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

Parallel Complexity of CFL-Reachability Problem: Tractable Cases

Ekaterina Shemetova · Alexander Okhotin · Semyon Grigorev

Received: date / Accepted: date

Abstract Whereas it has been shown that context-free language (CFL) reachability problem is P-complete, there are some subclasses of context-free languages, for which CFL-reachability lies in NC complexity class. We present two common classes which generalize known examples of such tractable subclasses: bounded-oscillation languages and context-free languages with a poly-slender storage languages. Polynomiality of the rational indices of languages in these classes is proved. Criterion for deciding whether a PDA has a poly-slender storage language is given. Closure properties of tractable subclasses in terms of polynomial rational index are investigated.

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

1 Introduction

The context-free language (CFL) reachability problem for a context-free grammar G and directed edge-labelled graph D consists of determining for pairs of nodes v and u whether v can reach u via a path labelled by a string in $L(G)$. That is, CFL-reachability is a kind of graph reachability problem with path constraints given by context-free languages. It is an important problem

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

A. Okhotin
St. Petersburg State University, 7/9 Universitetskaya nab., Saint Petersburg 199034, Russia,

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

underlying some fundamential static code analysis like data flow analysis and program slicing [35], alias analysis [7, 43], points-to analysis [28] and other [6, 22, 33], and graph database query evaluation [2, 17, 19, 44].

Unlike context-free language recognition, which is in NC (when context-free grammar is fixed), CFL-reachability is P-complete [34, 42]. Practically, it means that there is no efficient parallel algorithm for solving this problem (unless P $\neq$ NC).

While problem is not parallelizable in general, it is useful to develop more efficient parallel solutions for specific subclasses of context-free languages. For example, there are context-free languages which admit more efficient parallel algorithms in comparision with the general case of context-free recognition [23, 24, 30]. The same holds for CFL-reachability problem: there are some examples of context-free languages, for which CFL-reachability problem lies in NL complexity class (for example, linear and one-counter languages) [21, 26, 36].

CFL-reachability problem has long been known to be P-complete [16]. A parallel complexity of this problem is studied by both static code analysis [34, 35] and database communities [1, 39, 42]. First investigations of such type were made in terms of Datalog queries, because some classes of Datalog queries (logic programs without function symbols) can be represented via context-free grammars, while database can be considered as a graph. Important decidability result is obtained in [9]: given a context-free grammar (query) and an arbitrary graph (database), it is undecidable whether CFL-reachability problem for them is in NC or P-complete. However, Ulman and Van Gelder in [39] introduce a notion of a *polynomial fringe property* and show that a context-free grammars having this property are in NC. A context-free grammar G has the *polynomial fringe property* if and only if there is a polynomial p such that, for each regular language R recognized by an automaton with n states, $L(G) \cap R$ is either empty or contains a word shorter than $p(n)$. It is undecidable whether a context-free grammar has the polynomial fringe property. Important results from [39] can be reinterpreted in terms of CFL-reachability as follows:

1. CFL-reachability for linear languages and piecewise linear languages, and for arbitrary graphs is in NC, because corresponding grammars have the polynomial fringe property
2. The same holds for D_1 (the Dyck language on one kind of parentheses) and its GSM-mappings (one-counter languages)
3. CFL-reachability for D_2 (the Dyck language on two kinds of parentheses) is P-complete.

The third result is important because any context-free language can be represented via a regular language and D_2, which are combined by means of an intersection and a homomorphism, so it is the direct consequence of P-competeness of CLF-reachability problem in general. Also, using the fact that D_2 is included in many interesting subclasses of context-free languages, such as visibly pushdown languages [30], simple deterministic languages (defined by LL(1) grammars in Greibach normal form), we can state that CFL-reachability

for these languages is P-complete. Afrati et al. [1] investigate parallel complexity of Datalog simple chain queries and presents the Polynomial Stack Lemma which will be discussed in detail in Section 4.

The definition of polynomial fringe property coincides with the notion of a so called *rational index*: for a context-free language $L(G)$ having the polynomial rational index is the same as for G to have the polynomial fringe property. More precisely, rational index $\rho_L(n)$ is a function, which denotes the maximum length of the shortest word in $L(G) \cap R$, for arbitrary R recognized by an n-state automaton. The notion of rational index was introduced in [4] as a complexity measure for context-free languages and was investigated independently from the polynomial fringe property. In particular, it has been proved that the rational index of D_1 is in $O(n^2)$ [8]. Another important result concerns the rational index of languages, which generate all context-free languages (an example of such language is D_2). It states that the rational index of such languages is of the order $exp(\Theta(n^2/\ln n))$ [31] and, hence, this is the upper bound on the value of rational index for every context-free language. An example of a non-generating language with exponential rational index is given in [36]. Also it has been shown that for every algebraic number γ the language with the rational index in $\Theta(n^\gamma)$ exists [32].

The CFL-reachability problem is the same as the intersection non-emptiness problem for a context-free language (pushdown automaton) and a regular language (finite automaton), because a labelled graph is a special kind of a nondeterministic finite automata. Complexity of this problem is studied by Ganardi et al. [10] , Swernofsky et al. [37], Vyalyi [40].

Computational complexity of the language reachability for different variants of languages (regular, context-free, context-sensitive) and graphs (acyclic graphs, trees, grid graphs) is discussed in detail by Barret et al. [3], Holzer et al.[21], Komarath et al. [26].

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 languages, for which CFL-reachability is in NC. Effective subclasses can be useful in practice, because the general problem is not tractable [27]. For example, in case of graph databases it is important to know the complexity of a given context-free path query. Also it is natural to ask which properties of subclasses imply parallel effectiveness. Why some languages have polynomial rational indices? What is the difference between them and other subclasses of context-free languages?

The hierarchy of subfamilies of context-free languages, for which the CFL-reachability problem is in NC, is presented in Figure 1. Linear (and piecewise linear) languages and one-counter languages are uncomparable families of context-free languages, but both have polynomial rational indices (polynomial fringe property). These subfamilies have one thing in common: both are defind by strong restrictions on the stack in a pushdown automaton. Our main idea is to generalize known tractable classes by investigating the restrictions on the PDA store.

Our contributions. Our results can be summarized as follows:

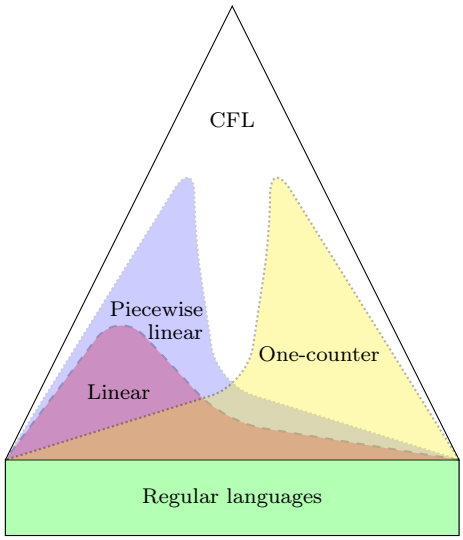

Fig. 1 The hierarchy of languages, for which CFL-reachability problem is in NC.

- We show that the CFL-reachability problem for bounded-oscillation languages of Ganty and Valput [11], is in NC (see Section 3). This class generalizes the case of linear languages.
- In Section 4 we introduce a new subclass of context-free languages — context-free languages with a poly-slender pushdown store languages. These languages are the natural generalization of one-counter languages, and the CFL-reachability problem for them is in NC. Also we show that deciding poly-slenderness of a pushdown store language is in P and give a criterion for testing whether a PDA has a storage language of polynomial or exponential density.
- Closure properties of the languages with polynomial rational indices are investigated in Section 5, particularly it is shown that the family of languages with polynomial rational is closed under Kleene star and insertion of a regular language.

2 Preliminaries

Formal languages. A *context-free grammar* is a 4-tuple $G = (\Sigma, N, P, S)$, where Σ is a finite set of alphabet symbols, N is a set of nonterminal symbols, P is a set of production rules and S is a start nonterinal. $L(G)$ is a context-free language generated by context-free grammar G. We use the notation $A \overset{*}{\Rightarrow} w$ to denote that the string $w \in \Sigma^*$ can be derived from a nonterminal 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: $A \to BC$, $A \to a$ or $S \to \varepsilon$, where $A, B, C \in N$ and $a \in \Sigma$.

The set of all context-free languages is identical to the set of languages 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 $a = a_1 a_2 ... a_k \in \Sigma^*$ $(k \geq 2)$, $h(a) = h(a_1)h(a_2)...h(a_k)$.

Given a homomorphism $h : \Sigma^* \to \Delta^*$ and a language L define

$$h(L) = \{h(w) | w \in L\} \subseteq \Delta^*.$$

Insertion of a language K into a language L is a language

$$L' = \{uxv | x \in K, u, v \in L\}.$$

A *full trio* is a family of languges is closed under arbitrary homomorphism and intersection with regular language. A *full AFL (abstract family of languages)* is a full trio closed under union, concatenation and the Kleene plus.

A *regular language* is a language that can be expressed with a regular expression or a deterministic or non-deterministic finite automata. A *nondeterministic finite automaton* (NFA) is represented by a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite set of input symbols, $\delta : Q \times \Sigma \to 2^{|Q|}$ is a transition function, $q_0 \in Q$ is a start state, $F \subseteq Q$ is a set of accepting (final) states. *Deterministic finite automaton* is a 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.

For a language L over an alphabet Σ, its rational index ρ_L is a function defined as follows:

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

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.

Context-free language reachability. A *directed labelled graph* is a triple $D = (Q, \Sigma, \delta)$, where Q is a finite set of nodes, Σ is a finite set of alphabet symbols, and $\delta \subseteq Q \times \Sigma \times Q$ is a finite set of labeled edges. Let $L(D)$ denote a graph language — a regular language, which is recognized by a NFA obtained from a directed labelled graph D.

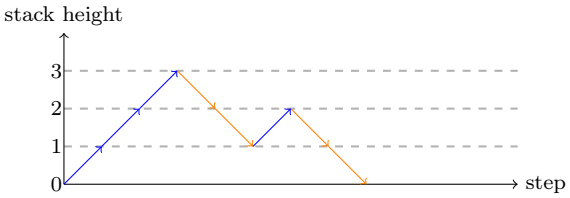

Fig. 2 Stack heights during the run of PDA.

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 $L \subseteq \Sigma^*$ be a context-free language and $D = (Q, \Sigma, \delta)$ 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 $l(\pi) \in L$.

For a context-free grammar $G = (\Sigma, N, P, S)$ and directed labelled graph $D = (Q, \Sigma, \delta)$, a triple (A, i, j) is *realizable* iff there is a path $i\pi j$ such that $A \overset{*}{\Rightarrow} l(\pi)$ for some nontermimal $A \in N$.

There are four varieties of CFL-reachability problems: all-pairs problem, single-source problem, single-target problem and single-source/single-target problem [35]. 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.

When this problem is restricted to some language L (not necessary context-free), it is called *L-reachability*.

3 Bounded-oscillation languages

Bounded-oscillation languages were introduced by Ganty and Valput [11]. 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 the stack height varies over time. Oscillation is defined using a hierarchy of *harmonics*. Let $\bar{a}$ be a *push*-move and a be a *pop*-move. Then a PDA run can be recursively described by well-nested subsequence of $\bar{a}$-s and a-s as follows:

- order 1 harmonic h_1 is $\bar{a}a\bar{a}a$ (*push pop push pop*)
- harmonic h_2 is $\bar{a}$ <order 1 harmonic> $a\bar{a}$ <order 1 harmonic> a
- $h_{(i+1)}$ 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. *Bounded-oscillation languages* are languages accepted by pushdown automata restricted to k-oscillating runs. It is important that the problem whether a given CFL is a bounded-oscillation language is undecidable [11].

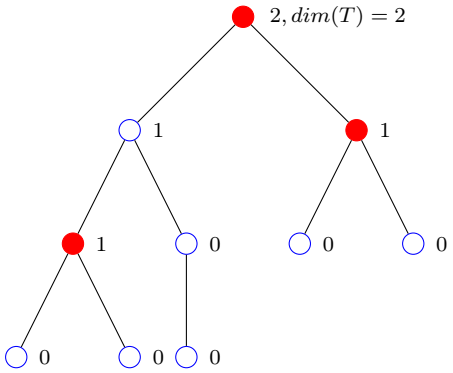

Fig. 3 A tree T with $dim(T) = 2$. Nodes having children without unique maximum are filled.

Example 1 Consider Figure 2. It shows how the stack height changes during the run of a PDA. Corresponding well-nested word is $\bar{a}\bar{a}\bar{a}aa\bar{a}aa$. The greatest harmonic in this word is order 1 harmonic (moves forming harmonic are marked in bold): $\bar{a}\bar{a}\mathbf{\bar{a}a}a\mathbf{\bar{a}a}a$, 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 $dim(v) = 0$
- If v is an internal node with k children $v_1, v_2, ..., v_k$ for $k \geq 1$, then

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

Dimension of a parse tree T $dim(T)$ is a dimension 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 3.

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

Lemma 1 ([11]) *Let a grammar $G = (\Sigma, N, P, S)$ be in Chomsky normal form and let T be a parse tree of G. Then $osc(T) - 1 \leq dim(T) \leq 2osc(T)$.*

Before we consider the value of the rational index for k-bounded-oscillation languages, we need to prove the following.

Lemma 2 *Let $G = (\Sigma, N, P)$ be a context-free grammar, $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Let w be the shortest string in $L(G) \cap L(D)$. Then a height of a parse tree for w does not exceed $|N|n^2$.*

Proof Assume that the parse tree for w 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 w 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 w' which is shorter than w. But w is the shortest, then we have a contradiction.

From Lemma 2 we have that rational index of linear languages is in $O(n^2)$.

Lemma 3 *Let G be a grammar $G = (\Sigma, N, P, S)$ in Chomsky normal form, such that every parse tree T has $dim(T) \leq d$, where d is some constant. Let $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then $\rho_{L(G)}$ is in $O((|N|n^2)^d)$.*

Proof Proof by induction on dimension $dim(T)$.
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 2). Every internal node of such tree has two children, one of which 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.
Inductive step. $dim = d + 1$.
Assume that $\rho_{L(G)}$ 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 $\rho_{L(G)}$ is no more than $O(h^d) = O((|N|n^2)^d)$ for every d.

Combining Lemma 1 and Lemma 3, we can deduce the following.

Theorem 1 *Let L be a k-bounded-oscillation language with grammar $G = (\Sigma, N, P, S)$ in Chomsky normal form and $D = (V, E, \Sigma)$ be a directed labelled graph with n nodes. Then $\rho_{L(G)}$ is in $O((|N|n^2)^{k/2})$.*

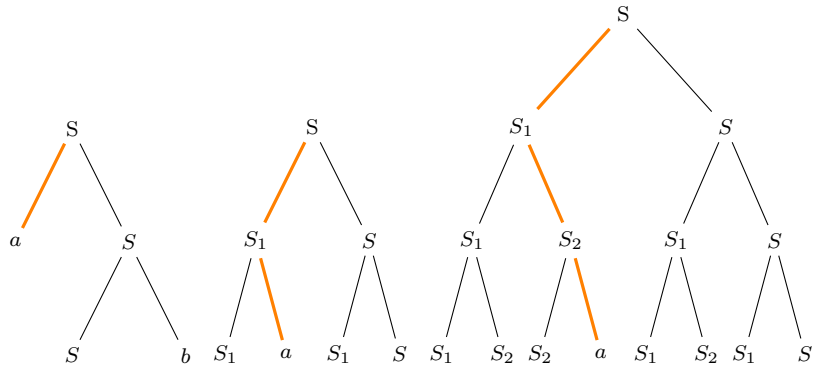

Fig. 4 A parse trees and critical paths for piecewise linear grammars in the Chomsky normal form for $|N| = 1, 2, 3$.

As we can see from the proof above, the family of linear languages is included in the family of bounded-oscillation languages. The reason is that the family of bounded-oscillation languages generalizes the family of languages accepted by finite-turn pushdown automata [11]. It is interesting that for arbitrary CFL, particularly for D_2, the value of osciillation is not constant-bounded: it depends on the length of input and does not exeed $O(\log n)$ for the input of length n [18, 41]. However, for some previously studied subclasses of context-free languages, oscillation is bounded by a constant.

Example 2 (Superlinear languages [5].)
A context-free grammar $G = (\Sigma, N, P, S)$ 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 $B \in N_L$, $\alpha \in \Sigma^*$.

From the grammar G it is observable that its parse trees have dimension at most 2. From Lemma 3, if dimensions of all parse trees are bounded by some k then the rational index of such language is polynomial, so CFL-reachability problem for superlinear languages and an arbitrary graph is in NC.

Example 3 (Piecewise linear languages.)
The family of piecewise linear queries is known to be a large class of Datalog queries which have polynomial fringe property [39]. Those queries can be described via piecewise linear grammars. A grammar $G = (\Sigma, N, P, S)$ is *piecewise linear* if every nonterminal symbol $A \in N$ generates a derivation with at most one A by any sequence of applying the production rules. A piecewise linear language is a language generated by piecewise linear grammar. We show that the family of piecewise linear languages is subclass of bounded-oscillation languages by the following Lemma.

Lemma 4 *Let $G = (\Sigma, N, P, S)$ be a piecewise linear grammar in Chomsky normal form. Then $dim(T) \leq |N| + 1$ for every parse tree T of G.*

Proof Recall that dimension of a parse tree is the maximum height of its perfect binary subtree. Let a *critical path* be the longest path in parse tree, such that all nonterminals along this path are distinct. The length of the critical path is obviously bounded by the number of nonterminals of grammar. Consider parse tree T of G and its arbitrary subtree T'. We show that every perfect binary subtree T' has a critical path. Suppose that the root of T' is labelled by some nonterminal S. The root has two children. One of the children and its descendants can not be labelled by S, otherwise G is not piecewise linear. Thus such child should have a different label S_1. Consider the subtree T'' of T' rooted by S_1. The root of T'' should have child, which is not labelled by S and S_1, so it is labelled by a distinct nonterminal S_2. Going from up to down, a distict nonterminal should be used until the path ends with a terminal symbol. So, such path is critical. Examples of parse trees and critical paths in them are shown in Figure 4. If T' is a perfect binary tree, its height is bounded by the length of the critical path. This completes the proof.

4 Languages with poly-slender storage languages

In the previous section restriction of PDA in terms of variability of stack height was described. But this is not the case for D_1, which is not k-oscillating CFL for any k, but has the polynomial rational index. In this section another kind of stack restriction is considered — poly-slenderness of a pushdown storage language as a measure of how stack contents vary along accepting computations of PDA.

For a PDA M, its *pushdown store language* $P(M)$ consists of all words occurring on the stack along accepting computations of M. It is well-known that the store language of any PDA is regular. The language D_1 is a one-counter language, so its pushdown store language is Z^*Z_0, where Z is a single pushdown symbol and Z_0 is a bottom symbol Z_0.

Afrati et al. [1] define the notion of *polynomial stack property* and show that if a PDA has the polynomial stack property, then corresponding query has the polynomial fringe property (and hence, lies in NC). A PDA has the polynomial stack property iff the largest possible number of different contents of the same height k along the any accepting computation of M is bounded by polynomial $O(k^d)$ for $d \geq 0$. For example, the usual PDA for D_1 has the polynomial stack property, because there is only one possible variant of contents for every stack height.

Generalizing an example of the family of one-counter languages, we can define the family of languages whose PDAs have the polynomial stack property — languages with a *poly-slender* pushdown store language (or storage language with polynomial density). The density of a language is a function $f(n)$ that shows the number of words of length n in language. A language $L \subseteq \Sigma^*$ is

called *poly-slender language (or with the polynomial density)* if the function $f(n)$ is bounded by $O(n^k)$ for some $k \geq 0$. For example, the language Z^*Z_0 is of polynomial density (even of a constant density), whereas the language $(Z_1 + Z_2)^*Z_0$ is of exponential density.

Thus we have the following corollary.

Corollary 1 *Let L be a context-free language and M be a PDA recognizing it. If the pushdown storage language $P(M)$ is poly-slender, then the CFL-reachability problem for L and an arbitrary given graph is in NC complexity class.*

The property of having a poly-slender storage language implies the polynomial stack property, but the converse is not true: there are PDAs with a storage language of an exponential density, which have the polynomial stack property. Consider the language of even-length palindromes $L = \{ww^R \mid w \in \{0,1\}^*\}$. It is easy to see that usual PDA M for this language has the storage language $P(M) = (0+1)^*$, which is of an exponential density. But the language L is linear and the PDA M is a finite-turn automaton, therefore M has bounded stack heights during every accepting computation, and, hence, has the polynomial stack property.

Whereas the polynomial fringe property of a query is undecidable [39], it is decidable in polynomial time whether a given PDA has a poly-slender storage language. At first, for a given NFA it is decidable whether its language has a polynomial or exponential density [20, 38] . The Lemma 5 gives a useful condition of poly-slenderness of a language using the notion of commutativity. A language $L \subseteq \Sigma^*$ is said to be *commutative* if there exists $\omega \in \Sigma^*$ such that $L \subseteq \omega^*$.

Lemma 5 ([12, 15]) *Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA. For every $q \in Q$ define an NFA $M_q = (Q, \Sigma, \delta, q, \{q\})$ with $L_q = L(M_q)$. Then the language $L(M)$ has polynomial density if and only if for every $q \in Q$, L_q is commutative.*

Gawrychowski et al. [12] give an algorithm for testing whether $L(M)$ is of polynomial or exponential density in $O(|Q| + |\delta|)$ time for an NFA $M = (Q, \Sigma, \delta, q_0, F)$. An NFA for pushdown store language of a given PDA $\mathcal{A} = (Q', \Sigma', \Gamma, \delta', q'_0, Z_0, F')$ can be constructed directly in $O(|Q'|^5|\Gamma|^2|\delta'|)$ time [29]. This construction uses the notion of meaningful triples, which form the states of NFA. A triple $[p, Z, q] \in Q' \times \Gamma \times Q'$ is *meaningful* if there exists a computation of $\mathcal{A}$ starting from state p with the sole symbol Z in the pushdown, and ending in q with the empty pushdown. By definition, there are at most $|\Gamma||Q'|^2$ meaningful triples, and, hence, states of NFA. Thus, we immediately deduce the following.

Corollary 2 *Given a pushdown automaton M, it can be decided whether $P(M)$ is poly-slender in polynomial time.*

We show a criteria for having a poly-slender storage language for a given PDA in normal form. A PDA is said to be in the *normal form* if any transition $(q, \omega) \in \delta(p, a, Z)$ satisfies $|\omega| \leq 2$. It is known that each PDA M accepting a

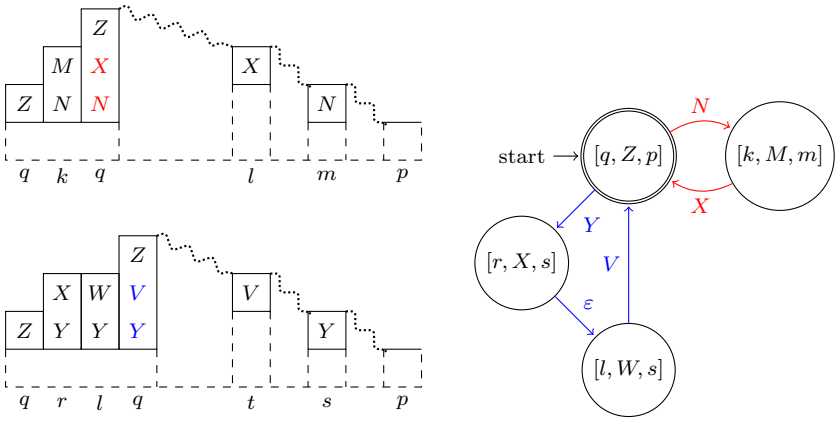

Fig. 5 A possible computations of the PDA M leading to the two different stack contents (left) and a corresponding part of NFA recognizing $L_{[q,Z,p]}$ (right).

non-emplty language can be converted into an equivalent PDA M' in normal form, preserving the same pushdown store language, with a state set $|Q'| \leq |Q|$, where Q and Q' are state sets of M amd M' respectively [13, 29].

Theorem 2 *Let M be a pushdown automaton in normal form with state set Q. Poly-slenderness of a pushdown storage language $P(M)$ can be tested as follows:*

1. *Find the set S of all meaningful triples of M;*
2. *For every pair (q, Z) of $s \in S$ find whether there is a computation started in the state q with a simbol Z on the top of the stack and ended in the same state with the same symbol on top, while the pushdown height does not decrease during the computation;*
3. *All states envolved in computation should form meaningful triples;*
4. *If such computations for every $[q, Z, p]$ leads to only variant of the stack contents, then $P(M)$ is of polynomial density, if there are at least two variants of contens for some meaningful triple — P(M) is of exponential density.*

Proof Consider an example of possible computation of a PDA M in Figure 5. Without loss of generality suppose that M has the following transitions from state q to q, satisfying the criteria from the theorem:

- $\delta(q, \sigma, Z) \in (r, YX), (k, NM)$;
- $\delta(r, \sigma', X) \in (l, W)$;
- $\delta(l, \sigma'', W) \in (q, VZ)$;
- $\delta(k, \sigma''', M) \in (q, XZ)$.

According to the construction of an NFA $\mathcal{A} = (S, \Gamma, \delta', [q_I, Z_0, q_f], t_f)$ recognizing pushdown store language of M [29], the NFA $\mathcal{A}$ has a state $[q, Z, p]$.

Consider a new NFA $\mathcal{A}_{[q,Z,p]} = (S, \Gamma, \delta', [q,Z,p], \{[q,Z,p]\})$ with $L_{[q,Z,p]} = L(\mathcal{A}_{[q,Z,p]})$, which is illustrated in Figure 5 (right). If there are at least two computations of PDA resulting in different stack contents for the same state q and top symbol Z, where $[q,Z,p]$ is meaningful triple, then there are two paths a and b from $[q,Z,p]$ to $[q,Z,p]$ in NFA $\mathcal{A}_{[q,Z,p]}$ such that $ab \neq ba$. Thus, $L(\mathcal{A}_{[q,Z,p]})$ is not commutative and by Lemma 5, $P(M)$ has exponential density. If there is only variant of the stack contents, than NFA $\mathcal{A}_{[q,Z,p]}$ has only one path from $[q,Z,p]$ to $[q,Z,p]$ or several paths satisfying $ab = ba$ for each pair a, b of these paths, therefore $L(\mathcal{A}_{[q,Z,p]})$ is commutative. If it holds for every meaningful triple, then $P(M)$ has polynomial density.

5 Closure properties of languages with polynomial rational indices

Given a context-free language L with a polynomial rational index, it is interesting to find which language operations preserve this property. Boasson et al. [4] give following useful relations for polynomial indices of two languages L and L'.

Theorem 3 ([4]) *Context-free languages with polynomial rational indices are closed under intersection with a regular language, union, concatenation, homomorphism and inverse homomorphism. More precisely,*

- $\rho_{L \cup L'} \leq \max(\rho_L, \rho_{L'})$
- $\rho_{LL'} \leq \rho_L + \rho_{L'}$
- $\rho_{L \cap R}(n) \leq \rho_L(nm)$, *where R is a regular language recognised by an m-state automaton*
- $\rho_{h(L)}(n) \leq \rho_L(n)$ *and* $\rho_{h^{-1}(L)}(n) < n(\rho_L(n) + 1)$, *where $h : \Sigma^* \to \Delta^*$ is a homomorphism.*

From the relations above it is easy to see that the family of context-free languages with polynomial rational indices is a full trio. Every full trio is closed under prefix and quotient with regular languages. Obviously, CFLs with polynomial rational indices languages are closed under reversal. Next we show that context-free languages with polynomial rational indices are closed under Kleene star and insertion of a regular language (or context-free language with a polynomial rational index).

Theorem 4 *Context-free languages with polynomial rational indices are closed under Kleene star and insertion of a regular language (or context-free language with a polynomial rational index). Particularly,*

- $\rho_{L^*}(n) \leq n(\rho_L(n))$
- $\rho_{L_{INSERT(K)}}(n) \leq \rho_L + \rho_K$

Proof Kleene star. Let $G = (\Sigma, N, P, S)$ be a grammar in Chomsky normal form and $L(G)$ be a language with polynomial rational index. Consider the language L^*. A grammar G' for L^* can be constructed from G as usual by

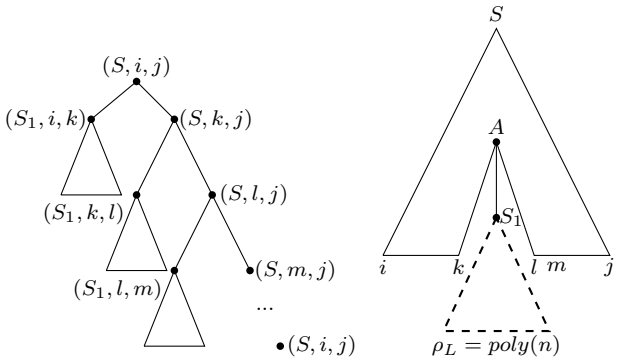

Fig. 6 Parse trees after applying Kleene star operation (left) and insersion of the language with a polynomial rational index (right).

renaming start nonterminal S to S_1 and adding a new start nonterminal S with production rules $S \to S_1 S$ and $S \to \varepsilon$. Consider a parse tree from G', which is illustrated in Figure 6 (left). Suppose that this tree is a parse tree for the shortest word w in the intersection of $L(G')$ and an arbitrary n-state NFA with the start and final states i and j respectively. Such tree has no more than n subtrees rooted by S_1, because there is no more than n triples in form of (S, x, j) for a fixed j (otherwise w is not the shortest). Every tree rooted by S_1 has at most polynomial number of leaves, because $L(G)$ has a polynomial rational index. Thus $\rho_{L^*}(n) \leq n(\rho_L(n))$.

Insertion of a regular language (or context-free language with a polynomial rational index). Proof by contradiction. Consider a parse tree T for the shortest word w from the intersection of $L_{INSERT(K)}$, where K is a language with a polynomial rational index, and an arbitrary n-state NFA $\mathcal{A}$ with the start and final states i and j respectively (see Figure 6 (right)). Let $w = l(i\pi j)$. Suppose that T has exponential on n number of leaves and, hence, a path $i\pi j$ has an exponential length. A subtree rooted by S_1 belongs to the grammar, which generates words from K, therefore this subtree has at most polynomial number of leaves by the definition of rational index. So, the whole tree, except the subtree rooted by S_1, has an exponential number of leaves. Suppose m is the next state in the shortest path from l to j in $i\pi j$. Then one can construct from $\mathcal{A}$ a new n-state NFA $\mathcal{A}'$, where state k has an edge to m labelled the same way as the transition from l to m in $\mathcal{A}$. Notice that the new edge does not affect the path from i to k, because this path is the shortest and, hence, does not contain any subpath from k to m. The same holds for the path from m to j. The the shortest word in the intersection of L and $\mathcal{A}'$ should have an exponetial length, but L has a polynomial rational index, a contradiction.

Family of languages with polynomial rational indices is a full trio closed under union, concatenation and Kleene star, therefore it is a full ALF. Full AFLs is known to be closed under substitution.

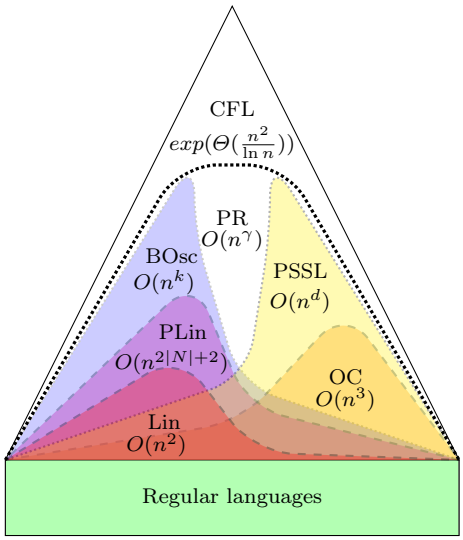

Fig. 7 The hierarchy of languages with polynomial rational indices and corresponding upper bounds on the value of rational index. PR — the family of CFLs with a polynomial rational indices, BOsc — bounded-oscillation languages, PSSL — CFLs with poly-slender storage languages, PLin — piecewise linear languages, OC — one-counter languages, Lin — linear languages, n — number of vertices in graph (NFA), $|N|$ — the number of non-terminals of grammar in Chomsky normal form, k — the oscillation value, d — degree of polynomial density of a pushdown storage language, γ — algrebraic number.

Using closure properties, it is easier to find new subclasses of context-free languages for which CFL-reachability problem is in NC.

Example 4 (Metalinear languages [14].)
Let $G = (\Sigma, N, P, S)$ be a context-free grammar. G is *metalinear* if all productions of P are of the following forms:

1. $S \to A_1 A_2 ... A_k$, where $A_i \in N - \{S\}$
2. $A \to u$, where $A \in N \setminus \{S\}$ and $u \in (\Sigma^*((N \setminus \{S\}) \cup \varepsilon)\Sigma^*)$

The width of a metalinear grammar is $max\{k \mid S \to A_1 A_2 ... A_k\}$. Metalinear languages of width 1 are obviously linear languages. It is easy to see that every metalinear language is a union of concatenations of k linear languages. Linear languages have polynomial rational index, CFLs with polynomial rational index are closed under concatenation and union, so metalinear languages have polynomial rational index.

6 Conclusions and open problems

We have obtained two classes, which extend the classes in the recent literature [1, 21, 26, 36, 39], for which CFL-reachability problem is in NC. The one is the class of bounded-oscillation languages, which generalizes the linear languages.

The second class is context-free languages with a poly-slender pushdown store languages, which is generalization of the one-counter languages. Recall that regular languages have polynomial fringe property (and are accepted by PDA with a bounded stack height), also it is known that L-reachibility for regular languages is in NL [26, 42]. Thereby it has been demonstrated that some natural restrictions on the pushdown storage implies polynomial rational index for the corresponding context-free languages: low variability of stack height during the PDA run (bounded-oscillation PDA) and limited number of possible stack contents (languages with poly-slender pushdown store languages). The updated hierarchy of tractable subclasses and corresponding upper bounds on the rational indices are illustrated in Figure 7.

It will be interesting to know whether there is another kind of stack restriction which implies polynomial rational index. Or is there a context-free language which does not belong to any of the above mentioned classes? Are there any other properties (except polynomial rational index) which make the CFL-reachability problem solvable in NC? For example there is a Datalog query, which does not have a polynomial fringe property but its evaluation is in NC [25]. One can also approach this question from another direction by looking for simple subfamilies of context-free languages that would have P-complete CFL-reachability problem.

We considered CFL-reachability problem for a fixed context-free languages and arbitrary graphs. What tractable cases can be obtained for a fixed graphs and an arbitrary context-free language? The known and trivial examples are acyclic graphs and trees. Can we have more complicated classes of graphs for which CFL-problem is in NC? Interesting algebraic properties of such graphs (NFA) are given in [10], but automata-theoretic characterizations of these properties remain to be found.

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. Azimov R, Grigorev S (2018) Context-free path querying by matrix multiplication. In: Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), ACM, New York, NY, USA, GRADES-NDA '18, pp 5:1–5:10, DOI 10.1145/3210259.3210264, URL http://doi.acm.org/10.1145/3210259.3210264
3. Barrett C, Jacob R, Marathe M (2000) Formal-language-constrained path problems. SIAM J Comput 30:809–837, DOI 10.1137/S0097539798337716
4. Boasson L, Courcelle B, Nivat M (1981) The rational index: a complexity measure for languages. SIAM Journal on Computing 10(2):284–296

5. Brzozowski JA (1968) Regular-like expressions for some irregular languages. In: 9th Annual Symposium on Switching and Automata Theory (swat 1968), pp 278–286, DOI 10.1109/SWAT.1968.24
6. Cai C, Zhang Q, Zuo Z, Nguyen K, Xu G, Su Z (2018) Calling-to-reference context translation via constraint-guided cfl-reachability. pp 196–210, DOI 10.1145/3192366.3192378
7. 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
8. 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
9. Gaifman H, Mairson H, Sagiv Y, Vardi M (1987) Undecidable optimization problems for database logic programs. vol 40, pp 106–115, DOI 10.1145/174130.174142
10. Ganardi M, Hucke D, König D, Lohrey M (2016) Circuit evaluation for finite semirings. 1602.04560
11. Ganty P, Valput D (2016) Bounded-oscillation pushdown automata. Electronic Proceedings in Theoretical Computer Science 226:178–197, DOI 10.4204/eptcs.226.13, URL http://dx.doi.org/10.4204/EPTCS.226.13
12. Gawrychowski P, Krieger D, Rampersad N, Shallit J (2008) Finding the growth rate of a regular of context-free language in polynomial time. In: Ito M, Toyama M (eds) Developments in Language Theory, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 339–358
13. Ginsburg S (1966) The Mathematical Theory of Context-Free Languages. McGraw-Hill, Inc., USA
14. Ginsburg S (1966) The Mathematical Theory of Context-Free Languages. McGraw-Hill, Inc., USA
15. Ginsburg S, Spanier EH (1966) Bounded regular sets. Proceedings of the American Mathematical Society 17(5):1043–1049
16. Greenlaw R, Hoover HJ, Ruzzo WL (1995) Limits to Parallel Computation: P-completeness Theory. Oxford University Press, Inc., New York, NY, USA
17. 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
18. 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
19. Hellings J (2015) Path results for context-free grammar queries on graphs. CoRR abs/1502.02242, 1502.02242

20. HIbarra O, Ravikumar B (1986) On sparseness, ambiguity and other decision problems for acceptors and transducers. In: Monien B, Vidal-Naquet G (eds) STACS 86, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 171–179
21. 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
22. Huang W, Dong Y, Milanova A, Dolby J (2015) Scalable and precise taint analysis for android. pp 106–117, DOI 10.1145/2771783.2771803
23. 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
24. 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
25. Kanellakis PC (1986) Logic programming and parallel complexity. In: Ausiello G, Atzeni P (eds) ICDT '86, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 1–30
26. Komarath B, Sarma J, Sunil KS (2014) On the complexity of l-reachability. In: Jürgensen H, Karhumäki J, Okhotin A (eds) Descriptional Complexity of Formal Systems, Springer International Publishing, Cham, pp 258–269
27. Kuijpers J, Fletcher G, Yakovets N, Lindaaker T (2019) An experimental study of context-free path query evaluation methods. In: Proceedings of the 31st International Conference on Scientific and Statistical Database Management, ACM, New York, NY, USA, SSDBM '19, pp 121–132, DOI 10.1145/3335783.3335791, URL http://doi.acm.org/10.1145/3335783.3335791
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. Malcher A, Meckel K, Mereghetti C, Palano B (2012) Descriptional complexity of pushdown store languages. J Autom Lang Comb 17(2):225–244
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. Rehof J, Fähndrich M (2001) Type-base flow analysis: From polymorphic subtyping to cfl-reachability. In: Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Association for Computing Machinery, New York,

NY, USA, POPL '01, pp 54–66, DOI 10.1145/360204.360208, URL https://doi.org/10.1145/360204.360208
34. 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
35. Reps TW (1997) Program analysis via graph reachability. Information & Software Technology 40:701–726
36. Rubtsov A, Vyalyi M (2015) Regular realizability problems and context-free languages. In: Shallit J, Okhotin A (eds) Descriptional Complexity of Formal Systems, Springer International Publishing, Cham, pp 256–267
37. 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
38. Szilard A, Yu S, Zhang K, Shallit J (1992) Characterizing regular languages with polynomial densities. In: Havel IM, Koubek V (eds) Mathematical Foundations of Computer Science 1992, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 494–503
39. 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
40. Vyalyi MN (2013) Universality of regular realizability problems. In: Bulatov AA, Shur AM (eds) Computer Science – Theory and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 271–282
41. 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
42. Yannakakis M (1990) Graph-theoretic methods in database theory. pp 230–242, DOI 10.1145/298514.298576
43. Zhang Q, Lyu MR, Yuan H, Su Z (2013) Fast algorithms for dyck-cfl-reachability with applications to alias analysis. SIGPLAN Not 48(6):435–446, DOI 10.1145/2499370.2462159, URL https://doi.org/10.1145/2499370.2462159
44. Zhang X, Feng Z, Wang X, Rao G, Wu W (2016) Context-free path queries on rdf graphs. In: Groth P, Simperl E, Gray A, Sabou M, Krötzsch M, Lecue F, Flöck F, Gil Y (eds) The Semantic Web – ISWC 2016, Springer International Publishing, Cham, pp 632–648