Lower Bound Technique for Length-reducing Automata*

Tomasz Jurdziński** and Krzysztof Loryś

Institute of Computer Science, Wrocław University,
Przesmyckiego 20, PL-51-151 Wrocław, Poland
{tju,lorys}@ii.uni.wroc.pl

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 [9]. GCSL and its deterministic counterpart called Church-Rosser Languages (CRL) complement the Chomsky hierarchy in a natural way [18], as the classes filling the gap between CFL and CSL. Interestingly, they possess characterizations by a natural machine model, length-reducing two-pushdown automata (lrTPDA). We present a lower bound technique applicable for lrTPDA. Using this method, we prove the conjecture that the set of palindromes is not in CRL [19]. This implies that CFL∩coCFL as well as UCFL∩coUCFLare not included in CRL, where UCFL denotes the class of unambiguous context-free languages (what solves an open problem from [1]). The another consequence of our result is that CRL is a strict subset of GCSL ∩ coGCSL.

1 Introduction and related work

Formalisms defining language classes located between context-free languages (CFL) and context-sensitive languages (CSL) have been intensively studies for many years. One of the motivations was to find families which possess both an acceptable computational complexity and sufficient expressibility, having simultaneously natural characterizations by grammars and a machine model. Neither CSL nor CFL fulfil these demands. For the first class the membership problem is PSPACE-complete what makes it in its full generality too powerful for practical applications. On the other hand context-free grammars are e.g. not powerful enough to express all syntactical aspects of programming languages.

One of the approaches was to restrict linearly lengths of context-sensitive derivations [2]. Another directions exploited extensions of machine model characterizations of context free languages (e.g. LOGCFL; flip-stack pushdown automata [13, 27]), or extensions of context free grammars (e.g., parallel communicating grammars [8], conjunctive and boolean grammars [25, 26]). Further, restrictions of linear space classes

* 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 [15].

** Corresponding author.

(characterizing CSL) were considered [12]. Very few of these models posses natural characterizations, both by machine models and grammars like structures, having simultaneously acceptable complexity.

One of the most interesting proposals was presented by Dahlhaus and Warmuth [9]. They consider grammars with strictly growing rules, i.e., such that the righthand side of the production is longer than the lefthand one. The resulting class of growing context-sensitive languages (GCSL) complements the Chomsky hierarchy in a natural way [18], it possesses several good properties justifying exploration of this class [29]. A linear bound on the length of derivations follows immediately from the definition. So, GCSL is contained in NSPACE(n). Dahlhaus and Warmuth showed a rather astonishing result that each languages generated by a grammar from GCSL can be recognized in deterministic polynomial time (and it is even included in LOGCFL). Buntrock and Loryś [4, 5] showed that this class is closed under many operations and forms an abstract family of languages. They also proved that it may be characterized by less restricted grammars. As shown by Niemann and Woinowski [24], GCSL may be also characterized by acyclic context sensitive grammars.

Buntrock and Otto [6, 3] give a characterization of GCSL by nondeterministic machine model, *shrinking two-pushdown automata* (sTPDA). Unexpectedly, it turned out that the class of languages recognized by deterministic sTPDA is equal to the class of Church-Rosser languages (CRL), that was introduced by McNaughton et al. [19] as languages defined by finite, length-reducing and confluent string-rewriting systems [6, 22, 20]. Moreover, an equivalence of shrinking and more restricted length-reducing two-pushdown automata has been shown by Niemann and Otto [22, 20]. Recently, Holzer and Otto considered generalizations of these models, they related these generalizations to other complexity of formal language classes [14].

The membership problem for CRL has linear time complexity and it contains many important languages that are not context-free. It is a strict superset of DCFL, but its definition is more intuitive than that of DCFL (as CRL has both machine model and the characterization by string rewriting systems). Recently, some new properties of CRL have been shown. They justify the applications of CRL in parsers construction [28]. Moreover, each language in CRL can be defined by a rewriting system in an elegant "normal form" [29]. However, weakness of CRL seems to be evident. Already McNaughton et al. [19] conjectured that CRL does not include all context-free languages. They stated also the conjecture that even a very simple context-free language, namely the set of palindromes of even length, is not in CRL. The incomparability of CRL and CFL was finally proved as a consequence of the fact that CRL is closed under complement (whereas CFL is not) and there are context-free languages whose complements are not even in GCSL [6]. The question about palindromes was restated in [22] and 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 palindromes is in CFL too, so the techniques used hitherto seem to be insufficient for proving the conjecture that palindromes are not in CRL.

2 Our Result

Lower bound techniques designed for other formal language classes fail in the case of language classes defined by length-reducing two-pushdown automata. Most of the known lower bounds for CRL were obtained by the fact that some fixed languages are not included in a larger class. We propose a direct lower bound technique designed particularly for length-reducing two-pushdown automata. It exploits a notion of so called computation graphs, cut and paste technique and a method of determining relationships between the current configuration and the input word. Finally, our method makes use of the incompressibility method (Kolmogorov complexity arguments) [16].

Specifically, we show that the set of palindromes of even lengths is not a Church-Rosser language, answering the open question stated in [19]. 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 such a length-reducing two-pushdown automata (lrDTPDAs) working on words from a certain set of palindromes has to fulfil 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, if we ensure that these palindromes are build out of blocks of high Kolmogorov complexity. This is impossible as the automaton is length reducing. As a technical tool designed particularly for this proof, we use a kind of pumping lemma. Note that even CRL languages over one-letter alphabet need not to be semilinear (e.g. $\{a^{2^n} | n \in \mathbb{N}\} \in$ CRL, [19]), so a pumping technique has to be used in a rather sophisticated way.

Observe that co-PAL, the complement of PAL, is also context-free. Moreover, PAL (as well as co-PAL) is an unambiguous context-free language (UCFL), i.e., there exists a context-free grammar for PAL such that every word from PAL has only one derivation tree (see [11]). Therefore, as a conclusion from Theorem 1, we obtain the following result solving the open problem from [1].

Corollary 1. *The classes* CFL$\cap$co-CFL *and* UCFL$\cap$co-UCFL *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 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 and state some open problems.

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, for $0 < i \leq j \leq |x|$. Let $[i,j] = \{l \in \mathbb{N} \mid i \leq l \leq j\}$. 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 $|x| = n$. 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 any function $f : X \to Y$ and any sequence $x = x_1 \dots x_n$ of elements of X, by $f(x)$ we denote a sequence $f(x_1)f(x_2)\dots f(x_n)$.

In the following, lower case letters $a, b, \dots, p$ and r, s denote natural numbers. The letter q usually denotes states of automata. Lower case letters u, v, w, x, y, z denote words over finite alphabets. Upper case letter C is used to denote configurations, 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 (CRL) are defined by finite, length-reducing and confluent string-rewriting systems [19]. 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 1 (Two-pushdown automaton. Length reducing automaton).
*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 $\Gamma_{\bot,j} = \Gamma^j \cup \{\bot\, v : |v| \leq j-1, v \in \Gamma^*\}$, $\Gamma_{j,\bot} = \Gamma^j \cup \{v\, \bot : |v| \leq j-1, v \in \Gamma^*\}$, $\mathcal{P}(Q \times \Gamma^* \times \Gamma^*)$ 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 (D)TPDA 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 $\bot$ occurs on both ends of the word uq_iv^R. As we denote configurations as 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. $uzqxv \vdash_M uz'q'x'v$ if $zqx \to z'q'x'$. Observe that the window of M in a configuration uqv ($q \in Q$) contains at most k symbols from the neighborhood of q. Further, $\vdash_M^*$ is a computation relation, defined as a transitive closure of $\vdash_M$.

For an input word $x \in \Sigma^*$, the corresponding *initial configuration* is $\perp q_0 x \perp$, 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, $L(M) = \{x \in \Sigma^* : \exists_{q \in F} \ \perp q_0 x \perp \vdash_M^* q\}$, 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 it can be removed from the pushdown only in the last step of the computation. Further, we assume that each transition $uqv \to u'qv'$ 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 may be characterized by two-pushdown length-reducing automata.

Theorem 2 ([20, 22]). *A language is accepted by a length-reducing TPDA if and only if it is a growing context-sensitive language.*

Theorem 3 ([20, 22]). *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 the length-reducing DTPDA to move its window through the whole word. As each step reduces the length of the configurations, so when moving through the whole word, we shorten the configuration by a linear multiplicative factor. Thus, we lose all information about the input word after logarithmic number of comparisons. 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 a "center" of the input word without destroying its content. The particular position is the good candidate for being a center (when an 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 such a candidate. Then, if the candidate is wrong (i.e. it is not a center), we will be unable to check other candidates.

On base of these observations we analyze computations of lrDTPDAs on inputs from a family $(ww^R)^*$, where the shortest description of w (i.e. its Kolmogorov complexity) is much larger then the size of the automaton. By analysis of computations on these inputs we also construct computations on other inputs using cut and paste technique and pumping. Note that each input from this family is a palindrome. However, the size of w makes unable to detect a periodic structure. On the other hand, the input

word contains many positions that are candidates for centers (positions on "borders" between w and w^R or w^R and w). When we compare symmetric positions starting from such a "candidate center", we do not realize quickly that our candidate is wrong, losing information about contents of the subword checked during this process.

Our formal analysis proceeds by partitioning computations into stages. One stage starts when one of pushdowns is (almost) empty and finishes when the opposite pushdown is (almost) empty. Note that the length of a configuration at the end of such stage is shorther than the length of a configuration at the beginning of the stage by a linear multiplicative factor. The automaton is not allowed to remove any information about the input during the first stage, because it does not even "know" the length of the input word. Recall that the choice of the length of w makes also unable to detect the periodic structure of the input. These restriction forces the automaton to leave the structure of 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. The periodicity makes unable to "detect" a center of the input word in the second stage and forces the automaton to work "similarly" as in the first stage. Applying this strategy to consecutive stages (that shorten the configuration linearly) we get a short configuration which does not store information about an original contents of the input. Simultaneously, no progress in the process of checking if the input word is a palindrom is done during these stages.

A formal proof following this strategy is presented in Section 9. It uses methodology introduced in Sections 5, 6, 7 and 8. In these sections we formulate conditions which allow to use a cut and paste technique and pumping for computations of lrTPDA's. We expect that these techniques, combined with the incompressibility method [16], are applicable for broader class of problems concerning language classes related to this machine model.

5 Computation Graphs and their Properties

We introduce the notion of a computation graph, similar to derivation graphs from [9] and [17]. Each computation of a length-reducing TPDA $M = (Q, \Sigma, \Gamma, q_0, \bot, F, \delta)$ corresponds to a planar directed acyclic graph defined in the following way. Vertices in such a graph will be labeled with the corresponding 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 are called symbol vertices, similarly vertices labeled with states and transitions are called state vertices and transition vertices, respectively.
A *computation graph* $G_{(j)} = (V_j, E_j)$ corresponding to the computation $C_0 \vdash C_1 \vdash \ldots \vdash C_j$ where C_0 denotes an initial configuration is defined inductively (see Figure 1 and Example 1):

- $j = 0$: Let $C_0 = \bot\, q_0 x_1 x_2 \ldots x_n \,\bot$ be an initial configuration, where $x_i \in \Sigma \cup Q$ for $i = 1, \ldots, n$. Then $G_{(0)} = (V_0, E_0)$ has the vertices $\rho_{-2}, \ldots, \rho_{n+2}$ such that $\omega(\rho_i) = x_i$ for $1 \leq i \leq n$, $\omega(\rho_i) = \bot$ 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 $C_0 \vdash C_1 \vdash_M \ldots \vdash_M C_{j-1}$ corresponds to the graph $G_{(j-1)}$, and a 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 $z = z_1 \dots z_p$ and $z' = z'_1 \dots z'_{p'}$ where $z_i, z'_i \in Q \cup \Gamma$ for each i. The graph $G_{(j)}$ is constructed from $G_{(j-1)}$ by adding:

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

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 ordering. In a similar way, there is a natural left to right ordering among the parents and children of each transition vertex. Further, this ordering induces a natural left to right order among the sinks of the computation graph (adequate to the ordering 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 remain as sources and sinks in all graphs $G_{(j)}$. Thus $\perp C_0 \perp = Src(G_{(j)})$, $\perp C_j \perp = Snk(G_{(j)})$. For a configuration C_j described by the sinks of $G_{(j)}$ we say 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 and 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 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 the first vertex is not a source or the last vertex is not a sink, we say about a *subpath*. Let σ be a path in G with a sink π. We say that σ is *short* if there is no path with sink π that is shorter than σ. The relation $\prec$ among vertices of the graph induces a left-to-right partial ordering 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) in a set of paths S if it is to the left (right) of every path $\sigma' \in S$.

Let us fix a length-reducing (D)TPDA $M = (Q, \Sigma, \Gamma, q_0, \perp, F, \delta)$ with the window of length $k > 2$. (One can simulate an automaton with the window of length 2 by an automaton with longer window.) The remaining part of this section concerns properties of computation graphs corresponding to computations of the automaton M.

Definition 2. *The **height** of a vertex π is the number of transition vertices in any short path with the sink in π. We say that a sink π is i-**successor** if one of short paths with the sink in π starts in ρ_i, the source vertex associated to the ith symbol of the input word.*

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

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

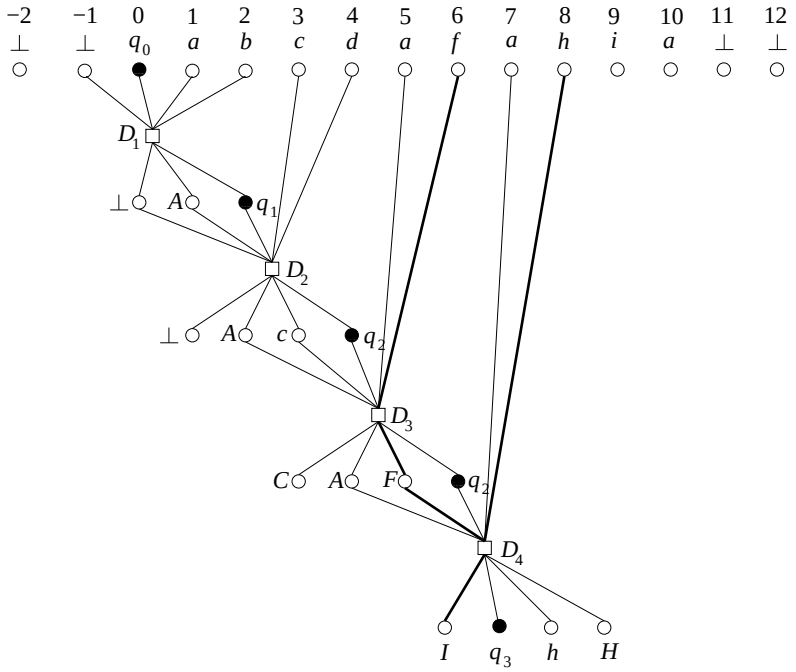


Fig. 1. A computation graph corresponding to the computation $C_0 \vdash_M^* C_4$

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

Transition vertices are denoted by squares and they have following labels describing consecutive transitions of the automaton M:

$\omega(D_1) = \langle \bot\, q_0 ab \to \bot\, A q_1 \rangle$,
$\omega(D_2) = \langle \bot\, A q_1 cd \to \bot\, A c q_2 \rangle$,
$\omega(D_3) = \langle A c q_2 a f \to C A F q_2 \rangle$,
$\omega(D_4) = \langle A F q_2 a h \to I q_3 h H \rangle$.

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

Observe, for example, that a path σ that starts in the eighth symbol of the input word (ρ_8) and consists of vertices with labels $(h, \omega(D_4), I)$ is short in $G_{(4)}$. A path σ' that starts in the 6th symbol of the input word and consists of vertices with labels $(f, \omega(D_3), F, \omega(D_4), I)$ is not short because it is longer than the path σ which has the same sink as σ'. □

Below, we enumerate some basic properties of computation graphs.

Proposition 1. *Let G be a computation graph, let $\rho_1, \ldots, \rho_n$ be sources of G corresponding to the input word.*

(a) Let π be a sink vertex in G. Then, there exists $-2 \leq i \leq n+2$ such that π is i-successor.
(b) Let D be a transition vertex in G and let π be a child of D. If π is i-successor then there exists π', a parent of D such that the shortest subpath that starts in a source of the graph and finishes in π' has ρ_i as a starting point.
(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 in π_{j-1} and finishes in π_j for $i = 1,2$ and $j \in [1, p+1]$ (with two exceptions: σ_i^1 starts in a source of σ_i and σ_i^{p+1} finishes in a sink of σ_i). Then, for every sequence $a_1, \ldots, a_p \in \{1,2\}^p$, a 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 $j_1 \leq j_2$ such that π is j_1-successor and π' is j_2-successor.
(e) Let σ be a short path in G, $G \vdash_M G'$ and let a sink of σ be also the sink of G'. Then σ is the short path in G'.

Proof. (a) There exists (at least one) path from a source vertex to π, for every vertex π. Let σ be a path with the minimum length among them, $\rho_i \in \sigma$ for some $-2 \leq i \leq n+2$. Thus, π is an i-successor.
(b) Observe that D is the only parent of π, by definition of the computation graph. So, D belongs to σ, the shortest path from ρ_i to π. Thus, there is π', a parent of D, that also belongs to σ. Thus, a subpath of σ that starts in a source of σ and finishes in π' is the shortest path from a source vertex to π' (otherwise σ would not be the shortest path with the sink in π).
(c) It is enough to show that $|\sigma_1^j| = |\sigma_2^j|$ for every $1 \leq j \leq p$. 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 $1 \leq j \leq p$. Let π be a sink of σ_2. If we replace σ_2^j by σ_1^j in σ_2 then we obtain a path that finishes in π and it is shorter than σ_2. Thus, σ_2 is not the shortest path with the endpoint in π. A similar argument works for the case $j = p+1$. Contradiction.

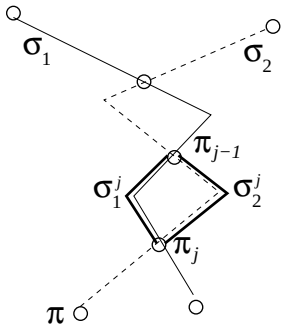


Fig. 2. Intersecting short paths.

(d) Let j_1 be a minimal value such that π is j_1-successor. For the sake of contradiction assume that the statement is false. Then, for each j_2, if π' is j_2-successor then $j_2 < j_1$. Let σ be a shortest path from ρ_{j_1} to π and σ' be the shortest path from ρ_{j_2} to 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 at ρ_{j_2} and finishes at μ by the appropriate subpath of σ then, by item (c) above, we obtain a short path with a source in ρ_{j_1} and the sink in π'. Thus, π' is j_1-successor – contradiction.
(e) Obvious. □

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

Proposition 1(c) has an important consequence which will be implicitely used in the paper.

Corollary 2. *Let G be a computation graph, let V_{sr} and V_{sn} be subsets of the set of sources of G and the set of sinks of G, respectively. Then, a set P of short paths with sources in V_{sr} and sinks in V_{sn} 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 1 (Heights Lemma). *Let G be a computation graph corresponding to the 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 $p_h \leq 6n\left(\frac{k}{k+1}\right)^h$, 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

$$\mathrm{sum}(D_\pi) = \Sigma_{\{\mu:(\mu,D_\pi)\in G\}} weight(\mu), \;\; \mathrm{out}(D_\pi) = |\{\mu : (D_\pi,\mu) \in G\}|$$

and D_π is a parent of π. In other words, the sum of weight of parents of D_π is uniformly distributed among the children of D_π. It can be shown by induction that if $height(\pi) \geq i$ then $weight(\pi) \geq ((k+1)/k)^i$. 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 $(\mu, D_\pi) \in G$, $height(\mu) \geq i-1$, so $weight(\mu) \geq (\frac{k+1}{k})^{i-1}$ by 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

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

since $\frac{\mathrm{in}(D_\pi)}{\mathrm{out}(D_\pi)} \geq \frac{k+1}{k}$ (what follows from the inequality $1 \leq \mathrm{out}(D_\pi) < \mathrm{in}(D_\pi) \leq k+1$, a consequence of the fact that M is length-reducing and its window length is equal to k). Moreover, the sum of 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 a set of sinks with height greater or equal to h and let $|P_h| = p_h$. Then,

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

so $p_h \leq (\frac{k}{k+1})^h (n+5) \leq 6n(\frac{k}{k+1})^h$ for $n \geq 1$. □

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

Corollary 3. *Let G be a computation graph corresponding to a configuration of M during the computation on an input word of length n. Then, there are no vertices with height greater than $\frac{\log(6n)}{\log((k+1)/k)} = O(\log n)$.*

By Corollary 3, the length of each short path is $O(\log n)$ where n is the length of the input word.

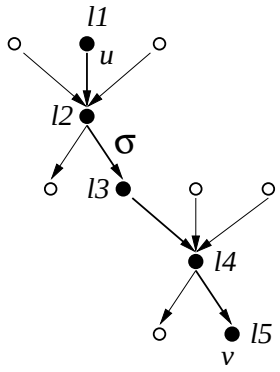
6 Cut and Paste Technique and Pumping

One of crucial properties of context free derivations is the independence of any subderivation starting in a particular nonterminal from the remaining part of the derivation. It allows to replace such subderivation by another subderivation starting from the same nonterminal, what gives another correct derivation of (possibly) another terminal word. Using this fact, one can cut and paste subderivations, obtaining different correct derivations. This property makes also possible to formulate very powerful pumping lemmata.

These conditions are not satisfied in case of growing grammars nor in case of TPDA. However, dependencies between subcomputations can be described by paths partitioning computation graphs into subgraphs. In consequence, cut and paste and pumping are possible under some conditions.

Definition 3 (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, $desc(\sigma)$, consists of the sequence $\omega(\pi_1), \ldots, \omega(\pi_{2l+1})$ of labels of consecutive vertices on the path and the sequence $p_1, p_2, \ldots, p_{2l}$ of numbers, such that for each even j, 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, where the numbering is according to the left to right order $\prec$ (see the figure and Example 2). We say that a path $\sigma \in G$ is a γ-path if $desc(\sigma) = \gamma$.



Labels: *l1, l2, l3, l4, l5*

Positions: 2, 2, 1, 2

Fig. 3. Description of a path.

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

Definition 4 (Decomposition). *Descriptions $\gamma_1, \ldots, \gamma_{l-1}$ (for $l > 1$) decompose 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 $desc(\sigma_i) = \gamma_i$ 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 are to the right of σ_{i-1} and to the left of σ_i or inside σ_{i-1} or inside σ_i (i.e., vertices from σ_i belong to G_{i-1} and to G_i).*

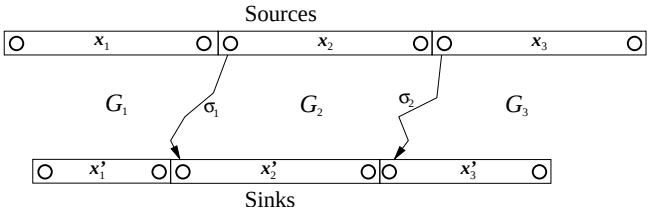


Fig. 4. Decomposition of a computation graph.

A computation graph G can be denoted as $G = G_1 \dots G_l$ if there exist descriptions $\gamma_1, \dots, \gamma_{l-1}$ which decompose G into subgraphs $G_1, \dots, G_l$.

We extend notions *Src* and *Snk* into subgraphs of computations 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 $G = G_1, \dots, G_l$ then $Src(G) = Src(G_1) \dots Src(G_l)$ and $Snk(G) = Snk(G_1) \dots Snk(G_l)$.

Example 2: A description of a path σ from Figure 1 that starts in the eighth symbol of the input word and consists of vertices with labels $(h, \omega(D_4), I)$ consists of this sequence and a sequence $(5,1)$, because h is the 5th parent of D_4 and I is the first child of D_4.

A description of a path σ' that starts in the 6th symbol of the input word and consists of vertices with labels $(f, \omega(D_3), F, \omega(D_4), I)$ is equal to this sequence of labels and a 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 $Src(J) = \perp\perp\, q_0abcda$, $Src(J') = fahia\ \perp\perp$, $Snk(J) = \perp\perp\, C$, $Snk(J') = Iq_3hHa\ \perp\perp$

□

Let $G_1G_2G_3$ be a computation graph which corresponds to the computation of a length-reducing (D)TPDA M with the window of length k. We say that the window *touches* a subgraph G_2 of $G_1G_2G_3$ if and only if $G_1G_2G_3 \vdash G'$ and at least one sink vertex of G_2 becomes non-sink vertex in G' (i.e., it is rewritten in the step). In other words, at least one sink of G_2 is inside the window of the automaton in the configuration described by (sinks of) $G_1G_2G_3$.

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 $\tau, \dots, \tau$, where τ is taken j times.

Lemma 2 (Cut and Paste Lemma). *Assume that descriptions γ_1, γ_2 decompose a graph G into G_1, G_2, G_3 and they decompose a 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, a graph $J = G_1H_2G_3$ is a 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, G_3.*

Lemma 3 (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 a 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 some computations, we introduce the notion of a correct graph. Let $M = (Q, \Sigma, \Gamma, q_0, \perp, F, \delta)$ 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 labelling of V. As in computation graph, vertices are called symbol, state and transition vertices. The graph G is correct with respect to M if it possesses the following properties:

(a) A set S of vertices with no incoming edges has at least five elements and is linearly ordered: $\rho_{-2} \prec \dots \prec \rho_{n+2}$, where $S = \{\rho_i\}_{i \in [-2, n+2]}$. Moreover, $\omega(\rho_i) \in \Sigma$ for $1 \leq$

$i \leq n$, $\omega(\rho_i) = \perp$ for $i \in \{-2,-1,n+1,n+2\}$ and $\omega(\rho_0) = q_0$. In other words, the sources describe an initial configuration of M for the input word $\omega(\rho_1)\dots\omega(\rho_n)$.

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

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

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 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 simple induction we can show the following proposition.

Proposition 2. *Let G be a correct graph with respect to (D)TPDA $M = (Q, \Sigma, \Gamma, q_0, \perp, F, \delta)$. Then, G is a computation graph.*

Proof. We show this fact by induction with respect to the number of transition vertices in a graph. The base step for graphs with no transition vertices is obvious. Assume that every correct graph with i transition vertices is a computation graph. Let G be a correct graph that contains $i+1$ transition vertices. Observe that there exists a transition vertex $D \in G$ such that there is no path from D to any other transition vertex. Indeed, otherwise there exists a cycle in G what contradicts correctness of G. Let D be a transition vertex such that there is no path from D to any other transition vertex. Then all children of D are sinks in G, since every non-sink state or symbol vertex has a child that is a transition vertex. Let G' be a graph obtained from G by deleting D, children of D and all edges incident to D. The graph G' contains i transition vertices and is correct, since we do not delete any vertices or edges that have influence on "local correctness" ((b)–(d) in the definition of correctness) of particular vertices, we do not delete any source vertices and we do not add any edges (making a cycle). So G' is a computation graph, by induction hypothesis. Then, by adding to G' the vertex D, its children and edges adjacent to D, we form a computation graph G, where $G' \vdash_M G$, according to the rules defining computation graphs. □

Proposition 2 substantially simplifies proofs of Cut and Paste Lemma and Pumping Lemma.

Proof of Lemma 2. First, we show that if G and H are correct graphs then $J = G_1 H_2 G_3$ is a correct graph as well, where $G_1 H_2 G_3$ is formed by identifying vertices on borders of H_2 with 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 descriptions of paths on borders of decompositions of G and H guarantees that it is also satisfied for all vertices on borders of the subgraphs of J. One can easily check that the set of sources of J satisfies the item (a) of the definition of

correctness. (Note that a path description determines whether the source state vertex is to the left or to the right of the path.) 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 Figure 6).

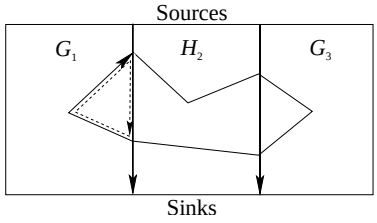


Fig. 5. The cycle in $G_1H_2G_3$ (solid lines) induces cycles in G_1 or H_2 or G_3 (dashed lines).

We showed that $J = G_1H_2G_3$ is the correct graph. So, by Proposition 2, J is the computation graph corresponding to the computation $x_1y_2x_3 \vdash_M^* x'_1y'_2x'_3$. □

Proof of Lemma 3. Observe that $G_1G_2^iG_3$ is a correct graph for each i what follows from the fact that the paths on both "borders" of G_2 have equal descriptions. Thus, the resulting graph is correct, so it is a computation graph by Proposition 2. □

7 How much information is stored on the pushdowns

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

We would like to treat subwords of configurations as representatives of subwords of the given 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 into 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.

Although such a partition of a configuration into independent parts describing appropriate subwords of the input word is natural in an initial configuration, we lose this condition during the computation when some parts of configurations become dependent of long subwords of the input word. As we have seen in the previous section, one can "extract" these dependencies using the notion of paths. Moreover, as each sink is also the endpoint of a short path (i.e., the path of logarithmic length, see Corollary 3), these paths do not add large overhead to information stored in the configuration.

It might seem that a good candidate for an image of a given subword of the input word could be a word stored between sinks of short paths starting from the left and right ends of this subword, respectively, together with appropriate short paths. However, we

should care about two additional circumstances. First, we would like to avoid a situation that images of two disjoint words overlap too much. Second, note that although for each sink, there exist short paths that have an endpoint in this vertex, there might exist source vertices which are not starting points of any short path (for example, the vertex ρ_2 in Figure 1). Thus, in order to obtain unique images and guarantee that images of disjoing subwords are (almost) disjoint, we define an image by deriving a smallest subgraph cutted by short paths and containing the whole considered subword (i.e., sources describing it). Then, an image is defined by these short paths and a sequence of sinks between them (these sinks describe some subword of the current configuration).

Definition 5. *Let G be a computation graph corresponding to a computation of* (D)TPDA *on an input word of length n. Then, $RL_G(i)$ for $i \in [0, n+1]$ denotes a rightmost symbol sink vertex in the set $\{\pi \mid \exists_{j \leq i} \ \pi$ is j-successor$\}$. Similarly $LR_G(i)$ denotes a leftmost symbol sink symbol vertex in the set $\{\pi \mid \exists_{j \geq i} \ \pi$ is j-successor$\}$.*

Definition 6 (Image). *Let G be a computation graph corresponding to a computation on an input word of length n, let $0 \leq l < r \leq n+1$ and $\pi_l = RL_G(l)$, $\pi_r = LR_G(r)$. If there is no symbol sink vertex π such that $\pi_l \prec \pi \prec \pi_r$ then an (l,r)-image is undefined in G. Otherwise, an (l,r)-image is defined in G and it is equal to $(\sigma_l, \sigma_r, \tau)$, where:*

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

Let length of an image $(\sigma_l, \sigma_r, \tau)$ be equal to the number of its sinks, i.e. $|\tau|$.
We say that an (l,r) image in a graph G and an (l',r') image in a graph G' are equivalent if both these images are undefined or they are equal to $(\sigma_l, \sigma_r, \tau)$ and $(\sigma'_l, \sigma'_r, \tau')$ respectively such that descriptions of the paths σ_l, σ_r are equal to descriptions of σ'_l and σ'_r resp., as well as $\omega(\tau) = \omega(\tau')$.

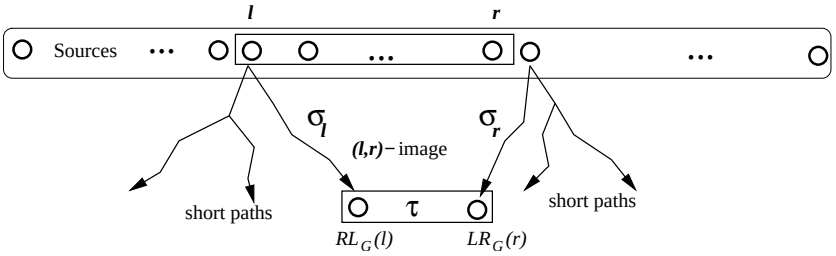


Fig. 6. Image

Example 3: Let $\rho_{-2},\ldots,\rho_{12}$ denote the sources of the graph $G_{(4)}$ presented in Figure 1. We have in particular the following images:

(l,r)	$desc(\sigma_l)$	$desc(\sigma_r)$	$\omega(\tau)$	$source(\sigma_l)$	$source(\sigma_r)$	image exists?
$(3,8)$	$(C,\omega(D_2),\perp)$	$(h,\omega(D_4),H)$	$\perp\perp CHq_3hH$	ρ_3	ρ_8	Y
$(2,6)$	$(\perp)$	$(f,\omega(D_3),C)$	$\perp\perp C$	ρ_{-2}	ρ_6	Y
$(3,5)$	$(C,\omega(D_2),\perp)$	$(a,\omega(D_3),C)$	$\perp C$	ρ_3	ρ_5	N
$(7,8)$	$(a,\omega(D_4),H)$	$(h,\omega(D_4),I)$	$\emptyset$	ρ_7	ρ_8	N
$(1,9)$	$(\perp)$	(i)	$\perp\perp CIq_3hHi$	ρ_{-2}	ρ_9	Y
$(9,11)$	(i)	$(\perp)$	$ia\perp$	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,σ_5). Further, the $(7,8)$ image is undefined (because the sink of σ_7 is to the right of the sink of σ_8).

□

The length of paths on "borders" of images do not influence significantly the size of the descriptions of images of long words, because short paths have only logarithmic lengths.

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, Section 5). As these vertices have no incident edges in each computation graph, they form two short paths at the left and the right end of the graph. This ensures that the values $RL_G(i)$ and $LR_G(i)$ are defined for each $i\in[0,n+1]$ 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, formal (and a bit tedious) proofs are presented for completeness. For the sake of the following propositions, observe that each state vertex (except the vertex denoting the final accepting/rejecting configuration) has a sibling symbol vertex such that the heights, in-neighbors and out-neighbors of both vertices are equal. So, we can assume that short paths do not contain internal state vertices.

Proposition 3. *Let G,G' be computation graphs corresponding to a computation on an input word of length n, $G\vdash_M G'$, and $0\le l<r\le n+1$. Assume that an (l,r)-image is defined in G and it is equal to (σ_l,σ_r,τ). Then,*

(a) For every $1<i<|\tau|$, if $\tau[i]$ is j-successor then $l<j<r$.
(b) 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.
(c) If $|\tau|>2k$ then an (l,r)-image is defined in G'.
(d) If the (l,r)-image is defined in G' then its length is not smaller than $|\tau|-k$ and not larger than $|\tau|+k$.
(e) Assume that the (l',r')-image is defined in G for $r<l'<r'\le n+1$ and it is equal to $(\sigma'_{l'},\sigma'_{r'},\tau')$. Then, $|\tau\cap\tau'|\le 2$, i.e., the (l,r) image and the (l',r') image overlap by at most two sinks.
(f) The paths σ_l and σ_r are disjoint.

Proof. (a) For the sake of contradiction assume that the statement does not hold. Then, a vertex $\tau[i]$ for some $1 < i < |\tau|$ is j-successor for $j \leq l$ or $j \geq r$. But this implies that $\tau[i] \preceq RL_G(l) = \tau[1]$ or $\tau[i] \succeq LR_G(r) = \tau[|\tau|]$ (i.e., $\tau[1]$ is not the rightmost j-successor for $j \leq i$ or $\tau[|\tau|]$ is not the leftmost j-successor for $j \geq r$) – contradiction.

(b) Indeed, all vertices of τ remain alive in G', so also $\tau[1] = RL_{G'}(l)$ and $\tau[|\tau|] = LR_{G'}(r)$ and the image does not change.

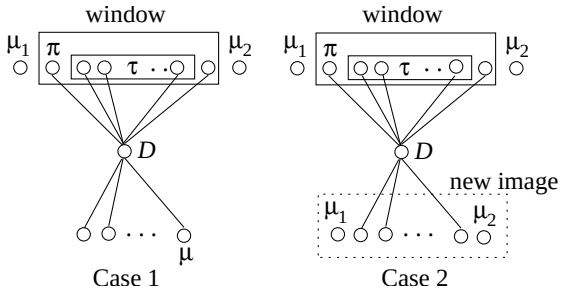
(c) and (d).
First we show that if the window does not contain $\tau[1]$ or it does not contain $\tau[|\tau|]$ then the sequence τ' between $RL_{G'}(l)$ and $LR_{G'}(r)$ is not shorter than $\max(0, |\tau| - k)$ and not longer than $|\tau| + k$. By (b) above, if the window does contain any vertex of τ in G then the (l,r)-image remains unchanged. Similarly, if the window is included in $\tau[2, |\tau| - 1]$, the image is shortened by one symbol. Now, assume that the window contains exactly one element from $\tau[1]$, $\tau[|\tau|]$. W.l.o.g. assume that the window contains $\tau[1]$ in G and it does not contain $\tau[|\tau|]$. Let $D \in G' \setminus G$ be the transition vertex "added" in the step $G \vdash_M G'$. Thus $\tau[1]$ is a parent of D. Let us consider the following cases:

- The minimum of heights of parents of D is achieved by a parent of D located to the left of $\tau[1]$ or by $\tau[1]$.
  Then, the rightmost child of D is equal to $RL_{G'}(l)$. Indeed, all children of D are j-successors for some $j \leq l$.
- The minimum of heights of parents of D is achieved only by a parent(s) of D located to the right of $\tau[1]$.
  Then, $RL_{G'}(l)$ is equal to a vertex directly preceding the leftmost child of D.

In both cases the length of the sequence between $RL_{G'}(l)$ and $LR_{G'}(r)$ differs by at most k from $|\tau|$. Note that if $|\tau| \geq 2k$ then it is impossible that the window (of length k) touches both $\tau[1]$ and $\tau[|\tau|]$. So, if $|\tau| \geq 2k$ then an (l,r)-image is defined in G' and satisfies inequalities stated in (d).

Finally, consider the case that the window includes $\tau[1]$ as well as $\tau[|\tau|]$ in G, what may happen only for $|\tau| \leq k + 1$. Let h be the minimum of heights of parents of a transition vertex $D \in G' \setminus G$ (i.e. vertices in the window in G). Let μ_1, μ_2 be sinks of G located directly to the left and directly to the right of the window in G, respectively. Then, either μ_1, μ_2 become endpoints of the paths of the (l,r) image in G' (when the minimum of heights of parents of D is achieved only by vertices π such that $\tau[1] \prec \pi \prec \tau[|\tau|]$ (so the length of the image is at most $k+2$) or the new image is undefined otherwise. More precisely, we have two cases (see figure):

Case 1. There exists π, a parent of D such that $\text{height}(\pi) = h$ and $\pi \preceq \tau[1]$ or $\pi \succeq \tau[|\tau|]$. W.l.o.g assume that $\pi \preceq \tau[1]$. Then a rightmost symbol child of D, say μ, satisfies $\mu \preceq RL_{G'}(l)$, while μ_2 that is directly to the right of μ satisfies $\mu_2 \succeq LR_{G'}(r)$. Thus there are no symbol vertices "between" $RL_{G'}(l)$ and $LR_{G'}(r)$, the image is undefined.



Case 2. Every parent π of D such that height$(\pi) = h$ satisfies $\tau[1] \prec \pi \prec \tau[|\tau|]$. Then all children of D are "inside" the (l,r)-image in G'. Indeed, they are not j-successors for $j \leq l$ nor $j \geq r$ (see Proposition 1(b) and the definition of the image). Moreover, μ_1, μ_2 are vertices on borders of the (l,r)-image in G'. Thus the length of the image is not larger than $k+2 < |\tau| + k$.

(e) Assume that τ and τ' overlap by more than two symbols. Then, there exists a symbol vertex μ which is internal (neither first nor last) for both τ and τ'. So, by item (a) of this lemma, μ is j_1-successor and j_2-successor for some $l < j_1 < r$ and $l' < j_2 < r'$. But the fact that μ is j_1-successor for $j_1 < r$ implies that the rightmost vertex that is j-successor for any $j \leq l'$ is equal to μ or some vertex to the right of μ (because $r < l'$). So, there are no vertices in the (l', r')-image to the left of μ. Contradiction.

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

Proposition 4. *Let G, G' be computation graphs corresponding to a computation on an input word of length n, $G \vdash G'$, and $0 \leq l < r \leq n+1$. Assume that an (l,r)-image is undefined in G. Then, the (l,r)-image is undefined in G' as well.*

Proof. Let $\pi_l = RL_G(l)$, $\pi_r = LR_G(r)$. The (l,r)-image is undefined in G, so there is no symbol sink vertex $\pi \in G$ such that $\pi_l \prec \pi \prec \pi_r$. If the window does not contain π_l nor π_r in G then $RL_{G'}(l) = RL_G(l) = \pi_l$ and $LR_{G'}(r) = LR_G(r) = \pi_r$ and there is no symbol sink vertex in G' between π_l and π_r either. Now, assume that the window contains exactly one vertex from $\{\pi_l, \pi_r\}$. W.l.o.g. we can assume that the window contains only π_l. Let $D \in G' \setminus G$ be a new transition vertex. We consider two cases:

Case 1. $\pi_r \preceq \pi_l$.
Note that the vertex π_r is the sink of G'. By Definition 5, π_r is j_1-successor for some $j_1 \geq r$, so $\pi_r \succeq LR_{G'}(r)$. The vertex π_l is j_2-successor for some $j_2 \leq l$, so π_r is j_2'-successor for some $j_2' \leq j_2$ (by Proposition 1(d)) what implies that $\pi_r \preceq RL_{G'}(r)$ So, $LR_{G'}(r) \preceq \pi_r \preceq RL_{G'}()$, i.e. there is no symbol sink vertex between $RL_{G'}(l)$ and $LR_{G'}(r)$ in G'.

Case 2. $\pi_l \prec \pi_r$.
There is no sink symbol vertex $\pi \in G$ such that $\pi_l \prec \pi \prec \pi_r$, because the (l,r)-image

is undefined in G. Thus, π_l is the rightmost parent of D among symbol vertices. All other parents of D are to the left of π_l, so each parent of D is j-successor for some $j \leq l$ (Proposition 1(d)). So, by Proposition 1(b), each child of D is j-successor for some $j \leq l$. Thus, the rightmost symbol child of D is equal to $RL_{G'}(l)$ and $LR_{G'}(r) = LR_{G'}(r)$ – there is no symbol sink vertex in G' between $RL_{G'}(l)$ and $LR_{G'}(r)$.

Finally, assume that the window contains both π_l and π_r in G. As there are no symbol vertices between π_l and π_r in this case, the (l,r)-image is undefined what has been shown in Proposition 3(c,d), Case 1 (we can adapt that proof by assuming that $\tau[1] = \pi_l$ and $\tau[|\tau|] = \pi_r$). □

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

Lemma 4. *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 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 an $(g + l, g + r)$-image in G and an $(h + l, h + r)$-image in H are equivalent for every $0 \leq l < r \leq |Src(G_2)|$.*

In other words, an image of any subword of the input word included in G_2 is also included in G_2 and independent of the remaining part of the graph.

Proof. Let σ_1, σ_2 be paths on borders of G_2 in G. Let $(\sigma'_1, \sigma'_2, \tau)$ be an $(g + l, g + r)$-image in $G = G_1 G_2 G_3$. We show that it is contained in the subgraph G_2 and 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, a source of σ'_1 is to the right of a source of σ_1, moreover σ_1 and σ'_1 do not cross, because σ'_1 is the rightmost short path with appropriate sink (see Proposition 1(c) and the definition of the image). By the same arguments one can show that, σ'_2 is to the left of σ_2. Thus, also τ is contained in G_2. 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. All considerations in this section concern computations of such automaton.

As our proof strategy is based on analysis of computations on periodic inputs (see Section 4), we introduce and analyze some notions and properties concerning such computations. More precisely, we analyze computations on inputs of the form w^* for $w \in \Sigma^*$, where consecutive (non-overlapping) occurences of w are called *blocks*. The ith copy of w is called the ith block of the input word. 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, sources of σ_1 and σ_2 start at the same positions in appropriate blocks.

First, we define a technical notion of *periodicity property* that specifies conditions under which we treat a computation graph as periodic. Let shift(u,l) denote a left cyclic shift of the word u by l positions, i.e. if $u \in \Sigma^n$ then shift$(u,l) = u[l+1]\dots u[n]u[1]\dots u[l]$.

Definition 7 (Periodicity property). *A word $w \in \Sigma^m$ satisfies periodicity property with with respect to a tuple $(G_1, G_2, G_3, \alpha, r, j)$, where $G_1G_2G_3$ is a computation graph and $\alpha, r, j \in \mathbb{N}^+$ iff*

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

(a) $m\alpha \leq r \leq m^j$, α is odd, r is even.
(b) $Src(G_2) = (shift(w,a))^r$ for some $a \in \mathbb{N}$.
*(c) There exists a description γ such that $\langle\gamma\rangle_{i+1}$ decompose $G_1G_2^iG_3$ into $G_1, \langle G_2\rangle_i, G_3$ for every $i > 0$. The description γ is **associated** with the parameter $(G_1, G_2, G_3, \alpha, r, j)$ and the word w.*
(d) A γ-path on the right border of $G_1G_2^l$ is short in $G_1G_2^iG_3$ for every $0 \leq l \leq i$ and $i > 0$.

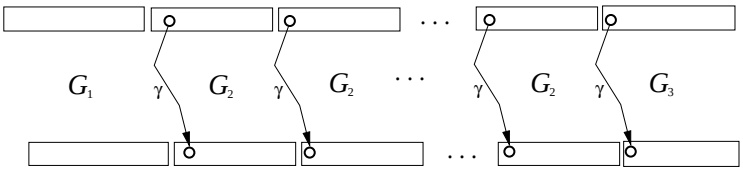


Fig. 7. Periodicity property.

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 that possibly lost a periodic structure. The item (a) ensures that these nonperiodic parts are much "smaller" than the periodic 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 the 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).

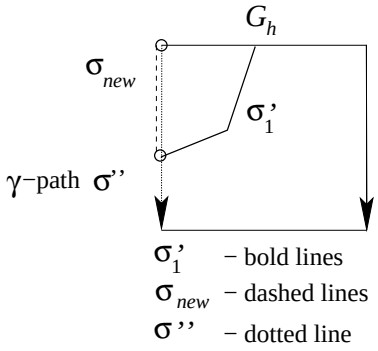
By adding a requirement that each block should have a long image, we define strong periodicity property.

Definition 8 (Strong periodicity property). *The word $w \in \Sigma^m$ satisfies strong periodicity property with respect to $(G_1, G_2, G_3, \alpha, r, j)$ if it satisfies periodicity property with respect to $(G_1, G_2, G_3, \alpha, r, j)$ and the image of each block is defined in $G_1G_2^iG_3$ for each $i > 0$, and its length is greater than $c'm$ in $G_1G_2^iG_3$, where c' is a fixed constant that depends only on the automaton M equal to $1/(8\lceil\log(|\Gamma| + |Q|\rceil))$.*

Now, we show a technical and intuitively clear result saying that if one applies Pumping Lemma for a decomposition of a graph based on short paths then paths defining an appropriate decomposition into "pumped" graphs are short as well. This fact strongly helps to show some properties of computations on periodic inputs.

Proposition 5. *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 $i \leq j$, a path on the right border of $G_1 G_2^i$ is short in the computation graph $G_1 G_2^j G_3$.*

Proof. (Sketch) Pumping Lemma implies that $\langle\gamma\rangle_{j+1}$ decompose $G_1 G_2^j G_3$ 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 $i \leq j$ such that the γ-path σ on the right border of $G_1 G_2^i$ is not short in the graph $G = G_1 G_2^j G_3$, i.e., there exists a path σ' in G that finishes 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 the shortest path in G which satisfies these 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 $G_1, \langle G_2\rangle_j, G_3$, 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). Now, let σ' be a path of length s satisfying above conditions, with minimal number of such subpaths. Then, σ'_1 starts in a source vertex and is included in G_h for $h \in \{1,2,3\}$. Let σ'' be a γ-path on a border of G_h at which σ' has its last vertex (see figure). Let σ_{new} be a subpath of σ'' that starts in a source and finishes in a 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 into σ_{new} obtaining a path 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 $G_1 G_2 G_3$. Thus, after replacing σ'_1 into σ'', we obtain a path with the same sink and length as σ'. However, this new path has a partition into smaller number of subpaths (contained in subgraphs G_l for $l = 1,2,3$) than a 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.



□

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

Proposition 6. *Assume that w satisfies 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 $\lceil(\alpha+r)/2\rceil \leq l \leq \alpha + ir - \lfloor(\alpha+r)/2\rfloor$. In particular, the middle block 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 images of blocks b and $b+2r$ in $G_1G_2^iG_3$ are equivalent.
(d) If there exists $a > 1$ such that an image of every block of $w^{\alpha+ar}$ is (defined and) longer than $c'm$ in $G_1G_2^aG_3$ then w satisfies strong periodicity property with respect to $(G_1, G_2, G_3, \alpha, r, j)$.

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

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

The second inequality follows from the fact that M is length reducing and the number of sources, $|Src(G_1)|$, is equal to the length of the input word plus 5; in the third inequality we use the fact that $\alpha \leq r \leq m^j$ by the item (a) of Definition 7.

(b) We show that the lth block is located to the right of the prefix G_1 (and one can show in analogous way that the lth block is to the left of G_3). Observe that

$$\begin{aligned} |Src(G_1)| &\leq |Src(G_1)| + |Src(G_3)| = (\alpha+ir)|w| - i|Src(G_2)| + 5 \\ &= m(\alpha+ir) - mir + 5 \ = m\alpha + 5. \end{aligned}$$

And the number of sources to the left of the lth block is

$$(l-1)m + 3 \geq ((\alpha+r)/2 - 1)\,m + 3 \geq ((\alpha+m\alpha)/2 - 1) \cdot m + 3 \geq m\alpha + 5$$

for each $l \geq \lceil(\alpha+r)/2\rceil$ (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. Let the block b be contained in b'th and $(b'+1)$th 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 positions of the blocks b and $b+2r$ in appropriate subgraphs G_2G_2 are equal. Thus, as all paths partitioning the graph are short by periodicity property, we can apply Lemma 4 to obtain equivalence of images.

(d) We have to show that the size of the image of each block w is greater than $c'm$ for each $b \geq 0$. 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, it follows from the fact that G_2 contains at least $2m$ sources by conditions (a) and (b) of periodicity property. As all paths on borders of each copy of G_2 are short (by periodicity property), the image of each block w is determined only by G_1G_2, G_2G_2 or G_2G_3 (Lemma 4). 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, all images of all blocks are longer than $c'm$ in $G_1G_2^bG_3$. □

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

Definition 9 (Opposite border graph). *Assume that w satisfies 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 (sinks of) the subgraph* G_1 *(*G_3 *resp.) for each* $i > 0$*. An opposite border graph with respect to* $G_1G_2^iG_3$ *for* $i > 0$ *is equal to* H' *such that* $G_1G_2^iG_3 \vdash_M^* G'G_3 \vdash_M H'$ *and at least one sink of* G_3 *(*G_1*) becomes an internal vertex in* H'*. In other words,* H' *is a first graph following* $G_1G_2^iG_3$ *in which the window touches* G_3 *(*G_1 *respectively).*

The condition (a) of the definition of periodicity property guarantees that the periodic part of the graph is large with respect to a "non-periodic" parts (G_1 and G_3). Especially, if the strong periodicity property is satisfied then the computation that "moves" 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 7. *Assume that w satisfies strong periodicity property with respect to* $(G_1, G_2, G_3, \alpha, r, j)$*. Let* $H(i)$ *be an opposite border graph with respect to* $G_1G_2^iG_3$ *for* $i > 0$*. Then, the computation*

$$G_1G_2^iG_3 \vdash_M^* H'$$

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 strong periodicity property. In other words, the configuration associated with* H' *is shorter by at least* $n \cdot c'/(8k)$ *from the configuration asscociated with* $G_1G_2^iG_3$*.*

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 7). Thus, by Lemma 4, it contains also 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 images of two consecutive blocks overlap by at most two sinks (Proposition 3(b)) and the image of each block is not shorter than $c'm$ (strong periodicity property). Next, $ir > \alpha$ by the condition (a) of the definition of periodicity property. In this way 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 $n \cdot c'/(8k)$. □

9 Proof of Theorem 1

Our proof exploits the notion of Kolmogorov complexity (cf. [16]). Recall that Kolmogorov complexity of a binary word x (denoted by $K(x)$), is 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 the length-reducing DTPDA $M = (Q, \Sigma, \Gamma, q_0, \bot, F, \delta)$ with the window of size k recognizes the language PAL. We will analyse computations of M on inputs which consists 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

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

Note that 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*. Let an image of the i-th block of $(ww^R)^j$ in a graph G be the $(m(i-1)+1, mi)$-image in G for $m = |ww^R|$.

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 strong periodicity property) and finishes in an opposite border graph, i.e., after moving the window through the whole periodic part. A crucial 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 at the end of each stage should be "long", in order to store long images of all blocks. This gives a contradiction.

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

Lemma 5 (Middle Block Lemma). *For every $d > 0$, there exists $m_d \in \mathbb{N}_+$ such that the following condition is satisfied for each $x \in \mathcal{W}_d$, where $x = (ww^R)^{2j-1}$, $|w| = m > m_d$, and $2j-1 \leq m^d$. If G is a computation graph corresponding to a computation of M on x and an image of a middle block is defined in G then images of all other blocks are also defined in G and the length of an image of each block (but the middle one, possibly) is greater than $c'm$, where $c' = 1/(8\lceil \log(|\Gamma| + |Q|)\rceil)$ (i.e., the constant c' from the definition of strong periodicity property).*

Proof. (Sketch) Recall that the polynomial bound (with respect to the length of a block) on the number of blocks guarantees that all short paths in the computation graph have only logarithmic length with respect to m. Using the fact that the image of the middle block is defined, we know that the image of each other block (if exists) cut out a subgraph which does not contain any vertex of the block symmetric to it (see properties of images, Proposition 3(e)). For the sake of contradiction assume that an image of the ith block is short in a graph describing a computation on $(ww^R)^{2j-1}$, $i \neq j$. Then, there exists another word, say y, of the length equal to the part of the input included in the image of the ith block that allows to construct a subgraph with the same sinks and paths on its border as in the image of the ith block. Indeed, otherwise the image would give a short description of w, what contradicts the assumption that w is incompressible. So, using Cut and Paste Lemma, we fool the automaton by replacing the subgraph corresponding

to the image of the ith block into appropriate subgraph containing y as the input (i.e. labels of consecutive source vertices). As we do not change the length of the input and positions symmetric to the ith block, the new input is not a palindrome. However, M accepts this input because we obtain the same configuration as on the palindrome from $\mathcal{W}_d$. The full formal proof is given in Subsection 9.1. □

By combining properties of periodic computation graphs described in Section 8 (in particular "parallelity" of images, Proposition 6(c)) with Middle Block Lemma, we show that strong periodicity property of a graph implies strong periodicity property in the opposite border graph (i.e. after moving the window through the whole periodic part), although the "level" of monotonicity is dropped down (see the growth of the last parameter, j in Lemma 6).

Lemma 6 (Periodicity Preserving Lemma). *For every $j \geq 1$, there exists $m'_j \in \mathbb{N}$ which satisfies the following conditions. Let $w \in \Sigma^*$ be an incompressible word such that $2|w| = m > m'_j$. Then, if the word ww^R satisfies 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 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 $i > 0$ and $H_1 H_2^i H_3$ is the opposite border graph with respect to $G_1 G_2^{g+if} G_3$.*

Proof. (Sketch) We show that the lemma is satisfied for each $m'_j > m_{80j}$ (and m larger that the constant that depends only on M), where m_l denotes the constant from Middle Block Lemma. So, the choice of m guarantees that Middle Block Lemma is applicable as long as $\alpha + ir \leq m^{80j}$. Let w, G_1, G_2, G_3, α, r, j and m satisfy the conditions stated in the lemma. W.l.o.g assume that the window in the configuration associated with $G_1 G_2^i G_3$ is located in G_1. Let $H(i)$ denote an opposite border graph with respect to $G_1 G_2^i G_3$. The outline of the proof is following:

1. We show that the image of the middle block of $(ww^R)^{\alpha+ir}$ is defined in $H(i)$ for every $i > 0$ such that $\alpha + ir \leq m^{40j}$. The proof is based on Middle Block Lemma, the strong periodicity of the configuration $G_1 G_2^i G_3$, and Proposition 6 (see Claim 1 in Section 9.2 for the full proof).
2. By the item 1 stated above, the image of each block (but the middle one, possibly) in $H(i)$ is longer than $c'm$ by Middle Block Lemma, if $\alpha + ir \leq m^{40j}$. Thus, the configurations associated with $H(i)$ are long (contains long images of blocks), we show by counting arguments that there exist two parallel short paths in $H(i)$ for some $i > 1$ such that $\alpha + ir \leq m^{5j}$ (see Claim 2 in Section 9.2).
3. Using Pumping Lemma and the existence of parallel paths in $H(i)$, we define subgraphs H_1, H_2, H_3 and numbers α', r', g, f such that ww^R satisfies periodicity property with respect to $(H_1, H_2, H_3, \alpha', r', 20j)$ where $H_1 H_2^i H_3$ is an opposite border graph with respect to $G_1 G_2^{g+if} G_3$ (for every $i > 0$), where $\alpha' = \alpha + gr$, $r' = fr$ (see Claim 3).
4. Finally, using item 1. above and Proposition 6 (in particular the facts (c)-(d) saying about "parallelity" of images of blocks), we show that ww^R satisfies **strong** periodicity property with respect to $(H_1, H_2, H_3, \alpha', r', 20j)$.

More details of the proof following this scenario are presented in Subsection 9.2. □

If we choose a long incompressible block w and appropriate parameters α, r, j, then strong periodicity property is satisfied for subgraphs G_1, G_2, G_3 which are subgraphs of the graph corresponding to the initial configuration. Then we apply Periodicity Preserving Lemma many times, obtaining consecutive "border" graphs (i.e. graphs describing configurations in which one pushdown is relatively short; interchangeably the left and the right one) where such periodic structure is preserved, and an image of each block is longer than $c'm$ in these graphs. On the other hand, M reduces the length of configurations in each step, what in consequence makes unable to preserve "long" images of blocks (required by Periodicity Preserving Lemma) and gives contradiction. Below, we present details of this idea.

Let $l \in \mathbb{N}$ be a constant, let $w \in \Sigma^*$ be an incompressible word such that $|ww^R| = m > m'_{20^{l+1}}$, where m'_j denotes the constant from Periodicity Preserving Lemma. The choice of $m > m'_{20^{l+1}}$ is done 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 an initial configuration on an input word $(ww^R)^{r_0+3}$. Let $G_{1,0}$, $G_{2,0}$, $G_{3,0}$ be subgraphs of G corresponding respectively to the first block ww^R, the next r_0 blocks and the last two blocks. One can easily verify that ww^R satisfies 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 $G_{1,i}, G_{2,i}, G_{3,i}$, α_i, r_i for $i \in \{1, \ldots, l\}$ such that ww^R satisfies 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) for any natural number p_l,

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

where $G_{1,0}G_{2,0}^{p_0}G_{3,0}$ is a computation graph associated with an initial configuration on an input word $(ww^R)^{\alpha_l + p_l r_l}$, $p_1 \geq \ldots p_{l-1} \geq p_l$ are some natural numbers (more precisely, $p_i = p_{i+1} \cdot \frac{r_{i+1}}{r_i} + \frac{\alpha_{i+1} - \alpha_i}{r_i}$ for $i < l$ by Periodicity Preserving Lemma).

(B) an 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 an opposite border graph with respect to $G_{1,i}G_{2,i}^{p_i}G_{3,i}$ for $i = 0, \ldots, l-1$; i.e. the window moves through the whole periodic infix $G_{2,i}^{p_i}$ during the subcomputation $G_{1,i}G_{2,i}^{p_i}G_{3,i} \vdash_M^* G_{1,i+1}G_{2,i+1}^{p_{i+1}}G_{3,i+1}$.

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 images of two consecutive blocks overlap by at most two symbols (Proposition 3(e)). So, the length of this configuration is larger than

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

where $n = (\alpha_l + p_l r_l)m$ is the length of the input word. On the other hand, a 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 7) for each $i = 0, \ldots, l-1$. As an 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 $l \geq 2 \cdot 8k/c'$. We get a 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 $m_d \in \mathbb{N}$, there exists an incompressible word w and a number $j \in \mathbb{N}$ such that

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

W.l.o.g. assume that $i < j$. Let G be a 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. Then, the image of the ith block is not shorter than $c'm-k$ in G, since an 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 cannot become undefined in one step if it is longer than $2k$ (Proposition 3(c,d)). Simultaneously, if the image of the middle block is defined in G, it has been defined during the whole computation described by G (Proposition 4).

Let $(\sigma_1, \sigma_2, \tau)$ be an image of the ith block in G, $\gamma_1 = \mathrm{desc}(\sigma_1)$, $\gamma_2 = \mathrm{desc}(\sigma_2)$, The path σ_1 is to the left of σ_2 (Proposition 3(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 positions of sources of σ_1, σ_2 in the input word (see figure below). By definition of the image, $p_1 \leq m(i-1)+1$ and $p_2 \geq mi$. We show that $p_2 < mj$ (i.e. p_2 is the position inside the middle block or to the left of it). Indeed, the image of the middle block is defined, so the rightmost vertex of τ is to the left of the rightmost vertex of the image of the middle block (otherwise images of the ith and the jth block overlap by more than two symbols, contradicting Proposition 3(d)). And there is no p-successor for each $p \geq mj$ to the left of the rightmost vertex of the image of the jth block, by definition of the image (see Proposition 3(a)). So, $p_2 < 2mj$ and G_2 does not contain any source vertex to the right of the middle block.

We claim that γ_1, γ_2, $\omega(\tau)$, p_1, p_2, j and M describe w. We provide an algorithm that determines w on base of these parameters. For every $v \in \Sigma^{m/2}$ we inspect all computation graphs that desribe computation of M on $x = (vv^R)^{2j-1}$ until we find such v and a computation graph G' describing a computation on x such that:

- γ_1, γ_2 decompose G' into G'_1, G'_2, G'_3,
- $Snk(G'_2) = \omega(\tau)$,
- the position of a source of a path on the border between G'_1 and G'_2 is equal to p_1 and the position of a source of a path on the border between G'_2 and G'_3 is equal to p_2.

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 figure below). 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 a palindrome $(ww^R)^{2j-1}$. So, M accepts also an input word z of the graph G'', obtained from $(ww^R)^{2j-1}$ by replacing its subword $(ww^R)^{2j-1}[p_1, p_2]$ by the appropriate subword of $(vv^R)^{2j-1}$. Thus, z is a palindrome. But $p_2 \leq 2mj$, so all blocks to the right of the middle

block remain unchanged, i.e. equal ww^R. On the other hand $p_1 \leq 2m(i-1)+1$, so the ith block ww^R is replaced by vv^R in z. Thus $v = w$, since z is the palindrome.

Now, we get a contradiction with incompressibility of w. We need only $O(\log n)$ bits in order to store $\gamma_1, \gamma_2,\ p_1, p_2,\ j$ and M, where n is the length of the input word $(ww^R)^{2j-1}$ (since γ_1, γ_2 describe short paths – see Corollary 3). The value $n = |ww^R|(2j-1) = m(2j-1) \leq m^{d+1}$, since $(ww^R)^{2j-1} \in \mathcal{W}_d$. So, all these data require $O(\log n) = O(\log m^{d+1}) = O(\log m)$ bits. Additionally, $|\omega(\tau)| < c'm$ and $\omega(\tau)$ is a word over alphabet $\Gamma \cup Q$. In order to store it binary we need at most $c'm\lceil \log(|\Gamma \cup Q|)\rceil \leq m/8$ bits. Together, we need $m/8 + O(\log m) < m/4$ bits for m large enough. This contradicts the assumption that w is incompressible (i.e., $K(w) > m/2 - 1$).

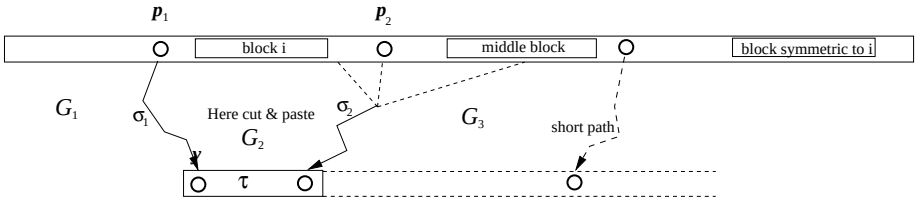


Fig. 8. Middle Block Lemma.

9.2 Proof of Periodicity Preserving Lemma

We show that the lemma is satisfied for each $m'_j > m_{80j}$ (and m is larger that the constant that depends only on M), where m_l denotes the constant from Middle Block Lemma. Assume that a word ww^R satisfies 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 $m > m'_j$. Let $H(i)$ denote an opposite border graph with respect to $G_1 G_2^i G_3$ (see Definition 9).

Claim 1 *For every $i > 1$ such that $\alpha + ir \leq m^{40j}$, 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 $\alpha + ir \leq m^{40j}$. Then there exists G' such that

$$init(ww^R)^{\alpha+ir} \vdash_M^* G_1 G_2^i G_3 \vdash_M^* G' G_3 \vdash_M^* H(i)$$

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 $G'G_3$ (see Proposition 3(c,d)). Let i' be a natural number such that $\alpha + i'r \leq m^{80j}$ and $(\alpha + i'r)/2 > m^{40j+2}$. Then

$$(ww^R)^{\alpha+i'r} \vdash_M^* G_1 G_2^i G_2^{i'-i} G_3 \vdash_M^* G' G_2^{i'-i} G_3,$$

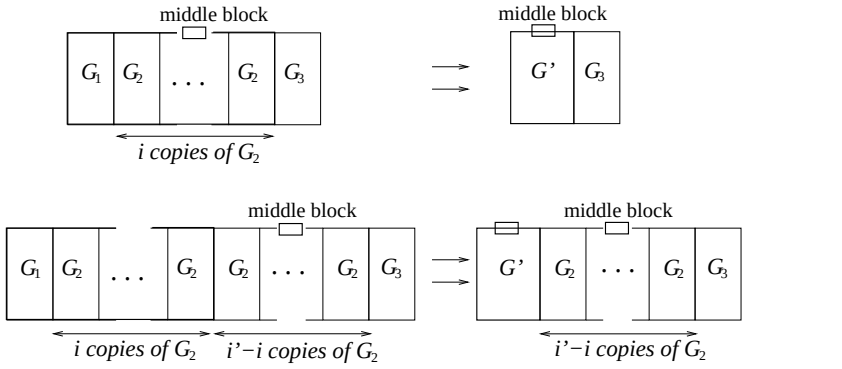
where the first subcomputation is obtained by strong periodicity property and the second subcomputation is obtained by Cut and Paste Lemma (see figure). Moreover,

(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,
$$(\alpha+i'r)/2-1 > m^{40j+2}\cdot m > (\alpha+ir)m,$$
so the number of sources of $G_1G_2^iG_3$ is smaller than the number of sources preceding the middle block of $(ww^R)^{\alpha+i'r}$ (recall that the automaton is length-reducing).

(b) The image of the middle block of $(ww^R)^{\alpha+i'r}$ is defined and 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 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 3(b).

(c) 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 shorter thatn $c'm-k$.
Indeed, this block is included in the prefix $G_1G_2^i$ of the graph $G_1G_2^iG_3$ (as well as $G_1G_2^{i'}G_3$) by Proposition 6(b). And, as the path on the border between $G_1G_2^i$ and G_3 (or $G_2^{i'-i}G_3$) is short, it remains short 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 4. So, they are shorter than $c'm-k$ by the assumption.

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



□

With help of the above claim, we show that the graphs $H(i)$ are really periodic. Our argument is based on the fact that, by Middle Block Lemma, the above claim and Proposition 3(e), the lengths 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 allows us to prove (by Heights Lemma 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 2 *There exists $i>1$ such that $\alpha+ir \le m^{5j}$ and there are two parallel short paths σ_1,σ_2 in the computation graph $H(i)$ with sinks located to the left of the window in $H(i)$. The distance (number of sinks) between sinks of σ_1,σ_2 is not smaller than k.*

Proof: Take i such that $\alpha+ir \leq m^{5j}$ and $\alpha+ir > m^{j+2}$ (note that, $|Src(G_1)|+|Src(G_2)| \leq m^{j+2}$ by Proposition 6(a)). 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 in $H(i)$ longer than two, by Claim 1. Moreover, we took $m > m_{80j}$, so Middle Block Lemma may be applied here. Thus, $|Snk(H(i))| > (\alpha+ir)c'm/4 = c'n/4$, because images of all blocks (but the middle one, possibly) are longer than $c'm$ and images of disjoint blocks may overlap by at most two (Proposition 3(e)). As $H(i)$ is a graph which corresponds to the first configuration in which the window touches G_3, there are at most

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

sinks to the right of the window in $H(i)$ (Proposition 6(a)). For m large enough, this number is smaller than $n \cdot c'/32$. Let $h = \lceil \log(c'/96)/\log(k/(k+1)) \rceil$. By Heights Lemma, the number of sinks of height greater or equal to h is in $H(i)$ less than $6n(k/(k+1))^h \leq nc'/16$. So, at least

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

sinks located to the left of the window have height less or equal to h. Let S be a set of these sinks. Let us choose the leftmost short path with the sink in s for each $s \in S$. Partition these paths into m groups such that the pth group contains paths that start at the pth position of any block (i.e., at the position equal to $p \bmod m$). 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 a constant independent of m, the number of possible descriptions 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 $\leq h$). And, by the choice of the group, these paths are parallel. □

Now, we are ready to show that

Claim 3 *There exist subgraphs H_1, H_2, H_3 such that the word ww^R satisfies 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, $H_1 H_2^l H_3$ is the opposite border graph with respect to $G_1 G_2^{g+lf} G_3$ for every $l > 0$,.*

Proof. By Claim 2, there exists i such that $\alpha+ir \leq m^{5j}$ and there are two parallel short paths σ_1, σ_2 in $H(i)$ such that $\mathrm{desc}(\sigma_1) = \mathrm{desc}(\sigma_2) = \gamma$. 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 G'_1, G'_2, G'_3 are subgraphs of $H(i)$ bordered by short paths σ_1 and σ_2. Thus, by Pumping Lemma, $\langle\gamma\rangle_{j+1}$ decompose a computation graph $G'_1 {G'_2}^j G'_3$ into $G'_1, \langle G'_2\rangle_j, G'_3$. Recall that σ_1, σ_2 on borders of G'_2 are short in $G'_1 G'_2 G'_3$ (Claim 2). So, by Proposition 5, paths on borders of the decomposition $G'_1, \langle G'_2\rangle_j, G'_3$ are short in $G'_1 {G'_2}^j G'_3$ 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 $a \leq \alpha + ir \leq m^{5j}$. So, the initial configuration of $G'_1 G'^{l}_2 G'_3$ corresponds to an input $(ww^R)^{(\alpha+ir)+a(l-1)}$ (because $Src(G'_1 G'_2 G'_3) = Src(H(i))$ 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 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 $\alpha' = \alpha + ir$, $r' = mra\alpha'$, $H_1 = G'_1 G'_2$, $H_2 = (G'_2)^{r'/a}$, $H_3 = G'_3$. Then:

- $\langle\gamma\rangle_{l+1}$ decompose $H_1 H_2^l H_3$ into $H_1, \langle H_2\rangle_l, H_3$, and paths on borders of this decomposition are short, (since paths on borders of decompositions $G'_1, \langle G'_2\rangle_*, G'_3$ are short, as showed above),
- $H_1 H_2^l H_3 = G'_1 G_2'^{r'l/a+1} G'_3$ and the initial configuration of the graph $H_1 H_2^l H_3$ corresponds to the input word $(ww^R)^{\alpha+ir+r'l} = (ww^R)^{\alpha'+lr'}$.
- $Src(H_2) = (Src(G'_2))^{r'/a} = \text{shift}((ww^R, p))^{r'}$
- $\alpha' = \alpha + ir \leq m^{5j} \leq m^{20j}$; moreover $r' = mra\alpha' \geq m\alpha'$ and $r' = mra\alpha' \leq m \cdot m^j(\alpha + ir)^2 \leq m^{1+j+10j} \leq m^{20j}$.
- $init((ww^R)^{\alpha'+lr'}) \vdash_M^* H_1 H_2^l H_3$ for every $l > 0$.

These conditions certify that ww^R satisfies 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 $H_1 H_2^l H_3$ is in fact an opposite border graph with respect to $G_1 G_2^{g+lf} G_3$ for each $l > 0$. It means that all sinks of G_3 should remain as sinks in a graph preceding $H_1 H_2^l H_3$ by one derivation step, and at least one of them should be an internal vertex of $H_1 H_2^l H_3$ (see Definition 9). Recall that $H_1 H_2^l H_3 = G'_1 (G'_2)^{r'l/a+1} G'_3$. Moreover, as $G'_1 G'_2 G'_3$ is the opposite border graph with respect to $G_1 G_2^i G_3$, there exists a subgraph J such that

$$G_1 G_2^i G_3 \vdash_M^* G'_1 J \vdash_M G'_1 G'_2 G'_3,$$

all sinks of G_3 remain the sinks of $G'_1 J$ and they belong to the subgraph J. Moreover, at least one of them is the internal vertex of $G'_1 G'_2 G'_3$. Indeed, the subgraph G'_1 appears already in the graph directly preceding $G'_1 G'_2 G'_3$, because G'_2 contains at least k sinks, while the window length is equal to k and the window is located in G'_3 (see Claim 2). 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 a computation

$$G'_1 (G'_2)^{r'l/a} J \vdash_M G'_1 (G'_2)^{r'l/a+1} G'_3 = H_1 H_2^l H_3,$$

because the description of the path on the right border of G'_1 and 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 $H_1 H_2^l H_3$. □

Finally, we show that ww^R satisfies **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 3. By Claim 3 we know that $H_1 H_2^l H_3 = H(g + lf)$ for every $l > 0$. Let us choose $l > 2$ such that $\alpha' + lr' = \alpha + (g + lf)r \leq m^{40j}$. (Such l exists by the fact that $\alpha', r' \leq m^{20j}$ – see Definition 7.) Then, the image of the middle block is defined in $H_1 H_2^l H_3$, by Claim 1, and images of all blocks (except the middle block, possibly) are defined and longer then

$c'm$ by Middle Block Lemma (as the choice of $m > m_{80j}$ and l such that $\alpha' + lr' \leq m^{40j}$ 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. Then, by Proposition 6(b), the block $\lceil(\alpha' + lr'/2)\rceil$ (the middle block) is included in the infix H_2^l. So, by Proposition 6(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 6(d).

10 Conclusions and Open Problems

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, what solves the open problem from [19]. 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 broader class of problems concerning language classes related to or defined by length-reducing two-pushdown automata. In particular, one can obtain a short and direct proof that the language $\{ww \mid w \in \Sigma^*\}$ is not growing-context sensitive. A generalization of this proof might for example help to answer the question whether there exists a strict intersection hierarchy for GCSL [5].

Acknowledgements. The authors thank Friedrich Otto and Gundula Niemann for helpful comments on the draft of this paper.

References

1. M. Beaudry, M. Holzer, G. Niemann, F. Otto, *McNaughton families of languages*, Theoretical Computer Science 290(3): 1581-1628 (2003).
2. R.V. Book, *Grammars with Time Functions*, Dissert., Harvard University, Cambridge, MA.
3. G. Buntrock, *Wachsende Kontextsensitive Sprachen*, Habilitationsschrift, Würzburg, 1995.
4. G. Buntrock, K. Loryś, *On growing context-sensitive languages*, Proc. of International Colloquium on Automata, Languages and Programming (ICALP), 1992, LNCS 623, 77–88.
5. G. Buntrock, K. Loryś, *The variable membership problem: Succintness versus complexity*, Proc. of STACS, 1994, LNCS 775, 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*, Journal of Computer Systems Sciences , 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. Lane A. Hemaspaandra, Proshanto Mukherji, Till Tantau, *Computation with Absolutely No Space Overhead*, In: Zoltan Esik, Zoltan Fulop (Eds.): Developments in Language Theory, 7th International Conference (DLT 2003), Szeged, Hungary, July 7-11, 2003, Proceedings. Lecture Notes in Computer Science 2710, 325–336.
13. Markus Holzer, Martin Kutrib, *Flip-Pushdown Automata: k+1 Pushdown Reversals Are Better than k*, In Proc. Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, Gerhard J. Woeginger (Eds.): Automata, Languages and Programming (ICALP 2003), Eindhoven. Lecture Notes in Computer Science 2719, 490–501.
14. Markus Holzer, Friedrich Otto, *Shrinking Multi-pushdown Automata*, Proc. of Fundamentals of Computation Theory, International Symposium (FCT), 2005, LNCS 3623, 305–316.
15. T. Jurdzinski, K. Lorys, *Church-Rosser Languages vs. UCFL*, in Proc. of International Colloquium on Automata, Languages and Programming (ICALP), 2002, LNCS 2380, 147–158.
16. M. Li, P. Vitanyi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag 1993.
17. J. Loecks, *The parsing of general phase-structure grammars*, Information and Control 16 (1970), 443–464.
18. 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, Berlin, 1999, 204–212.
19. R. McNaughton, P. Narendran, F. Otto, *Church-Rosser Thue systems and formal languages*, Journal of the Association Computing Machinery, 35 (1988), 324–344.
20. G. Niemann, *Church-Rosser Languages and Related Classes*, PhD Thesis, Universitaet Kassel, 2002.
21. 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; 49–62.
22. 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.
23. G. Niemann, F. Otto, *Restarting Automata and Prefix-rewriting Systems*, Mathematische Schriften Kassel 18/99, Universitt-GH Kassel, Dezember 1999.
24. G. Niemann, Jens R. Woinowski, *The growing context-sensitive languages are the acyclic context-sensitive languages*, in Proc. Developments in Language Theory, 5th International Conference (DLT 2001), Vienna, Austria, July 16-21, 2001. LNCS 2295, 197–205.
25. Alexander Okhotin, *An overview of conjunctive grammars*, Formal Language Theory Column. Bulletin of the EATCS 79, 145-163 (2003).
26. Alexander Okhotin, *Boolean grammars*, Information and Computation 194(1): 19-48 (2004)
27. Palash Sarkar, *Pushdown Automaton with the Ability to Flip its Stack*, Electronic Colloquium on Computational Complexity (ECCC)(081): (2001).
28. Jens R. Woinowski, *Prefix Languages of Church-Rosser Languages*, Foundations of Software Technology and Theoretical Computer Science, 20th Conference (FSTTCS), 2000, LNCS 1974, 516–530.
29. Jens R. Woinowski, *The context-splittable normal form for Church-Rosser language systems*, Information and Computation, 183(2), 2003, 245–274.