Re-pairing brackets

Dmitry Chistikov
Centre for Discrete Mathematics and its Applications (DIMAP)
Department of Computer Science
University of Warwick
Coventry, United Kingdom
d.chistikov@warwick.ac.uk

Mikhail Vyalyi
National Research University Higher School of Economics
Moscow, Russia
Dorodnicyn Computing Centre, FRC CSC RAS
Moscow, Russia
Moscow Institute of Physics and Technology
Dolgoprudny, Moscow Region, Russia
vyalyi@gmail.com

Abstract

Consider the following one-player game. Take a well-formed sequence of opening and closing brackets (a Dyck word). As a move, the player can *pair* any opening bracket with any closing bracket to its right, *erasing* them. The goal is to re-pair (erase) the entire sequence, and the cost of a strategy is measured by its *width:* the maximum number of nonempty segments of symbols (separated by blank space) seen during the play.

For various initial sequences, we prove upper and lower bounds on the minimum width sufficient for re-pairing. (In particular, the sequence associated with the complete binary tree of height n admits a strategy of width sub-exponential in $\log n$.) Our two key contributions are (1) lower bounds on the width and (2) their application in automata theory: quasi-polynomial lower bounds on the translation from one-counter automata to Parikh-equivalent nondeterministic finite automata. The latter result answers a question by Atig et al. (2016).

CCS Concepts: • **Mathematics of computing → Combinatoric problems**; • **Theory of computation → Automata extensions**; **Regular languages**.

Keywords: automata theory, one-counter automata, Parikh image, Dyck language, balanced parentheses, combinatorics

ACM Reference Format:
Dmitry Chistikov and Mikhail Vyalyi. 2020. Re-pairing brackets. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3373718.3394752

1 Introduction

Consider the following one-player game. Take a well-formed sequence of opening and closing brackets; that is, a word in the (1-)Dyck language. As a move, the player can *pair* any opening bracket with any closing bracket to its right, *erasing* them. The two brackets do not need to be adjacent or matched to each other in the word. The goal is to re-pair (erase) the entire word, and the cost of a play is measured by its *width:* the maximum number of nonempty segments of symbols ('islands' separated by blank space) seen during the play. Here is an example:

```
  Initial word:  (()())
After move 1:  (   ())
After move 2:  (     )
After move 3:
```

Note that move 2 pairs up two brackets not matched to each other in the initial word; such moves are permitted without restrictions. At the beginning, there is a single segment, which splits into two after the first move. Both segments disappear simultaneously after the third move; the width of the play is equal to 2. In this example, width 1 is actually sufficient: a better strategy is to erase the endpoints first and then erase the two matched pairs in either order.

For a word σ, the *width* of σ is the minimum width sufficient for re-pairing it. Is it true that all well-formed (Dyck) words, no matter how long, have bounded cost, i.e., can be re-paired using width at most c, where c is independent of the word? The answer to this simply formulated combinatorial question turns out to be negative, but there does not appear to be a simple proof for this: re-pairing strategies, perhaps surprisingly, turn out quite intricate. In the present paper, we study this and related questions.

Motivation. First of all, we find the re-pairing problem interesting in its own right—as a curious combinatorial game, easily explained using just the very basic concepts in discrete mathematics. So our motivation is, in part, driven by the appeal of the problem itself, and in fact the results that we obtain add to this motivation further.

Second, and perhaps most importantly, we use the re-pairing game as a key abstraction when we prove a lower bound in another problem, answering an open question from automata theory. This question is about the complexity of translation of one-counter automata (OCA) on finite words into Parikh-equivalent nondeterministic finite automata (NFA) [7]. The translation arose in model checking, namely in the context of availability expressions [1, 26], which extend usual regular expressions by constraints counting occurrences of letters. It is more generally motivated by recent lines of research on the classic Parikh theorem [43]: on its applications in verification (see, e.g., [17, 20, 25]) and on its extensions and refinements required for them [16, 32, 33]. It has been unknown [7] whether the translation in question can be made polynomial, and in this paper we answer this question negatively: using our results on the re-pairing problem, we obtain a quasi-polynomial lower bound on the blowup in the translation. We are not aware of any other way to prove this lower bound; the reader focused on automata theory can view the present paper as one presenting a solution to the OCA to Parikh-equivalent NFA problem. (This is how we have first identified the re-pairing problem.)

Finally, viewed from a different angle, the re-pairing problem can be seen as a curious case study in the theory of *non-uniform models of computation.* As it turns out, restricted strategies, where paired brackets are always matched to each other in the word, have a close connection with the black-and-white pebble game on binary trees, a classic setting in computational complexity (see, e.g., surveys by Nordström [41, 42]). We show that unrestricted strategies in the re-pairing game make it significantly more complex than pebbling, strengthening this model.

Our contribution

In this paper, we define and study the *re-pairing* problem (game), as sketched above. Our two key contributions are (i) lower bounds on the width of Dyck words in this problem and (ii) the connection to automata theory: lower bounds on the translation from OCA to Parikh-equivalent NFA. Before giving a precise formulation of our results (see below), we explain why we would like to highlight these two.

Our lower bounds on the width are obtained by bounding the set of (Dyck) words that can be re-paired using width k (for each k). To put this into context, classic models of matrix grammars [47] and deterministic two-way transducers [45], as well as a more recent model of streaming string transducers [4, 5], extend our model of computation with additional finite-state memory. (We refer the reader to surveys [19, 38, 39] for more details on transducers.) In terms of transducers, our technique would correspond to determining the expressive power of machines of bounded size. Existing results of this kind (see [19, 38]) apply to variants of the model where *concatenation is restricted:* with [6] or without [13] restrictions on output and in the presence of nondeterminism [8]. Here our result makes a step towards a first lower bound against the *unrestricted* model.

In the application of the re-pairing problem to automata theory, our lower bounds on the size of NFA for the Parikh image can be viewed as lower bounds on the size of commutative NFA, i.e., nondeterministic automata over the free commutative monoid (cf. [15, 24, 30–33]). To the best of our knowledge, we are the first to develop lower bounds (on description size) for this simple non-uniform model of computation. It is well-known that, even for usual NFA, obtaining lower bounds on the size (number of states) is challenging and, in fact, provably (and notoriously) hard; and that the available toolbox of techniques is limited (see, e.g., [22, 27] and [29]). From the 'NFA perspective', we first develop a lower bound for a different model of computation and then import this result to NFA using combinatorial tools, which we thus bring to automata theory: the Birkhoff—von Neumann theorem on doubly stochastic matrices (see, e.g., [50, p. 301]) and the Nisan—Wigderson construction of a family of sets with pairwise low intersection [40]. The obtained lower bounds point to a limitation of NFA that does not seem to have the form of the usual communication complexity bottleneck (cf. [51, Theorem 3.11.4], [29], and the book by Hromkovic [28]); exploring and exploiting this further is a possible direction for future research.

Summary of our results. Our main results for the re-pairing game are as follows:

1. We show that every well-formed (Dyck) word σ has a re-pairing of width $O(\log|\sigma|)$, where $|\sigma|$ is the length of σ.
   This re-pairing always pairs up brackets that are matched to each other in σ; we call re-pairings with this property *simple.* It is standard that Dyck words are associated with trees; for words σ associated with *binary* trees, we show that the minimum width of a simple re-pairing is equal (up to a constant factor) to the *minimum number of pebbles in the black-and-white pebble game* on the associated tree, a quantity that has been studied in computational complexity [11, 35, 41, 42] and captures the amount of space used by nondeterministic computation.
   In particular, this means [35–37] that for the word $Z(n)$ associated with a complete binary tree of height n (see definition in Section 2), the minimum width of simple re-pairings is $\Theta(n)$, which is logarithmic in the length of $Z(n)$.
2. For $Z(n)$, we show how to beat this bound, giving a (non-simple) recursive re-pairing strategy of width $2^{O(\sqrt{\log n})}$. This is a function sub-exponential in $\log n$; it grows faster than all $(\log n)^k$, but slower than all n^ε, $\varepsilon > 0$.
3. For $Z(n)$ and for a certain 'stretched' version of it, $Y(\ell)$ (see definition in Section 5), we prove lower bounds

on the width of re-pairings:

$$\text{width}(Z(n)) = \Omega\left(\frac{\log\log|Z(n)|}{\log\log\log|Z(n)|}\right) = \Omega\left(\frac{\log n}{\log\log n}\right),$$
$$\text{width}(Y(\ell)) = \Omega\left(\sqrt{\frac{\log|Y(\ell)|}{\log\log|Y(\ell)|}}\right) = \Omega(\ell). \quad (1)$$

We further show how to use the re-pairing game in automata theory—the application that motivated our study of this game in the first place:

4. As an application of our lower bounds in the re-pairing game, we prove that there is no polynomial-time translation from one-counter automata (OCA) on finite words to Parikh-equivalent nondeterministic finite automata (NFA). This shows that optimal translations must be quasi-polynomial, answering a question by Atig et al. [7].
   To prove this result, we consider OCA from (a variant of) a specific *complete* family, $(\mathcal{H}_n)_{n\geq 2}$, identified by Atig et al. [7]. (There is a polynomial translation from (all) OCA to Parikh-equivalent NFA if and only if these OCA $\mathcal{H}_n$ have Parikh-equivalent NFA of polynomial size.) We prove, for *every* Dyck word σ_n of length $\leq \sqrt{n/8}$, a lower bound of $n^{\Omega(\text{width}(\sigma_n))}$ on the minimum size of NFA accepting regular languages Parikh-equivalent to $L(\mathcal{H}_n)$. Based on the words $Y(\ell)$, we get a lower bound of
   $$n^{\Omega\left(\sqrt{\log n/\log\log n}\right)}$$
   on the size of NFA. Note that this holds for NFA that accept not just a specific regular language, but *any* language Parikh-equivalent to the one-counter language $L(\mathcal{H}_n)$ (there are infinitely many such languages for each n).

In this extended abstract, we introduce main definitions and show the main results for the re-pairing game, including proof ideas and sketches. We show the proof of the theorem linking re-pairings to the OCA to Parikh-equivalent NFA problem (Theorem 5) in more detail. Full proofs for all our results can be found in the full version of the paper [10].

Background and related work

Parikh image of one-counter languages. The problem of re-pairing brackets in well-formed (Dyck) words is linked to the following problem in automata theory.

The *Parikh image* (or commutative image) of a word u over an alphabet Σ is a vector of dimension $|\Sigma|$ in which the components specify how many times each letter from Σ occurs in u. The Parikh image of a language $L \subseteq \Sigma^*$ is the set of Parikh images of all words $u \in L$. It is well-known [43] that for every context-free language L there exists a regular language R with the same Parikh image (*Parikh-equivalent* to L). If L is generated by a context-free grammar of size n, then there is a nondeterministic finite automaton (NFA) of size exponential in n that accepts such a regular language R (see [16]); the exponential in this translation is necessary in the worst case.

When applying this translation to a language from a proper subclass of context-free languages, it is natural to ask whether this blowup in description size can be avoided. For languages recognized by *one-counter automata* (OCA; a fundamental subclass of pushdown automata), the exponential construction is suboptimal [7]. If an alphabet Σ is fixed, then for every OCA with n states over Σ there exists a Parikh-equivalent NFA of polynomial size (the degree of this polynomial depends on $|\Sigma|$). And even in general, if the alphabet is not fixed, for every OCA with n states over an alphabet of cardinality at most n there exists a Parikh-equivalent NFA of size $n^{O(\log n)}$, quasi-polynomial in n. Whether this quasi-polynomial construction is optimal has been unknown, and we prove in the present paper a quasi-polynomial lower bound.

We note that the gap between NFA of polynomial and quasi-polynomial size grows to exponential when the translation is applied iteratively, as is the case in Abdulla et al. [1].

Matrix grammars of finite index and two-way transducers. The question of whether all well-formed (Dyck) words can be re-paired using bounded width can be linked to a question on *matrix grammars*, a model of computation studied since the 1960s [2]. Matrix grammars are a generalization of context-free grammars in which productions are applied in 'batches' prescribed by the grammar. This formalism subsumes many classes of rewriting systems, including controlled grammars, L systems, etc. (see, e.g., [12]).

The *index* of a derivation in a matrix grammar is the maximum number of nonterminals in a sentential form in this derivation (this definition applies to ordinary context-free grammars as well) [9, 21]. Bounding the index of derivations, i.e., restricting grammars to *finite index* is known to reduce the class of generated languages; this holds both for ordinary context-free [21, 23, 48] and matrix grammars [9]. Languages generated by finite-index matrix grammars have many characterizations: as languages output by deterministic two-way transducers with one-way output tape [44], or produced by EDT0L systems of finite index [34, Proposition I.2]; images of monadic second-order logic (MSO) transductions [14]; and, most recently, output languages of streaming string transducers [4, 5]. (See also the survey by Filiot and Reynier [19].)

Encoding the rules of our re-pairing problem in the matrix grammar formalism leads to a simple sequence of grammars with increasing indices for subsets of the Dyck language D_1; the question of whether all Dyck words can be re-paired using bounded width is the same as asking if any of these grammars has in fact *(bounded-index) derivations for all Dyck words.* A 1987 paper by Rozoy [47] is devoted to the proof that, in fact, no matrix grammar can generate all words in

D_1 using bounded-index derivations without also generating some words outside D_1. This amounts to saying that no finite-index matrix grammar generates D_1; and a non-constant lower bound on the width in the re-pairing problem could be extracted from the proof.

Unfortunately, the proof in that paper, as well as in an earlier paper [46], seems to be flawed. Fixing the argument does not seem possible as far as we can see, and we are not aware of an alternative proof. We discuss Rozoy's proof in the full version of the present paper [10].

2 Basic definitions

The Dyck language. We use non-standard notation for brackets in words from the Dyck language: the opening bracket is denoted by $+$ and the closing bracket by $-$; we call these symbols pluses and minuses, accordingly. Moreover, in some contexts it is convenient to interpret $+$ and $-$ as integers $+1$ and -1.

Let N be an even integer. A word $\sigma = (\sigma(1), \ldots, \sigma(N))$, $\sigma(i) \in \{+1, -1\}$, is a *Dyck word* if it has an equal number of $+1$ and -1 and for every $1 \le k \le N$ the inequality $\sum_{i=1}^{k} \sigma(i) \ge 0$ is satisfied.

We assume that a *position* in a word points to a place *between* symbols. Formally, a position corresponds to a partitioning of the word w into a prefix and suffix: $w = p \cdot s$. Thus, an n-symbol word has $n+1$ positions, including the start position (empty prefix) and the end position (empty suffix). The positions are numbered left to right, starting from 0 (the symbols are also numbered left to right, but starting from 1). The *height* of a position i in a Dyck word σ is $h(i) = \sum_{j=1}^{i} \sigma(j)$. As usual, $|\sigma|$ denotes the length of the word σ (the number of symbols in it).

Dyck words are naturally *associated* with *ordered rooted forests* (i.e., with sequences of ordered rooted trees). E.g., words $Z(n)$ defined by

$$Z(1) = +-; \quad Z(n+1) = +Z(n)Z(n)- \tag{2}$$

can be associated with complete binary trees of height $n-1$. Recall that the height of a rooted tree is the maximum length of a path (number of edges) from the root to a leaf.

Note that we described a re-pairing of the word $Z(2)$ in Section 1.

Re-pairings and their width. A *re-pairing* of a Dyck word σ is a sequence of pairs

$$p = (p_1, \ldots, p_{N/2}), \quad \text{where } p_i = (\ell_i, r_i)$$

and the following properties are satisfied:

(R1) $\sigma(\ell_i) = +1$, $\sigma(r_i) = -1$, $\ell_i < r_i$ for all i;
(R2) every number from the interval $[1, N]$ occurs in exactly one pair p_j.

(We use the word 'interval' to refer to a set of the form $[a, b] = \{x \in \mathbb{Z} : a \le x \le b\}$.)

The intuition is that the index i corresponds to discrete time, and at time i the two symbols $\sigma(\ell_i)$ and $\sigma(r_i)$ are *(re-)paired* (or *erased*). Denote by

$$B_t(p) = \{b \in [1, N] : (b = \ell_i) \text{ or } (b = r_i),\ i \le t\}$$

the set of points from $[1, N]$ that correspond to symbols erased at times $[1, t]$.

It is easy to see that re-pairings exist for every Dyck word. By induction on the length of the word one can prove a stronger statement: a sequence $(p_1, \ldots, p_t)$ can be extended to a re-pairing iff all numbers in the pairs $p_i = (\ell_i, r_i)$ are different, the property (R1) is satisfied, and the remaining signs (those which have not been erased) constitute a Dyck word. We now define the following quantities:

- *The width of a set S of integers*, $\mathrm{width}(S)$, is the smallest number of intervals the union of which is equal to S.
- *The width of a re-pairing p at time t* is $\mathrm{width}(B_t(p))$.
- *The width of a re-pairing p* of a Dyck word σ, $\mathrm{width}(p)$, is $\max_t \mathrm{width}(B_t(p))$, i.e., the maximum of the width of this re-pairing over all time points.
- *The width of a Dyck word σ*, $\mathrm{width}(\sigma)$, is the minimum of $\mathrm{width}(p)$ over all re-pairings p of σ.

We will look into how big the width of a Dyck word of length N can be, that is, we are interested in $\max_\sigma \mathrm{width}(\sigma)$, where the maximum is over all Dyck words of length N.

Remark 1. Section 1 discussed the minimization of the maximum number of the **"surviving" (non-erased) intervals**. This quantity cannot differ from the width defined above (using **erased intervals**) by more than 1.

Remark 2. A tree-based representation of re-pairings is described in Section 5. For more details please refer to the full version of the paper.

3 Simple bounds and simple re-pairings

In this section we establish several basic facts on the width of Dyck words and re-pairings. A careful use of bisection (see Subsection 3.1) leads to the following upper bound:

Theorem 1. $\mathrm{width}(\sigma) = O(\log|\sigma|)$ *for all Dyck words* σ.

We call a re-pairing of a Dyck word σ *simple* if at all times it pairs up two signs that are matching in the word σ. The re-pairing that the proof of Theorem 1 constructs is simple.

We now show a link between simple re-pairings and strategies in the following game. Let G be an acyclic graph (in our specific case it will be a tree with edges directed from leaves to root). Define a *black-and-white pebble game* on G (see, e.g., [35, 42]) as follows. There is only one player, and black and white pebbles are placed on the nodes of the graph. The following moves are possible:

(M1) place a black pebble on a node, provided that all its immediate predecessors carry pebbles;
(M2) remove a black pebble from any node;

(M3) place a white pebble on any node; and
(M4) remove a white pebble from a node, provided that all its immediate predecessors carry pebbles.

(In a tree, immediate predecessors are immediate descendants, i.e., children. Rules (M1) and (M4) are always applicable to all sources, i.e., leaves of G.) At the beginning there are no pebbles on any nodes. A sequence of moves in the game is a *strategy*; it is *successful* if it achieves the goal: reaching a configuration in which all sinks of the graph (i.e., nodes of outdegree 0; the root in the case of trees) carry pebbles and there are no white pebbles on any nodes. By $\mathrm{bw}(G)$ we will denote the minimum number of pebbles sufficient for a successful strategy in the black-and-white pebble game on G.

Theorem 2. *Suppose the tree D associated with a Dyck word σ is binary. Then the minimum width of a simple re-pairing for σ is $\Theta(\mathrm{bw}(D))$.*

Since D is a tree, it follows from the results of the papers [35–37] (see also [49, pp. 526–528]) that the value of $\mathrm{bw}(D)$ at most doubles if the strategies are not allowed to use any white pebbles. The optimal number of (black) pebbles in such strategies is determined by the so-called Strahler number (see, e.g., [18] and [35]):

Corollary 1. *For binary trees, the following two quantities are within a constant factor from each other: the minimum width of a simple re-pairing for σ and the maximum height of a complete binary tree which is a graph-theoretic minor of the tree D.*

Upper bounds provided by Theorem 1 and Corollary 1 are similar. Note that the former gives a simple re-pairing too, but also holds for non-binary trees D.

The lower bound in Theorem 2 relies on the re-pairing being simple. For instance, for the word $Z(n)$ associated with a complete binary tree (see equation (2)), the minimum width of a simple re-pairing is $\Theta(n)$, but the (usual) width is $o(n^\varepsilon)$ for all $\varepsilon > 0$ (Section 4).

3.1 Proof of the Theorem 1

In the proof we use several observations.

Claim 1. *Let $\sigma = \sigma_1 \cdot \sigma_2 \cdot \ldots \cdot \sigma_t$ be a factorization of a Dyck word into Dyck words. Then*

$$\mathrm{width}(\sigma) \le 1 + \max_{1\le i\le t} \big(\mathrm{width}(\sigma_i)\big).$$

Proof. Denote by p_i an optimal re-pairing of a word σ_i.

Consider a re-pairing p of σ which erases the words σ_1, σ_2, …, σ_t consecutively and according to the optimal re-pairings $p_1, \ldots, p_t$. The width of this re-pairing at all times when the word σ_i is being re-paired cannot exceed the number of erased intervals in the re-pairing p_i plus possibly an additional interval that consists of the fully erased word $\sigma_1 \cdot \ldots \cdot \sigma_{i-1}$. □

Claim 2. *Let a Dyck word σ factorize as $L \cdot \pi_1 \cdot R$, where π_1 is a Dyck word. Then the word $\pi_2 = LR$ is also a Dyck word and*

$$\mathrm{width}(\sigma) \le \max(\mathrm{width}(\pi_1), 1 + \mathrm{width}(\pi_2)).$$

Proof. The first statement is obvious.

Consider a re-pairing p of the word σ which first erases the word π_1 (optimally, using a re-pairing p_1) and then the word π_2 (also optimally, using a re-pairing p_2).

Before the beginning of π_2's re-pairing, the width of p does not exceed $\mathrm{width}(p_1)$. From that point on, the word π_1 is erased completely, so the width of p_2 can be increased by at most 1 (which corresponds to the erased interval π_1). □

Proof of Theorem 1. Use induction on the length of the Dyck word, $N = |\sigma|$. Construct a sequence of nested factors of σ in the following way.

Any Dyck word is a concatenation of Dyck primes, i.e. words of the form $+w-$, where w is a Dyck word. If $\sigma = \sigma_1 \ldots \sigma_m$, where σ_i are Dyck primes, then pick the factor $\sigma^{(1)} = \sigma_j$ of maximum width. Since $\sigma^{(1)}$ is a Dyck prime, $\sigma^{(1)} = +\sigma_1^{(1)} \ldots \sigma_{m(1)}^{(1)}-$, where $\sigma_i^{(1)}$ are Dyck primes. If some $\sigma_j^{(1)}$ has length greater than $N/2$, set $\sigma^{(2)} = \sigma_i^{(1)}$. Note that $\sigma^{(1)} = L_2\sigma^{(2)}R_2$, and $|L_2R_2| < N/2$.

Repeating this procedure produces a sequence of Dyck primes $\sigma^{(i)}$: if $\sigma^{(i)} = +\sigma_1^{(i)} \ldots \sigma_{m(i)}^{(i)}-$, where $\sigma_k^{(i)}$ are Dyck primes, and $|\sigma_j^{(i)}| > N/2$ for some $1 \le j \le m(i)$, then $\sigma^{(i+1)} = \sigma_j^{(i)}$. It is easy to check that $\sigma^{(1)} = L_{i+1}\sigma^{(i+1)}R_{i+1}$ and $|L_{i+1}R_{i+1}| < N/2$. The process stops on a Dyck prime $\sigma^{(f)}$ such that

$$\sigma^{(f)} = +\sigma_1^{(f)} \ldots \sigma_{m(f)}^{(f)}-$$

where $|\sigma_j^{(f)}| \le N/2$ for all $1 \le j \le m(f)$. We have $\sigma^{(1)} = L_f\sigma^{(f)}R_f$ and $|L_fR_f| < N/2$.

Applying Claim 1 to the factorization of σ, Claim 2 to the factorization $\sigma^{(1)} = L_f\sigma^{(f)}R_f$, and then again Claims 2 and 1 to the factorization of $\sigma^{(f)}$, obtain the inequality

$$\mathrm{width}(\sigma) \le 1 + \mathrm{width}(\sigma^{(1)}) \le 1 + 1 + 1 + \max_{|\tau|\le N/2} \mathrm{width}(\tau).$$

It is now easy to see that $\max\limits_{|\sigma|=N} \mathrm{width}(\sigma) \le 3\log_2 N$. □

4 Upper bound for complete binary trees

In this section we construct re-pairings of the words $Z(n)$ associated with complete binary trees of height $n-1$ and defined by equation (2) on page 4.

Theorem 3. $\mathrm{width}(Z(n)) = 2^{O(\sqrt{\log n})}$.

The upper bounds from Section 3 give $\mathrm{width}(Z(n)) = O(n)$, whilst the functions $f(n) = 2^{a\sqrt{\log n}}$ for $a > 0$ are such that $(\log n)^k = o(f(n))$ and $f(n) = o(n^\varepsilon)$ for all $k, \varepsilon > 0$.

To prove Theorem 3 we need a family of *framed words* $Z(n)^{(k)}$. Denote by $\sigma^{(k)}$ the word

$$\underbrace{++\ldots++}_{k}\sigma\underbrace{--\ldots--}_{k}. \qquad (3)$$

Using the brackets terminology, this is the word σ which is enclosed by k pairs of opening and closing brackets. We will call such words k*-framed.*

Remark 3. If $k \geq |\sigma|/2$, then $\operatorname{width}(\sigma^{(k)}) \leq 2$, because a re-pairing can erase the signs of σ from left to right, pairing each $+$ with a $-$ from the suffix and each $-$ with a $+$ from the prefix. This re-pairing is, of course, *not* simple.

We construct a family of re-pairings $p(q, n, k)$ of framed words $Z(n)^{(k)}$, where $k \leq n$ and $1 \leq q \leq (n+1)/2$ are parameters. The definition will be recursive, and q will control the 'granularity' of the recursion.

Overview. On each step of the re-pairing $p(q, n, k)$ the leftmost remaining $-$ is erased. For $n \leq 2$, it is paired with the leftmost remaining $+$. For $n > 2$, it is paired with the $+$ that we choose using the following recursive definition.

At each step of the re-pairing $p(q, n, k)$, we define an auxiliary subsequence of the word $Z(n)^{(k)}$ that forms a word $Z(q)^{(k')}$. If the leftmost remaining minus is not in the subsequence, then we pair it with the *leftmost* remaining plus. Otherwise we consider the re-pairing $p(q', q, k')$ of the word $Z(q)^{(k')}$, where we pick q' and k' below, and pair the minus using this re-pairing (more details to follow).

Stages of the re-pairing $p(q, n, k)$. The re-pairing is divided into stages, indexed by $t = 1, \ldots, 2^{n-q}$. Denote by Z_t, $1 \leq t \leq N = 2^{n-q}$, the tth leftmost occurrence (factor) of $Z(q)$ in the word $Z(n)^{(k)}$. Stage t begins at the moment when all minuses to the left of the start position i_t of the factor Z_t have been erased, and ends when stage $t+1$ begins. Define an integer sequence k_t as follows:

$$k_{2^q \cdot s+1} = 0,\ k_{2^q \cdot s+a} = \lceil \log_2 a \rceil - 1 \text{ for } 1 < a \leq 2^q,\ 0 \leq s. \quad (4)$$

At the beginning of stage t, consider the subsequence Z'_t of $Z(n)^{(k)}$ formed by the $k' = k_t$ rightmost non-erased pluses to the left of i_t; followed by the symbols of the factor Z_t; followed by the k_t leftmost non-erased minuses to the right of the end position of Z_t. (There is a separate proof that this is well-defined.) The symbols of Z'_t, written together, form the word $Z(q)^{(k_t)}$.

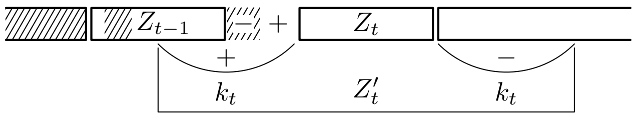

Figure 1. Beginning of stage t, with erased signs dashed.

Choose $1 \leq q' \leq q/3$ such that the width of the re-pairing $p(q', q, k_t)$ is minimal. At the first part of stage t, the re-pairing $p(q, n, k)$ pairs the signs in Z'_t according to the re-pairing $p(q', q, k_t)$. The first part ends when either all minuses to the left of the factor Z_{t+1} are erased or the sequence Z'_t is exhausted (whichever is earlier). In the latter case the second and final part of stage t is started. At each step of this part, the leftmost non-erased minus is paired with the leftmost non-erased plus.

Claim 3. *Re-pairings $p(q, n, k)$ are well-defined.*

Define $W_n = \min_q \max_{0 \leq k \leq n} \operatorname{width}(p(q, n, k))$, where the minimum is over $15 \leq q \leq n/3$ for $n \geq 45$ and over $1 \leq q \leq n/3$ for $3 \leq n < 45$.

Claim 4. $W_n \leq \min\limits_{15 \leq q \leq n/3} \left(\dfrac{2n}{q} + 2W_q + 3 \right)$ *for* $n \geq 45$.

Somewhat strangely, we have been unable to find solutions to recurrences of this form in the literature.

Claim 5. $W_n = 2^{O(\sqrt{\log n})}$.

Since $\operatorname{width}(Z(n)) \leq W_n$, Theorem 3 follows.

Proof idea for Claim 4. Assume $15 \leq q \leq n/3$. We notice that at each step at most two factors Z_t, Z_{t+1} are partially erased. (All other factors $Z_{t'}$ either have been erased completely ($t' < t$) or are yet untouched ($t' > t+1$).) Furthermore, non-erased signs to the left of the partially erased factors Z_t, Z_{t+1} form several intervals; each of them, except possibly the leftmost, has size at least q.

Note that, at each moment in time, the non-erased signs form a Dyck word, so the height of each position in $Z(n)^{(k)}$ with respect to these signs only is nonnegative. Since the height of positions in the word $Z(n)^{(k)}$ cannot exceed $n+k \leq 2n$, it follows that a partially erased factor Z_t can be preceded by at most $2n/q + 1$ non-erased intervals (runs of pluses). This leads to the recurrence of Claim 4.

5 Lower bounds

Theorem 4. *There exists a sequence of Dyck words W_n with*

$$\operatorname{width}(W_n) = \Omega(\sqrt{\log |W_n| / \log\log |W_n|}).$$

The words in this sequence are similar to the words $Z(n)$ associated with complete binary trees. They are associated with a 'stretched' version of the complete binary tree, i.e., one in which every edge is subdivided into several edges. More precisely, let $a_0, a_1, \ldots, a_k$ be a finite sequence of positive integers. Define the following sequence of Dyck words inductively:

$$X(a_0) = +^{a_0}-^{a_0},$$
$$X(a_0, \ldots, a_k) = +^{a_k} X(a_0, \ldots, a_{k-1}) X(a_0, \ldots, a_{k-1}) -^{a_k}.$$

The words we use to prove Theorem 4 have the form $Y(m, \ell) = X(a_0, \ldots, a_{m\ell-1})$, where $a_i = 2^{\lfloor i/\ell \rfloor}$, $m \geq 1$, and

$\ell \geq 1$. In particular, $Y(\ell) = Y(\lfloor \ell \cdot \log \ell \rfloor, \ell)$. (Notice that $Z(n) = X(1, \ldots, 1) = Y(1, n)$.) Our method applies both to $Y(\ell)$ and $Z(n)$, giving the bounds in equation (1) on page 3.

We give a proof overview below; details are provided in the full version of the paper. We use a **tree representation of re-pairings.** Informally, this tree tracks the sequence of mergers of erased intervals. This sequence, indeed, is naturally depicted as an ordered rooted (binary) tree as shown in Fig. 2(a).

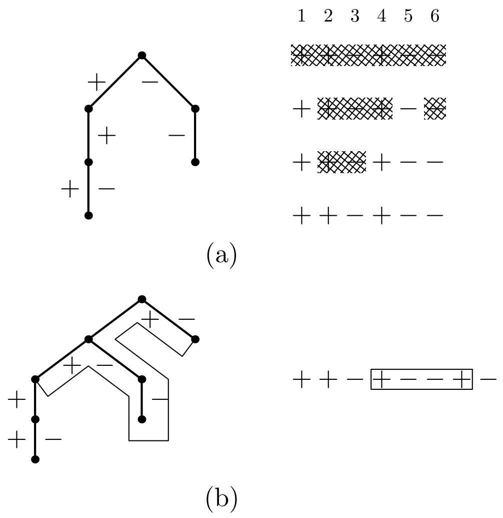

Figure 2. (a) Tree representation of the re-pairing (2, 3), (4, 6), (1, 5) for the word $Z(2)$; (b) for the word $Z(2)$+−, a fragment of the tree associated with the factor +−−+.

This tree is essentially a derivation tree for a word in an appropriate matrix grammar. Edges of a rooted tree are divided into levels according to the distance to the root. We think of this distance as a moment of *time* in the derivation process. The derived word can be read off the tree by following the left-to-right depth-first traversal. Formal definitions can be found in the full version of the paper.

Our proof of Theorem 4 is inductive, and one of the ideas is *what the induction should be over.* Observe that every factor w of a Dyck word W induces a connected subgraph, which we call a *fragment*; see Fig. 2(b). The width of a tree or a fragment is defined in natural way: it is the maximum number of edges at a level of the tree. For example, the fragment shown in Fig. 2(b) has width 2.

Our **inductive statement** applies to fragments. Fix a Dyck word W; in the sequel we specialize the argument to $Z(n)$ and $Y(\ell)$. Denote by $L(W, k)$ the maximum length of a factor w associated with a fragment of width at most k in trees that derive the word W. Put differently, given W, consider all possible trees that derive W. Fragments of width at most k in these trees are associated with factors of the word W, and $L(W, k)$ is the maximum length of such a factor. Note that in this definition the width of the (entire) trees is not restricted.

It is clear from the definition that the sequence of numbers $L(W, k)$ is non-decreasing: $L(W, 1) \leq L(W, 2) \leq \cdots \leq L(W, k)$. We obtain *upper* bounds on the numbers $L(W, k)$ by induction. For $Z(n)$, we show that $L(Z(n), k) \leq O(n) \cdot k^{O(k)} \cdot L(Z(n), k-1)^{O(k)}$ for big enough n and k. Here and below, implicit constants in the asymptotic notation do not depend on n and k. From this we get

$$L(Z(n), k) \leq 2^{k^{O(k)} \cdot \log n}.$$

We observe that if width$(W) \leq k$, then $|W| \leq L(W, k+1)$. Since $|Z(n)| = \Theta(2^n)$, it follows that every derivation tree of the word $Z(n)$ must have width k satisfying $n \leq k^{O(k)} \cdot \log n$, that is, width$(Z(n)) = \Omega(\log n / \log \log n)$. (Notice that this lower bound is doubly logarithmic in $|Z(n)|$, as stated in, e.g., equation (1).)

For $Y(\ell)$, we show a stronger inequality, $L(Y(\ell), k) \leq \mathrm{poly}(\ell, k) \cdot (ck)^\ell \cdot L(Y(\ell), k-1)$, which is sufficient for a lower bound of width$(Y(\ell)) = \Omega(\ell)$.

To prove the inductive upper bound on $L(W, k)$, we need to show that narrow fragments cannot be associated with long factors. For this purpose we use two ideas.

Combinatorial properties of increases and drops in $Z(n)$ and $Y(\ell)$. Denote by $\Delta(u)$ the difference $h(j) - h(i)$, where i and j are the start and end positions of the factor u. (Recall that $h(\cdot)$ denotes the height of a position in the word.) The value $\Delta(u)$ is the increase in height on the factor.

The first property is that every factor of $Z(n)$ of length x contains a sub-factor $-^d$ with $d \geq \log x - O(1)$. The second combinatorial property of $Z(n)$ is as follows: for sufficiently large x and every two factors u and $-^x$ of the word $Z(n)$, if $\Delta(u) \geq x$ and u is located to the left of $-^x$, then the distance between these factors is at least 2^x. Here and below the *distance* between the factors is the length of the smallest factor of W containing both of them.

For the word $Y(\ell)$, similar properties hold, but the functions $\log x$ and 2^x are replaced by the functions $\Omega(\ell \cdot (x/9)^{1/\ell})$ and $\Omega((x/2\ell)^\ell)$, respectively.

Balance within a single time period. Consider a factor w of the word W associated with a fragment of width at most k (in a tree derivation that generates W). Denote this fragment F. We will assume here that w is the maximal factor associated with F, i.e., all signs derived by F are included in w. (In the example shown in Fig. 2(b), the factor should be +−−+− not +−−+.)

Notice that, in a Dyck word, every − is matched by a + somewhere to the left of it. Thus, for a factor $-^d$, there exists a factor u to the left of $-^d$ with a matching height increase: $\Delta(u) \geq d$. We strengthen this balance observation and "relativize" it, ensuring that both factors appear inside w. More precisely, we identify a pair of matching factors $-^d$,

u (with a slightly smaller height increase in u) which also satisfies the following conditions:

(a) d is large enough (of magnitude indicated by the first combinatorial property, relative to w),
(b) the factors u and $-^d$ are derived during overlapping time intervals,
(c) the factor u sits to the left of $-^d$ and inside w, and
(d) the sub-fragment associated with the factor between u and $-^d$ has width strictly smaller than the width of the entire fragment F.

These conditions enable us to upper-bound the distance between u and $-^d$ through a function of $L(W, k-1)$. On the other hand, this distance is lower-bounded by the second combinatorial property. Comparing the bounds shows how to bound $|w|$, and thus $L(W, k)$, from above by a function of $L(W, k-1)$.

6 An application: Lower bounds for commutative NFA

In this section we link the re-pairing problem for Dyck words to the descriptional complexity (number of states in NFA) of the Parikh image of languages recognized by one-counter automata (OCA).

We will use the standard definition of nondeterministic finite automata (NFA). A formal definition of OCA (see, e.g., [7]) is not required for our arguments, and we explain the relevant intuition below.

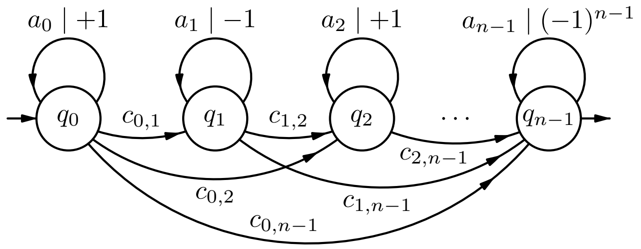

Figure 3. One-counter automaton $\mathcal{H}_n$.

We consider a slightly simplified version of *complete* languages introduced by Atig et al. [7] (see Section 1). Each of them is over the alphabet

$$\{c_{ij} : 0 \le i < j < n\} \cup \{a_i : 0 \le i < n\}$$

and can be recognized by an OCA $\mathcal{H}_n$ with n states in Fig. 3. We will assume throughout that n is even. In what follows, we need only the Parikh image H_n of $L(\mathcal{H}_n)$. We call H_n the *hard set* (for each n). This is the set of $n(n+1)/2$-dimensional vectors $(\boldsymbol{y}, \boldsymbol{x}) = (y_{ij} \colon 0 \le i < j < n;\ x_i \colon 0 \le i < n)$ of nonnegative integers that satisfy the following conditions:

(H1) $y_{ij} \in \{0, 1\}$ and the directed graph on vertices $[n] = \{0, 1, \ldots, n-1\}$ with the set of edges

$$\{(i, j) \text{ such that } y_{ij} = 1\}$$

consists of a monotone path from 0 to $n-1$ (i.e., one with $i < j$ in each edge) and possibly some isolated vertices; we call such paths *chains*;

(H2) (*balance*) the vector $\boldsymbol{x} = (x_i)$ belongs to *the cone K of balanced vectors:*

$$K = \Big\{(x_0, \ldots, x_{n-1}) \colon \sum_{i=0}^{n-1} (-1)^i x_i = 0; \\ \sum_{i=0}^{k} (-1)^i x_i \ge 0,\ 0 \le k < n-1\Big\};$$

(H3) (*compatibility*) if $x_j > 0$ for some $j > 0$, then $y_{ij} = 1$ for some i; if $x_j > 0$ for some $j < n-1$, then $y_{jk} = 1$ for some k.

Example 1. Let $n = 6$. Consider a chain with the edge set $\{(0, 1); (1, 4); (4, 5)\}$. We have $y_{01} = y_{14} = y_{45} = 1$, and the other y_{ij} are zero. For this setting of $\boldsymbol{y}$, the variables x_0, x_1, x_4, and x_5 may assume non-zero values, and $x_2 = x_3 = 0$ by the compatibility condition. The balance condition is then equivalent to

$$x_0 - x_1 + x_4 - x_5 = 0;\ x_0 - x_1 \ge 0. \quad \square$$

The meaning of the numbers x_i and y_{ij} is that they specify the number of occurrences of letters a_i and c_{ij} on accepting paths of $\mathcal{H}_n$ (see Fig. 3). Chains defined in condition (H1) are just paths in the transition diagram of $\mathcal{H}_n$ from the initial state to the final state. Recall that an OCA is an automaton equipped with a nonnegative integer counter, initialized with 0. A computation traverses the transition diagram of $\mathcal{H}_n$, incrementing and decrementing the counter according to the labels of the loops. Negative values are prohibited, i.e., decrement is not possible if the counter value is zero; whereas at the end of the computation, the counter value must be zero (condition (H2)). Clearly, to read a symbol a_i, the automaton should visit state q_i (condition (H3)).

An NFA *recognizes a language with Parikh image* H_n iff for each vector $(\boldsymbol{y}, \boldsymbol{x}) \in H_n$ the NFA has an accepting path with these counts of occurrences, and for all other vectors no such path exists. There exists [7] an NFA with $n^{O(\log n)}$ states that recognizes a language with Parikh image H_n. Our goal is to prove that this superpolynomial dependency on n is unavoidable.

Theorem 5. *Let σ_n be an arbitrary Dyck word of length $\le \sqrt{n/8}$. Suppose an NFA $\mathcal{A}_n$ recognizes a language with Parikh image H_n. Then the number of states of $\mathcal{A}_n$ is at least $n^{\Omega(\mathrm{width}(\sigma_n))}$.*

Corollary 2. *If an NFA $\mathcal{A}_n$ recognizes a language with Parikh image H_n, then its number of states is $n^{\Omega(\sqrt{\log n / \log\log n})}$.*

Corollary 2 follows from Theorems 4 and 5.

Since H_n is the Parikh image of a language recognized by an OCA with n states, it follows that there is no polynomial translation from OCA to Parikh-equivalent NFA.

Our proof of Theorem 5 makes three steps:

1. In the NFA for H_n we find accepting paths π_F, parameterized by sets F ($F \subseteq [n]$, $|F| = |\sigma_n|$), and extract re-pairings of σ_n from them. Roughly speaking, the parameter F determines the set of indices l for which the path π_F has $y_{kl} = 1$ or $y_{lk} = 1$ for some k; conceptually this is the set of states visited by the OCA $\mathcal{H}_n$.
Intuitively, as π_F goes through any strongly connected component (SCC) in the NFA, the re-pairing erases (some) pairs $(2i, 2j+1)$, where $\{2i, 2j+1\} \subseteq F$, such that a cycle in this SCC reads letters a_{2i} and a_{2j+1}. To find a bijection between even and odd indices (which is a necessary step in the construction of the re-pairing), we use the Birkhoff—von Neumann theorem on doubly stochastic matrices (see, e.g., [50, p. 301]).
2. With every SCC V in the NFA, we associate an auxiliary set $B(V) \subseteq [n]$. Intuitively, $B(V)$ corresponds to the information that the NFA must keep in its memory when it is in V. These sets $B(V)$ have the following property: $B(V) \subseteq F$ for every path π_F which visits V. We show that each path π_F must visit an SCC V_F for which $|B(V_F)| \geq \text{width}(\sigma_n)/3$.
3. By making F range in a family $\mathcal{F}$ of sets with low intersection, we ensure that no other path $\pi_{F'}$ can visit the SCC V_F. So the NFA has at least $|\mathcal{F}|$ SCCs, and therefore at least $|\mathcal{F}|$ states. The low intersection property means that $|F_1 \cap F_2| \leq d$ for all distinct $F_1, F_2 \in \mathcal{F}$. We choose $d = \text{width}(\sigma_n)/3 - 1$; the family $\mathcal{F}$ of size $n^{\Omega(d)}$ can be obtained by the Nisan—Wigderson construction [40].

We give more details in the following subsections.

6.1 Graph of strongly connected components

We first make several observations about the structure of the transition graph of an NFA $\mathcal{A} = \mathcal{A}_n$ which recognizes a language $L(\mathcal{A})$ with Parikh image H_n. Denote by $Q_0, Q_1, \ldots, Q_s$ the (strongly connected) components in the transition graph of the automaton $\mathcal{A}$ which are reachable from the initial state and from which a final state is reachable. Assume without loss of generality that $\mathcal{A}$ has no other strongly connected components. Suppose the initial state of $\mathcal{A}$ belongs to the component Q_0.

The first observation is that every edge labeled c_{ij} on a path from the initial state to a final state goes from one strongly connected component to another because $y_{ij} \leq 1$ by the condition (H1).

The second observation concerns directed cycles in the transition graph of the automaton $\mathcal{A}$. Let C be such a cycle. Denote by $x_i^{(C)}$ the number of edges in this cycle that are labeled by the symbol a_i. The balance condition (H2) implies the following statement.

Claim 6. *For the edges of every cycle C in the transition graph of the automaton $\mathcal{A}$, the vector $(x_i^{(C)} : 0 \leq i < n)$ belongs to the cone of balanced vectors K.*

We construct, based on the transition graph of $\mathcal{A}$, another graph $\mathcal{B}$—the condensation of $\mathcal{A}$, also known as the *graph of strongly connected components.* The vertices of this graph are strongly connected components in the transition graph of $\mathcal{A}$. The edges of $\mathcal{B}$ correspond to the edges of $\mathcal{A}$ between different components. An edge in $\mathcal{B}$ has label (i, j) if the corresponding transition in $\mathcal{A}$ has label $c_{i,j}$; and no label if the transition in $\mathcal{A}$ has label ε or a_i for some i. By construction, the graph $\mathcal{B}$ may have parallel edges and is acyclic.

Vertices of $\mathcal{B}$ that contain initial (resp. final) state(s) of the NFA $\mathcal{A}$ are called *initial* and *final*, respectively. A path from the initial vertex to a final vertex in $\mathcal{B}$ is *a complete path.*

Recall that chains are defined earlier in this section, on page 8, when we describe condition (H1).

Claim 7. *Let π be a complete path in the graph $\mathcal{B}$. Then the labels on (the edges of) this path induce a chain on the set $[n]$.*

By Claim 7, the edges of every complete path π in the graph $\mathcal{B}$ induce a chain. We will denote the set of elements of $[n]$ visited by this chain by $C(\pi)$. For every complete path, $\{0, n-1\} \subseteq C(\pi)$. Note that since every chain is essentially a monotone path, it is uniquely determined by the set of numbers from the set $[n]$ that it visits (i.e., the set of elements of $[n]$ that are incident to at least one edge on the chain).

6.2 Sets associated with strongly connected components

Consider, in the graph $\mathcal{B}$, a path from the initial vertex to a vertex v. Take all labels from the edges of this (incomplete) path. These labels form a subgraph of a chain and are therefore a disjoint union of non-intersecting paths: from number ℓ_0 to r_0; from ℓ_1 to r_1; $\ldots$; from ℓ_t to r_t.

Claim 8. *Let v be a vertex in the graph $\mathcal{B}$. Then for every path in $\mathcal{B}$ from the initial vertex to v the set of numbers*

$$\ell_0 < r_0 < \ell_1 < r_1 < \cdots < \ell_t < r_t \tag{5}$$

is the same.

Sketch of the proof. Observe that any path from the initial vertex to v and any path from v to a final vertex are completions of each other.

Assume for the sake of contradiction that for paths π and π' from the initial vertex to v the sequences

$$\ell_0 < r_0 < \ell_1 < r_1 < \cdots < \ell_t < r_t,$$
$$\ell'_0 < r'_0 < \ell'_1 < r'_1 < \cdots < \ell'_{t'} < r'_{t'}$$

are different. Consider the first difference and suppose it is the left endpoint of a segment. Without loss of generality, $\ell_i < \ell'_i$. Take some path π'' from v to a final vertex. The unions $\pi \cup \pi''$ and $\pi' \cup \pi''$ should induce chains on $[n]$; but a simple case analysis shows that this is impossible. □

The set (5) specifies a system of (closed) intervals inside $[n]$: $[\ell_0, r_0]$, $[\ell_1, r_1]$, …, $[\ell_t, r_t]$. Numbers in the interior of any of these intervals, i.e., inside the interval but except the two endpoints, are *internal* for a vertex v; and numbers outside the intervals are *external.*

Example 2. If the endpoints of the intervals are $\{1, 3, 6, 9\}$, then numbers $2, 7, 8$ are internal, $0, 4, 5$ are external, and the numbers $1, 3, 6, 9$ are neither.

A path from the initial vertex to v always visits all endpoints of the intervals, and may also visit some numbers internal for v. Similarly, a path from v to a final vertex always visits all endpoints of the intervals, and may also visit some numbers external for v.

For each vertex v, we now define four subsets of $[n]$:

- $S(v)$ is the set of all numbers $i \in [n]$ such that the corresponding component in $\mathcal{A}$ contains an edge with label a_i;
- $I(v)$ is the set (5) from Claim 8;
- $L(v)$ is the set of all internal numbers for v that are visited by all paths from the initial vertex to v;
- $R(v)$ is the set of all external numbers for v that are visited by all paths from v to final vertices.

Finally, we define

$$B(v) = S(v) \cup I(v) \cup L(v) \cup R(v).$$

Claim 9. *For every vertex v in the graph $\mathcal{B}$ and every complete path π that visits v, the inclusion $B(v) \subseteq C(\pi)$ holds.*

Sketch of the proof. We have $I(v) \subseteq C(\pi)$ by Claim 8, and $L(v) \subseteq C(\pi)$ and $R(v) \subseteq C(\pi)$ directly from the definitions. The inclusion $S(v) \subseteq C(\pi)$ requires a separate proof, which is based on standard cut-and-paste arguments. $\square$

Intuitively, Claim 9 links the information the NFA $\mathcal{A}$ must "remember" in each control state v to features of the paths that v belongs to.

It will be important in the sequel that the sets $L(v)$, $R(v)$, and $S(v)$ for vertices v on a complete path are related. This connection is characterized as follows.

Claim 10. *Let a vertex v_1 be visited by some complete path before another vertex v_2. (The vertices v_1 and v_2 may coincide.) Then the set $L(v_1)$ contains all numbers from $S(v_2)$ that are internal for v_1, and the set $R(v_2)$ contains all numbers from $S(v_1)$ that are external for v_2.*

Proof. Suppose v_1 occurs before v_2 on some complete path π; also suppose that the number i is internal for v_1 and belongs to $S(v_2)$. We show by contradiction that $i \in L(v_1)$. Assume that there is a path π' from the initial vertex to v_1 that has no edges with labels of the form (x, i) and (i, x). Denote by π'' the part of the path π from v_1 to the final vertex. Then $\pi' \cup \pi''$ is a complete path, the vertex v_2 is visited by this path, but $i \notin C(\pi' \cup \pi'')$, because i is internal for v_1. This contradicts Claim 9, and in particular the inclusion $S(v) \subseteq C(\pi)$.

The second assertion is proved in a similar way. $\square$

6.3 Strategy for the rest of the proof

In the following subsection, we describe a construction that produces paths π in the graph $\mathcal{B}$ with 'predetermined' $C(\pi)$. Our construction will have the following properties:

- Each path π is originally chosen based (in a certain way) on a Dyck word σ; we show how to obtain from π a re-pairing of σ (based on Claim 12 below).
- The width of this re-pairing will bound the cardinality of the set $B(u)$ from below, for *some* vertex $u = u(\pi)$ on the path π (Claim 13).
- Many (different) paths π will be chosen and it will be ensured that the sets $C(\pi)$ have low pairwise intersection (Subsection 6.5).

Now suppose that the vertex $u = u(\pi)$ for a path π is visited by *another* such path, say π'. Then the intersection of the sets $C(\pi')$ and $C(\pi)$ includes $B(u)$ by Claim 9. On the one hand, the cardinality $|B(u)|$ is greater than or equal to the width of the re-pairing (times a constant factor); on the other hand, no two sets $C(\pi')$ and $C(\pi)$ may overlap a lot. Therefore, with a careful choice of parameters, vertices $u = u(\pi)$ are not shared by the paths π (that is, each u can only belong to one π), and so the NFA $\mathcal{A}$ should have at least as many strongly connected components as we can choose paths π.

6.4 From NFA and Dyck word to re-pairing

Let σ be a Dyck word of length $s \le \frac{n}{2}$. As everywhere in this section, we assume that n is even. Based on the word σ, we identify a family of sets that will be useful throughout in the sequel. Conceptually, for each set F in this family, the OCA $\mathcal{H}_n$ visits exactly the states $F \subseteq [n] = \{0, 1, \ldots, n-1\}$ in some accepting computation.

Definition 1 (well-formed sets)**.** We call a set $F \subseteq [n]$ of size s *well-formed* if, for some auxiliary set $G \subseteq [\frac{n}{2}]$ of the same size with $\{0, \frac{n}{2} - 1\} \subseteq G$, the set F is determined by the following rule. Sort the elements of G in ascending order. Suppose the ith least element is equal to j; then F contains the number $2j$ if $\sigma(i) = +1$ and the number $2j + 1$ otherwise.

Note that, here and below, the dependence on σ is not reflected in the notation. Since σ, as a Dyck word, begins with a +1 and ends with a −1, we have $\{0, n-1\} \subseteq F$ for every well-formed F.

Remark 4. The correspondence between the elements of the set F and the symbols of the word σ defined in this way is a bijection. We will refer to the symbols in the word σ by specifying the corresponding numbers from the set F.

We now associate with the word σ a family of vectors from the hard set H_n as follows. These vectors will be determined by some well-formed set F and some nonnegative integer λ; we will denote them $(\boldsymbol{y}_F, \boldsymbol{x}_F(\lambda))$.

Definition 2 (vectors $(\boldsymbol{y}_F, \boldsymbol{x}_F(\lambda))$). Given a well-formed F and $\lambda \geq 0$, we set $y_{ij} = 1$ if i and j are two adjacent elements (in ascending order) in the set F, otherwise $y_{ij} = 0$; $x_i = \lambda$ if $i \in F$, otherwise $x_i = 0$.

Example 3. Let $\sigma = Z(2) = {+}{+}{-}{+}{-}{-}$; $s = |\sigma| = 6$ and $n = 16$; $s \leq \frac{n}{2} = 8$. The set $F = \{0, 2, 7, 10, 13, 15\}$ is well-formed, as witnessed by the auxiliary set $G = \{0, 1, 3, 5, 6, 7\} \subseteq [8]$. The vector $(\boldsymbol{y}_F, \boldsymbol{x}_F(\lambda))$ satisfies the equations $y_{0,2} = y_{2,7} = y_{7,10} = y_{10,13} = y_{13,15} = 1$ and $x_0 = x_2 = x_7 = x_{10} = x_{13} = x_{15} = \lambda$, and all other components of this vector are 0.

The vectors $(\boldsymbol{y}_F, \boldsymbol{x}_F(\lambda))$ defined in this way belong to the hard set H_n: conditions (H1) and (H3) hold by construction, and condition (H2) by the fact that the word σ is a Dyck word.

Let F be any well-formed set, $F \subseteq [n]$. We now find, for this F, a complete path π_F in the graph $\mathcal{B}$, as follows.

For every λ, the transition graph of the NFA $\mathcal{A}$ has at least one path $\tau(\lambda)$ from the initial state to a final state on which the labels form a word with Parikh image $(\boldsymbol{y}_F, \boldsymbol{x}_F(\lambda))$, i.e., each symbol c_{ij} occurs y_{ij} times and each a_i occurs x_i times.

Decompose the path $\tau(\lambda)$ into cycles and a simple path in a greedy way, by reading the sequence of states $q_0, q_1, \ldots, q_t$ of the automaton $\mathcal{A}$ specified by the path $\tau(\lambda)$, *from left to right*. Suppose we have constructed a partial decomposition

$$\tau(\lambda) = q_0\gamma_0(\lambda)q_1(\lambda)\gamma_1(\lambda)\ldots q_m(\lambda)q_{i+1}\ldots q_t,$$

where $q_m(\lambda) = q_i$. We find the last occurrence of q_i in the sequence $\tau(\lambda)$: $q_k = q_i$ and $q_j \neq q_i$ for all $j > k$. The subsequence $q_{i+1}, q_{i+2}, \ldots, q_k = q_i$ forms a cycle in the transition graph of the automaton $\mathcal{A}$; denote this cycle $\gamma_m(\lambda)$. After this, set $q_{m+1}(\lambda) = q_{k+1}$. Alternatively, if the state q_i never occurs further on the path $\tau(\lambda)$, we set $\gamma_m(\lambda)$ be the empty cycle and $q_{m+1}(\lambda) = q_{i+1}$. In this way we obtain a total decomposition

$$\tau(\lambda) = q_0\gamma_0(\lambda)q_1(\lambda)\gamma_1(\lambda)\ldots q_T(\lambda)\gamma_T(\lambda), \tag{6}$$

where all $q_m(\lambda)$ are distinct states of $\mathcal{A}$ which together form a simple path in the transition graph. In particular, T does not exceed $|Q(\mathcal{A})|$, the number of states of the automaton $\mathcal{A}$.

Since the set of possible values of the parameter λ is infinite, there exists a sequence $\tau_F^* = (q_0, q_1, \ldots, q_T)$ of distinct states of the NFA $\mathcal{A}$ that coincides with infinitely many sequences $(q_0, q_1(\lambda), \ldots, q_T(\lambda))$ that occur in the decomposition (6). (Note that here the set F is fixed and the parameter λ ranges over nonnegative integers.)

The sequence τ_F^* corresponds to a complete path in the graph $\mathcal{B}$ up to the choice of parallel edges. Choose these edges, and thus a complete path π_F, in such a way that π_F corresponds (in the graph $\mathcal{B}$) to infinitely many values of the parameter λ. Denote the vertices of this path

$$v_0^{(F)},\ v_1^{(F)},\ \ldots,\ v_T^{(F)}.$$

The following claim holds by definition of the vectors $\boldsymbol{y}_F$.

Claim 11. *$C(\pi_F) = F$ for every well-formed F.*

We next connect accepting paths in the automaton $\mathcal{A}$ and re-pairings of the word σ. We construct such a re-pairing based on the sets $S(v_i^{(F)})$, where the vertices $v_i^{(F)}$ are as above and the sets $S(\cdot)$ are defined in Subsection 6.2.

Claim 12. *Every well-formed set F can be decomposed as a union of disjoint sets $\{\ell_i, r_i\}_{i=1}^{s/2}$ such that for all i there is a k with $\{\ell_i, r_i\} \subseteq S(v_k^{(F)})$. For each pair (ℓ_i, r_i), the corresponding pair of signs in σ is formed by a plus and a minus, and the plus occurs to the left of the minus.*

Proof. For each label (i, j) on the edges of the path π_F, we have $\{i, j\} \subseteq F$ by Definition 2. By Claim 9, $S(v_k^{(F)}) \subseteq C(\pi_F)$, and $C(\pi_F) = F$ by Claim 11.

It is clear that each cycle $\gamma_m(\lambda)$ lies in some strongly connected component of the transition graph of $\mathcal{A}$. Note that the balance condition holds for every cycle in the transition graph of the automaton $\mathcal{A}$ (by Claim 6), and thus for each $\gamma_m(\lambda)$. Since the vector $\boldsymbol{x}_F(\lambda)$ belongs to the cone of balanced vectors as well, the balance condition also holds for the multiplicities of the labels a_i on the other edges of the path $\tau(\lambda)$. Moreover, the number of these (other) edges cannot exceed $|Q(\mathcal{A})|$, simply because the states in the sequence τ_F^* are all distinct.

Therefore, if we take the expansion of the vector $\boldsymbol{x}_F(\lambda)$ as a nonnegative linear combination of the generators of the cone K, this expansion can be split into two terms as follows:

$$\boldsymbol{x}_F(\lambda) = \sum_k \sum_{(i,j):\{i,j\}\subseteq S(v_k^{(F)})} u_{ij}^{(k)}(\lambda)\boldsymbol{e}_{i,j} + \sum_{(i,j)} v_{ij}(\lambda)\boldsymbol{e}_{i,j}. \tag{7}$$

Here the vectors

$$\boldsymbol{e}_{i,j} = (\underbrace{0,\ldots,0}_{i}, 1, \underbrace{0,\ldots,0}_{j-i-1}, -1, 0, \ldots, 0),\ i \text{ even},\ j \text{ odd},\ i < j \tag{8}$$

generate the cone of balanced vectors (the proof can be found, e.g., in the full version of the paper), the outer summation in the first term enumerates all vertices $v_k^{(F)}$ on the path, and the second term puts together the contribution of the edges not included in the cycles $\gamma_i(\lambda)$. We thus have $v_{ij}(\lambda) \leq |Q(\mathcal{A})|$.

Denote by Z_F^{even} the intersection of F with the set of even integers and by Z_F^{odd} the intersection of F with the set of odd integers. Notice that if $u_{ij}^{(k)}(\lambda) > 0$ or $v_{ij}(\lambda) > 0$, then $i \in Z_F^{\text{even}}$ and $j \in Z_F^{\text{odd}}$, and that $|Z_F^{\text{even}}| = |Z_F^{\text{odd}}| = s/2$. Also denote

$$u_{ij}(\lambda) = \sum_k u_{ij}^{(k)}(\lambda),$$

then the coefficients of the expansion (7) satisfy the equations

$$\sum_{i\in Z_F^{\text{even}}} u_{ij}(\lambda) + \sum_{i\in Z_F^{\text{even}}} v_{ij}(\lambda) = \lambda, \quad j \in Z_F^{\text{odd}},$$

$$\sum_{j\in Z_F^{\text{odd}}} u_{ij}(\lambda) + \sum_{j\in Z_F^{\text{odd}}} v_{ij}(\lambda) = \lambda, \quad i \in Z_F^{\text{even}},$$

by definition of the vector $\boldsymbol{x}_F(\lambda)$. (Each equation in this system corresponds to one coordinate of $\boldsymbol{x}_F(\lambda)$.)

Since there are infinitely many possible values of λ, and the coefficients $v_{ij}(\lambda)$ are upper-bounded by the number of states of the automaton $\mathscr{A}$, it follows that the matrices $(u_{ij}(\lambda)/\lambda)$ of dimension $(s/2) \times (s/2)$ (in which the rows are indexed with even numbers from F, and the columns by odd numbers from F) have a limit point, (u^*_{ij}), as $\lambda \to \infty$. (Recall that we picked the path through $\mathscr{B}$ in such a way that it corresponds to infinitely many values of λ.) We see from expansion (7) that if $u^*_{ij} > 0$, then the set $\{i, j\}$ is contained in some $S(v_k^{(F)})$. Moreover, we have

$$\sum_{i \in Z_F^{\text{even}}} u^*_{ij} = 1, \qquad \sum_{j \in Z_F^{\text{odd}}} u^*_{ij} = 1.$$

These conditions mean that (u^*_{ij}) is a doubly stochastic matrix of size $(s/2) \times (s/2)$. By the Birkhoff—von Neumann theorem (see, e.g., [50, p. 301]), it is a convex combination of permutation matrices. Take some permutation matrix that occurs in this convex combination with a positive coefficient.

This permutation matrix specifies a bijection $\alpha \colon Z_F^{\text{even}} \to Z_F^{\text{odd}}$ between even and odd indices from F. The bijection has the following properties. If $\alpha(2i) = 2j + 1$, then $2j + 1 > 2i$, since $u_{2i,2j+1}(\lambda) = 0$ for $i > j$ (see equation (8)). Furthermore, every pair $\{2i, \alpha(2i)\}$ is included in some set $S(v_k^{(F)})$ by expansion (7). Since $Z_F^{\text{even}} \cup Z_F^{\text{odd}} = F$, we obtain the equality

$$\bigcup_k S(v_k^{(F)}) = F$$

and the required partitioning of the set F into pairs $(2i, \alpha(2i))$. Now recall that even numbers in F correspond to pluses in the word σ and odd numbers to minuses. This correspondence is bijective by the construction of the set F. This completes the proof. □

Define a linear order on the pairs of signs identified by Claim 12. Roughly speaking, this will be the order in which the pairs $\{\ell_i, r_i\}$ occur along the path π_F. More precisely, fix for each i some specific index k_i such that $\{\ell_i, r_i\} \subseteq S(v_{k_i}^{(F)})$; it exists by Claim 12. If $k_i < k_j$ then the pair (ℓ_i, r_i) must appear in the linear order before the pair (ℓ_j, r_j). If $k_i = k_j$ then the pairs can be ordered arbitrarily. This linear order on the pairs gives us a re-pairing p_F of the word σ.

Claim 13. $\operatorname{width}(p_F) \le 3 \max_k \left|B(v_k^{(F)})\right|$.

Proof. In this argument, we identify the signs of the word σ with the corresponding numbers from the set F. Recall that $B(v) = S(v) \cup I(v) \cup L(v) \cup R(v)$. The idea behind the assertion is as follows. When the re-pairing has (just) erased all the pairs included in the sets $S(v_1^{(F)}), S(v_2^{(F)}), \ldots, S(v_t^{(F)})$, the set of all erased signs is the union of intervals whose endpoints are specified by the set $I(v_t^{(F)})$, plus perhaps the signs that correspond to $R(v_t^{(F)})$, but except the signs that correspond to $L(v_t^{(F)})$. (This property is based on Claim 10.) At all other points in time, the set of all erased signs is almost the same—except possibly for signs in one of the sets $S(v_k^{(F)})$. We will now make this precise and provide justification.

Consider, for some t, the first time point in the re-pairing when the pairs included in the sets $S(v_1^{(F)}), S(v_2^{(F)}), \ldots, S(v_t^{(F)})$ have all been erased. The set $I(v_t^{(F)})$ defines intervals

$$[\ell_0, r_0];\ [\ell_1, r_1];\ \ldots\ [\ell_{m-1}, r_{m-1}], \quad m = \frac{1}{2}\left|I(v_t^{(F)})\right|.$$

Suppose the number $i \in F$ belongs to one of these intervals and has not been erased yet. We then obtain from Claim 10 that $i \in L(v_t^{(F)}) \cup I(v_t^{(F)})$. Similarly, if a $j \in F$ does not belong to these intervals and has been erased already, then $j \in R(v_t^{(F)})$ by the same claim. Therefore, every sign in the word σ that has been erased by this time either is covered by one of the m intervals, or belongs to the set $R(v_t^{(F)})$; all the signs inside these m intervals have been erased, except maybe the elements of the set $L(v_t^{(F)})$. But this means that the set of all erased signs is a union of at most

$$\begin{aligned}\frac{1}{2}\left|I(v_t^{(F)})\right| + \left|L(v_t^{(F)})\right| + \left|R(v_t^{(F)})\right| &\le \\ \left|I(v_t^{(F)})\right| + \left|L(v_t^{(F)})\right| + \left|R(v_t^{(F)})\right| &= \\ \left|I(v_t^{(F)}) \cup L(v_t^{(F)}) \cup R(v_t^{(F)})\right|&\end{aligned}$$

intervals; so the width of the re-pairing at this time point does not exceed this quantity. (Whether the endpoints of the intervals are erased is irrelevant.)

Now consider an intermediate time point, when the pairs included in the sets $S(v_1^{(F)}), S(v_2^{(F)}), \ldots, S(v_{t-1}^{(F)})$ have all been erased, and so have *some* of the pairs included in the set $S(v_t^{(F)})$. The width of the re-pairing at this time point cannot differ by more than $2\left|S(v_t^{(F)})\right|$ from the same width at the time point when the pairs included in the sets $S(v_1^{(F)}), S(v_2^{(F)}), \ldots, S(v_t^{(F)})$ have all been erased, because, when a pair is erased, this can change (increase or decrease) the number of intervals by at most 2.

In summary, at no point in the re-pairing can its width exceed the quantity

$$\begin{aligned}\left|I(v_t^{(F)}) \cup L(v_t^{(F)}) \cup R(v_t^{(F)})\right| + 2\left|S(v_t^{(F)})\right| &\le \\ 3\left|I(v_t^{(F)}) \cup L(v_t^{(F)}) \cup R(v_t^{(F)}) \cup S(v_t^{(F)})\right|&,\end{aligned}$$

where t is chosen appropriately depending on the time point. In particular, taking the maximum over all time points, we obtain the desired inequality. □

6.5 Putting everything together: Proof of Theorem 5

We now obtain a lower bound on the number of states of the NFA $\mathscr{A}$ from the construction described in the previous subsections. We will pick in the set $[\frac{n}{2}]$ an appropriate big enough family $\mathcal{G}$ of subsets with low pairwise intersection: whenever $G_1, G_2 \in \mathcal{G}$ and $G_1 \neq G_2$, it should be the case that $|G_1 \cap G_2| \leq d$. The parameter d will be chosen later.

There exist such families of cardinality $n^{\Omega(d)}$. We will use a construction by Nisan and Wigderson [40], modified slightly.

We will assume n to be sufficiently large, because for small n the statement of the theorem holds trivially (since the constant in $\Omega(\cdot)$ can be chosen appropriately).

Pick an (odd) prime p in the interval between $\sqrt{n/8 - 1/2}$ and $\sqrt{n/2 - 2}$. For sufficiently large n, there are primes in this interval due to Bertrand's postulate (see, e.g., [3]). Also pick a subset D of size $|\sigma_n| - 2$ in the finite field $\mathbb{F}_p$, where σ_n is the Dyck word from the statement of the theorem. This is possible, because $|\sigma_n| - 2 \leq \sqrt{n/8} - 2 \leq \sqrt{n/8 - 1/2}$.

Embed $\mathbb{F}_p \times \mathbb{F}_p$ into $[\frac{n}{2}] \setminus \{0, \frac{n}{2} - 1\}$ arbitrarily. The members of the family $\mathcal{G}$ are obtained from the graphs of univariate polynomials of degree (strictly) less than $d - 1$ restricted to the subset D, by applying this embedding and adding the numbers 0 and $\frac{n}{2} - 1$. There are p^{d-1} such polynomials in total; if $d - 1 \leq p$, then different polynomials give rise to different subsets in the family $\mathcal{G}$, because the graphs of any two polynomials of degree $< d - 1$ can have an overlap of size $< d - 1$ only. This gives us a family $\mathcal{G}$ of $p^{d-1} = n^{\Omega(d)}$ subsets of $[\frac{n}{2}]$, each of size $|D| + 2 = |\sigma_n| = O(n^{1/2})$, with pairwise intersection of at most d.

Let now $\mathcal{F}$ be the family of all well-formed sets obtained according to Definition 1 from auxiliary sets $G \in \mathcal{G}$. We have $F \subseteq [n]$ and $|F| = |\sigma_n|$ for all $F \in \mathcal{F}$. Moreover, $|\mathcal{F}| = |\mathcal{G}|$ and $\mathcal{F}$ also has pairwise intersection of at most d.

Denote $w_n = \text{width}(\sigma_n)$. We have constructed in Subsection 6.4 a path π_F in the graph $\mathcal{B}$ for each well-formed set F, and in particular for each $F \in \mathcal{F}$. By Claim 13, this path satisfies the inequality

$$\max_k \left|B(v_k^{(F)})\right| \geq w_n/3.$$

Therefore, there is a vertex $u^{(F)} = v_k^{(F)}$ for which $|B(u^{(F)})|$ is at least $w_n/3$. By Claim 9, this set is included into $C(\pi_F) = F$ (see Claim 11). Pick

$$d = w_n/3 - 1,$$

then the vertex $u^{(F)}$ cannot be visited by any other path $\pi_{F'} \neq \pi_F$ due to the upper bound on the pairwise intersection of the subsets from $\mathcal{F}$. (If it is, then $B(u^{(F)}) \subseteq C(\pi_{F'}) = F'$ and $|F \cap F'| \geq w_n/3 = d + 1$. Therefore, $|F \cap F'| \geq d + 1$ and $F = F'$, which is a contradiction.) Note that $d - 1 \leq p$, as needed in the argument from the previous paragraph, because $p = \Omega(n^{1/2})$ and $d = O(\log \sqrt{n/8}) = O(\log n)$ by Theorem 1.

We have thus identified for each path π_F, $F \in \mathcal{F}$, a unique vertex of the graph $\mathcal{B}$ that is visited by no other path from this family. So the number of vertices in the graph $\mathcal{B}$, i.e., the number of strongly connected components in the transition graph of the NFA $\mathcal{A}$, cannot be less than the number of members of the family $\mathcal{F}$, that is,

$$p^{d-1} = n^{\Omega(w_n)} = n^{\Omega(\text{width}(\sigma_n))}.$$

The number of states of $\mathcal{A}$ cannot be less than that either, and this is exactly the assertion of Theorem 5.

7 Open problems

Our work suggests several directions for future research. The first is computing the width of $Z(n)$ as well as of other words, closing the gap between the upper and lower bounds. Obtaining super-constant lower bounds (for infinite families of words, both constructively and non-constructively) seems particularly difficult.

Our lower bound on the width of $Y(n)$ leaves a gap between $n^{\Omega(\sqrt{\log n / \log\log n})}$ and $n^{O(\log n)}$ for the size of blowup in an OCA to Parikh-equivalent NFA translation, and our second problem is to close this gap. We do not know if small-width re-pairings can be converted into small NFA; thus, stronger lower bounds on the NFA size (avoiding re-pairings) might be possible.

The third problem is to recover a proof of Rozoy's statement that the Dyck language D_1 is not generated by any matrix grammar of finite index [47], or equivalently by any two-way deterministic transducer with one-way output tape [45]. We expect that our lower bound construction for the width can be extended appropriately.

Last but not least, our re-pairing game corresponds to the following family of deterministic two-way transducers $\mathcal{T}_k$ generating Dyck words. The input to a transducer $\mathcal{T}_k$ encodes a derivation tree of width k, in the sense defined in Section 5. Symbols correspond to layers of the tree; there are $O(k^2)$ symbols in the alphabet that encode the branching and $O(k^2)$ symbols that encode the positions of a pair of brackets (+ and −). The transducer $\mathcal{T}_k$ simulates a traversal of the tree and outputs the generated word; it has $O(k)$ states. All words of width at most k are generated by $\mathcal{T}_k$.

Our final problem is to determine if there exist smaller transducers that generate all Dyck words of length n, $D_1 \cap \{+, -\}^n$, and do not generate any words outside D_1. Here n is such that all words of length n have width at most k.

Acknowledgments

We are grateful to Georg Zetzsche for the reference to Rozoy's paper [47].

This research has been supported by the Royal Society (IEC\R2\170123). The research of the second author has been funded by the Russian Academic Excellence Project '5-100'. Supported in part by RFBR grant 17-51-10005 and by the state assignment topic no. 0063-2016-0003.

References

[1] Parosh Aziz Abdulla, Mohamed Faouzi Atig, Roland Meyer, and Mehdi Seyed Salehi. 2015. What's Decidable about Availability Languages?. In *FSTTCS'15 (LIPIcs)*, Vol. 45. 192–205.

[2] Samuel Abraham. 1965. Some questions of phrase-structure grammars I. *Computational Linguistics* 4 (1965), 61–70.

[3] Martin Aigner and Günter M. Ziegler (Eds.). 2009. *Proofs from THE BOOK*. Springer.

[4] Rajeev Alur and Pavol Cerný. 2010. Expressiveness of streaming string transducers. In *FSTTCS'10*. 1–12.

[5] Rajeev Alur and Pavol Cerný. 2011. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *POPL'11*. 599–610.

[6] Rajeev Alur and Mukund Raghothaman. 2013. Decision Problems for Additive Regular Functions. In *ICALP'13 (Proceedings, Part II)*. 37–48.

[7] Mohamed Faouzi Atig, Dmitry Chistikov, Piotr Hofman, K. Narayan Kumar, Prakash Saivasan, and Georg Zetzsche. 2016. The complexity of regular abstractions of one-counter languages. In *LICS'16*. 207–216.

[8] Félix Baschenis, Olivier Gauwin, Anca Muscholl, and Gabriele Puppis. 2016. Minimizing Resources of Sweeping and Streaming String Transducers. In *ICALP'16*. 114:1–114:14.

[9] Barron Brainerd. 1967. An Analog of a Theorem about Context-Free Languages. *Information and Control* 11, 5/6 (1967), 561–567.

[10] Dmitry Chistikov and Mikhail Vyalyi. 2019. Re-pairing brackets. arXiv:1904.08402 [cs.FL]

[11] Stephen A. Cook and Ravi Sethi. 1976. Storage Requirements for Deterministic Polynomial Time Recognizable Languages. *J. Comput. Syst. Sci.* 13, 1 (1976), 25–37.

[12] Jürgen Dassow, Gheorghe Păun, and Arto Salomaa. 1997. Grammars with Controlled Derivations. In *Handbook of Formal Languages, Volume 2. Linear Modeling: Background and Application*. Springer, 101–154.

[13] Laure Daviaud, Pierre-Alain Reynier, and Jean-Marc Talbot. 2016. A Generalised Twinning Property for Minimisation of Cost Register Automata. In *LICS'16*. 857–866.

[14] Joost Engelfriet and Hendrik Jan Hoogeboom. 2001. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.* 2, 2 (2001), 216–254.

[15] Javier Esparza. 1997. Petri Nets, Commutative Context-Free Grammars, and Basic Parallel Processes. *Fundam. Inform.* 31, 1 (1997), 13–25.

[16] Javier Esparza, Pierre Ganty, Stefan Kiefer, and Michael Luttenberger. 2011. Parikh's theorem: A simple and direct automaton construction. *Inf. Process. Lett.* 111, 12 (2011), 614–619.

[17] Javier Esparza, Pierre Ganty, and Tomás Poch. 2014. Pattern-Based Verification for Multithreaded Programs. *ACM Trans. Program. Lang. Syst.* 36, 3 (2014), 9:1–9:29.

[18] Javier Esparza, Michael Luttenberger, and Maximilian Schlund. 2014. A Brief History of Strahler Numbers. In *LATA'14 (Lecture Notes in Computer Science)*, Vol. 8370. 1–13.

[19] Emmanuel Filiot and Pierre-Alain Reynier. 2016. Transducers, logic and algebra for functions of finite words. *SIGLOG News* 3, 3 (2016), 4–19.

[20] Pierre Ganty and Rupak Majumdar. 2012. Algorithmic Verification of Asynchronous Programs. *ACM Trans. Program. Lang. Syst.* 34, 1, Article 6 (May 2012), 48 pages.

[21] Seymour Ginsburg and Edwin H. Spanier. 1968. Derivation-Bounded Languages. *J. Comput. Syst. Sci.* 2, 3 (1968), 228–250.

[22] Hermann Gruber and Markus Holzer. 2006. Finding Lower Bounds for Nondeterministic State Complexity is Hard. *Electronic Colloquium on Computational Complexity (ECCC)* 13, 027 (2006). Conference version in: *Developments in Language Theory (DLT) 2006*; Lecture Notes in Computer Science, vol. 4036, pp. 363–374, Springer.

[23] Jozef Gruska. 1971. A Few Remarks on the Index of Context-Free Grammars and Languages. *Information and Control* 19, 3 (1971), 216–223.

[24] Christoph Haase and Piotr Hofman. 2016. Tightening the Complexity of Equivalence Problems for Commutative Grammars. In *STACS (LIPIcs)*, Vol. 47. 41:1–41:14.

[25] Matthew Hague and Anthony Widjaja Lin. 2012. Synchronisation- and Reversal-Bounded Analysis of Multithreaded Programs with Counters. In *CAV (Lecture Notes in Computer Science)*, Vol. 7358. Springer, 260–276.

[26] Jochen Hoenicke, Roland Meyer, and Ernst-Rüdiger Olderog. 2010. Kleene, Rabin, and Scott Are Available. In *CONCUR (Lecture Notes in Computer Science)*, Vol. 6269. Springer, 462–477.

[27] Markus Holzer and Martin Kutrib. 2008. Nondeterministic Finite Automata-Recent Results on the Descriptional and Computational Complexity. In *CIAA'08*. 1–16.

[28] Juraj Hromkovic. 1997. *Communication Complexity and Parallel Computing*. Springer.

[29] Juraj Hromkovic, Holger Petersen, and Georg Schnitger. 2009. On the limits of the communication complexity technique for proving lower bounds on the size of minimal NFA's. *Theor. Comput. Sci.* 410, 30-32 (2009), 2972–2981.

[30] Dung T. Huynh. 1983. Commutative Grammars: The Complexity of Uniform Word Problems. *Information and Control* 57, 1 (1983), 21–39.

[31] Dung T. Huynh. 1985. The Complexity of Equivalence Problems for Commutative Grammars. *Information and Control* 66, 1/2 (1985), 103–121.

[32] Eryk Kopczynski. 2015. Complexity of Problems of Commutative Grammars. *Logical Methods in Computer Science* 11, 1 (2015).

[33] Eryk Kopczynski and Anthony Widjaja To. 2010. Parikh Images of Grammars: Complexity and Applications. In *LICS*. 80–89.

[34] Michel Latteux. 1979. Substitutions dans le EDT0L Systèmes Ultralinéaires. *Information and Control* 42, 2 (1979), 194–260.

[35] Thomas Lengauer and Robert Endre Tarjan. 1980. The Space Complexity of Pebble Games on Trees. *Inf. Process. Lett.* 10, 4/5 (1980), 184–188.

[36] Michael Conrad Loui. 1979. *The space complexity of two pebble games on trees*. Technical memorandum TM-133. Laboratory for Computer Science, Massachusetts Institute of Technology (MIT).

[37] Friedhelm Meyer auf der Heide. 1981. A Comparison of two Variations of a Pebble Game on Graphs. *Theor. Comput. Sci.* 13 (1981), 315–322.

[38] Anca Muscholl. 2017. A Tour of Recent Results on Word Transducers. In *FCT'17*. 29–33.

[39] Anca Muscholl and Gabriele Puppis. 2019. The Many Facets of String Transducers (Invited Talk). In *STACS (LIPIcs)*, Vol. 126. 2:1–2:21.

[40] Noam Nisan and Avi Wigderson. 1994. Hardness vs Randomness. *J. Comput. Syst. Sci.* 49, 2 (1994), 149–167.

[41] Jakob Nordström. 2013. Pebble Games, Proof Complexity, and Time-Space Trade-offs. *Logical Methods in Computer Science* 9 (2013), 1–63. Issue 3.

[42] Jakob Nordström. 2015. New Wine into Old Wineskins: A Survey of Some Pebbling Classics with Supplemental Results. http://csc.kth.se/~jakobn/research/PebblingSurveyTMP.pdf

[43] Rohit J. Parikh. 1966. On Context-Free Languages. *J. ACM* 13, 4 (1966), 570–581.

[44] Vaclav Rajlich. 1972. Absolutely Parallel Grammars and Two-Way Finite State Transducers. *J. Comput. Syst. Sci.* 6, 4 (1972), 324–342.

[45] Brigitte Rozoy. 1985. About two-way transducers. In *FCT'85*. 371–379.

[46] Brigitte Rozoy. 1986. Outils et Résultats Pour Les Transducteurs Boustrophedons. *ITA* 20, 3 (1986), 221–249.

[47] Brigitte Rozoy. 1987. The Dyck Language $D_1'^*$ Is Not Generated by Any Matrix Grammar of Finite Index. *Inf. Comput.* 74, 1 (1987), 64–89.

[48] Arto Salomaa. 1969. On the Index of a Context-Free Grammar and Language. *Information and Control* 14, 5 (1969), 474–477.

[49] John E. Savage. 1998. *Models of Computation: Exploring the Power of Computing*. Addison-Wesley. http://cs.brown.edu/people/jsavage/book/

[50] Alexander Schrijver. 2003. *Combinatorial optimization*. Springer.

[51] Jeffrey Shallit. 2009. *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press.