Noname manuscript No.
(will be inserted by the editor)

(Parallel) Efficient Subclasses Of CFL-Reachability Queries

Ekaterina Shemetova · Semyon Grigorev

Received: date / Accepted: date

Abstract The context-free language (CFL) reachability problem for a context-free grammar G and directed edge-labelled graph D consists of determining for every pair of nodes v and u whether v can reach u via a path labelled by a string in $L(G)$. Whereas it has been shown that CFL-reachability problem is P-complete, there are subclasses of context-free languages and graphs, for which CFL-reachability lies is in NC complexity class.

We investigate parallel complexity of variants of CFL-reachability problem with interesting subclasses of context-free grammars and graphs. A Boolean circuit for this problem is presented. We obtain characteristic affecting the effectiveness of Boolean circuit and observe lower and upper bounds on its value.

As the result, we generalize known cases of context-free languages and graphs, for which CFL-reachability problem is in NC class; new examples of such subclasses are also obtained. We demonstrate connection between different restrictions on pushdown automata or graph and effectiveness of an appropriate Boolean circuit.

Particularly, CFL-reachability problem for languages of finite-turn pushdown automata, oscillation-bounded automata, one-counter automata and for acyclic graphs with self-loops lies is in NC and, hence, can be effectively parallelized.

Keywords CFL-reachability · parallel complexity · Boolean circuit · graphs · regular languages · context-free languages · context-free path queries

E. Shemetova
St. Petersburg Academic University, ul. Khlopina, 8, Saint Petersburg 194021, Russia, and JetBrains Research
E-mail: katyacyfra@gmail.com

S. Grigorev
St. Petersburg State University, 7/9 Universitetskaya nab., Saint Petersburg 199034, Russia, and JetBrains Research
E-mail: s.v.grigoriev@spbu.ru

1 Introduction

The context-free language (CFL) reachability is an important problem underlying some fundamential static code analysis like inter-procedural data flow analysis, inter-procedural slicing and other [9, 34, 28], and graph database query evaluation [18, 44, 16, 3]. CFL-reachability is some kind of graph reachability problem with path constraints in terms of context-free languages. Unlike context-free language recognition, which is in NC, CFL-reachability is P-complete [43, 33]. Practically, it means that there is no efficient parallel algorithm for solving this problem (unless P $\neq$ NC).

There are two important models for parallel computation: circuits and PRAMs. Complexity class NC is a class of problems which can be parallelized effectively. It is closely related with the boolean circuit compexity. Also it is known that PRAM algorithm can be simulated by a circuit family. In this work we will use the computational model of boolean circuits.

While problem is not parallelizable in general, it is useful to develop more efficient parallel solutions for interesting classes of grammars and graphs. For example, there are context-free languages which admit more efficient parallel algorithms in compare with the general case of context-free recognition [21, 22]. The same holds for CFL-reachability problem: there are some examples of context-free languages and graphs, for which CFL-reachability problem lies in NL space complexity class (for example linear languages, trees, acyclic graphs) [36, 24]. It is well known that $NL \subseteq NC^2$, so an effective parallel circuit for some subclasses of graph and context-free languages exists. Above mentioned works study space complexity, so they don't purpose any parallel algorithms for an "easy" subclasses.

Our focus is on investigating the parallel complexity of CFL-reachability. Especially we are interested in generalization of "easy" subclasses and discovering new examples of context-free grammars and graphs, for which CFL-reachability is in NC. Effective subclasses can be useful in practice, because general problem is hard [26]. Also it is natural to ask which properties of such subclasses implies P-completeness or effectiveness of an appropriate circuit. [36, 1] mentions a rational index and a fringe property as an example of such properties, but there are no characteristics in terms of pushdown automata, grammar or graph for subclasses with the good rational index and fringe property. The second goal of this work is to construct effective circuit for solving CFL-reachability problem for an "easy" cases.

Related work. CFL-reachability can be considered as an extention of graph reachability problem with path constraints in terms of context-free languages or as context-free recognition problem with non-linear, graph input. Graph reachability is a classical example of highly parallelizable problem, it lies in NC^2 complexity class [19]. It is well-known that context-free recognition is in NC^2 too [8, 37]. Moreover, there are subclasses of context-free languages, like unambigious context-free languages, visibly pushdown languages and other, which in NC^1 [21, 22, 35, 30].
However, CFL-reachability problem has long been known to be P-complete

[14]. A parallel complexity of this problem is studied by both static code analysis [34, 33] and database communities [1, 43, 40]. There are works investigating computational complexity of the language reachability for different variants of languages (regular, context-free) and graphs (acyclic graphs, trees, grid graphs)[4, 24, 20]. In [36] focus is on the context-free languages and their subclasses with respect to the rational index measure ([6]).
On the other hand, CFL-reachability problem can be observed as an intersection non-emptiness problem for context-free language (pushdown automaton) and regular language, because labelled graph is a special kind of nondeterministic finite automata. Complexity of such problem is studied in [39, 41, 11].

Our contributions. Our results can be summarized as follows:

- We adapt the Brent-Goldschlager-Rytter parallel algorithm for solving CFL-reachability problem. The resulting circuit has polynomial number of elements and its depth (hence, parallel time) depends on the special parameter denoted by $\mathcal{L}$. $\mathcal{L}$ is defined as follows. Fix two nodes i and j of the input graph. For a grammar $G = (\Sigma, N, P)$ consider the shortest path between i and j, such that a word obtained from concatenation of edge labeles of this path is in $L(G_A)$ (assume that such path exists), where $A \in N$ and G_A is the input grammar G with fixed start nonterminal A. $\mathcal{L}$ is the maximum length of such paths in all triples (A, i, j). We show that context-free language or graph with polynomial on the number of input graph nodes $\mathcal{L}$ lies in NC complexity class.
  Notice that the said parameter $\mathcal{L}$ is also useful for estimating the complexity of context-free path queries with single-path query semantics [18]. While CFL-reachability problem consists of finding the binary relation between pairs of graph nodes, the mentioned queries evaluate to a single path, which provide a proof that two nodes is related. For example, if there is a path π between graph nodes i and j, such that a concatenation of edge labeles $l(\pi)$ of π is in $L(G)$, CFL-reachability algorithm should return true value for i and j. Context-free path query with single-path query semantics evaluates to π in this case. Clearly, $\mathcal{L}$ is exactly an upper bound on the minimum length of such path for all pair of graph nodes and all nonterminals of the input grammar.
- We show that languages of pushdown automata with a natural restrictions on stack, like finite-turn pushdown automata (linear, metalinear, superlinear languages), oscillation-bounded automata (oscillation-bounded languages) and automata with a single stack symbol (one-counter languages) have polynomial $\mathcal{L}$, and hence, CFL-reachability for such languages and an arbitrary graphs is in NC^2. The same holds for the substitution closures of such languages. In contrast, languages with deterministic properties, like LL(k) languages, have exponential $\mathcal{L}$ in the worst case, which implies P-completeness of CFL-reachability for these languages.
- We obtain that CFL-reachability problem for directed acyclic graphs with loops, which is associated with partially ordered finite automata, and for

arbitrary context-free languages lies in NC^2 complexity class. It means that partial order on the input graph nodes implies polynomial $\mathcal{L}$.

Organization. This paper is organized as follows. In Section 2, we present the basic definitions from formal languages and complexity theory used in this paper. In Section 3 we present Boolean circuit for solving CFL-reachability problem and bounds on size and depth of the resulting circuits. In Section 4 we investigate bounds on $\mathcal{L}$ for interesting classes of context-free languages and arbitratrary graphs. In Section 5 our focus is on the estimating $\mathcal{L}$ for arbitrary context-free languages and fixed subtypes of graphs. We conclude in Section 6 with a summary of our findings.

2 Preliminaries

Formal languages A *context-free grammar* is a triple $G = (\Sigma, N, P)$, where Σ is a finite set of alphabet symbols, N is a set of nonterminal symbols, P is a set of production rules. $L(G)$ is a context-free language generated by context-free grammar G. Notice that usually context-free grammar is defined with a fixed start nonterminal. G_A denotes a context-free grammar with fixed start nonterminal $A \in N$, $L(G_A)$ is a language generated by a grammar G_A, respectively. We use the notation $A \overset{*}{\Rightarrow} w$ to denote that the string $w \in \Sigma^*$ can be derived from a non-terminal A by sequence of applying the production rules from P. A *parse tree* is an entity which represents the structure of the derivation of a terminal string from some nonterminal.

A grammar G is said to be is in the *Chomsky normal form*, if all production rules of P are of the form: $A \to BC$ or $A \to a$, where $A, B, C \in N$ and $a \in \Sigma$.

A context-free grammar is in the *Greibach normal form*, if all production rules are of the form: $A \to aA_1A_2...A_k$, where A is a nonterminal symbol, a is a terminal symbol, $A_1A_2...A_k$ is a (possibly empty) sequence of nonterminal symbols.

Notice that the rules of the form $A \to \varepsilon$, where ε denotes an empty string, are omitted. This is because it is trivial to find an empty string in the case of graph input.

Context-free languages are accepted by pushdown automata (PDA). *Pushdown automaton* is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, where Q is a finite set of states, Σ is a input alphabet, Γ is a finite set which is called the stack alphabet, δ is a finite subset of $Q \times (\Sigma \cap \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$, $q_0 \in Q$ is the start state, $Z \in \Gamma$ is the initial stack symbol and $F \subseteq Q$ is the set of accepting states.

Some operations on languages will be mentioned during this paper.
A *homomorphism* is a function $h : \Sigma^* \to \Delta^*$ defined as follows:

- $h(\varepsilon) = \varepsilon$ and for $a \in \Sigma$, $h(a)$ is any string in Δ^*,
- for $a = a_1a_2...a_k \in \Sigma^*$ $(k \geq 2)$, $h(a) = h(a_1)h(a_2)...h(a_k)$.

Given a homomorphism $h : \Sigma^* \to \Delta^*$ and a language L define $h(L) = \{h(w) \mid w \in L\} \subseteq \Delta^*$.

Substitution closure of a class of languages L is the least class containing all substitutions of languages from L to the languages from L.

A *regular language* is a language that can be expressed with a regular expression or a deterministic or non-deterministic finite automata. An *nondeterministic finite automata* (NFA) is represented by a 5-tuple, $(Q, \varSigma, \varDelta, q_0, F)$, where Q is a finite set of states, $\varSigma$ is a finite set of input symbols, $\varDelta : Q \times \varSigma \to 2^{|Q|}$ is a transition function, $q_0 \in Q$ is a start state, $F \subseteq Q$ is a set of accepting (final) states. *Deterministic fininite automaton* is NFA with the following restrictions: each of its transitions is uniquely determined by its source state and input symbol, and reading an input symbol is required for each state transition.

A rational transducer M is a 6-tuple $(Q, \varSigma, \varDelta, \tau, q_0, F)$ where Q is a finite set of states, $\varSigma$ and $\varDelta$ are the input and output alphabets respectively, and is as a finite subset of $S \times \varSigma^* \times S \times \varDelta^*$, $q_0 \in Q$ is initial state and $F \subseteq Q$ is non-empty set of finite states.

Given a rational transducer M, for every input word w, *rational trunsduction RT_M* is a function defined as follows: $RT_M(w) = \{v \in \varDelta^* | (t, v) \in \tau(q_0, w)$ is a final output configuration$\}$.

CFL-reachability A *directed labelled graph* is a triple $D = (Q, \varSigma, \delta)$, where Q is a finite set of nodes, $\varSigma$ is a finite set of alphabet symbols, and $\delta \subseteq Q \times \varSigma \times Q$ is a finite set of labeled edges.

Let $i\pi j$ denote a unique path between nodes i and j of the input graph and $l(\pi)$ denote a unique string which is obtained from concatenation of edge labels along the path π. Then the general formulation of CFL-reachability can be stated as follows.

Definition 1 Let $G = (\varSigma, N, P)$ be a context-free grammar and $D = (Q, \varSigma, \delta)$ be a directed labelled graph. Given two nodes i and j we say that j is *reachable* from i if there exists a path $i\pi j$, such that $l(i\pi j) \in L(G)$.

When this problem is restricted to some language L (not necessary context-free), it is called *L-reachability*. There are four varieties of CFL-reachability problems: all-pairs problem, single-source problem, single-target problem and single-source/single-target problem [34]. In this paper we consider all-pairs problem. The *all-pairs problem* is to determine all pairs of nodes i and j such that j is reachable from i. Notice that our definition of context-free grammar doesn't include a start nonterminal: therefore we are interested in reachability for every $L(G_A)$, where $A \in N$. In other words, we need to find all triples (A, i, j) for which j is reachable from i via path labelled from a string from $L(G_A)$.

Boolean circuits and complexity classes *Boolean circuit* is a directed acyclic graph, along with various labels associated with its nodes, satisfying the following constraints:

- There are set of *input nodes* with n members, labelled $x_1, ..., x_n$. Each input node and must have in-degree 0.

- There are set of *gate nodes (gates)*. Each gate node of the circuit is labelled by an element of the set $\{\vee, \wedge, \neg\}$.
- There are set of *output nodes* with m members, labelled $y_1, ..., y_m$.
- All three sets above are disjoint.

A Boolean circuit having n inputs and m outputs computes a Boolean function of the form: $f : \{0,1\}^n \to \{0,1\}^m$.

Two important characteristics of the Boolean circuit are its size and depth. *Size* of the Boolean circuit is number of its elements, and *depth* is the length of the longest path in the circuit from an input to an output node.

NC^i is the complexity class of decision problems decidable by uniform Boolean circuits with a polynomial number of gate nodes of at most two inputs and depth $O(\log^i n)$, or the class of decision problems solvable in time $O(\log^i n)$ on a parallel computer with a polynomial number of processors. NC is the union of all NC^i.

A decision problem is P-complete (complete for the complexity class P) if it is in P and every problem in P can be reduced to it by an appropriate reduction. Practically, P-complete problems are problems, which are difficult to parallelize effectively.

3 Boolean circuit for CFL-reachability

In this section we build generalization of classical Boolean circuit for context-free recognition from the Brent-Goldschlager-Rytter algorithm [8, 37] in order to evaluate CFL-reachability problem.

3.1 Circuit description

The idea of the Brent-Goldschlager-Rytter parallel algorithm is based on replacement of ordinary parse trees with a equivalent system of logical dependencies. We can adapt this system for graph input as follows (context-free grammar is in the Chomsky normal form):

1. $\frac{i \xrightarrow{a} j}{A(i,j)}$ — if edge from the node i to the node j has label "a" and $A \to a \in P$, then there is a parse tree with nonterminal A at the root deriving a path from i to j.

2. $\frac{B(i,j)}{\frac{A}{C}(i,j::z)}$ — creating a gap on the right: if a parse tree with the nonterminal B at the root deriving a path from i to j exists and $A \to BC \in P$, then a parse tree with nonterminal A at the root containing a smaller subtree with the nonterminal C deriving path from j to z ("gap"), where $1 \leq z \leq n$, may exist.

3. $\frac{C(j,z)}{\frac{A}{B}(i::j,z)}$ — creating a gap on the left: if a parse tree with the nonterminal C at the root deriving a path from j to z exists and $A \to BC \in P$, then a parse tree with nonterminal A at the root containing a smaller subtree with the nonterminal B deriving a path from i to j ("gap"), where $1 \leq i \leq n$, may exist.

4. $\frac{B(i,j), \frac{A}{B}(i::j,z)}{A(i,z)}$ — filling the gap: if a parse tree with the nonterminal A at the root deriving a path from i to z, which cointains the gap represented by a node labelled B deriving a path from i to j and a parse tree with the nonterminal B at the root deriving a path from i to j exist, then the whole parse tree with the nonterminal A at the root deriving a path from i to z exists.

5. $\frac{\frac{A}{E}(i,j::w,z), \frac{E}{D}(j,u::v,w)}{\frac{A}{D}(i,u::v,z)}$ — combination of conditional propositions: if there is a parse tree with the nonterminal A at the root deriving a path from i to z with the gap represented by a node labelled E deriving a path from j to w and there is a parse tree with the nonterminal E at the root deriving a path from j to w, which contains gap represented by some node D, then there is a parse tree with the nonterminal A at the root deriving a path from i to z, which contains the gap represented by the above mentioned node D.

Using logical dependencies above, elements of the circuit can be described with two types of elements: $x_{A,i,j}$, which is true when a parse tree with the nonterminal A at the root deriving a path from i to j exists, and $y_{A,i,j,B,k,l}$ for conditional propositions, which is true when there is a parse tree with the nonterminal A at the root deriving a path from i to j containg a hole instead of a subtree with the nonterminal B at the root deriving a path from k to l.

Therefore the circuit contains the following gates:

- *Input*: input gates, one gate for every edge of the input graph and alphabet symbol; evaluates to true if the corresponding edge has an appropriate label
- internal AND-gates for evaluating the combinations of conditional propositions and filling the gaps
- internal OR-gates for creating gaps: conditional proposition $y_{A,i,j,E,k,l}$ can be obtained by taking the gap from the right subtree (if $x_{B,i,z}$ is true and $y_{C,z,j,E,k,l}$ is true) or by taking the gap from the left subtree ($y_{B,i,z,E,k,l}$ is true and $x_{C,z,j}$ is true) for rule $A \to BC \in P$. Notice that if start and end nodes of the tree with the gap coincide with start and end nodes of the gap ($i = k, z = l$ for the left case, $z = k, j = l$ for the right case), then the gap just added to the left or right side of another tree.
- *Output*: output gates, one gate for every pair of graph nodes and nonterminal; evaluates to true if appropriate parse tree exists.

3.2 Bounds on the circuit size and depth

Circuit size

Lemma 1 *Let $G = (\Sigma, N, P)$ be a context-free grammar in Chomky normal form and let n be a number of nodes in the input graph. Then the circuit described in 3.1 has $O(n^6)$ elements.*

Proof Let's count circuit gates by type. The number of input gates is $|\Sigma|n^2$ and the number of output gates is $|N|n^2$. Also there are $|N|^2n^5$ OR-gates for creating the gaps and $|N|^2n^4$ AND-gates for filling gaps. It is easy to see that the maximal number of elements is needed for combinations of conditional propositions: there are $|N|^3n^6$ such elements ($|N|^2n^4 \times |N|n^2$). Therefore the circuit contains at most $O(n^6)$ elements.

Circuit depth Before we analyze the depth of the circuit we shall prove the following statement.

Lemma 2 *Let $G = (\Sigma, N, P)$ be a context-free grammar in Chomky normal form. Then every parse tree with k leaves contains a middle node ("critical" node) that spans over more than $\frac{1}{3}k$ and at most $\frac{2}{3}k$ leaves.*

Proof Because the grammar is in the Chomsky normal form, every node of a parse tree has two children. Going from the root to leaves, we can choose the largest of two subtrees at each node while the current subtree has more than $\frac{2}{3}k$ leaves. Obviously, the first node that has at most $\frac{2}{3}k$ leaves is bound to have more than $\frac{1}{3}k$ leaves due to binary branching.

Lemma 3 *Let l be a length of the shortest path from node i to node j labelled by string from $L(G_A)$. Then proposition $A(i,j)$ or $\frac{A}{D}(i, u :: v, j)$ has a proof of height $O(\log l)$.*

Proof Proof by induction on l.
Induction hypotesis: every proposition with no more than $\frac{2}{3}l$ leaves has a proof of logarithmic height $(O(\log l))$.

At first, consider the proposition $A(i,j)$. By Lemma 2, a corresponding tree has a critical node E with two children B and C (for $E \to BC \in P$). Then we can write the proof tree as follows.

$$\cfrac{\cfrac{B(u,l) \qquad C(l,v)}{E(u,v)} \qquad \frac{A}{E}(i, u :: v, j)}{A(i,j)}$$

Because E is the critical node, each of the propositions $B(u,l), C(l,v)$, $\frac{A}{E}(i, u :: v, j)$ has no more than $\frac{2}{3}l$ leaves and therefore has a proof of logarithmic height by inductive hypothesis.

There are two cases for the second proposition $\frac{A}{D}(i, u :: v, j)$.

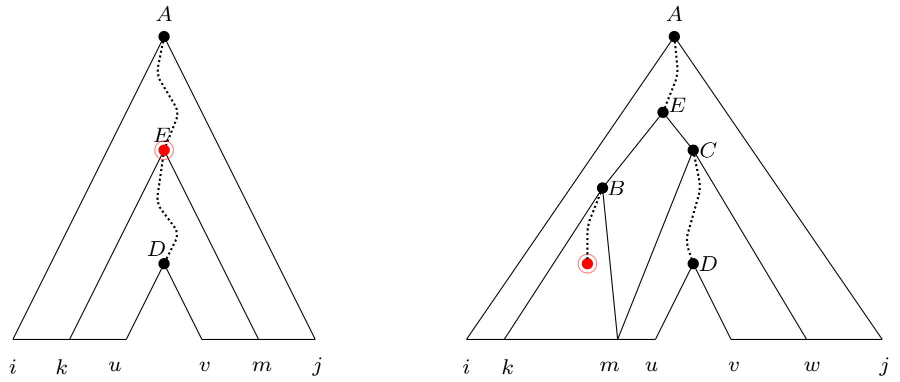

Fig. 1 Cases for conditional proposition $\frac{A}{D}(i, u :: v, j)$: (left) a subtree with the critical node at the root contains the hole; (right) a subtree with the critical node at the root contains only leaves corresponding to the path $i \to u$.

1. *A subtree with the critical node at the root contains the hole $u :: v$.*
   Let E be a label of the critical node. Then a corresponding subtree splits the path from the vertex i to the vertex u onto two paths: $i \to k$ and $k \to u$ and the path from the vertex v to the vertex j onto $v \to m$ and $m \to j$ respectively (as illustrated in Figure 1 (left)). The next proof tree represents the proof of the proposition $\frac{A}{D}(i, u :: v, j)$ in this case.

$$\frac{\frac{A}{E}(i, k :: m, j) \qquad \frac{E}{D}(k, u :: v, m)}{\frac{A}{E}(i, u :: v, j)}$$

   Conditional proposition $\frac{A}{E}(i, k :: m, j)$ and $\frac{E}{D}(k, u :: v, m)$ have no more than $\frac{2}{3}l$ leaves, so $\frac{A}{E}(i, u :: v, j)$ has a proof of logarithmic height.
2. *A subtree with the critical node at the root contains only leaves corresponding to the path from the vertex i to the vertex u in the input graph.*
   This case is illustrated in Figure 1 (right). Consider the biggest subtree which contains a subtree with the critical node at the root and has leaves corresponding only to the path from the vertex i to the vertex u of the input graph. Let B be a label at the root of this tree and $k \to m$ be a corresponding path in the input graph. Let E be a parent node of node B and let C be a second child of E (for $E \to BC \in P$). A parse tree with the root labelled by C has some leaves corresponding to the path $v \to j$, so let w be a vertex which splits this path on two. Now we are able to write the proof of $\frac{A}{E}(i, u :: v, j)$.

$$\frac{\frac{A}{E}(i, k :: w, j) \qquad \dfrac{\dfrac{B(k, m)}{\frac{E}{C}(k, m :: w)} \qquad \frac{C}{D}(m, u :: v, w)}{\frac{E}{D}(k, u :: v, w)}}{\frac{A}{D}(i, u :: v, j)}$$

The subtree with the root labelled B obviously contains at least $\frac{2}{3}l$ leaves, so each of the other propositions $\frac{C}{D}(m, u :: v, w)$ and $\frac{A}{E}(i, k :: w, j)$ has no more then $\frac{2}{3}l$ leaves. By induction hypothesis they have a proof of logarithmic height. Thus, proposition $\frac{A}{D}(i, u :: v, j)$ can be proved via 4 steps using propositions with the proofs of logarithmic depth.

The case when the subtree with the critical node at the root contains only leaves corresponding to the path from the vertex v to the vertex j in the input graph is held symmetrically.

Before we will make a conclusion, the following definition should be introduced.

Definition 2 Let $G = (\Sigma, N, P)$ be a context-free grammar and D be a directed labelled graph. Then $\mathcal{L}$ is the length of longest shortest path $i\pi j$ for all triples (A, i, j), where j is reachable from i via path labelled by $l(i\pi j) \in L(G_A)$ and $A \in N$.

Using lemmas above, we can conclude the following.

Corollary 1 *Let $G = (\Sigma, N, P)$ be a context-free grammar in Chomky normal form and let $\mathcal{L}$ be the length of the maximum in all nonterminals and all pairs of graph nodes shortest path labelled by a string from $L(G)$ (see Definition 2). Then there is a uniform family of circuits for solving CFL-reachability problem on the graphs with n nodes, which are of depth $O(\log n \log \mathcal{L})$ and contain $O(n^6)$ elements.*

The depth and therefore the efficiency of the circuit depends on the value of parameter $\mathcal{L}$. We investigate bounds for $\mathcal{L}$ in the next two sections.

4 Efficient subclasses of context-free languages

In this section our focus is on estimating the value of $\mathcal{L}$ for some fixed context-free languages and arbitrary input graphs.

4.1 General case

Hellings in [18] gave the worst-case lower and upper bounds on the $\mathcal{L}$ parameter.

Theorem 1 (Hellings) *Let $G = (\Sigma, N, P)$ be a context-free grammar and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. In the worst case, we have $n^2 2^{|N|} \le \mathcal{L} \le 2^{|N|n^2-1}$.*

Using the bounds above and Corollary 1, we can get the maximum depth of our circuit in the worst case: $O(\log n \log \mathcal{L}) = O(\log n \log 2^{|N|n^2-1}) = = O(|N|n^2 \log n)$. The circuit can have a depth which is polynomial on the input length in case of arbitrary context-free grammar. It is not surprising because of the nature of P-completeness: we don't have an efficient parallel algorithm or circuit with a polylogarithmic on the input length depth for CFL-reachability problem because it is P-complete problem (unless $P \neq NC$).

But we also have a polynomial lower bound on $\mathcal{L}$, which give us the effective circuit with depth $O(\log^2 n)$ in some cases. Our next goal is to investigate important subclasses of context-free languages, and find those for which CFL-reachability is in the class NC.

4.2 Rational index

It is known that the value of $\mathcal{L}$ is different for various types of context-free languages, in particular it can be bounded from above by the polynomial or exponential function on the number of graph nodes [10, 31, 32]. Boasson et al. in [6] introduced definition of such function called *rational index.* For a language L over an alphabet Σ, its rational index ρ_L is a function defined as follows:

$$\rho_L(n) = \max\{\min\{|w| : w \in L \cap K\}, K \in Rat_n, L \cap K \neq \emptyset\}, \qquad (1)$$

where $|w|$ is the length of a word w and Rat_n denotes the set of regular languages on an alphabet Σ, recognized by a finite nondeterministic automation with at most n states.

It is easy to see that in the case of fixed context-free language and arbitrary graph, $\mathcal{L}$ is some kind of the rational index: we can represent arbitrary graphs with n nodes via nondeterministic automations with at most n states, then $\mathcal{L}$ is exactly rational index. So, we have the following corollary.

Corollary 2 *Let L be a context-free language on an alphabet Σ, which has a polynomially bounded from above rational index, and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then, CFL-reachability problem for L and D is in NC^2.*

For example, language $L = \{a^n b^m, n \neq m\}$ has polynomial rational index $\rho_L(n) = 2n - 1$ [32], so CFL-reachability problem for L and arbitrary directed labelled graph with n nodes is in NC^2.

4.3 Linear, metalinear and superlinear languages

A *linear language* is a language generated by some *linear grammar*. A *linear grammar* is a context-free grammar that has at most one nonterminal in the right hand side of each of its productions. The linear grammar $G = (\Sigma, N, P)$ is said to be in the Chomsky normal form, when all of its production rules are of the form: $A \to aB$, or $A \to Ba$, or $A \to a$, where $B \in N$ and $a \in \Sigma$.

Before we consider the value of $\mathcal{L}$ for linear grammars, we shall prove the following.

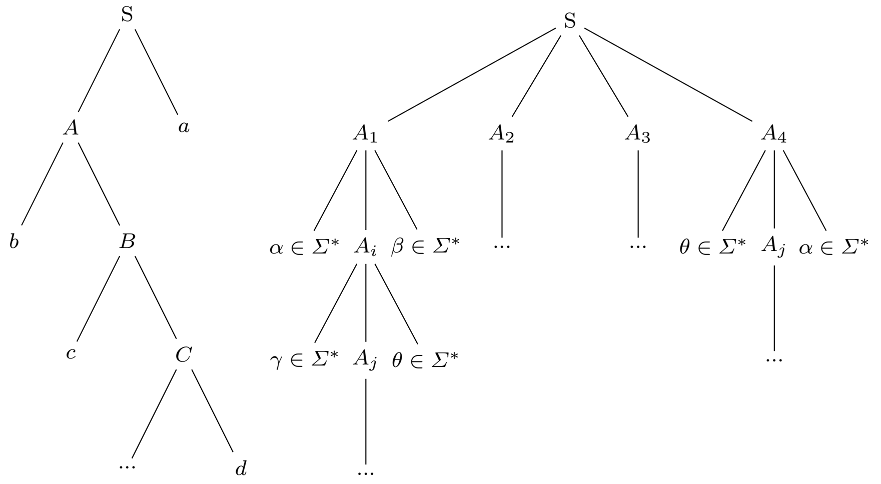
S
A
a
b
B
c
C
...
d
S
A_1
A_2
A_3
A_4
$\alpha \in \Sigma^*$ A_i $\beta \in \Sigma^*$
...
...
$\theta \in \Sigma^*$ A_j $\alpha \in \Sigma^*$
$\gamma \in \Sigma^*$ A_j $\theta \in \Sigma^*$
...
...

Fig. 2 A parse tree for linear grammar in the Chomsky normal form (left) and for metalinear grammar with width $m = 4$ (right).

Lemma 4 *Let $G = (\Sigma, N, P)$ be a context-free grammar, $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes and π be the maximum in all nonterminals and all pairs of graph nodes shortest path labelled by a string from $L(G)$. Then a height of a parse tree for $l(\pi)$ does not exceed $|N|n^2$.*

Proof Assume that the parse tree for $l(\pi)$ has a height of more than $|N|n^2$. There are $|N|n^2$ unique labels (A, i, j) for nodes of the parse tree, so according to the pigeonhole principle, the parse tree for $l(\pi)$ contains at least one subtree T with label (A, i, j) at the root, which has a subtree T' with the same label. Then we can change T with T' and get a new string $l(\pi')$ which is shorter than $l(\pi)$. But $l(\pi)$ is the shortest, then we have a contradiction.

Theorem 2 *Let $G = (\Sigma, N, P)$ be a linear grammar and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then the length of the maximum in all nonterminals and all pairs of graph nodes shortest path labelled by a string from $L(G)$ ($\mathcal{L}$) is $O(|N|n^2)$.*

Proof From Lemma 4, a height of a parse tree is no more then $|N|n^2$. Let's construct a parse tree of such height for linear grammar in the Chomsky normal form. Every rule for linear grammar has at most one nonterminal in the right hand side of each of its productions, so if the grammar is in the Chomsky normal form, then there is the only one terminal symbol at the every level of the parse tree (as illustrated in Figure 2 (left)). The tree has no more then $|N|n^2$ levels, so the length of any shortest path labelled by a string from $L(G)$ is no more than $O(|N|n^2)$.

Combining Theorem 2 and Corollary 1 we are able to conclude the following.

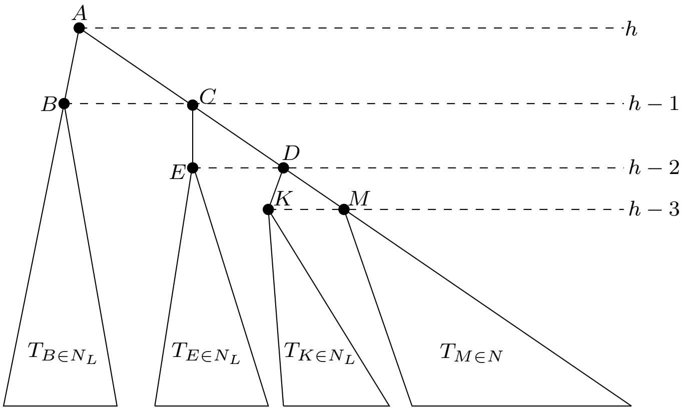

Fig. 3 A worst-case parse tree for superlinear grammar.

Corollary 3 *Let $G = (\Sigma, N, P)$ be a linear grammar and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then CFL-reachability problem for G and D is in NC^2.*

There are natural extensions of linear languages: *metalinear* and *superlinear* languages. Just like linear languages, these classes of context-free languages are known to be easy to parse [23]. We will show that they are "easy" for our circuit too.

Metalinear languages Let $G = (\Sigma, N, P, S)$ be a context-free grammar. G is *meatalinear* if all productions of P are of the following forms:

1. $S \to A_1A_2...A_m$, where $A_i \in N - \{S\}$
2. $A \to u$, where $A \in N - \{S\}$ and $u \in (\Sigma^*((N - \{S\}) \cup \varepsilon)\Sigma^*)$

The width of a metalinear grammar is $max\{m \mid S \to A_1A_2...A_m\}$. Metalinear languages of width 1 are obviously linear languages.

Lemma 5 *Let $G = (\Sigma, N, P)$ be a metalinear grammar, where G_S has the width m, k be a length of the longest sequence of terminal symbols in the right-hand-side of rules and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then the length of the maximum in all nonterminals and all pairs of graph nodes shortest path labelled by a string from $L(G)$ ($\mathcal{L}$) is $O(mk|N|n^2)$.*

Proof Consider the highest parse tree for metalinear grammar. By Lemma 4, it's height is no more then $|N|n^2$. Obviously, if the root of the tree is labelled by some $A_i \in N - \{S\}$, value of $\mathcal{L}$ in this case is the same as value of $\mathcal{L}$ for the case of linear grammar — $O(|N|n^2)$ for constant value of k. If the root of the tree is labelled by S, the tree looks like as illustrated in Figure 2 (right). It is easy to see that there are no more than km terminal symbols on every level of such tree. Thus, we have $\mathcal{L} = O(mk|N|n^2)$.

Applying Lemma 5 and Corollary 1, we deduce with the corollary.

Corollary 4 *Let $G = (\Sigma, N, P)$ be a metalinear grammar and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then CFL-reachability problem for G and D is in NC^2.*

Superlinear languages Let $G = (\Sigma, N, P)$ be a context-free grammar. G is *superlinear* if all productions of P satisfy these conditions:

1. there is a subset $N_L \subseteq N$ such that every $A \in N_L$ has only linear productions $A \to aB$ or $A \to Ba$, where $B \in N_L$ and $a \in \Sigma$.
2. if $A \in N \backslash N_L$, then A can have non-linear productions of the form $A \to BC$ where $B \in N_L$ and $C \in N$, or linear productions of the form $A \to \alpha B \mid B\alpha \mid \alpha$ for $B \in N_L$, $\alpha \in \Sigma^*$.

It can be seen from conditions above, that superlinear grammars contain the metalinear grammars.

Lemma 6 *Let $G = (\Sigma, N, P)$ be a superlinear grammar and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then the length of the maximum in all nonterminals and all pairs of graph nodes shortest path labelled by a string from $L(G)$ ($\mathcal{L}$) is $O(|N^2|n^4)$.*

Proof Let's construct a parse tree of maximum height h from the root to leaves. Let A be a nonterminal at the root of the parse tree T_A. We have two cases to consider:

1. $A \in N_L$ or $A \in N \setminus N_L$ and A has linear productions: then the tree with A at the root is the same tree as parse tree for linear grammar and it has $O(h)$ leaves.
2. $A \in N \setminus N_L$ and $A \to BC \in P$: then number of leaves in the tree $le(T_A)$ can be expressed with the following reccurent formula: $le(T_A) = le(T_B) + le(T_C) = h(T_B) + le(T_C) = h(T_A) - 1 + le(T_C)$.

In the worst case, we have to add tree with i leaves on every i-th level of the parse tree (counting from $h - 1$ to 0) (see Figure 6). By Lemma 4, $h \leq |N|n^2$, so in the worst case the highest tree will have $|N|n^2 - 1 + |N|n^2 - 2... + 1 = O(|N^2|n^4)$ leaves.

Combining Lemma 6 and Corollary 1 we are able to conclude the following.

Corollary 5 *Let $G = (\Sigma, N, P)$ be a superlinear grammar and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then CFL-reachability problem for G and D is in NC^2.*

4.4 Dyck languages

Dyck language over Σ_k is the set of well-balanced words over alphabet $\Sigma_k = \{(_1, (_2, ..., (_k,)_1,)_2, ...,)_k\}$. Dyck language can be defined by the following rules: $S \to SS \mid S \to \varepsilon \mid S \to (_1 S)_1 \mid S \to (_2 S)_2 \mid ... \mid S \to (_k S)_k$, where S is the start symbol, and ε is the empty string. We will denote the Dyck language over Σ_i by D_i.

Dyck languages play an important role in formal languages theory. The famous Chomsky-Schützenberger theorem states that every context-free language L can be represented via a regular language R and a Dyck language D_n, which are combined by means of an intersection and a homomorphism h.

$$L = h(D_n \cap R) \tag{2}$$

Because every context-free language can be represented with Dyck language, it is natural to expect that CFL-reachability is P-complete for Dyck languages. P-completeness for the similiar problem LGAP in case of D_2 was proved in [14, 24, 36]. We will show that $\mathcal{L}$ is exponential for D_k where $k \geq 2$, so our circuit is not effective in that case.

Lemma 7 (Pierre) *The rational indexes of generators of context-free languages belongs to* $exp(\Theta(n^2/\ln n))$.

Theorem 3 *Let L be a Dyck language over $k \geq 2$ brackets and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then, $\mathcal{L}$ belongs to* $exp(\Theta(n^2/\ln n))$.

Proof By definition, generator is a context-free language which can transform into any context-free language through a rational transduction [31]. Notice that for $k > 2$ there is a homomorphism g such that $D_k = g^{-1}(D_2)$ can be constructed. Thanks to the Chomsky-Schützenberger theorem, we have the following equation.

$$L = h(g^{-1}(D_2) \cap R) \tag{3}$$

Thus, every context-free language can be generated by D_k, where $k \geq 2$. By 7 we can conclude that $\mathcal{L}$ for D_k belongs to $exp(\Theta(n^2/\ln n))$.

We remark that strict upper bound on $\mathcal{L}$ for arbitrary context-free grammar is obtained from Lemma 7, so it answers to the corresponding open question stated in [18].

It is known that D_1 is a generator of *one-counter languages* [15]. *One-counter* languages are the languages recognized by *one-counter automata* — pushdown automata with a single stack symbol. For example, $L = \{w\#w' \mid |w| \neq |w'|\}$ is one-counter language.

Lemma 8 (Deleage, Pierre) *The rational index of D_1 is in $O(n^2)$.*

Theorem 4 *Let L be a one-counter language, and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then, CFL-reachability problem for L and D is in NC^2.*

Proof Because D_1 is a generator of one-counter languages [15], D_1 rationally dominates any one-counter language. If L rationally dominates L', then there exists an integer c such that for every $n \in \mathbb{N}$ $\rho_{L'(n)} < cn(\rho_L(cn) + 1)$ [6]. Thus, the rational index of any one-counter language doesn't exceed $O(n^3)$, and by Corollary 1, the depth of the circuit for one-counter languages is $O(\log^2 n)$.

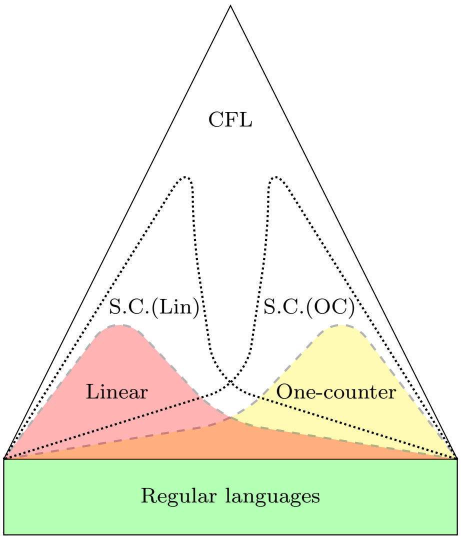

Fig. 4 A hierarchy of linear languages (Lin), one-counter languages (OC) and their substitution closures (S.C).

Notice that there is an effective sequential algorithm for CFL-reachability for D_1 based on matrix multiplication [7]. We believe that using Theorem 4 it can be extended for working with one-counter languages.

Results for the case of one-counter languages can not be generalized to bigger number of counters. It is well-known, that two-counter automaton is already too powerfull: it can simulate the Turing machine, which accepts context-sensitive languages.

4.5 Greibach languages and substitution closures

Consider families of linear and one-counter languages. We can see that they are "easy" for CFL-reachability (Theorem 4, Corollary 3). One reason for this is that they are defined by natural restrictions on grammars or pushdown automatons (PDA). Restricting a PDA such that the height of its stack is only allowed to increase and then to decrease, thus performing only one turn, leads to the definition of one-turn PDAs. It is known that these PDAs can be characterized by linear grammars [27]. One-counter languages are recognized by PDA with only one stack symbol. The same restrictions holds for substition closures of these languages, because substitution corresponds to a nesting of the pushdown stack [13]. It is easily observed that substitution closure of polinomially-bounded $\mathcal{L}$ language yields polinomially-bounded $\mathcal{L}$ language.

But some of the restrictions on the pushdown store or on the grammar cannot be simulated by others. It is exactly the case of linear and one-counter languages: they and their iterated substitution closures are distinct families of

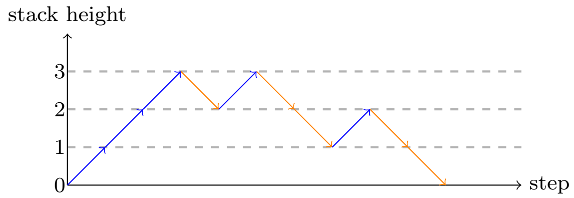

Fig. 5 Stack heights during the run of PDA.

context-free languages [5]. Hierarchy of linear languages, one-counter languages and their substitution closures is illustrated in Figure 4.

Family of *Greibach languages* is the substitution closure of linear and one-counter languages. It is a large strict subfamily of the family of context-free languages: it does not contain the language D_2 [2]. Because it is the substitution closure of two polinomially-bounded $\mathcal{L}$ languages, we have the following corollary.

Corollary 6 *Let L be a Greibach language and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then CFL-reachability problem for L and D is in NC^2.*

4.6 Oscillation-bounded languages

Oscillation-bounded languages were introduced in [12]. Just like one-counter and linear languages, it is defined by restriction on the pushdown automata. This restriction is based on the notion of *oscillation*, a special measure of how variable is the stack height over time. Oscillation is defined using a hierarchy of *harmonics*. Let $\bar{a}$ be a *push*-move and a be a *pop*-move. Then PDA run can be described by well-nested word of $\bar{a}$-s and a-s. Order 1 harmonic h_1 is $\bar{a}a$ $\bar{a}a$ (*push pop push pop*), order 2 harmonic h_2 is $\bar{a}$ <order 1 harmonic> a $\bar{a}$ <order 1 harmonic> a, so $(i+1)$-st harmonic is $\bar{a}$ h_i a $\bar{a}$ h_i a.

PDA run r is *k-oscillating* if the harmonic of order k is the greatest harmonic that is contained in r. *Oscillation-bounded languages* are languages accepted by pushdown automata restricted to k-oscillating runs.

Example 1 Consider Figure 5. It shows how the stack height changes during the run of PDA. Corresponding well-nested word is $\bar{a}$ $\bar{a}$ $\bar{a}a$ $\bar{a}a$ a $\bar{a}a$ a. The greatest harmonic in this word is 1 order harmonic: $\mathbf{\bar{a}}$ $\bar{a}$ $\mathbf{\bar{a}a}$ $\mathbf{\bar{a}a}$ a $\bar{a}a$ $\mathbf{a}$, therefore oscillation of the run is 1.

Oscillation of the run is closely related with the *dimension* of the corresponding parse tree. For each node v in a tree T a dimension $dim(v)$ is inductively defined as follows:

- If v is a leaf, then $dim(v) = 0$

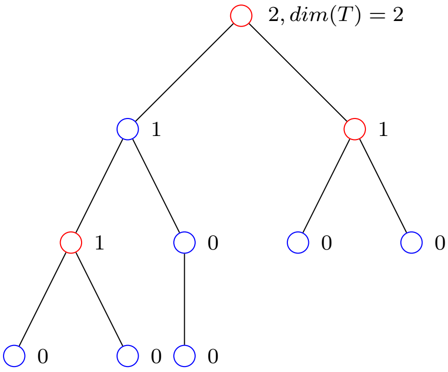

Fig. 6 A tree T with $dim(T) = 2$.

- If v is an internal node with k children $v_1, v_2, ..., v_k$ for $k \geq 1$, then

$$dim(v) = \begin{cases} \max_{i\in\{1\ldots k\}} dim(v_i) & \text{if there is a unique maximum} \\ \max_{i\in\{1\ldots k\}} dim(v_i) + 1 & \text{otherwise} \end{cases} \quad (4)$$

Dimension of a parse tree T $dim(T)$ is a dimention of it's root. It is observable from the definition that dimension of a tree T is the height of the largest perfect binary tree, which can be obtained from T by contracting edges and accordingly identifying vertices. A tree with dimension $dim(T) = 2$ is illustrated in Figure 6.

It is known that the dimension of parse trees and the oscillation defined on PDA runs are in linear relationship.

Lemma 9 (Ganty, Valput) *Let a grammar $G = (\Sigma, N, P)$ be in Chomsky normal form and let T be a parse tree of G. We have that $osc(T) - 1 \leq dim(T) \leq 2osc(T)$.*

It is not hard to see that if $L(G)$ is k-oscillation-bounded language, parse trees of G always have a bounded dimension.

Lemma 10 *Let L be a k-oscillation-bounded language with grammar $G = (\Sigma, N, P)$ in Chomsky normal form and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then the length of the maximum in all nonterminals and all pairs of graph nodes shortest path labelled by a string from $L(G)$ ($\mathcal{L}$) is no more than $O((|N|n^2)^{k/2})$.*

Proof Proof by induction on dimension $dim(T)$ (if oscillation is bounded by k, then dimension value is at most $2k$).
Basis. $dim = 1$.
Consider the worst-case tree T with the dimension $dim(T) = 1$. The root of the tree has the same dimension and has two children (because the grammar in Chomsky normal form). There are two cases: first, when both of child nodes have dimension equal to 0, then the tree has only two leaves and second, when

one of children has a dimension 1, and the second child has a dimension equal to 0. For the second case we can recursively construct a tree of maximum height $|N|n^2$ (Lemma 4). Every internal node of such tree has two children, one of wich has dimension equal to 0 and therefore has only one leaf. This tree is exactly the worst-case tree for linear grammar in Chomsky normal form, so the number of leaves in such tree is $O(h) = O(|N|n^2)$, where h is the height of the tree. Thanks to Lemma 10, the maximum value of oscillation for $dim(T) = 1$ is 2, so $\mathcal{L}$ is no more than $O(h^{k/2}) = O((|N|n^2)^{k/2}) = O(|N|n^2)$.
Inductive step. $dim = d + 1$.
Assume $\mathcal{L}$ is no more than $O(h^d)$ for every d, where h is the height of the tree. We have two cases for the root node with dimension equal to $d + 1$: 1) both of children have a dimension equal to d, then by proposition the tree of heght h has no more than $O(h^d)$ leaves; 2) one of children has a dimension $d+1$, and the second child v has a dimension $dim(v) \leq d$. Again, a tree of maximum height with the maximum number of leaves can be constructed recursively: each node of such tree has two children u and v with dimension $d + 1$ and d respectively (the more value of dimension of the node, the more leaves in the corresponding tree). By proposition we have no more than $(h-1)^d+(h-2)^d+(h-3)^d+...+1 = O(h^{d+1})$ leaves, so proposition holds for $dim = d + 1$. Finally, we have $\mathcal{L}$ is no more than $O(h^d) = O((|N|n^2)^{k/2})$ for every k.

As we can see from the proof above, the family of linear languages is included in the family of oscillation-bounded languages. The reason is that the family of oscillation-bounded languages generalizes the family of languages accepted by finite-turn pushdown automata [12]. It is interesting that for arbitrary CFL, particularly for D_2 , the value of osciilation is not constant-bounded: it depends on the length of input and does not exeed $O(\log n)$ for the input of length n [42, 17].

Combining Lemma 10 and Corollary 1, we have the following.

Corollary 7 *Let L be a k-oscillation-bounded language and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then CFL-reachability problem for L and D is in NC^2.*

4.7 LL(k) languages and a determinism

LL(k) languages represent another kind of restriction: unlike previous cases, which describe restrictions on PDA, *LL(k) grammars* have constraints for their productions.

For a natural number $k \geq 0$, a context-free grammar $G = (\Sigma, N, P)$ is an *LL(k) grammar* if for each terminal symbol string $w \in \Sigma^*$ of length up to k, for each nonterminal symbol $A \in V$, and for each terminal symbol string $w_1 \in \Sigma^*$ there is at most one production rule $p \in P$ such that for some terminal symbol strings $w_2, w_3 \in \Sigma^*$ the following conditions hold:

- the string w_1Aw_3 can be derived from the start symbol $A \in N$,

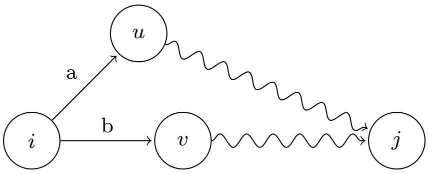

Fig. 7 A labelled graph with distinct paths from i to j.

- w_2 can be derived from A after first applying rule p,
- the first k symbols of w and of w_2w_3 agree.

LL(k) languages are connected with *simple deterministic languages*. A context-free grammar $G = (\Sigma, N, P)$ is simple if it is in Greibach normal form and if, for each pair $(A, \alpha) \in N$, there is at most one rule of the form $A \to \alpha Y_1 Y_2 ... Y_n$. A language is simple deterministic if it can be generated by a simple grammar. Simple deterministic languages are included in the family of LL(1) languages. It is known that D_n is simple deterministic language [25], so we have exponential value of $\mathcal{L}$ for LL(k) languages in the worst case (because LL(0) $\subsetneq$ LL(1) $\subsetneq$ LL(2) $\subsetneq$ ... $\subsetneq$ LL(k)). Example 2 shows how an advantage of having the unique production is lost in case of graph input even for LL(1) grammars.

Example 2 Let $G = (\Sigma, N, P)$ be a LL(1) grammar with rules p_1:$A \to aX_1...X_k$ and p_2 :$B \to bY_1...Y_m$, $p_1, p_2 \in P$. Consider the graph illustrated in Figure 7. We want to make a proposition about nonterminal, which derives $l(i\pi j)$. Because the node i has two outcoming edges with labels a and b, both rules p_1 and p_2 are suitable.

5 Efficient subclasses of graphs

In this section a case when the input graph (regular language or finite automata, respectively) is fixed and context-free grammar is arbitrary, will be considered.

5.1 Directed acyclic graphs and trees

The most trivial example of "easy" for our circuit graphs are directed acyclic graphs. It is not difficult to see that all words accepted by an appropriate finite automata have length no more than $O(n)$, where n is number of nodes in graph. So, we obtain the following.

Corollary 8 *Let $G = (\Sigma, N, P)$ be a context-free grammar and $D = (V, E, \Sigma)$ be a directed acyclic labelled graph with n nodes. Then CFL-reachability problem for G and D is in NC^2.*

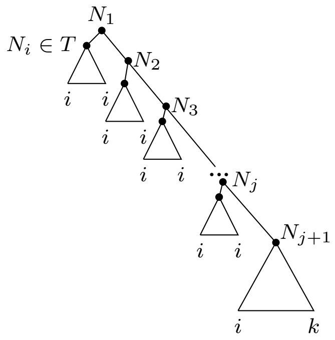

Fig. 8 A derivation tree for $l(i\pi k)$, $T \in N$ is the set of terminals.

5.2 Partially ordered automata

Partially ordered automata generalize the previous case. *Partially ordered automata* are finite automata where the transition relation has a partial order on states. It means that as soon as a state is left during the run, it is never visited again. As a consequence, all cycles of partially ordered automata are only self-loops.

Notice that the lengths of accepted words are not bounded in this case because of existence of self-loops. We will show that the value of $\mathcal{L}$ still stays polynomial despite infinite lengths of accepted words.

Lemma 11 *Let $G = (\Sigma, N, P)$ be a context-free grammar in Chomsky normal form and $\mathcal{A}$ be a partially ordered automaton with n states. Then the length of the maximum in all nonterminals and all pairs of states shortest path labelled by a string from $L(G)$ ($\mathcal{L}$) is no more than $O(n)$.*

Proof (Sketch). Assume that there is loop on some vertex i. Consider the maximum possible length of path $i\pi i$ from i to i. There two cases of parse subtrees for $l(i\pi i)$. First case is when $l(i\pi i)$ has a distinct parse subtree. Then, the height of such tree is no more than $|N|$, because there are only $|N|$ unique triples in form of (A, i, i), where $A \in N$. The maximum number of leaves in a parse tree of height h (if the grammar is in Chomsky normal form) is $2^h = 2^{|N|}$. The second case is when the $l(i\pi i)$ doesn't have a distinct parse tree. It means that there is no derivation $A \overset{*}{\Rightarrow} l(i\pi i)$ for $A \in N$. Let T be the smallest parse tree which yields a string $l(i\pi_1 k)$, where $l(i\pi i)$ is substring of $l(i\pi_1 k)$. Because the grammar is in the Chomsky normal form, derivation tree for $l(i\pi k)$ looks as illustrated in Figure 8. The maximum length of $l(i\pi i)$ is equal to the maximum height of a parse tree for string $l(i\pi_1 k)$. It is equal to number of unique triples $(A, i, k) : |N|$. Case for $l(m\pi_1 i)$ is hold symmetrically. Finally we have the $\mathcal{L}$ in $O(2^{|N|}n) + O(n) = O(n)$ in the worst case (there are loops on every vertex).

Applying Lemma 11 and Corollary 1, we have the following corollary.

Corollary 9 *Let $G = (\Sigma, N, P)$ be a context-free in Chomsky normal form and $\mathcal{A}$ be a partially ordered automaton with n states. Then CFL-reachability problem for G and $\mathcal{A}$ is in NC^2.*

Famous examples of language accepted by partially ordered automata are *piecewise testable languages* [38, 29]. A regular language over an alphabet Σ is *piecewise testable* if it is a finite Boolean combination of languages of the form $\Sigma^* a_1 \Sigma^* a_2 \Sigma^* ... \Sigma^* a_k$, where $a_1, a_2, ..., a_k \in \Sigma$, $k \geq 0$.

6 Conclusions and open problems

We have constructed a Boolean circuit for CFL-reachability problem and have shown that parallel complexity of CFL-reachability depends on the parameter $\mathcal{L}$ — the length of the maximum in all nonterminals and all pairs of graph nodes shortest path labelled by a string from $L(G)$, where G is the input grammar. CFL-reachability is in NC^2 for context-free languages and graphs with polynomial $\mathcal{L}$, and is P-complete, when the value of $\mathcal{L}$ is exponential.

We have shown that natural restrictions on the pushdown storage implies polynomial $\mathcal{L}$ for corresponding context-free language. The most trivial case of such restriction is a bounded height of the stack. Pushdown automata with bounded stack accept regular languages. It is known that L-reachibility for regular languages is in $NL \subseteq NC^2$ [24, 43]. We have observed another restrictions on the stack like constant number of turns, single stack symbol (one-counter automata) and constant oscillation of PDA run. CFL-reachability for languages of such pushdown automata and arbitrary graphs is in NC. Also operations like substitution closure, which is related with PDA stack nesting, applied to a language with polynomial $\mathcal{L}$, give a new language with polynomial $\mathcal{L}$. However, one restriction on the pushdown storage can not be simulated by others: for example, Dyck language D_1, which is accepted by one-counter automata, is a k-oscillating language for no k. This leads to the question of having the common property for pushdown automata accepted languages with polynomial $\mathcal{L}$. In other words, is there any common characteristic for PDA or grammar, which implies the polynomial value of $\mathcal{L}$?

In the case of an arbitrary context-free grammar and fixed graph, we have shown that CFL-reachability for directed labelled graphs, which can be associated with partially ordered automata, is in NC^2. It means that CFL-reachability for directed acyclic graphs with loops and arbitrary context-free language is in NC.

References

1. Afrati F, Papadimitriou C (1987) The parallel complexity of simple chain queries. In: Proceedings of the Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM, New

York, NY, USA, PODS '87, pp 210–213, DOI 10.1145/28659.28682, URL http://doi.acm.org/10.1145/28659.28682

2. Autebert JM, Berstel J, Boasson L (1997) Context-Free Languages and Pushdown Automata, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 111–174
3. Azimov R, Grigorev S (2017) Graph parsing by matrix multiplication. CoRR abs/1707.01007, URL http://arxiv.org/abs/1707.01007, 1707.01007
4. Barrett C, Jacob R, Marathe M (2000) Formal-language-constrained path problems. SIAM J Comput 30:809–837, DOI 10.1137/S0097539798337716
5. Beauquier J, Berstel J (1981) More about the geography of context-free languages. Information and Control 49(2):91 – 108
6. Boasson L, Courcelle B, Nivat M (1981) The rational index: a complexity measure for languages. SIAM Journal on Computing 10(2):284–296
7. Bradford PG (2018) Efficient exact paths for dyck and semi-dyck labeled path reachability. CoRR abs/1802.05239, 1802.05239
8. Brent R, M Goldschlager L (1983) A parallel algorithm for context-free parsing
9. Chatterjee K, Choudhary B, Pavlogiannis A (2017) Optimal dyck reachability for data-dependence and alias analysis. Proc ACM Program Lang 2(POPL):30:1–30:30, DOI 10.1145/3158118, URL http://doi.acm.org/10.1145/3158118
10. Deleage JL, Pierre L (1986) The rational index of the dyck language ${D_1}^*$. Theoretical Computer Science 47:335 – 343, DOI 10.1016/0304-3975(86)90158-1
11. Ganardi M, Hucke D, KÃűnig D, Lohrey M (2016) Circuit evaluation for finite semirings. 1602.04560
12. Ganty P, Valput D (2016) Bounded-oscillation pushdown automata. Electronic Proceedings in Theoretical Computer Science 226:178âĂŞ197, DOI 10.4204/eptcs.226.13, URL http://dx.doi.org/10.4204/EPTCS.226.13
13. Ginsburg S (1975) Algebraic and Automata-Theoretic Properties of Formal Languages. Elsevier Science Inc., New York, NY, USA
14. Greenlaw R, Hoover HJ, Ruzzo WL (1995) Limits to Parallel Computation: P-completeness Theory. Oxford University Press, Inc., New York, NY, USA
15. Greibach SA (1967) An infinite hierarchy of context-free languages. In: 8th Annual Symposium on Switching and Automata Theory (SWAT 1967), pp 32–36, DOI 10.1109/FOCS.1967.5
16. Grigorev S, Ragozina A (2017) Context-free path querying with structural representation of result. In: Proceedings of the 13th Central & Eastern European Software Engineering Conference in Russia, ACM, New York, NY, USA, CEE-SECR '17, pp 10:1–10:7, DOI 10.1145/3166094.3166104, URL http://doi.acm.org/10.1145/3166094.3166104
17. Gundermann T (1985) A lower bound on the oscillation complexity of context-free languages. In: Fundamentals of Computation Theory, FCT '85, Cottbus, GDR, September 9-13, 1985, pp 159–166, DOI 10.1007/BFb0028800, URL https://doi.org/10.1007/BFb0028800

18. Hellings J (2015) Path results for context-free grammar queries on graphs. CoRR abs/1502.02242, 1502.02242
19. Hirschberg DS (1976) Parallel algorithms for the transitive closure and the connected component problems. In: Proceedings of the Eighth Annual ACM Symposium on Theory of Computing, ACM, New York, NY, USA, STOC '76, pp 55–57, DOI 10.1145/800113.803631, URL http://doi.acm.org/10.1145/800113.803631
20. Holzer M, Kutrib M, Leiter U (2011) Nodes connected by path languages. In: Mauri G, Leporati A (eds) Developments in Language Theory, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 276–287
21. Ibarra OH, Jiang T, Ravikumar B (1988) Some subclasses of context-free languages in nc1. Information Processing Letters 29(3):111 – 117, DOI https://doi.org/10.1016/0020-0190(88)90047-6, URL http://www.sciencedirect.com/science/article/pii/0020019088900476
22. Ibarra OH, Jiang T, Chang JH, Ravikumar B (1991) Some classes of languages in nc1. Information and Computation 90(1):86 – 106, DOI https://doi.org/10.1016/0890-5401(91)90061-6, URL http://www.sciencedirect.com/science/article/pii/0890540191900616
23. Jayaram R, Saha B (2017) Approximating language edit distance beyond fast matrix multiplication: Ultralinear grammars are where parsing becomes hard! In: ICALP
24. Komarath B, Sarma J, Sunil KS (2017) On the complexity of l-reachability. CoRR abs/1701.03255, URL http://arxiv.org/abs/1701.03255, 1701.03255
25. Korenjak AJ, Hopcroft JE (1966) Simple deterministic languages. In: Proceedings of the 7th Annual Symposium on Switching and Automata Theory (Swat 1966), IEEE Computer Society, Washington, DC, USA, SWAT66, pp 36–46, DOI 10.1109/SWAT.1966.22
26. Kuijpers J, Fletcher G, Yakovets N, Lindaaker T (2019) An experimental study of context-free path query evaluation methods. pp 121–132, DOI 10.1145/3335783.3335791
27. Kutrib M, Malcher A (2007) Finite turns and the regular closure of linear context-free languages. Discrete Applied Mathematics 155(16):2152 – 2164
28. Lu Y, Shang L, Xie X, Xue J (2013) An incremental points-to analysis with cfl-reachability. In: Jhala R, De Bosschere K (eds) Compiler Construction, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 61–81
29. Masopust T, KrÃűtzsch M (2019) Partially ordered automata and piecewise testability. 1907.13115
30. Okhotin A, Salomaa K (2014) Complexity of input-driven pushdown automata. SIGACT News 45:47–67
31. Pierre L (1992) Rational indexes of generators of the cone of context-free languages. Theoretical Computer Science 95(2):279 – 305, DOI 10.1016/0304-3975(92)90269-L
32. Pierre L, Farinone JM (1990) Context-free languages with rational index in $\theta(n^\gamma)$ for algebraic numbers γ. RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications 24(3):275–322

33. Reps T (1996) On the sequential nature of interprocedural program-analysis problems. Acta Inf 33(5):739–757, DOI 10.1007/BF03036473, URL http://dx.doi.org/10.1007/BF03036473
34. Reps TW (1997) Program analysis via graph reachability. Information & Software Technology 40:701–726
35. Rossmanith P, Rytter W (1992) Observations on log (n) time parallel recognition of unambiguous cfl's. Information Processing Letters 44(5):267 – 272, DOI https://doi.org/10.1016/0020-0190(92)90212-E, URL http://www.sciencedirect.com/science/article/pii/002001909290212E
36. Rubtsov AA, Vyalyi MN (2015) Regular realizability problems and context-free languages. CoRR abs/1503.00295, 1503.00295
37. Rytter W (1985) On the recognition of context-free languages. In: Skowron A (ed) Computation Theory, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 318–325
38. Simon I (1972) Hierarchies of events with dot-depth one. PhD thesis, Thesis (Ph. D.)–University of Waterloo
39. Swernofsky J, Wehar M (2015) On the complexity of intersecting regular, context-free, and tree languages. In: Halldórsson MM, Iwama K, Kobayashi N, Speckmann B (eds) Automata, Languages, and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 414–426
40. Ullman JD, Van Gelder A (1986) Parallel complexity of logical query programs. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), pp 438–454, DOI 10.1109/SFCS.1986.40
41. Vyalyi MN (2012) On complexity of regular realizability problems. ArXiv abs/1211.0606
42. Wechsung G (1979) The oscillation complexity and a hierarchy of context-free languages. In: Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arthmetic, and Categorial Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979., pp 508–515
43. Yannakakis M (1990) Graph-theoretic methods in database theory. pp 230–242, DOI 10.1145/298514.298576
44. Zhang X, Feng Z, Wang X, Rao G (2015) Context-free path queries on RDF graphs. CoRR abs/1506.00743, 1506.00743