}}{r_i} + \frac{\alpha_{i+1}-\alpha_i}{r_i}$ for $i < l$.

(B) The image of each block is longer than $c'm$ in the graphs $\{G_{1,i}G_{2,i}^{p_i}G_{3,i}\}_{i=0,\ldots,l}$.

(C) $G_{1,i+1}G_{2,i+1}^{p_{i+1}}G_{3,i+1}$ is the opposite border graph with respect to $G_{1,i}G_{2,i}^{p_i}G_{3,i}$ for $i \in [0, l-1]$; i.e., the window moves through the whole periodic infix $G_{2,i}^{p_i}$ during the subcomputation $G_{1,i}G_{2,i}^{p_i}G_{3,i} \vdash_M^* G_{1,i+1}G_{2,i+1}^{p_{i+1}}G_{3,i+1}$ (see Periodicity Preserving Lemma).

The condition (B) implies that the length of the configuration associated with the graph $G_{1,i}G_{2,i}^{p_i}G_{3,i}$ for $i = 0, \ldots, l$ is not smaller than $(\alpha_l + p_l r_l)(c'm/2 - 2)$, because the images of two consecutive blocks overlap by at most two symbols (Proposition 20(d)). So, the length of this configuration is larger than

$$(\alpha_l + p_l r_l)(c'm/2 - 2) > (\alpha_l + p_l r_l) \cdot c'm/4 = n \cdot c'/4,$$

where $n = (\alpha_l + p_l r_l)m$ is the length of the input word. On the other hand, the computation $G_{1,i}G_{2,i}^{p_i}G_{3,i} \vdash_M^* G_{1,i+1}G_{2,i+1}^{p_{i+1}}G_{3,i+1}$ makes the configuration shorter by at least $n \cdot c'/(8k)$ symbols (Proposition 27) for each $i \in [0, l-1]$. As the initial configuration has the length $n + 5 < 2n$, the length of the configuration associated with the graph $G_{1,l}G_{2,l}^{p_l}G_{3,l}$ is not larger than $2n - l \cdot nc'/(8k)$. Thus, $n \cdot c'/4 < 2n - l \cdot n \cdot c'/(8k)$. However, this inequality is false for $l \geqslant 2 \cdot 8k/c'$. In this way we get contradiction, what finishes the proof of Theorem 1.

9.1. Proof of Middle Block Lemma

For the sake of contradiction assume that the lemma is not true. That is, for each $s_d \in \mathbb{N}$, there exists an incompressible word w and a number $j \in \mathbb{N}$ such that

- $|ww^R| = m > s_d$,
- $2j - 1 \leqslant m^d$,
- there exists a graph G that describes a computation on $(ww^R)^{2j-1}$ such that the image of the middle block of $(ww^R)^{2j-1}$ (i.e., the jth block) is defined in G and, for some $i \neq j$, the image of the ith block is undefined or shorter than $c'm$ in G.

Let G be the first graph (with respect to the order induced by the computation relation $\vdash_M$) during the computation on $(ww^R)^{2j-1}$ that satisfies the above conditions for some i. W.l.o.g. assume that $i < j$ and $c's_d > 2k$. Then, the image of the ith block is not shorter than $c'm - k$ in G, since the image of each block has length m in the graph corresponding to the initial configuration, it may be shortened by at most k in one step and it cannot become undefined in one step if it is longer than $2k$ (Proposition 20(b,c)). Simultaneously, if the image of the middle block is defined in G, it has been defined during the whole computation described by G (Proposition 21).

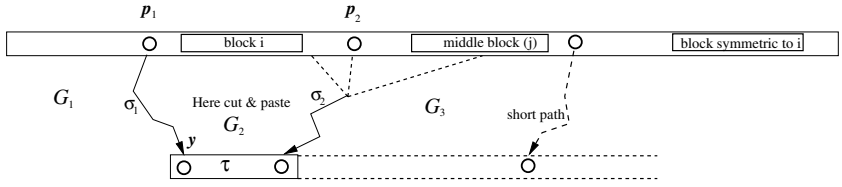


Fig. 8. Middle Block Lemma.

Let $(\sigma_1, \sigma_2, \tau)$ be the image of the ith block in G, let $\gamma_1 = \mathrm{desc}(\sigma_1)$ and $\gamma_2 = \mathrm{desc}(\sigma_2)$. The path σ_1 is located to the left of σ_2 (Proposition 20(e)), so γ_1, γ_2 decompose G into G_1, G_2, G_3 for some subgraphs G_1, G_2, G_3. Let p_1, p_2 be the positions of the sources of σ_1, σ_2 in the input word (see Fig. 8). By the definition of the image, $p_1 \leqslant m(i-1)+1$ and $p_2 \geqslant mi$. We show that $p_2 < mj$ (i.e., p_2 is the position located to the left of the middle block or inside it). Indeed, the image of the middle block is defined, so the rightmost vertex of τ is located to the left of the rightmost vertex of the image of the middle block (otherwise, the images of the ith and the jth block overlap by more than two symbols, contradicting Proposition 20(d)). And, for each $p \geqslant mj$, there is no p-successor located to the left of the rightmost vertex of the image of the jth block (see the definition of the image). So, $p_2 < mj$ and G_2 does not contain any source vertex located to the right of the middle block.

We claim that $\gamma_1, \gamma_2, \omega(\tau), p_1, p_2, j, m$ and M uniquely describe w. We present an algorithm that determines w on the basis of these parameters. For every $v \in \Sigma^{m/2}$ we inspect the computation graphs that describes the computation of M on $y = (vv^R)^{2j-1}$ until we find a word v and a computation graph G' describing the computation on y such that:

- γ_1, γ_2 decompose G' into G'_1, G'_2, G'_3,
- $Snk(G'_2) = \omega(\tau)$,
- p_1 and p_2 are positions of the sources of γ_1 and γ_2, respectively.

Then, $G'' = G_1 G'_2 G_3$ is a computation graph and γ_1, γ_2 decompose G'' into G_1, G'_2, G_3 (see Cut and Paste Lemma and Fig. 8). Observe that the last configurations described by G and G'', i.e., $Snk(G)$ and $Snk(G'')$, are equal. The automaton M accepts starting from the configuration $Snk(G)$, since G describes the computation on the palindrome $(ww^R)^{2j-1}$. So, M accepts also the input word z of the graph G'', obtained from $(ww^R)^{2j-1}$ by replacing its subword $(ww^R)^{2j-1}[p_1, p_2]$ with the appropriate subword of $(vv^R)^{2j-1}$. Thus, z is a palindrome. But $p_2 \leqslant mj$, so all blocks located to the right of the middle block remain unchanged, i.e., they are equal to ww^R. On the other hand $p_1 \leqslant m(i-1)+1$, so the ith block ww^R is replaced with vv^R in z. Thus $v = w$, since z is a palindrome.

Now, we get a contradiction with incompressibility of w. We need only $O(\log n)$ bits in order to store γ_1, γ_2, p_1, p_2, j, m and M, where n is the length of the input word $(ww^R)^{2j-1}$. (It follows from the fact that γ_1, γ_2 describe short paths—see Corollary 10.) Moreover,

$$n = |ww^R|(2j-1) = m(2j-1) \leqslant m^{d+1},$$

since $(ww^R)^{2j-1} \in \mathcal{W}_d$. So, all these data require $O(\log m^{d+1}) = O(\log m)$ bits. Additionally, $|\omega(\tau)| < c'm$ and $\omega(\tau)$ is a word over the alphabet $\Gamma \cup Q$. In order to store it binary we need at most $c'm\lceil \log(|\Gamma \cup Q|)\rceil \leqslant m/8$ bits. Finally, we encode w using $m/8 + O(\log m) < m/4$ bits for m large enough. This contradicts the assumption that w is incompressible, i.e., $K(w) > m/2 - 1$.

9.2. Proof of Periodicity Preserving Lemma

We show that the lemma is satisfied for each s'_j which is larger than the maximum of s_{80j} and a constant that depends only on M, where s_l denotes the constant from Middle Block Lemma for $l \in \mathbb{N}_+$. Assume that the word

ww^R satisfies the strong periodicity property with respect to $(G_1, G_2, G_3, \alpha, r, j)$, where w is an incompressible word of length $m/2$ such that $m > s'_j$. W.l.o.g assume that the window in the configuration associated with $G_1G_2^iG_3$ is located in G_1. Let $H(i)$ denote the opposite border graph with respect to $G_1G_2^iG_3$ (see Definition 26). Note that $H(i)$ exists for every $i > 0$, because each computation of M should finish with empty stacks. So, all symbols corresponding to the sinks of G_3 will finally be removed.

Claim 30. *For every $i > 1$ such that $\alpha + ir \leqslant m^{40j}$, the image of the middle block of $(ww^R)^{\alpha+ir}$ is defined in $H(i)$.*

Proof. For the sake of contradiction, assume that the image of the middle block of $(ww^R)^{\alpha+ir}$ is undefined in $H(i)$ for some $i > 1$ such that $\alpha + ir \leqslant m^{40j}$. Then, there exists a subgraph G' such that

$$init(ww^R)^{\alpha+ir} \vdash_M^* G_1G_2^iG_3 \vdash_M^* G'G_3 \vdash_M^* H(i),$$

where the image of the middle block (i.e., $\lceil(\alpha + ir)/2\rceil$-th block) is defined but shorter than $c'm - k$ in $G'G_3$. Indeed, the size of the image of the middle block is larger than $c'm$ in $G_1G_2^iG_3$, what follows from the strong periodicity property of ww^R with respect to $(G_1, G_2, G_3, \alpha, r, j)$. On the other hand, the image of this block is undefined in $H(i)$. Proposition 20(b,c) implies that the size of the image may change by at most k in one step and the image may not become undefined in one step when its size is larger than $2k$. So, there exists a computation graph $G'G_3$ in the derivation $G_1G_2^iG_3 \vdash^* H(i)$ such that the image of the middle block is defined in $G'G_3$ and its size is smaller than $c'm - k$ (if m is large enough).

Let i' be a natural number such that $\alpha + i'r \leqslant m^{80j}$ and $\lfloor(\alpha + i'r)/2\rfloor > m^{40j+2}$. Then

$$(ww^R)^{\alpha+i'r} \vdash_M^* G_1G_2^iG_2^{i'-i}G_3 \vdash_M^* G'G_2^{i'-i}G_3,$$

where the first subcomputation is obtained by the periodicity property of ww^R with respect to $(G_1, G_2, G_3, \alpha, r, j)$ and the second subcomputation is obtained by Cut and Paste Lemma (see Fig. 9). Moreover,

(a) The middle block of $(ww^R)^{\alpha+i'r}$ is included in the subgraph $G_2^{i'-i}G_3$ of the graph $G_1G_2^{i'}G_3$.
Indeed,

$$m \cdot \lfloor(\alpha + i'r)/2\rfloor - 1 > m^{40j+2} \cdot m > (\alpha + ir)m = |(ww^R)^{\alpha+ir}|,$$

so the number of sources of $G_1G_2^i$ is smaller than the number of sources preceding the middle block of $(ww^R)^{\alpha+i'r}$.

(b) The image of the middle block of $(ww^R)^{\alpha+i'r}$ is defined and it is longer than $c'm$ in $G'G_2^{i'-i}G_3$.
Indeed, it is longer than $c'm$ in $G_1G_2^{i'}G_3$ by the strong periodicity property and it remains unchanged during the subcomputation $G_1G_2^{i'}G_3 \vdash_M^* G'G_2^{i'-i}G_3$ by (a) above and Proposition 20(a).

(c) The image of the block $\lceil(\alpha + ir)/2\rceil$ of $(ww^R)^{\alpha+ir}$ in $G'G_3$ and the image of the block $\lceil(\alpha + ir)/2\rceil$ of $(ww^R)^{\alpha+i'r}$ in $G'G_2^{i'-i}G_3$ are shorter than $c'm - k$.
Indeed, this block is included in the prefix $G_1G_2^i$ of the graph $G_1G_2^iG_3$ (as well as in the prefix $G_1G_2^i$ of $G_1G_2^{i'}G_3$) by Proposition 25(b). And, as the path on the border between $G_1G_2^i$ and G_3 in $G_1G_2^iG_3$ (and the path between $G_1G_2^i$ and $G_2^{i'-i}G_3$ in $G_1G_2^{i'}G_3$) is short, it remains short in $G'G_3$ and in $G'G_2^{i'-i}G_3$ as the path on the right border of G'. Thus, the image of the block $\lceil(\alpha + ir)/2\rceil$ in $G'G_3$ and the image of this block in $G'G_2^{i'-i}G_3$ are equivalent by Lemma 22. So, they are shorter than $c'm - k$ by the contrary assumption.

The conditions (b) and (c) contradict Middle Block Lemma for the graph $G'G_2^{i'-i}G_3$. □

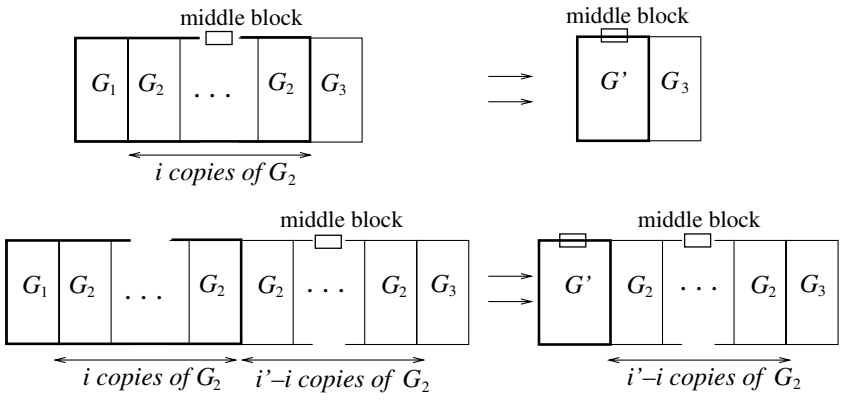


Fig. 9. The images in the periodic graph.

Using the claim above, we show that the graphs $H(i)$ are really periodic. Our argument is based on the fact that the length of the configuration corresponding to $H(i)$, i.e., $Snk(H(i))$ is large since it contains long and "almost disjoint" images of all blocks. (This fact follows from Middle Block Lemma, the above claim and Proposition 20(d).) This allows us to prove (by Lemma 9 and counting arguments) that there are two parallel short paths in $H(i)$ for some i. Then, we apply Pumping Lemma for decomposition of $H(i)$ induced by these paths.

Claim 31. *There exists $i > 1$ such that $\alpha + ir \leqslant m^{5j}$ and the computation graph $H(i)$ contains two parallel short paths σ_1, σ_2 with sinks located to the left of the window in $H(i)$. The distance (the number of sinks) between the sink of σ_1 and the sink of σ_2 is not smaller than k.*

Proof. Let us take i such that $\alpha + ir \leqslant m^{5j}$ and $\alpha + ir > m^{j+2}$ (note that $|Src(G_1)| + |Src(G_2)| \leqslant m^{j+2}$ by the periodicity property). Let $n = m(\alpha + ir)$ denote the length of the input word in the computation described by $H(i)$. The image of the middle block is defined in $H(i)$, by Claim 30. Moreover, we took $m > s_{80j}$, so Middle Block Lemma may be applied for $(ww^R)^{\alpha+ir}$. Thus, $|Snk(H(i))| > (\alpha + ir)c'm/4 = c'n/4$, because the images of all blocks (but the middle one, possibly) are longer than $c'm$ in $H(i)$ and images of disjoint blocks may overlap by at most two sink vertices (Proposition 20(d)). As $H(i)$ corresponds to the first configuration in which some sink of G_3 becomes the internal vertex, there are at most

$$|Snk(G_3)| + k < 2m^{j+2} < 2(\alpha + ir) = 2n/m$$

sinks located to the right of the window in $H(i)$ (Proposition 25(a)). For m large enough (i.e., $m > 64/c'$), this number is smaller than $n \cdot c'/32$.

Let $h = \lceil \log(c'/96)/\log(k/(k+1)) \rceil$. By Lemma 9, the number of sinks of height greater or equal to h is in $H(i)$ less than $6n(k/(k+1))^h \leqslant nc'/16$. So, at least

$$|Snk(H(i))| - \frac{c'}{16}n - \frac{c'}{32}n \geqslant \frac{c'}{4}n - \frac{c'}{16}n - \frac{c'}{32}n \geqslant \frac{c'}{32}n$$

sinks located to the left of the window in $H(i)$ have height less than h. Let S be the set of these sinks. Let us choose the leftmost short path with the sink in s for each $s \in S$. We partition the set of these paths into m groups such that a path starting with the source vertex at the pth position of the input word belongs to the group $p \bmod m$. The largest group contains at least

$$\frac{1}{m} \cdot \frac{c'}{32}n = \frac{c'}{32}\frac{(\alpha + ir)m}{m} > \frac{c'}{32}m^{j+2}$$

paths, where the last inequality follows from the assumption $\alpha + ir > m^{j+2}$. As the length of each path in the group is bounded by the constant h which is independent of m, the number of possible descriptions of these paths

is constant as well. So, there exist at least k paths with the same description in the group of $c'm^{j+2}/32$ paths for m large enough (i.e., m such that $c'm^{j+2}/32$ is at least $2k$ times larger than the number of possible descriptions of paths with the height $\leqslant h$). And, by the choice of the group, these paths are parallel. We choose the leftmost path and the rightmost path among them as σ_1 and σ_2, respectively. □

Now, we are ready to show that

Claim 32. *There exist subgraphs H_1, H_2, H_3 such that the word ww^R satisfies the periodicity property with respect to $(H_1, H_2, H_3, \alpha', r', 20j)$, where $\alpha' = \alpha + gr$, $r' = fr$ for some $g, f \in \mathbb{N}$. Moreover, $H_1H_2^lH_3$ is the opposite border graph with respect to $G_1G_2^{g+lf}G_3$ for every $l > 0$.*

Proof. By Claim 31, there exists i such that $\alpha + ir \leqslant m^{5j}$ and there are two parallel short paths σ_1, σ_2 in $H(i)$, that is $\mathrm{desc}(\sigma_1) = \mathrm{desc}(\sigma_2) = \gamma$. And, the paths σ_1, σ_2 are located to the left of the position of the window. So, γ, γ decompose $H(i)$ into G'_1, G'_2, G'_3, where σ_1 and σ_2 are the paths on the left border and on the right border of G'_2 in $H(i) = G'_1G'_2G'_3$. Thus, by Pumping Lemma, $\langle\gamma\rangle_{j+1}$ decompose the computation graph $G'_1(G'_2)^jG'_3$ into $G'_1, \langle G'_2\rangle_j, G'_3$. Recall that σ_1, σ_2 on the borders of G'_2 are short in $G'_1G'_2G'_3$ (Claim 31). So, by Proposition 16, the paths on the borders of the decomposition $G'_1, \langle G'_2\rangle_j, G'_3$ are short in $G'_1(G'_2)^jG'_3$ as well. Moreover, the parallelity of the paths σ_1, σ_2 implies that $Src(G'_2) = (\mathrm{shift}(ww^R, p))^a$ for some $p < m$ and $a \leqslant \alpha + ir \leqslant m^{5j}$. So, the initial configuration of $G'_1(G'2)^lG'_3$ corresponds to the input $(ww^R)^{(\alpha+ir)+a(l-1)}$ (because $Src(G'_1G'_2G'_3) = Src(H(i))$ corresponds to the input word $(ww^R)^{\alpha+ir}$ and the subgraph G'_2 "adds" a cyclic shifts of a copies of ww^R). We can show the periodicity property on the basis of these facts, but we should care about requirements of the claim. In particular, r' and $(\alpha' - \alpha)$ should be divisible by r. So, let $\alpha' = \alpha + ir$, $r' = mra\alpha'$, $H_1 = G'_1G'_2$, $H_2 = (G'_2)^{r'/a}$, $H_3 = G'_3$. Then:

- $\langle\gamma\rangle_{l+1}$ decompose $H_1H_2^lH_3$ into $H_1, \langle H_2\rangle_l, H_3$, and paths on borders of this decomposition are short, (since the paths on the borders of the decompositions $G'_1, \langle G'_2\rangle_*, G'_3$ are short, as showed above),
- $H_1H_2^lH_3 = G'_1(G'_2)^{r'l/a+1}G'_3$ and the initial configuration of the graph $H_1H_2^lH_3$ corresponds to the input word $(ww^R)^{\alpha+ir+r'l} = (ww^R)^{\alpha'+lr'}$.
- $Src(H_2) = (Src(G'_2))^{r'/a} = \mathrm{shift}((ww^R, p))^{r'}$
- $\alpha' = \alpha + ir \leqslant m^{5j} \leqslant m^{20j}$; moreover, $r' = mra\alpha' \geqslant m\alpha'$ and $r' = mra\alpha' \leqslant m \cdot m^j(\alpha + ir)^2 \leqslant m^{1+j+10j} \leqslant m^{20j}$.
- $init((ww^R)^{\alpha'+lr'}) \vdash_M^* G_1G_2^{i+ma\alpha'\cdot l}G_3 \vdash_M^* H_1H_2^lH_3$ for every $l > 0$.

These conditions certify that ww^R satisfies the periodicity property with respect to $(H_1, H_2, H_3, \alpha', r', 20j)$, where $\alpha' = \alpha + gr$, $r' = fr$ for $g = i$ and $f = ma(\alpha + ir)$.

It remains to verify that $H_1H_2^lH_3$ is in fact the opposite border graph with respect to $G_1G_2^{g+lf}G_3$ for each $l > 0$. It means that all sinks of G_3 in $G_1G_2^{i+lf}G_3$ should remain as sinks in the graph preceding $H_1H_2^lH_3$ by one derivation step, and at least one of them should be an internal vertex of $H_1H_2^lH_3$ (see Definition 26). Recall that $H_1H_2^lH_3 = G'_1(G'_2)^{r'l/a+1}G'_3$. Moreover, as $G'_1G'_2G'_3$ is the opposite border graph with respect to $G_1G_2^iG_3$, there exists a subgraph J such that

$$G_1G_2^iG_3 \vdash_M^* G'_1J \vdash_M G'_1G'_2G'_3,$$

all sinks of G_3 remain the sinks of G'_1J and they belong to the subgraph J; moreover, at least one of these sinks is an internal vertex of $G'_1G'_2G'_3$. Indeed, the subgraph G'_1 appears already in the graph directly preceding $G'_1G'_2G'_3$, because G'_2 contains at least k sinks (by Claim 31), while the window length is equal to k and the window is located in the sinks of G'_3 of the graph $G'_1G'_2G'_3$. On the other hand, one step of the computation changes only the status of vertices included in the window.

Thus, by Cut and Paste Lemma, we obtain the (sub)computation

$$G'_1(G'_2)^{r'l/a}J \vdash_M G'_1(G'_2)^{r'l/a+1}G'_3 = H_1H_2^lH_3,$$

because the description of the path on the right border of G'_1 and the description of the path on the right border of $G'_1(G'_2)^{r'l/a}$ are equal. And, all sinks of G_3 remain the sinks in J and at least one of them is internal in $H_1H_2^lH_3$. □

Finally, we show that ww^R satisfies the **strong** periodicity property with respect to $(H_1, H_2, H_3, \alpha', r', 20j)$ where $H_1, H_2, H_3, \alpha', r'$ are chosen according to Claim 32. By Claim 32 we know that $H_1H_2^lH_3 = H(g + lf)$ for every $l > 0$ (i.e., $H_1H_2^lH_3$ is the opposite border graph with respect to $G_1G_2^{g+lf}G_3$). Let us choose $l > 2$ such that $\alpha' + lr' = \alpha + (g + lf)r \leqslant m^{40j}$. (Such l exists by the fact that $\alpha', r' \leqslant m^{20j}$—see Definition 23.) Then, the image of the middle block is defined in $H_1H_2^lH_3$, by Claim 30, and images of all blocks (except the middle block, possibly) are defined and longer than $c'm$ by Middle Block Lemma (as the choice of $m > s_{80j}$ and l such that $\alpha' + lr' \leqslant m^{40j}$ guarantee that Middle Block Lemma is applicable). It remains to show that the image of the middle block is longer than $c'm$ as well. Indeed, by Proposition 25(b), the block $\lceil(\alpha' + lr'/2)\rceil$ (the middle block) is included in the infix H_2^l. So, by Proposition 25(c), it is equivalent to the image of the block $\lceil(\alpha' + lr')/2\rceil + 2r'$ or $\lceil(\alpha' + lr')/2\rceil - 2r'$, which is longer than $c'm$. Thus, images of all blocks are longer than $c'm$ in $H_1H_2^lH_3$ and the lemma holds by Proposition 25(d).

10. Conclusions

We introduced a direct lower bound technique designed particularly for length-reducing two-pushdown automata. Next, we applied this method to the proof of the fact that the set of palindromes is not a Church–Rosser Language; this solves the open problem from [21]. This fact implies that CRL is incomparable with the set of unambiguous context-free languages.

We expect that our proof technique might be applicable for a broader class of problems concerning language classes related to or defined by length-reducing two-pushdown automata. Recently, an infinite intersection hierarchy for Church–Rosser and growing context-sensitive languages has been shown with the help of this technique [15]. Moreover, some separation results for language classes defined by randomized lrTPDAs make use of this technique as well [16].

Acknowledgments

The authors thank Friedrich Otto and Gundula Niemann for helpful comments on the draft of this paper, and to anonymous referees for valuable comments on the submitted version of the paper.

References

[1] M. Beaudry, M. Holzer, G. Niemann, F. Otto, McNaughton families of languages, Theoretical Computer Science 290 (3) (2003) 1581–1628.
[2] R.V. Book, Grammars with Time Functions, Dissertation, Harvard University, Cambridge, MA.
[3] G. Buntrock, Wachsende Kontextsensitive Sprachen, Habilitationsschrift, Würzburg, 1995.
[4] G. Buntrock, K. Loryś, On growing context-sensitive languages, in: Proc. ICALP'92, LNCS, vol. 623, Springer, 1992, pp. 77–88.
[5] G. Buntrock, K. Loryś, The variable membership problem: Succintness versus complexity, in: Proc. STACS'94, LNCS, vol. 775, Springer, 1994, pp. 595–606.
[6] G. Buntrock, F. Otto, Growing context-sensitive languages and Church–Rosser languages, Information and Computation 141 (1) (1998) 1–36.
[7] N. Chomsky, On certain formal properties of grammars, Information and Control 2 (2) (1959) 137–167.
[8] E. Csuhaj-Varju, J. Dassow, J. Kelemen, G. Paun, A Grammatical Approach to Distribution and Cooperation, Gordon and Breach, London, 1994.
[9] E. Dahlhaus, M.K. Warmuth, Membership for growing context-sensitive grammars is polynomial, JCSS 33 (3) (1986) 456–472.
[10] A. Gladkij, On the complexity of derivations for context-sensitive grammars, Algebri i Logika 3 (1964) 29–44, [In Russian].
[11] M.A. Harrison, Introduction to Formal Language Theory, Addison-Wesley, 1978.
[12] L.A. Hemaspaandra, P. Mukherji, T. Tantau, Computation with absolutely no space overhead, in: Proc. DLT'03 (Developments in Language Theory), LNCS, vol. 2710, Springer, 2003, pp. 325–336.
[13] M. Holzer, M. Kutrib, Flip-pushdown automata: k+1 pushdown reversals are better than k, in: Proc. ICALP'03, LNCS, vol. 2719, Springer, 2003, pp. 490–501.
[14] M. Holzer, F. Otto, Shrinking multi-pushdown automata, in: Proc. FCT'05, LNCS, vol. 3623, Springer, 2005, pp. 305–316.

[15] T. Jurdziński, The boolean closure of growing context-sensitive languages, in: Proc. Developments in Language Theory (DLT), LNCS, vol. 4036, Springer, 2006, pp. 248–259.
[16] T. Jurdziński, Probabilistic length-reducing automata, in: Proc. MFCS'06, LNCS, vol. 4162, Springer, 2006, pp. 561–572.
[17] T. Jurdziński, K. Loryś, Church–Rosser languages vs. UCFL, in: Proc. ICALP'02, LNCS, vol. 2380, Springer, 2002, pp. 147–158.
[18] M. Li, P. Vitanyi, An Introduction to Kolmogorov Complexity and its Applications, Springer-Verlag, 1993.
[19] J. Loecks, The parsing of general phase-structure grammars, Information and Control 16 (1970) 443–464.
[20] R. McNaughton, An insertion into the Chomsky hierarchy?, in: J. Karhumaki, H.A. Maurer, G. Paun, G. Rozenberg (Eds.), Jewels are Forever, Contributions on Theoretical Computer Science in Honour of Arto Salomaa, Springer-Verlag, 1999, pp. 204–212.
[21] R. McNaughton, P. Narendran, F. Otto, Church–Rosser Thue systems and formal languages, Journal of the Association Computing Machinery 35 (1988) 324–344.
[22] G. Niemann, Church–Rosser Languages and Related Classes, PhD Thesis, Universitaet Kassel, 2002.
[23] G. Niemann, F. Otto, Restarting automata, Church–Rosser languages, and confluent internal contextual languages, Developments in Language Theory (DLT), Preproceedings, Aachener Informatik-Berichte 99-5, RWTH Aachen (1999) 9–62.
[24] G. Niemann, F. Otto, The Church–Rosser languages are the deterministic variants of the growing context-sensitive languages, Information and Computation 197 (1–2) (2005) 1–21.
[25] G. Niemann, J.R. Woinowski, The growing context-sensitive languages are the acyclic context-sensitive languages, in: Proc. DLT'01 (Developments in Language Theory), LNCS, vol. 2295, Springer, 2002, pp. 197–205.
[26] A. Okhotin, An overview of conjunctive grammars, Formal Language Theory Column. Bulletin of the EATCS 79 (2003) 145–163.
[27] A. Okhotin, Boolean grammars, Information and Computation 194 (1) (2004) 19–48.
[28] P. Sarkar, Pushdown Automaton with the Ability to Flip its Stack, Electronic Colloquium on Computational Complexity (ECCC)(081): (2001).
[29] J.R. Woinowski, Prefix Languages of Church–Rosser Languages, FSTTCS'00 (Foundations of Software Technology and Theoretical Computer Science), LNCS 1974, 2000, Springer, pp. 516–530.
[30] J.R. Woinowski, The context-splittable normal form for Church–Rosser language systems, Information and Computation 183 (2) (2003) 245–274.