Visibly Pushdown Languages over Sliding Windows

Moses Ganardi
Universität Siegen, Germany
ganardi@eti.uni-siegen.de

Abstract

We investigate the class of visibly pushdown languages in the sliding window model. A sliding window algorithm for a language L receives a stream of symbols and has to decide at each time step whether the suffix of length n belongs to L or not. The window size n is either a fixed number (in the fixed-size model) or can be controlled by an adversary in a limited way (in the variable-size model). The main result of this paper states that for every visibly pushdown language the space complexity in the variable-size sliding window model is either constant, logarithmic or linear in the window size. This extends previous results for regular languages.

2012 ACM Subject Classification Theory of computation → Streaming models

Keywords and phrases visibly pushdown languages, sliding windows, rational transductions

Digital Object Identifier 10.4230/LIPIcs.STACS.2019.27

Funding The author is supported by the DFG project LO 748/13-1.

1 Introduction

The sliding window model. A *sliding window algorithm (SWA)* is an algorithm which processes a stream of data elements $a_1a_2a_3\cdots$ and computes at each time instant t a certain value that depends on the suffix $a_{t-n+1}\cdots a_t$ of length n where n is a parameter called the *window size*. This streaming model is motivated by the fact that in many applications data elements are outdated or become irrelevant after a certain time. A general goal in the area of sliding window algorithms is to avoid storing the window content explicitly (which requires $\Omega(n)$ bits) and to design space efficient algorithms, say using polylogarithmic many bits in the window size n.

A prototypical example of a problem considered in the sliding window model is the Basic Counting problem. Here the input is a stream of bits and the task is to approximate the number of 1's in the last n bits (the *active window*). In [15], Datar, Gionis, Indyk and Motwani present an approximation algorithm using $O(\frac{1}{\epsilon}\log^2 n)$ bits of space with an approximation ratio of ϵ. They also prove a matching lower bound of $\Omega(\frac{1}{\epsilon}\log^2 n)$ bits for any deterministic (and even randomized) algorithm for Basic Counting. Other works in the sliding window model include computing statistics [2, 3, 8], optimal sampling [9] and various pattern matching problems [10, 12, 13, 14].

There are two variants of the sliding window model, cf. [2]. One can think of an adversary who can either insert a new element into the window or remove the oldest element from the window. In the *fixed-size* sliding window model the adversary determines the window size n in the beginning and the initial window is set to a^n for some default known element a. At every time step the adversary inserts a new symbol and then immediately removes the oldest element from the window. In the *variable-size* sliding window model the window size is initially set to $n = 0$. Then the adversary is allowed to perform an arbitrary sequence of insert- and remove-operations. A remove-operation on an empty window leaves the window empty. We also mention the timestamp-based model where every element carries a timestamp (many elements may have the same timestamp) and the active window at time t contains

36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019).
Editors: Rolf Niedermeier and Christophe Paul; Article No. 27; pp. 27:1–27:23
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

only those elements whose timestamp is at least $t - t_0$ for some parameter t_0 [9]. Both the fixed-size and the timestamp-based model can be simulated in the variable-size model.

Regular languages. In a recent series of works we studied the membership problem to a fixed regular language in the sliding window model. It was shown in [20] that in both the fixed-size and the variable-size sliding window model the space complexity of any regular language is either constant, logarithmic or linear (a *space trichotomy*). In a subsequent paper [18] a characterization of the space classes was given: A regular language has a fixed/variable-size SWA with $O(\log n)$ bits if and only if it is a finite Boolean combination of regular left ideals and regular length languages. A regular language has a fixed-size SWA with $O(1)$ bits if and only if it is a finite Boolean combination of suffix testable languages and regular length languages. A regular language has a variable-size SWA with $O(1)$ bits if and only if it is empty or universal.

Context-free languages. A natural question is whether the results above can be extended to larger language classes, say subclasses of the context-free languages. More precisely, we pose the questions: (i) Which language classes have a "simple" hierarchy of space complexity classes (like the space trichotomy for the regular languages), and (ii) are there natural descriptions of the space classes? A positive answer to question (i) seems to be necessary to answer question (ii) positively. In [21] we presented a family of context-free languages $(L_k)_{k\geq 1}$ which have space complexity $\Theta(n^{1/k})$ in the variable-size model and $O(n^{1/k}) \setminus o(n^{1/k})$ in the fixed-size model, showing that there exists an infinite hierarchy of space complexity classes inside the class of context-free languages. Intuitively, this result can be explained with the fact that a language and its complement have the same sliding window space complexity; however, the class of context-free languages is not closed under complementation (in contrast to the regular languages) and the analysis of co-context-free languages in this setting seems to be very difficult. Even in the class of deterministic context-free languages, which is closed under complementation, there are example languages which have sliding window space complexity $\Theta((\log n)^2)$ [21].

Visibly pushdown languages. Motivated by these observations in this paper we will study the class of *visibly pushdown languages*, introduced by Alur and Madhusudan [1]. They are recognized by *visibly pushdown automata* where the alphabet is partitioned into *call letters*, *return letters* and *internal letters*, which determine the behavior of the stack height. Since visibly pushdown automata can be determinized, the class of visibly pushdown languages turns out to be very robust (it is closed under Boolean operations and other language operations) and to be more tractable in many algorithmic questions than the class of context-free languages [1]. In this paper we prove a space trichotomy for the class of visibly pushdown languages in the variable-size sliding window model, stating that the space complexity of every visibly pushdown language is either $O(1)$, $\Theta(\log n)$ or $O(n) \setminus o(n)$. The main technical result is a growth theorem (Theorem 6) for rational transductions. A natural characterization of the $O(\log n)$-class as well as a study of the fixed-size model are left as open problems.

Let us mention some related work in the context of streaming algorithms for context-free languages. Randomized streaming algorithms were studied for subclasses of context-free languages (DLIN and LL(k)) [4] and for Dyck languages [25]. A streaming property tester for visibly pushdown languages was presented by François et al. [17].

2 Preliminaries

We define $\log n = \lfloor \log_2 n \rfloor$ for all $n \geq 1$, which is the minimum number k of bits required to encode n elements using bit strings of length *at most* k. If $w = a_1 \cdots a_n$ is a word then any word of the form $a_i \cdots a_n$ $(a_1 \cdots a_i)$ is called *suffix (prefix)* of w. A prefix (suffix) v of w is *proper* if $v \neq w$. A *factor* of w is any word of the form $a_i \cdots a_j$. A *factorization* of w is formally a sequence of possibly empty factors $(w_0, \dots, w_m)$ with $w = w_0 \cdots w_m$. We call w_0 the *initial* factor and $w_1, \dots, w_m$ the *internal* factors. The *reversal* of w is $w^{\mathsf{R}} = a_n a_{n-1} \cdots a_1$. For a language $L \subseteq \Sigma^*$ we denote by $\mathsf{Suf}(L)$ the set of suffixes of words in L. If $L = \mathsf{Suf}(L)$ then L is *suffix-closed*.

Automata. An *automaton* over a monoid M is a tuple $A = (Q, M, I, \Delta, F)$ where Q is a finite set of *states*, $I \subseteq Q$ is a set of *initial states*, $\Delta \subseteq Q \times M \times Q$ is the *transition relation* and $F \subseteq Q$ is the set of *final states*. A *run* on $m \in M$ from q_0 to q_n is a sequence of transitions of the form $\pi = (q_0, m_1, q_1)(q_1, m_2, q_2) \cdots (q_{n-1}, m_n, q_n) \in \Delta^*$ such that $m = m_1 \cdots m_n$. We usually depict π as $q_0 \xrightarrow{m_1} q_1 \xrightarrow{m_2} q_2 \cdots q_{n-1} \xrightarrow{m_n} q_n$, or simply $q_0 \xrightarrow{m} q_n$. It is *initial* if $q_0 \in I$ and *accepting* if $q_n \in F$. The *language* defined by A is the set $\mathsf{L}(A)$ of all elements $m \in M$ such that there exists an initial accepting run on m. A subset $L \subseteq M$ is *rational* if $L = \mathsf{L}(A)$ for some automaton A. We only need the case where M is the free monoid Σ^* over an alphabet Σ or where M is the product $\Sigma^* \times \Omega^*$ of two free monoids. In these cases we change the format and write $(Q, \Sigma, I, \Delta, F)$ and $(Q, \Sigma, \Omega, I, \Delta, F)$, respectively. Subsets of Σ^* are called *languages* and subsets of $\Sigma^* \times \Omega^*$ are called *transductions*. Rational languages are usually called *regular languages*.

In this paper we will also use *right automata*, which read the input from right to left. Formally, a right automaton $A = (Q, M, F, \Delta, I)$ has the same format as a (left) automaton where the sets of initial and final states are swapped. Runs in right automata are defined from right to left, i.e. a run on $m \in M$ from q_n to q_0 is a sequence of transitions of the form $(q_0, m_1, q_1)(q_1, m_2, q_2) \cdots (q_{n-1}, m_n, q_n) \in \Delta^*$ such that $m = m_1 \cdots m_n$. In the graphic notation we write the arrows from right to left. It is initial (accepting) if $q_n \in I$ $(q_0 \in F)$.

Right congruences. For any equivalence relation $\sim$ on a set X we write $[x]_\sim$ for the $\sim$-class containing $x \in X$ and $X/{\sim} = \{[x]_\sim \mid x \in X\}$ for the set of all $\sim$-classes. The *index* of $\sim$ is the cardinality of $X/{\sim}$. We denote by $\nu_\sim \colon X \to X/{\sim}$ the function with $\nu_\sim(x) = [x]_\sim$. A subset $L \subseteq X$ is *saturated* by $\sim$ if L is a union of $\sim$-classes. An equivalence relation $\sim$ on the free monoid Σ^* over some alphabet Σ is a *right congruence* if $x \sim y$ implies $xz \sim yz$ for all $x, y, z \in \Sigma^*$. The *Myhill-Nerode right congruence* $\sim_L$ of a language $L \subseteq \Sigma^*$ is the equivalence relation on Σ^* defined by $x \sim_L y$ if and only if $x^{-1}L = y^{-1}L$ where $x^{-1}L = \{z \mid xz \in L\}$. It is indeed the coarsest right congruence on Σ^* which saturates L. We usually write ν_L instead of $\nu_{\sim_L}$. A language $L \subseteq \Sigma^*$ is regular iff $\sim_L$ has finite index.

Rational transductions. Rational transductions are accepted by automata over $\Sigma^* \times \Omega^*$, which are called finite state transducers. In this paper, we will use a slightly extended but equivalent definition. A *transducer* is a tuple $A = (Q, \Sigma, \Omega, I, \Delta, F, o)$ such that $(Q, \Sigma^* \times \Omega^*, I, \Delta, F)$ is an automaton over $\Sigma^* \times \Omega^*$ and a *terminal output function* $o \colon F \to \Omega^*$. To omit parentheses we write runs $p \xrightarrow{(x,y)} q$ in the form $p \xrightarrow{x|y} q$ and depict $o(q) = y$ by a transition $q \xrightarrow{|y}$ without input word and target state. If π is a run $p \xrightarrow{x|y} q$ we define $\mathsf{out}(\pi) = y$ and $\mathsf{out}_F(\pi) = y\, o(q)$. The transduction defined by A is the set $\mathsf{T}(A)$ of all pairs $(x, \mathsf{out}_F(\pi))$ such that π is an initial accepting run $p \xrightarrow{x|y} q$. Since the terminal output function can be eliminated by ε-transitions, a transduction is rational if and only if it is of the form $\mathsf{T}(A)$ for some transducer A. In this paper we will mainly use *rational functions*, which are partial functions $t\colon \Sigma^* \to \Omega^*$ whose graph $\{(x, t(x)) \mid x \in \mathrm{dom}(t)\}$ is a rational transduction.

A transducer A is *trim* if every state occurs on some accepting run. If every word $x \in \Sigma^*$ has at most one initial accepting run $p \xrightarrow{x|y} q$ for some $y \in \Omega^*$ then A is *unambiguous*. If $\Delta \subseteq Q \times \Sigma \times \Omega^* \times Q$ then A is *real-time*. It is known that every rational function is defined by a trim unambiguous real-time transducer [6, Corollary 4.3]. If A is unambiguous and trim then for every word $x \in \Sigma^*$ and every pair of states $(p, q) \in Q^2$ there exists at most one run from p to q with input word x. Therefore, the state pair (p, q) and the input word x uniquely determine the run (if it exists) and we can simply write $p \xrightarrow{x} q$. Similarly to [28], we define for a real-time transducer A the parameter $\mathsf{iml}(A) = \max(\{|y| \mid (q, a, y, p) \in \Delta\} \cup \{|o(q)| \mid q \in Q\})$. For every run π on a word $x \in \Sigma^*$ we have $|\mathsf{out}(\pi)| \le \mathsf{iml}(A) \cdot |x|$ and $|\mathsf{out}_F(\pi)| \le \mathsf{iml}(A) \cdot (|x|+1)$.

The following closure properties for rational transductions are known [6]: The class of rational transductions is closed under inverse, reversal and composition where the *inverse* of T is $T^{-1} = \{(y, x) \mid (x, y) \in T\}$, the *reversal* of T is $T^{\mathsf{R}} = \{(x^{\mathsf{R}}, y^{\mathsf{R}}) \mid (x, y) \in T\}$, and the composition of two transductions T_1, T_2 is $T_1 \circ T_2 = \{(x, z) \mid \exists y : (x, y) \in T_1 \text{ and } (y, z) \in T_2\}$. If $T \subseteq \Sigma^* \times \Omega^*$ is rational and $L \subseteq \Sigma^*$ is regular then the restriction $\{(x, y) \in T \mid x \in L\}$ is also rational. If $K \subseteq \Sigma^*$ is regular (context-free) and $T \subseteq \Sigma^* \times \Omega^*$ is rational then $TK = \{y \in \Omega^* \mid (x, y) \in T \text{ for some } x \in K\}$ is also regular (context-free).

A *right transducer* is a tuple $A = (Q, \Sigma, \Omega, F, \Delta, I, o)$ such that $(Q, \Sigma^* \times \Omega^*, F, \Delta, I)$ is a right automaton over $\Sigma^* \times \Omega^*$ and a *terminal output function* $o\colon F \to \Omega^*$. We depict $o(q) = y$ by a transition $\xleftarrow{|y} q$. If π is a run $q \xleftarrow{x|y} p$ we define $\mathsf{out}(\pi) = y$ and $\mathsf{out}_F(\pi) = o(q)\, y$. All other notions on transducers are defined for right transducers in a dual way.

Growth functions. A function $\gamma\colon \mathbb{N} \to \mathbb{N}$ grows *polynomially* if $\gamma(n) \in O(n^k)$ for some $k \in \mathbb{N}$; we say that γ grows *exponentially* if there exists a number $c > 1$ such that $\gamma(n) \ge c^n$ for infinitely many $n \in \mathbb{N}$. A function $\gamma(n)$ grows exponentially if and only if $\log \gamma(n) \notin o(n)$.

We will define a generalized notion of growth. Let $t\colon \Sigma^* \to Y$ be a partial function and let $X \subseteq \mathrm{dom}(t)$ be a language. The *t-growth* of X is the function $\gamma(n) = |t(X \cap \Sigma^{\le n})|$, i.e. it counts the number of output elements on input words from X of length at most n. The *growth* of X is simply the id_X-growth of X, i.e. $\gamma(n) = |X \cap \Sigma^{\le n}|$. It is known that every context-free language has either polynomial or exponential growth [22]. Furthermore, a context-free language L has polynomial growth if and only if it is *bounded*, i.e. $L \subseteq w_1^* \cdots w_k^*$ for some words $w_1, \dots, w_k$ [22]. We need the fact that if L is a bounded language and K is a set of factors of words in L then K is bounded [23, Lemma 1.1(c)].

3 Visibly pushdown languages

A *pushdown alphabet* is a triple $\tilde{\Sigma} = (\Sigma_c, \Sigma_r, \Sigma_{int})$ consisting of three pairwise disjoint alphabets: a set of *call letters* Σ_c, a set of *return letters* Σ_r and a set of *internal letters* Σ_{int}. We identify $\tilde{\Sigma}$ with the union $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_{int}$. The set of *well-matched* words W over Σ is defined as the smallest set which contains $\{\varepsilon\} \cup \Sigma_{int}$, is closed under concatenation, and if w is well-matched, $a \in \Sigma_c$, $b \in \Sigma_r$ then also awb is well-matched. A word is called *descending (ascending)* if it can be factorized into well-matched factors and return (call) letters. The set of descending words is denoted by D. A *visibly pushdown automaton (VPA)* has the form $A = (Q, \tilde{\Sigma}, \Gamma, \bot, q_0, \delta, F)$ where Q is a finite state set, $\tilde{\Sigma}$ is a pushdown alphabet, Γ is the finite stack alphabet containing a special symbol $\bot$ (representing the empty stack), $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and $\delta = \delta_c \cup \delta_r \cup \delta_{int}$ is the transition function where $\delta_c\colon Q \times \Sigma_c \to (\Gamma \setminus \{\bot\}) \times Q$, $\delta_r\colon Q \times \Sigma_r \times \Gamma \to Q$ and $\delta_{int}\colon Q \times \Sigma_{int} \to Q$. The set of *configurations* Conf is the set of all words αq where $q \in Q$ is a state and $\alpha \in \bot(\Gamma \setminus \{\bot\})^*$ is the *stack content*. We define $\delta\colon \mathrm{Conf} \times \Sigma \to \mathrm{Conf}$ for each $p \in Q$ and $a \in \Sigma$ as follows:

- If $a \in \Sigma_c$ and $\delta(p, a) = (\gamma, q)$ then $\delta(\alpha p, a) = \alpha\gamma q$.
- If $a \in \Sigma_{int}$ and $\delta(p, a) = q$ then $\delta(\alpha p, a) = \alpha q$.
- If $a \in \Sigma_r$, $\delta(p, a, \gamma) = q$ and $\gamma \in \Gamma \setminus \{\bot\}$ then $\delta(\alpha\gamma p, a) = \alpha q$.
- If $a \in \Sigma_r$ and $\delta(p, a, \bot) = q$ then $\delta(\bot p) = \bot q$.

As usual we inductively extend δ to a function $\delta\colon \mathrm{Conf} \times \Sigma^* \to \mathrm{Conf}$ where $\delta(c, \varepsilon) = c$ and $\delta(c, wa) = \delta(\delta(c, w), a)$ for all $w \in \Sigma^*$ and $a \in \Sigma$. The *initial* configuration is $\bot q_0$ and a configuration c is *final* if $c \in \Gamma^* F$. A word $w \in \Sigma^*$ is *accepted* from a configuration c if $\delta(c, w)$ is final. The VPA A *accepts* w if w is accepted from the initial configuration. The set of all words accepted by A is denoted by $\mathsf{L}(A)$; the set of all words accepted from c is denoted by $\mathsf{L}(c)$. A language L is a *visibly pushdown language (VPL)* if $L = \mathsf{L}(A)$ for some VPA A. To exclude some pathological cases we assume that $\Sigma_c \neq \emptyset$ and $\Sigma_r \neq \emptyset$. In fact, if $\Sigma_c = \emptyset$ or $\Sigma_r = \emptyset$ then any VPL over that pushdown alphabet would be regular.

One can also define nondeterministic visibly pushdown automata in the usual way, which can always be converted into deterministic ones [1]. This leads to good closure properties of the class of all VPLs, as closure under Boolean operations, concatenation and Kleene star.

The set W of well-matched words forms a submonoid of Σ^*. Notice that a VPA can only see the top of the stack when reading return symbols. Therefore, the behavior of a VPA on a well-matched word is determined only by the current state and independent of the current stack content. More precisely, there exists a monoid homomorphism $\varphi\colon W \to Q^Q$ into the finite monoid of all state transformations $Q \to Q$ such that $\delta(\alpha p, w) = \alpha\varphi(w)(p)$ for all $w \in W$ and $\alpha p \in \mathrm{Conf}$.

4 Sliding window algorithms and main results

In our context a *streaming algorithm* is a deterministic algorithm A which reads an input word $a_1 \cdots a_m \in \Sigma^*$ symbol by symbol from left to right and outputs after every prefix either 1 or 0. We view A as a deterministic (possibly infinite) automaton whose states are encoded by bit strings and thus abstract away from the actual computation, see [18] for a formal definition. A *variable-size sliding window algorithm* for a language $L \subseteq \Sigma^*$ is a streaming algorithm A which reads an input word $a_1 \cdots a_m$ over the extended alphabet $\overline{\Sigma} = \Sigma \cup \{\downarrow\}$. The symbol $\downarrow$ is the operation which removes the oldest symbol from the window. At time $0 \le t \le m$ the algorithm has to decide whether the *active window* $\mathrm{wnd}(a_1 \cdots a_t)$ belongs to L which is defined by

$$\begin{aligned} \mathrm{wnd}(\varepsilon) &= \varepsilon & \mathrm{wnd}(u\!\downarrow) &= \varepsilon \text{ if } \mathrm{wnd}(u) = \varepsilon \\ \mathrm{wnd}(ua) &= \mathrm{wnd}(u)a & \mathrm{wnd}(u\!\downarrow) &= v \text{ if } \mathrm{wnd}(u) = av \end{aligned}$$

for $u \in \Sigma^*$, $a \in \Sigma$. For example, a variable-size sliding window algorithm A for the language $L_a = \{w \in \{a, b\}^* \mid w \text{ contains } a\}$ maintains the window length n and the position i (from the right) of the most recent a-symbol in the window (if it exists): We initialize $n := 0$ and $i := \infty$. On input a we increment n and set $i := 1$, on input b we increment both n and i. On input $\downarrow$ we decrement n, unless $n = 0$, and then set $i := \infty$ if $i > n$.

The *space complexity* of A is the function which maps n to the maximum number of bits used when reading an input $a_1 \cdots a_m$ where the window size never exceeds n, i.e. $|\mathrm{wnd}(a_1 \cdots a_t)| \le n$ for all $0 \le t \le n$. Notice that this function is monotonic. For every language L there exists a space optimal variable-size sliding window algorithm [19, Lemma 3.1] and we write $V_L(n)$ for its space complexity. Clearly we have $V_L(n) \in O(n)$. For example the example language L_a above satisfies $V_{L_a}(n) \in O(\log n)$ because the algorithm above only maintains two numbers using $O(\log n)$ bits. The main result of this paper states:

▶ **Theorem 1** (Trichotomy for VPL). *If L is a visibly pushdown language then $V_L(n)$ is either $O(1)$, $\Theta(\log n)$ or $O(n) \setminus o(n)$.*

In the rest of this section we will give an overview of the proof of Theorem 1.

Suffix expansions. Let $\sim$ be an equivalence relation on Σ^*. The *suffix expansion* of $\sim$ is the equivalence relation $\approx$ on Σ^* defined by $a_1 \cdots a_n \approx b_1 \cdots b_m$ if and only if $n = m$ and $a_i \cdots a_n \sim b_i \cdots b_n$ for all $1 \leq i \leq n$. Notice that $\approx$ saturates each subset $\Sigma^{\leq n}$. Furthermore, if $\sim$ is a right congruence then so is $\approx$ since $|u| = |v|$ implies $|ua| = |va|$ and $a_i \cdots a_n \sim b_i \cdots b_n$ implies $a_i \cdots a_n a \sim b_i \cdots b_n a$. We also define suffix expansions for partial functions $t \colon \Sigma^* \to Y$ with suffix-closed domain $\mathrm{dom}(t)$. The *suffix expansion* of t is the total function $\overleftarrow{t} \colon \mathrm{dom}(t) \to Y^*$ defined by $\overleftarrow{t}(a_1 \cdots a_n) = t(a_1 \cdots a_n)\, t(a_2 \cdots a_n) \cdots t(a_{n-1}a_n)\, t(a_n)$ for all $a_1 \cdots a_n \in \Sigma^*$. Here the range of $\overleftarrow{t}$ is the free monoid (alternatively, the set of all sequences) over Y. If $\sim$ is an equivalence relation on Σ^* then its suffix expansion $\approx$ is the *kernel* of $\overleftarrow{\nu}_\sim$, i.e. $x \approx y$ if and only if $\overleftarrow{\nu}_\sim(x) = \overleftarrow{\nu}_\sim(y)$. The space complexity in the variable-size model is captured by the suffix expansion $\approx_L$ of the Myhill-Nerode right congruence $\sim_L$ or alternatively by the suffix expansion $\overleftarrow{\nu}_L$ of ν_L.

▶ **Theorem 2** ([18, Theorem 4.1]). *For all $\emptyset \subsetneq L \subsetneq \Sigma^*$ we have $V_L(n) = \log |\Sigma^{\leq n}/{\approx_L}| = \log |\overleftarrow{\nu}_L(\Sigma^{\leq n})|$. In particular, $V_L(n) = \Omega(\log n)$ for every non-trivial language.*

If L is empty or universal, then $V_L(n) \in O(1)$ and otherwise $V_L(n) = \Omega(\log n)$. Hence to prove Theorem 1 it suffices to show that either $V_L(n) \in O(\log n)$ or $V_L(n) \notin o(n)$ holds for every VPL L. If L is a regular language and A is the minimal DFA of L with state set Q, one can identify $\nu_L(x)$ with the state $q \in Q$ reached on input x. Hence, $\overleftarrow{\nu}_L(x)$ is represented by a word over Q. Using the transition monoid of A one can show that $\overleftarrow{\nu}_L \colon \Sigma^* \to Q^*$ is rational (in fact *right-subsequential*, see Section 6) and hence the image $\overleftarrow{\nu}_L(\Sigma^*) \subseteq Q^*$ is regular [19, Lemma 4.2]. Since the growth of $\overleftarrow{\nu}_L(\Sigma^*)$ is either polynomial or exponential this implies that $V_L(n) \in O(\log n)$ or $V_L(n) \notin o(n)$.

Restriction to descending words. The approach above for regular languages can be extended to visibly pushdown languages L if we restrict ourselves to the set D of descending words. If a VPA with state set Q reads a descending word $x \in D$ from the initial configuration it reaches some configuration $\bot q$ with empty stack. Notice that there may be distinct configurations $\bot p \neq \bot q$ with $\mathsf{L}(\bot p) = \mathsf{L}(\bot q)$, in which case we need to pick a single representative. Since every suffix of x is again descending we can represent $\overleftarrow{\nu}_L(x)$ by a word $\sigma_0(x) \in Q^*$ and in fact we will prove that $S_0 = \sigma_0(D)$ is a context-free language (Lemma 10). By the growth theorem for context-free languages the growth of S_0 is either polynomial or exponential. If S_0 grows exponentially we obtain an exponential lower bound on $|\overleftarrow{\nu}_L(\Sigma^{\leq n})|$ (Lemma 11). Hence, the interesting case is that S_0 has polynomial growth, i.e. S_0 is bounded.

Representation by rational functions. In order to simulate a VPA by a finite automaton on arbitrary words we will "flatten" the input word in the following way. The input word w is factorized $w = w_0 w_1 \cdots w_m$ into a descending prefix w_0, and call letters and well-matched factors $w_1, \ldots, w_m$. The descending prefix w_0 is replaced by $\sigma_0(w_0)$ and each well-matched factor w_i is replaced by a similar information $\sigma_1(w_i)$ which describes the behavior of the

VPA on the factor w_i and on each of its suffixes. The set Flat of all flattenings is a context-free language. Furthermore, there exists a rational function ν_f such that, if a flattening s represents a word $w \in \Sigma^*$ then $\nu_f(s)$ is a configuration representing the Myhill-Nerode class $\nu_L(w)$ (Proposition 9). Hence, we can reduce proving the main theorem to the question whether the $\overleftarrow{\nu}_f$-growth of Flat is always either polynomial or exponential.

This question is resolved positively as follows. We prove that for every rational function t with suffix-closed domain $X = \mathrm{dom}(t)$ the $\overleftarrow{t}$-growth of X is either polynomial or exponential (Theorem 6). In the case that S_0 has polynomial growth we can overapproximate Flat by a regular superset RegFlat. If the $\overleftarrow{\nu}_f$-growth of RegFlat is polynomial then the same holds trivially for the subset Flat. If the $\overleftarrow{\nu}_f$-growth of RegFlat is exponential then the proper choice of RegFlat ensures that Flat also has exponential $\overleftarrow{\nu}_f$-growth (Proposition 14).

Dichotomy for rational functions. The main technical result of this paper states that for every rational function $t\colon \Sigma^* \to \Omega^*$ with suffix-closed domain $X = \mathrm{dom}(t)$ the $\overleftarrow{t}$-growth of X is either polynomial or exponential. We emphasize that the range of $\overleftarrow{t}$ is not Ω^* but the free monoid over Ω^* (consisting of all finite sequences of words over Ω). There are in fact two reasons for exponential $\overleftarrow{t}$-growth: (i) The image $t(X)$ has exponential growth, and (ii) X contains a so called linear fooling set. We need these lower bounds in the more general setting where $X \subseteq \mathrm{dom}(t)$ is a context-free subset, namely $X = \mathrm{Flat}$.

▶ **Proposition 3.** *Let $t\colon \Sigma^* \to \Omega^*$ be rational with suffix-closed domain. If $X \subseteq \mathrm{dom}(t)$ is context-free and $t(X)$ has exponential growth then X has exponential t-growth and exponential $\overleftarrow{t}$-growth.*

▶ **Example 4.** Consider the transduction $f\colon \{a,b\}^* \to a^*$ defined by

$$f = \{(a^n, a^n) \mid n \in \mathbb{N}\} \cup \{(a^n b w, a^n) \mid n \in \mathbb{N}, w \in \{a,b\}^*\},$$

which projects a word over $\{a,b\}$ to its left-most (maximal) a-block and is rational. Its image $\overleftarrow{f}(\{a,b\}^*)$ can be identified with the set of all sequences of natural numbers which are concatenations of monotonically decreasing sequences of the form $(k, k-1, \dots, 0)$. There are exactly 2^n of such sequences of length n and hence $\{a,b\}^*$ has exponential $\overleftarrow{f}$-growth.

A *linear fooling scheme* for a partial function $t\colon \Sigma^* \to Y$ is a tuple (u_2, v_2, u, v, Z) where $u_2, v_2, u, v \in \Sigma^*$ and $Z \subseteq \Sigma^*$ such that u_2 is a suffix of u and v_2 is a suffix of v, $|u_2| = |v_2|$, $\{u_2, v_2\}\{u, v\}^* Z \subseteq \mathrm{dom}(t)$ and for all $n \in \mathbb{N}$ there exists a word $z_n \in Z$ of length $|z_n| \le O(n)$ such that $t(u_2 w z_n) \neq t(v_2 w z_n)$ for all $w \in \{u, v\}^{\le n}$. The set $\{u_2, v_2\}\{u, v\}^* Z$ is called a *linear fooling set* for t. Notice that the definition implies that $u_2 \neq v_2$ and hence u is not a suffix of v, and vice versa, i.e. $\{u, v\}$ is a *suffix code*. Therefore $\{u, v\}^n$ contains 2^n words of length $O(n)$ and thus $\{u_2, v_2\}\{u, v\}^*$ has exponential growth.

▶ **Proposition 5.** *Let $t\colon \Sigma^* \to \Omega^*$ be a partial function with suffix-closed domain. If $X \subseteq \mathrm{dom}(t)$ contains a linear fooling set for t then the $\overleftarrow{t}$-growth of X is exponential.*

Proof. Let (u_2, v_2, u, v, Z) be a linear fooling scheme with $\{u_2, v_2\}\{u, v\}^* Z \subseteq X$. Let $n \in \mathbb{N}$ and let $z_n \in Z$ with the properties from the definition. Consider two distinct words $w, w' \in \{u, v\}^n$. Without loss of generality the words have the form $w = w_1 u w_2$ and $w' = w_3 v w_2$ for some $w_1, w_2, w_3 \in \{u, v\}^*$. Hence w has the suffix $u_2 w_2$ and w' has the suffix $v_2 w_2$, which are suffixes of the same length. By assumption we have $t(u_2 w_2 z_n) \neq t(v_2 w_2 z_n)$ and hence also $\overleftarrow{t}(w z_n) \neq \overleftarrow{t}(w' z_n)$. This implies that $|\overleftarrow{t}(u_2 \{u, v\}^n z_n)| \ge 2^n$ for all $n \in \mathbb{N}$. Since all words in $u_2 \{u, v\}^n z_n \subseteq X$ have length $O(n)$ there exists a number $c > 1$ such that $|\overleftarrow{t}(X \cap \Sigma^{\le cn})| \ge 2^n$ for sufficiently large n. ◀

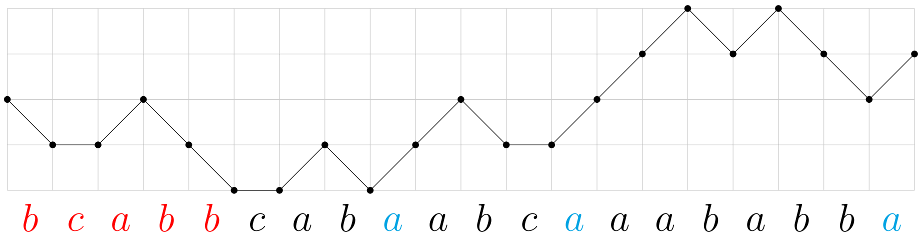

Figure 1 The stack height function for a word ($\Sigma_c = \{a\}$, $\Sigma_r = \{b\}$, $\Sigma_{int} = \{c\}$) and a monotonic factorization *bcabb cab a abc a aababb a*.

The following dichotomy theorem will be proved in Section 6.

▶ **Theorem 6.** *Let $t\colon \Sigma^* \to \Omega^*$ be rational and with suffix-closed domain $X = \mathrm{dom}(t)$. If X contains no linear fooling set for t and $t(X)$ is bounded then the $\overleftarrow{t}$-growth of X is polynomial. Otherwise the $\overleftarrow{t}$-growth of X is exponential.*

5 Reduction to transducer problem

Fix a VPA $A = (Q, \tilde{\Sigma}, \Gamma, \bot, q_0, \delta, F)$ and let $\emptyset \subsetneq L = \mathsf{L}(A) \subsetneq \Sigma^*$ for the rest of this section.

Monotonic factorization. A factorization of $w = w_0 w_1 \cdots w_m \in \Sigma^*$ into factors $w_i \in \Sigma^*$ is *monotonic* if w_0 is descending (possibly empty) and for each $1 \le i \le m$ the factor w_i is either a call letter $w_i \in \Sigma_c$ or a non-empty well-matched factor. If $w_0 w_1 \cdots w_m$ is a monotonic factorization then $w_i' w_{i+1} \cdots w_j$ is a monotonic factorization for any $0 \le i \le j \le m$ and suffix w_i' of w_i. To see that every word $w \in \Sigma^*$ has at least one monotonic factorization consider the set of non-empty maximal well-matched factors in w (maximal with respect to inclusion). Observe that two distinct maximal well-matched factors in a word cannot overlap because the union of two overlapping well-matched factors is again well-matched. Since every internal letter is well-matched the remaining positions contain only return and call letters. Furthermore, every remaining call letter must be to the right of every remaining return letter, which yields a monotonic factorization of w. Figure 1 shows a monotonic factorization $w = w_0 w_1 \cdots w_m$ where the descending prefix w_0 is colored red and call letters w_i are colored green. The *stack height function* for the word w increases (decreases) by one on call (return) letters and stays constant on internal letters.

Representation of Myhill-Nerode classes. To apply Theorem 2 we need a suitable description of the $\sim_L$-classes. We follow the approach in [5] of choosing length-lexicographic minimal representative configurations. Since their definition slightly differs from ours (according to their definition, a VPA may not read a return letter if the stack contains $\bot$ only) we briefly recall their argument (in the appendix). Let $\mathrm{rConf} = \{\delta(\bot q_0, w) \mid w \in \Sigma^*\}$ be the set of all *reachable* configurations in A, which is known to be regular [7, 11]. Two configurations $c_1, c_2 \in \mathrm{rConf}$ are *equivalent*, denoted by $c_1 \sim c_2$, if $\mathsf{L}(c_1) = \mathsf{L}(c_2)$. By fixing arbitrary linear orders on Γ and Q we can consider the length-lexicographical order on rConf and define the function $\mathsf{rep}\colon \mathrm{rConf} \to \mathrm{rConf}$ which chooses the minimal representative from each $\sim$-class, i.e. for all $c \in \mathrm{rConf}$ we have $\mathsf{rep}(c) \sim c$ and for any $c' \in \mathrm{rConf}$ with $c \sim c'$ we have $\mathsf{rep}(c) \le_{llex} c'$. The set of representative configurations is denoted by $\mathrm{Rep} = \mathsf{rep}(\mathrm{rConf})$.

▶ **Lemma 7** ([5]). *The function* rep *is rational.*

Finally we define $\nu_A \colon \Sigma^* \to \mathrm{Rep}$ by $\nu_A(w) = \mathsf{rep}(\delta(\bot q_0, w))$ for all $w \in \Sigma^*$. It represents $\sim_L$ in the sense that $\mathsf{L}(\nu_A(w)) = w^{-1}\mathsf{L}(A)$ for all $w \in \Sigma^*$ and hence $\nu_A(u) = \nu_A(v)$ if and only if $u \sim_L v$. Therefore we have $V_L(n) = \log |\bar{\nu}_A(\Sigma^{\le n})|$ by Theorem 2.

Flattenings. Since we cannot compute ν_A using a finite state transducer we choose a different representation of the input. Define the alphabet $\Sigma_f = \Sigma_c \cup Q \cup Q^Q$. A *flattening* is a word $s_0 s_1 \cdots s_m \in \Sigma_f^*$ where $s_0 \in Q^*$ and $s_i \in \Sigma_c \cup Q^Q Q^*$ for all $1 \le i \le m$. Notice that the factorization $s = s_0 s_1 \cdots s_m$ is unique. The set of all flattenings is $\mathrm{AllFlat} = Q^*(\Sigma_c \cup Q^Q Q^*)^*$. We define a function $t_f \colon \mathrm{AllFlat} \to \mathrm{rConf}$ as follows. Let $s = s_0 s_1 \cdots s_m \in \Sigma_f^*$ be a flattening and we define $t_f(s)$ by induction on m:

- If $s_0 = \varepsilon$ then $t_f(s_0) = \bot q_0$. If $s_0 = q_1 \cdots q_n \in Q^+$ then $t_f(s_0) = \bot q_1$.
- If $s_m \in \Sigma_c$ then $t_f(s_0 \cdots s_m) = \delta(t_f(s_0 \cdots s_{m-1}), s_m)$.
- If $s_m = \tau q_2 \cdots q_m \in Q^Q Q^*$ and $t_f(s_0 \cdots s_{m-1}) = \alpha q$ then $t_f(s) = \alpha\tau(q)$.

Define the function $\nu_f \colon \mathrm{AllFlat} \to \mathrm{Rep}$ by $\nu_f = \mathsf{rep} \circ t_f$.

▶ **Lemma 8.** *The functions t_f and ν_f are rational.*

Proof. We first define a transducer A_1 which handles flattenings where the initial factor is empty. Let $A_1 = (Q, \Sigma_f, Q \cup \Gamma, \{q_0\}, \Delta', Q, o)$ with the following transitions:

- $p \xrightarrow{q|\varepsilon} p$ for all $p, q \in Q$
- $p \xrightarrow{a|\gamma} q$ for all $\delta(p, a) = (\gamma, q)$ where $a \in \Sigma_c$
- $p \xrightarrow{\tau|\varepsilon} \tau(p)$ for all $p \in Q, \tau \in Q^Q$

and $o(q) = q$. For each $p \in Q$ let t_p be the rational function defined by A_1 with the only initial state p. One can easily show that for all $s \in \mathrm{AllFlat}$ we have $t_f(s) = \bot t_{q_0}(s)$ and $t_f(q_1 \cdots q_k s) = \bot t_{q_1}(s)$ for all $q_1 \cdots q_k \in Q^+$. Hence we can prove that t_f is rational by providing a transducer for t_f: First it verifies whether the input word belongs to the regular language $\mathrm{AllFlat} \subseteq \Sigma_f^*$. Simultaneously, it verifies whether the input word starts with a state $q \in Q$. If so, it memorizes q and simulates A_1 on s' from q, and otherwise A_1 is directly simulated on s from q_0. Since rep is rational by Lemma 7, ν_f is also rational. ◀

If $w = a_1 \cdots a_n \in D$ is a descending word then $\delta(\bot q_0, w) = \bot p$ for some $p \in Q$. By definition of ν_A there exists a state $q \in Q$ with $\nu_A(w) = \bot q$. Since each suffix of w is also descending we have $\bar{\nu}_A(w) = \bot q_1 \bot q_2 \cdots \bot q_n$ for some $q_1, \ldots, q_n \in Q$. We define $\sigma_0(w) = q_1 \cdots q_n \in Q^*$, i.e. we remove the redundant $\bot$-symbols from $\bar{\nu}_A(w)$. If w is non-empty and well-matched we additionally define $\sigma_1(w) = \tau q_2 \cdots q_n \in Q^Q Q^*$ where $\tau = \varphi(w)$. We define the sets $S_0 = \sigma_0(D)$ and $S_1 = \sigma_1(W \setminus \{\varepsilon\})$. Notice that S_0 is exactly the set of proper suffixes of words from S_1 since descending words are exactly the (proper) suffixes of well-matched words. We say that $s = s_0 s_1 \cdots s_m \in \mathrm{AllFlat}$ *represents* a word $w \in \Sigma^*$ if there exists a monotonic factorization $w = w_0 w_1 \cdots w_m \in \Sigma^*$ such that $s_0 = \sigma_0(w_0)$, and for all $1 \le i \le m$ if w_i is well-matched, then $s_i = \sigma_1(w_i)$, and if $w_i \in \Sigma_c$ then $s_i = w_i$. Since a word may have different monotonic factorizations, it may also be represented by many flattenings. We define the suffix-closed set $\mathrm{Flat} = S_0(\Sigma_c \cup S_1)^*$, containing all flattenings which represent some word.

▶ **Proposition 9.** *If $s \in \mathrm{AllFlat}$ represents $w \in \Sigma^*$ then $\nu_f(s) = \nu_A(w)$. Therefore, $\nu_f(\mathrm{Flat}) = \mathrm{Rep}$ and $V_L(n) = \log |\bar{\nu}_f(\mathrm{Flat} \cap \Sigma_f^{\le n})|$.*

▶ **Lemma 10.** *The languages S_0 and S_1 are context-free.*

Proof. Since S_0 is the set of all proper suffixes of words from S_1 it suffices to consider S_1. We will prove that $\{w \otimes \sigma_1(w) \mid w \in W\}$ is a VPL over the pushdown alphabet

$(\Sigma_c \times \Sigma_f, \Sigma_r \times \Sigma_f, \Sigma_{int} \times \Sigma_f)$. Since the class of context-free languages is closed under projections it then follows that S_1 is context-free. A VPA can test whether the first component $w = a_1 \cdots a_n$ is well-matched and whether the second component has the form $\tau q_2 \cdots q_n \in Q^Q Q^*$. Since VPLs are closed under Boolean operations, it suffices to test whether $\tau \neq \varphi(w)$ or there exists a state q_i with $\nu_A(a_i \cdots a_n) \neq \bot q_i$. To guess an incorrect state we use a VPA whose stack alphabet contains all stack symbols of A and a special symbol # representing the stack bottom. We guess and read a prefix of the input word and push/pop only the special symbol # on/from the stack. Then at some point we store the second component q_i in the next symbol and simulate A on the remaining suffix. Finally, we accept if and only if the reached state is q and $\mathsf{rep}(\bot q) \neq \bot q_i$. Similarly, we can verify τ by testing whether there exists a state $p \in Q$ with $\varphi(w)(p) \neq \tau(p)$. ◀

▶ **Lemma 11.** *The language S_0 is bounded if and only if S_1 is bounded. If S_0 is not bounded then the $\breve{\nu}_A$-growth of Σ^* is exponential and therefore $V_L(n) \notin o(n)$.*

Proof. Assume that $S_0 \subseteq s_1^* \cdots s_k^*$ is bounded. Since $S_1 \subseteq \bigcup\{\tau S_0 \mid \tau \in Q^Q\}$ we have $S_1 \subseteq \tau_1^* \cdots \tau_m^* s_1^* \cdots s_k^*$ for any enumeration $\tau_1, \ldots, \tau_m$ of Q^Q. Conversely, if S_1 is bounded then each word in S_0 is a factor, namely a proper suffix, of a word from S_1. Therefore S_0 must be also bounded.

If the context-free language $S_0 = \sigma_0(D) \subseteq Q^*$ is not bounded then its growth must be exponential. Recall that $\breve{\nu}_A(w)$ and $\sigma_0(w)$ are equal for all $w \in D$ up to the $\bot$-symbol. Hence $|\breve{\nu}_A(\Sigma^{\leq n})| \geq |\breve{\nu}_A(D \cap \Sigma^{\leq n})| = |\sigma_0(D \cap \Sigma^{\leq n})| = |S_0 \cap Q^{\leq n}|$, which proves the growth bound. ◀

Bounded overapproximation. By Lemma 11 we can restrict ourselves to the case that S_0 and S_1 are bounded languages, which will be assumed in the following. We define $\Psi(a_1 \cdots a_n) = \{(a_1, n), (a_2, n-1) \ldots, (a_n, 1)\}$ and $\Psi(L) = \bigcup_{w \in L} \Psi(w)$.

▶ **Lemma 12.** *Let K be a bounded context-free language. Then there exists a bounded regular superset $R \supseteq K$ such that $\{|w| \mid w \in K\} = \{|w| \mid w \in R\}$ and $\Psi(K) = \Psi(R)$, called a* bounded overapproximation *of K.*

Proof. We use Parikh's theorem [26], which implies that for every context-free language $K \subseteq \Sigma^*$ the set $\{|w| \mid w \in K\}$ is *semilinear*, i.e. a finite union of arithmetic progressions, and hence $\{v \in \Sigma^* \mid \exists w \in K : |v| = |w|\}$ is a regular language. Assume that $K \subseteq w_1^* \cdots w_k^*$ for some $w_1, \ldots, w_k \in \Sigma^*$. We define

$$R = (w_1^* \cdots w_k^*) \cap \{v \in \Sigma^* \mid \exists w \in K : |v| = |w|\} \cap \{w \in \Sigma^* \mid \Psi(w) \subseteq \Psi(K)\}.$$

Clearly, K is contained in R and it remains to verify that the third part is regular. It suffices to show that for each $a \in \Sigma$ the set $P_a = \{i \mid (a, i) \in \Psi(K)\}$ is semilinear because then an automaton can verify the property $\Psi(w) \subseteq \Psi(K)$. Consider the transducer

$$T_a = \{(a_1 \cdots a_n, \Box^{n-i+1}) \mid a_1 \cdots a_n \in \Sigma^*, \, a_i = a\}.$$

It is easy to see that T_a is rational and $T_a K = \{\Box^i \mid i \in P_a\}$. The claim follows again from Parikh's theorem. ◀

For each $\tau \in Q^Q$ let R_τ be a bounded overapproximation of $\tau^{-1} S_1$ and let $R_1 = \bigcup_{\tau \in Q^Q} (\tau R_\tau)$. Let $R_0 = \bigcup_{\tau \in Q^Q} \mathsf{Suf}(R_\tau)$, which is the set of all proper suffixes of words in R_1. Both R_0 and R_1 are also bounded languages. Finally, set $\mathrm{RegFlat} = R_0(\Sigma_c \cup R_1)^*$, which is the same as $\mathsf{Suf}((\Sigma_c \cup R_1)^*)$ and is suffix-closed. According to the definition of bounded overapproximations we can approximate a word $v = \tau q_2 \cdots q_k \in R_1$ in two possible ways: Firstly, define $\mathsf{apx}_\ell(v)$ to be any word of the form $\mathsf{apx}_\ell(v) = \tau p_2 \cdots p_k \in S_1$ with $|v| = |\mathsf{apx}_\ell(v)|$. Secondly, for any position $2 \le i \le k$ define $\mathsf{apx}_i(v)$ to be any word $\mathsf{apx}_i(v) = \tau s' q_i p_{i+1} \cdots p_k \in S_1$ where $s', p_{i+1} \cdots p_k \in Q^*$. If $r = r_0 r_1 \cdots r_m \in \mathrm{RegFlat}$ then we can replace any internal factor $r_i \in R_1$ by $\mathsf{apx}_\ell(r_i)$ or any $\mathsf{apx}_j(r_i)$ without changing the value of $\nu_f(r)$.

▶ **Proposition 13.** $\nu_f(\mathrm{Flat}) = \nu_f(\mathrm{RegFlat}) = \mathrm{Rep}$.

▶ **Proposition 14.** *If* RegFlat *contains a linear fooling set for ν_f then also* Flat *contains a linear fooling set for ν_f.*

Proof of Theorem 1. If $L = \emptyset$ or $L = \Sigma^*$ then $V_L(n) \in O(1)$. Now assume $\emptyset \subsetneq L \subsetneq \Sigma^*$, in which case we have $V_L(n) = \Omega(\log n)$. Furthermore we know that $V_L(n) = \log |\overleftarrow{\nu}_f(\mathrm{Flat} \cap \Sigma_f^{\le n})|$ by Proposition 9. If the constructed language S_0 is not bounded then $V_L(n) \notin o(n)$ by Lemma 11. Now assume that S_0 is bounded, in which case we can construct the regular language RegFlat. By Theorem 6 the $\overleftarrow{\nu}_f$-growth of RegFlat is either polynomial or exponential (formally, we have to restrict the domain of ν_f to the regular language RegFlat). If the $\overleftarrow{\nu}_f$-growth of RegFlat is polynomial then the same holds for its subset Flat, and hence $V_L(n) \in O(\log n)$. If the $\overleftarrow{\nu}_f$-growth of RegFlat is exponential then by Theorem 6 either the image $\nu_f(\mathrm{RegFlat})$ is not bounded or RegFlat contains a linear fooling set for ν_f. By Proposition 13 we have $\nu_f(\mathrm{RegFlat}) = \nu_f(\mathrm{Flat}) = \mathrm{Rep}$. Hence, if Rep has exponential growth then Proposition 3 implies that Flat has exponential $\overleftarrow{\nu}_f$-growth and hence $V_L(n) \notin o(n)$. If RegFlat contains a linear fooling set for ν_f then also Flat contains one by Proposition 14. By Proposition 5 the $\overleftarrow{\nu}_f$-growth of Flat is exponential and hence $V_L(n) \notin o(n)$. ◀

6 Dichotomy for rational functions

In this section we will prove Theorem 6. Let $t \colon \Sigma^* \to \Omega^*$ be a rational function with suffix-closed domain $X = \mathrm{dom}(t)$. By Proposition 3 the interesting case is where the image $t(X)$ is polynomial growing, i.e. a bounded language. There are two further necessary properties in order to achieve polynomial $\overleftarrow{t}$-growth. Since we apply the rational function to all suffixes, it is natural to consider right transducers, reading the input from right to left. The first property states that t has to resemble so called right-subsequential functions, which are defined by deterministic finite right transducers. Here we will make use of a representation of rational functions due to Reutenauer and Schützenberger, which decomposes the rational function t into a right congruence $\mathcal{R}_t$ and a right-subsequential transducer B [27]. Secondly, we demand that B is well-behaved, which means that, roughly speaking, the output produced during a run inside a strongly connected component only depends on its entry state and the length of the run. We will prove that in fact these properties are sufficient for the polynomial $\overleftarrow{t}$-growth and in all other cases X contains a linear fooling set.

The case of finite-index right congruences. Let $\sim$ be a finite index right congruence on Σ^* and $\approx$ its suffix expansion. We will characterize those finite index right congruences $\sim$ where $\Sigma^{\le n}/\approx$ is polynomially bounded, which can be viewed as a special case of Theorem 6 since $\nu_\sim \colon \Sigma^* \to \Sigma^*/\sim$ is rational. First assume that $\sim$ is the Myhill-Nerode right congruence $\sim_L$ of a regular language L. Since $\log |\Sigma^{\le n}/\approx|$ is exactly the space complexity $V_L(n)$ by Theorem 2, this case was characterized in [18] using so called critical tuples in the minimal DFA for L. We slightly adapt this definition for right congruences. A *critical tuple* in a right congruence $\sim$ is a tuple of words $(u_2, v_2, u, v) \in (\Sigma^*)^4$ such that $|u_2| = |v_2| \geq 1$, there exist $u_1, v_1 \in \Sigma^*$ with $u = u_1 u_2$, $v = v_1 v_2$, and $u_2 w \not\sim v_2 w$ for all $w \in \{u, v\}^*$.

▶ **Proposition 15.** *If $\sim$ has a critical tuple then $|\Sigma^{\leq n}/\approx|$ grows exponentially and there exists a critical tuple (u_2, v_2, u, v) in $\sim$ such that $u_2 u \sim u_2 w u$ and $v_2 u \sim v_2 w u$ for all $w \in \{u, v\}^*$.*

Proof. If (u_2, v_2, u, v) is critical tuple in a right congruence $\sim$ then we claim that $|\Sigma^{\leq n}/\approx|$ grows exponentially. Let $n \in \mathbb{N}$ and let $w \neq w' \in \{u, v\}^n$. There exists a word $z \in \{u, v\}^*$ such that w and w' have the suffixes $u_2 z$ and $v_2 z$ of equal length. By the definition of critical tuples we have $u_2 z \not\sim v_2 z$, which implies $w \not\approx w'$. Therefore $|\Sigma^{\leq cn}/\approx| \geq 2^n$ where $c = \max\{|u|, |v|\}$.

The second part is based on the proof of [19, Lemma 7.4]. Let $\equiv$ be the syntactic congruence on Σ^* defined by $x \equiv y$ if and only if $\ell x \sim \ell y$ for all $\ell \in \Sigma^*$. Since $\sim$ is a right congruence $\equiv$ is a congruence on Σ^* of finite index satisfying $\equiv \subseteq \sim$. Define the monoid $M = \Sigma^*/\equiv$. It is known that there exists a number $\omega \in \mathbb{N}$ such that m^ω is idempotent for all $m \in M$, i.e. $m^\omega \cdot m^\omega = m^\omega$. Now let (u_2, v_2, u, v) be a critical tuple and define $u' = (v^\omega u^\omega)^\omega$ and $v' = (v^\omega u^\omega)^\omega v^\omega$. Since u_2 is a suffix of u', v_2 is a suffix of u' and $u', v' \in \{u, v\}^*$ the tuple (u_2, v_2, u', v') is again a critical tuple in $\sim$. Furthermore we have $u'u' = (v^\omega u^\omega)^\omega (v^\omega u^\omega)^\omega \equiv (v^\omega u^\omega)^\omega = u'$ and $v'u' = (v^\omega u^\omega)^\omega v^\omega (v^\omega u^\omega)^\omega \equiv (v^\omega u^\omega)^\omega = u'$, and therefore $u' \equiv w u'$ for all $w \in \{u', v'\}^*$. Since $\equiv$ is a congruence this implies $u_2 u' \equiv u_2 w u'$ and $v_2 u' \equiv v_2 w u'$ for all $w \in \{u', v'\}^*$, and thus also $u_2 u' \sim u_2 w u'$ and $v_2 u' \sim v_2 w u'$, which concludes the proof. ◀

▶ **Theorem 16.** *Let $L \subseteq \Sigma^*$ be regular. Then $V_L(n) \in O(\log n)$ if and only if $|\Sigma^{\leq n}/\approx_L|$ is polynomially bounded if and only if $\sim_L$ has no critical tuple.*

Proof. The first equivalence follows from Theorem 2. By Proposition 15 the existence of a critical tuple in $\sim$ implies exponential growth of $|\Sigma^{\leq n}/\approx|$.

Now assume that $V_L(n) \notin O(\log n)$. By [18, Lemma 7.2] there exist words $u_2, v_2, u, v \in \Sigma^*$ such that u_2 is a suffix of u, v_2 is a suffix of v, $|u_2| = |v_2|$ and $u_2 w \not\sim_L v_2 w'$ for all $w, w' \in \{u, v\}^*$ (one needs the fact that $x \sim_L y$ if and only if x and y reach the same state in the minimal DFA for L). Since in particular $u_2 w \not\sim_L v_2 w$ for all $w \in \{u, v\}^*$ the tuple (u_2, v_2, u, v) constitutes a critical tuple. ◀

We generalize this theorem to arbitrary finite index right congruences (Theorem 18). Given equivalence relations $\sim$ and $\sim'$ on a set X, we say that $\sim'$ is *coarser* than $\sim$ if $\sim \subseteq \sim'$, i.e. each $\sim'$-class is a union of $\sim$-classes. The *intersection* $\sim \cap \sim'$ is again an equivalence relation on X.

▶ **Lemma 17.** *Let $\sim$ and $\sim'$ be right congruences.*

(a) *If $\sim'$ is coarser than $\sim$ and $\sim$ has no critical tuple, then $\sim'$ also has no critical tuple.*

(b) *If $\sim$ and $\sim'$ have no critical tuple then $\sim \cap \sim'$ is also a right congruence which has no critical tuple*

Proof. Closure under coarsening is clear because the property "$\sim$ has no critical tuple" is *positive* in $\sim$: $\forall u = u_1 u_2 \, \forall v = v_1 v_2 (|u_2| = |v_2| \to \exists w \in \{u, v\}^* : u_2 w \sim v_2 w)$.

Consider two right congruences $\sim$, $\sim'$ which have no critical tuples. One can verify that their intersection $\sim \cap \sim'$ is again a right congruence. Let $u = u_1 u_2$ and $v = v_1 v_2$ with $|u_2| = |v_2|$. Because $\sim$ has no critical tuple there exist a word $w \in \{u, v\}^*$ with $u_2 w \sim v_2 w$. Now consider the condition for the words $u_1(u_2 w)$ and $v_1(v_2 w)$. Because $\sim'$ has no critical tuple there exists a word $x \in \{uw, vw\}^*$ such that $u_2 w x \sim' v_2 w x$. Since $\sim$ is a right congruence we also have $u_2wx \sim v_2wx$ and thus $u_2wx\ (\sim \cap \sim')\ v_2wx$. This proves that $\sim \cap \sim'$ has no critical tuple. ◀

▶ **Theorem 18.** *$|\Sigma^{\le n}/{\approx}|$ is polynomially bounded if and only if $\sim$ has no critical tuple.*

Proof. Let $u_1, \dots, u_m$ be representatives from each $\sim$-class. Observe that $\sim = \bigcap_{i=1}^m \sim_{[u_i]_\sim}$ because $\sim$ saturates each class $[u_i]_\sim$ and $\bigcap_{i=1}^m \sim_{[u_i]_\sim}$ also saturates each class $[v]_\sim$. Let us write $\sim_i$ instead of $\sim_{[u_i]_\sim}$ and let $\approx_i$ be its suffix expansion $\approx_{[u_i]_\sim}$. Then we have $\sim = \bigcap_{i=1}^m \sim_i$ and $\approx = \bigcap_{i=1}^m \approx_i$. This implies that

$$\max_{1\le i\le m} |\Sigma^{\le n}/{\approx_i}| \le |\Sigma^{\le n}/{\approx}| \le \prod_{i=1}^m |\Sigma^{\le n}/{\approx_i}|. \tag{1}$$

($\Rightarrow$): If $|\Sigma^{\le n}/{\approx}|$ is polynomially bounded then the same holds for $|\Sigma^{\le n}/{\approx_i}|$ for all $1 \le i \le k$ by (1). By Theorem 16 $\sim_{[u_i]_\sim}$ has no critical tuple for all $1 \le i \le k$ and therefore Lemma 17(b) implies that $\sim = \bigcap_{i=1}^m \sim_{[u_i]_\sim}$ has no critical tuple.

($\Leftarrow$): If $\sim$ has no critical tuple then each congruence $\sim_i$ has no critical tuple by Lemma 17(a) because $\sim_i$ is coarser than $\sim$. Theorem 16 implies that $|\Sigma^{\le n}/{\approx_i}|$ is polynomially bounded for all $1 \le i \le k$. By (1) also $|\Sigma^{\le n}/{\approx}|$ is polynomially bounded. ◀

Regular look-ahead. A result due to Reutenauer and Schützenberger states that every rational function f can be factorized as $f = r \circ \ell$ where ℓ and r are *left- and right-subsequential*, respectively [27]. A rational function is left- or right-subsequential if the input is read in a deterministic fashion from left to right and right to left, respectively. In the literature the order of the directions is usually reversed, i.e. one decomposes t as $f = r \circ \ell$. Often this is described by the statement that every rational function is (left-)subsequential with regular look-ahead. Furthermore, this decomposition is canonical in a certain sense.

We follow the notation from the survey paper [16]. A *right-subsequential transducer* $B = (Q, \Sigma, \Omega, F, \Delta, \{q_{in}\}, o)$ is a real-time right transducer which is *deterministic*, i.e. q_{in} is the only initial state and for every $p \in Q$ and $a \in \Sigma$ there exists at most one transition $(p, a, y, q) \in \Delta$. Clearly, right-subsequential transducers define rational functions, the so called *right-subsequential functions*, but not every rational function is right-subsequential. Let $\mathcal{R}$ be a right congruence on Σ^* with finite index. The *look-ahead extension* is the injective function $e_\mathcal{R} \colon \Sigma^* \to (\Sigma \times \Sigma^*/\mathcal{R})^*$ defined by

$$e_\mathcal{R}(a_1 \cdots a_n) = (a_1, [\varepsilon]_\mathcal{R})(a_2, [a_1]_\mathcal{R})(a_3, [a_1a_2]_\mathcal{R}) \cdots (a_n, [a_1 \cdots a_{n-1}]_\mathcal{R}).$$

Let $f \colon \Sigma^* \to \Omega^*$ be a partial function. The partial function $f[\mathcal{R}] \colon (\Sigma \times \Sigma^*/\mathcal{R})^* \to \Omega^*$ with $\mathrm{dom}(f[\mathcal{R}]) = e_\mathcal{R}(\mathrm{dom}(f))$ is defined by $f[\mathcal{R}](e_\mathcal{R}(x)) = f(x)$. Furthermore we define a right congruence $\mathcal{R}_f$ on Σ^*. For this we need the distance function $\|x, y\| = |x| + |y| - 2|x \wedge y|$ where $x \wedge y$ is the longest common suffix of x and y. Equivalently, $\|x, y\|$ is the length of the reduced word of xy^{-1} in the free group generated by Σ. Notice that $\|\cdot,\cdot\|$ satisfies the triangle inequality. We define $u\ \mathcal{R}_f\ v$ if and only if (i) $u \sim_{\mathrm{dom}(f)} v$ and (ii) $\{\|f(uw), f(vw)\| \mid uw, vw \in \mathrm{dom}(f)\}$ is finite. One can verify that $\mathcal{R}_f$ is a right congruence on Σ^*. As an example, recall the rational transduction f from Example 4. The induced right congruence $\mathcal{R}_f$ has two classes, which are a^* and $a^*b\{a, b\}^*$.

▶ **Theorem 19** ([27]). *A partial function $f \colon \Sigma^* \to \Omega^*$ is rational if and only if $\mathcal{R}_f$ has finite index and $f[\mathcal{R}_f]$ is right-subsequential.*

For the rest of the section let $B = (Q, \Sigma \times \Sigma^*/\mathcal{R}_t, \Omega, F, \Delta, \{q_{in}\}, o)$ be a trim right-subsequential transducer for $t[\mathcal{R}_t]$. One obtains an unambiguous real-time right transducer A for t by projection to the first component, i.e. $A = (Q, \Sigma, \Omega, F, \Lambda, \{q_{in}\}, o)$ where $\Lambda = \{(q, a, y, p) \mid (q, (a, b), y, p) \in \Delta\}$. Notice that every run $q \xleftarrow{x|y} p$ in A induces a corresponding run $q \xleftarrow{(x,z)|y} p$ in B for some $z \in (\Sigma^*/\mathcal{R}_t)^*$ and that this correspondence is a bijection between the sets of all runs in A and B. We need two auxiliary lemmas which concern the right congruence $\mathcal{R}_t$.

▶ **Lemma 20** (Short distances). *Let $u, v, w \in \Sigma^*$ with $uw, vw \in X$. If $u \; \mathcal{R}_t \; v$ then $\|t(uw), t(vw)\| \le O(|u| + |v|)$.*

Two partial functions $t_1, t_2 \colon \Sigma^* \to \Omega^*$ are *adjacent* if $\sup\{\|t_1(w), t_2(w)\| \mid w \in \mathrm{dom}(t_1) \cap \mathrm{dom}(t_2)\} < \infty$ where $\sup \emptyset = -\infty$. We remark that two functions are adjacent in our definition if and only if their reversals are adjacent according to the original definition [27]. Notice that $u \; \mathcal{R}_t \; v$ if and only if $u \sim_X v$ and the functions $w \mapsto t(uw)$ and $w \mapsto t(vw)$ are adjacent.

▶ **Lemma 21** (Short witnesses). *Let $t_1, t_2 \colon \Sigma^* \to \Omega^*$ be rational functions which are not adjacent. Then there are words $x, y, z \in \Sigma^*$ such that $xy^*z \subseteq \mathrm{dom}(t_1) \cap \mathrm{dom}(t_2)$ and $\|t_1(xy^kz), t_2(xy^kz)\| = \Omega(k)$. In particular, for each $k \in \mathbb{N}$ there exists a word $x \in \mathrm{dom}(t_1) \cap \mathrm{dom}(t_2)$ of length $|x| \le O(k)$ such that $\|t_1(x), t_2(x)\| \ge k$.*

▶ **Proposition 22.** *If $\mathcal{R}_t$ has a critical tuple then X contains a linear fooling set.*

Proof. Let (u_2, v_2, u, v) be a critical tuple in $\mathcal{R}_t$ with $u = u_1u_2$ and $v = v_1v_2$. By Proposition 15 we can assume that $u_2u \; \mathcal{R}_t \; u_2wu$ and $v_2u \; \mathcal{R}_t \; v_2wu$ for all $w \in \{u, v\}^*$. By assumption we know that $(u_2u, v_2u) \notin \mathcal{R}_t$. Furthermore, we claim that $u_2u \sim_X v_2u$: Let $z \in \Sigma^*$ and assume that $u_2uz \in X$. Then $u_2v_1v_2uz \in X$ because $u_2u \sim_X u_2v_1v_2u$, and thus $v_2uz \in X$ because X is suffix-closed. The other direction follows by a symmetric argument.

Let $n \in \mathbb{N}$ and define

$$N = \max_{x \in \{u_2, v_2\}} \max_{w \in \{u,v\}^{\le n}} \sup\{\|t(xuz), t(xwuz)\| \mid xuz, xwuz \in X\} < \infty.$$

By Lemma 20 we have $N \le O(n)$. Since $(u_2u, v_2u) \notin \mathcal{R}_t$ and $u_2u \sim_X v_2u$, the functions $z \mapsto t(u_2uz)$ and $z \mapsto t(v_2uz)$ are not adjacent. By Lemma 21 there exists a word $z_n \in (u_2u)^{-1}X$ with $\|t(u_2uz_n), t(v_2uz_n)\| \ge 2N + 1$ and $|z_n| \le O(N) \le O(n)$. We claim that $t(u_2wuz_n) \ne t(v_2wuz_n)$ for all $w \in \{u, v\}^{\le n}$: By the triangle inequality we have

$$\begin{aligned} 2N + 1 &\le \|t(u_2uz_n), t(v_2uz_n)\| \\ &\le \|t(u_2uz_n), t(u_2wuz_n)\| + \|t(u_2wuz_n), t(v_2wuz_n)\| + \|t(v_2wuz_n), t(v_2uz_n)\| \\ &\le 2N + \|t(u_2wuz_n), t(v_2wuz_n)\| \end{aligned}$$

which implies $\|t(u_2wuz_n), t(v_2wuz_n)\| \ge 1$ and in particular $t(u_2wuz_n) \ne t(v_2wuz_n)$. We have proved that for each $n \in \mathbb{N}$ there exists a word z_n of length $O(n)$ such that $t(u_2wuz_n) \ne t(v_2wuz_n)$ for all $w \in \{u, v\}^{\le n}$. If Z is the set of all constructed z_n for $n \in \mathbb{N}$ then $\{u_2, v_2\}\{u, v\}^*uZ \subseteq X$ and (u_2, v_2, u, v, uZ) is a linear fooling scheme. ◀

Well-behaved transducers. Let $(Q, \preceq)$ be the quasi-order defined by $q \preceq p$ iff there exists a run from p to q in A or equivalently in B. Its equivalence classes are the strongly connected components (SCCs) of A and B. A word $w \in \Sigma^*$ is *guarded* by a state $p \in Q$ if there exists a run $q' \xleftarrow{w} p$ in A such that $p \preceq q'$, i.e. p and q' belong to the same SCC. Notice that the set of all words which are guarded by a fixed state p is suffix-closed. A run $q \xleftarrow{w} p$ in A is *guarded* if w is guarded by p. We say that A is *well-behaved* if for all $p \in Q$ and all guarded accepting runs π, π' from p with $|\pi| = |\pi'|$ we have $\mathsf{out}_F(\pi) = \mathsf{out}_F(\pi')$.

▶ **Proposition 23.** *If A is not well-behaved then X contains a linear fooling set.*

Proof. Assume there exist states $p, q, r, q', r' \in Q$, and accepting runs $q \xleftarrow{u_2} p$ and $r \xleftarrow{v_2} p$ with $|u_2| = |v_2|$ and $\mathsf{out}_F(q \xleftarrow{u_2} p) \neq \mathsf{out}_F(r \xleftarrow{v_2} p)$. Furthermore let $p \xleftarrow{u_1} q' \xleftarrow{u_2} p$, $p \xleftarrow{v_1} r' \xleftarrow{v_2} p$ and $p \xleftarrow{s} q_{in}$ be runs. Let $u = u_1u_2$ and $v = v_1v_2$ and consider any word $w \in \{u, v\}^*$. Since $t(u_2ws) = \mathsf{out}_F(q \xleftarrow{u_2} p)\,\mathsf{out}(p \xleftarrow{ws} q_{in})$ and $t(v_2ws) = \mathsf{out}_F(r \xleftarrow{v_2} p)\,\mathsf{out}(p \xleftarrow{ws} q_{in})$, we have $t(u_2ws) \neq t(v_2ws)$. This shows that $(u_2, v_2, u, v, \{s\})$ is a linear fooling scheme. ◀

If π is a non-empty run $p \xleftarrow{a_1 \cdots a_n} q$ in A and $p \xleftarrow{(a_1,\rho_1)\cdots(a_n,\rho_n)} q$ is the corresponding run in B then we call ρ_1 the *key* of π. The following lemma justifies the name, stating that π is determined by the state q, the word $a_1 \cdots a_n$ and the key ρ_1.

▶ **Lemma 24.** *If $p \xleftarrow{w} q$ and $p' \xleftarrow{w} q$ are non-empty runs in A with the same key then the runs must be identical.*

Proof. Assume that $w = a_1 \cdots a_n$ and let $p \xleftarrow{(a_1,\rho_1)\cdots(a_n,\rho_n)} q$ and $p' \xleftarrow{(a_1,\rho'_1)\cdots(a_n,\rho'_n)} q$ be the corresponding runs in B with $\rho_1 = \rho'_1$. We proceed by induction on n. If $n = 1$ then this statement is trivial because B is deterministic. Now assume $n \geq 2$ and let $p \xleftarrow{a_1} r \xleftarrow{a_2 \cdots a_n} q$ and $p' \xleftarrow{a_1} r' \xleftarrow{a_2 \cdots a_n} q$. Since B is trim there exist an accepting run on $e_{\mathcal{R}_t}(u)$ from p and an accepting run on $e_{\mathcal{R}_t}(u')$ from p' for some words $u, u' \in \Sigma^*$. By definition of $t[\mathcal{R}_t]$ we have $[u]_{\mathcal{R}_t} = \rho_1 = \rho'_1 = [u']_{\mathcal{R}_t}$ and therefore $\rho_2 = [ua_1]_{\mathcal{R}_t} = [u'a_1]_{\mathcal{R}_t} = \rho'_2$. By induction hypothesis we know that the runs $r \xleftarrow{a_2 \cdots a_n} q$ and $r' \xleftarrow{a_2 \cdots a_n} q$ are identical. Since $p \xleftarrow{(a_1,\rho_1)} r$ and $p' \xleftarrow{(a_1,\rho'_1)} r'$ and B is deterministic we must have $p = p'$. ◀

Let π be any run on a word $y \in \Sigma^*$. If π is not guarded, we can factorize $\pi = \pi'\pi''$ such that π'' is the shortest suffix of π which is unguarded, and then iterate this process on π'. This yields unique factorizations $\pi = \pi_0\pi_1 \cdots \pi_m$ and $y = y_0y_1 \cdots y_m$ where π_i is a run on y_i from a state q_i to a state q_{i-1} such that y_i is the shortest suffix of $y_0 \cdots y_i$ which is not guarded by q_i for all $1 \leq i \leq m$ and π_0 is guarded. The factorization $\pi = \pi_0\pi_1 \cdots \pi_m$ is the *guarded factorization* of π.

▶ **Proposition 25.** *Assume that $t(X)$ is bounded, A is well-behaved and $\mathcal{R}_t$ has no critical tuple. Then the $\overleftarrow{t}$-growth of X is polynomially bounded.*

Proof. We will describe an encoding of $\overleftarrow{t}(w)$ for $w \in X$ using $O(\log |w|)$ bits. For each word $w \in \Sigma^*$ and each state $q \in Q$ we define a tree $T_{q,w}$ recursively, which carries information at the nodes and edges. If w is guarded by q then $T_{q,w}$ consists of a single node labelled by the pair $(q, |w|)$. Otherwise let $w = uv$ such that v is the shortest suffix of w which is not guarded by q. Then $T_{q,w}$ has a root which is labelled by the tuple $(q, |w|, |v|, \breve{\nu}_{\mathcal{R}_t}(u))$. For each run $p \xleftarrow{v} q$ we attach $T_{p,u}$ to the root as a direct subtree. The edge is labelled by the pair $(\rho, \mathsf{out}(p \xleftarrow{v} q))$ where ρ is the key of $p \xleftarrow{v} q$. By Lemma 24 distinct outgoing edges from the root are labelled by distinct keys.

The tree $T_{q,w}$ can be encoded using $O(\log |w|)$ bits: Since we have $p \prec q$ for every unguarded run $p \xleftarrow{v} q$ the tree $T_{q,w}$ has height at most $|Q|$ and size at most $|Q|^{|Q|}$. All occurring numbers have at most magnitude $|w|$, and the states and keys can be encoded by $O(1)$ bits. The output words $\mathsf{out}(p \xleftarrow{v} q)$ are factors of words from the bounded language

$t(X)$ and have length at most $\mathsf{iml}(A) \cdot |v|$. Thus they can be encoded using $O(\log |w|)$ bits. The node label $\overleftarrow{\nu}_{\mathcal{R}_t}(u)$ can be encoded using $O(\log |w|)$ bits by Theorem 18 since $\mathcal{R}_t$ has no critical tuple.

Let $w = xy \in \Sigma^*$, $q \in Q$ and let π be an accepting run on y from q. We show that $T_{q,w}$ and $|y|$ determine $\mathsf{out}_F(\pi)$ by induction on the length of the guarded factorization $\pi = \pi_0\pi_1 \cdots \pi_m$. Since $T_{q_{in},w}$ determines the length $|w|$, the tuple $\overleftarrow{t}(w)$ is determined by $T_{q_{in},w}$ for all $w \in X$. If $m = 0$ then y is guarded by q. Since A is well-behaved $\mathsf{out}_F(\pi)$ is determined by q (which is part of the label of the root of $T_{q,w}$) and $|y|$ only. Now assume $m \geq 1$ and suppose that π_i is a run $q_{i-1} \xleftarrow{y_i} q_i$ for all $1 \leq i \leq m$ with $q_m = q$. Then y_m is the shortest suffix of w which is not guarded by q. The root of $T_{q,w}$ is labelled by $(q, |y_m|, \overleftarrow{\nu}_{\mathcal{R}_t}(xy_0 \cdots y_{m-1}))$. Since $|y_m|$ and $|y|$ are known, we can also determine $|y_0 \cdots y_{m-1}|$. From $\overleftarrow{\nu}_{\mathcal{R}_t}(xy_0 \cdots y_{m-1})$ and $|y_0 \cdots y_{m-1}|$ we can then determine $[y_0 \cdots y_{m-1}]_{\mathcal{R}_t}$, which is the key of π_m. By Lemma 24 we can find the unique edge which is labelled by $([y_0 \cdots y_{m-1}]_{\mathcal{R}_t}, \mathsf{out}(\pi_m))$. It leads to the direct subtree $T_{q_{m-1},xy_0 \cdots y_{m-1}}$ of $T_{q,w}$. By induction hypothesis $T_{q_{m-1},xy_0 \cdots y_{m-1}}$ and $|y_0 \cdots y_{m-1}|$ determine $\mathsf{out}_F(\pi_0 \cdots \pi_{m-1})$. Finally, we can determine $\mathsf{out}_F(\pi_0 \cdots \pi_m) = \mathsf{out}_F(\pi_0 \cdots \pi_{m-1})\,\mathsf{out}(\pi_m)$, concluding the proof. ◀

Now we can prove Theorem 6: If X contains no linear fooling set for t then A must be well-behaved by Proposition 23 and $\mathcal{R}_t$ has no critical tuple by Proposition 22. If additionally $t(X)$ is bounded then the $\overleftarrow{t}$-growth of X is polynomially bounded by Proposition 25. Otherwise, if either X contains a linear fooling set or $t(X)$ is not bounded then the $\overleftarrow{t}$-growth of X is exponential by Proposition 5 and by Proposition 3.

References

1 Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 202–211, 2004. URL: `http://doi.acm.org/10.1145/1007352.1007390`, `doi:10.1145/1007352.1007390`.

2 Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. In *Proceedings of PODS 2004*, pages 286–296. ACM, 2004.

3 Brian Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of PODS 2003*, pages 234–243. ACM, 2003.

4 Ajesh Babu, Nutan Limaye, Jaikumar Radhakrishnan, and Girish Varma. Streaming algorithms for language recognition problems. *Theor. Comput. Sci.*, 494:13–23, 2013. URL: `https://doi.org/10.1016/j.tcs.2012.12.028`, `doi:10.1016/j.tcs.2012.12.028`.

5 Vince Bárány, Christof Löding, and Olivier Serre. Regularity problems for visibly pushdown languages. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, pages 420–431, 2006. URL: `https://doi.org/10.1007/11672142_34`.

6 Jean Berstel. *Transductions and context-free languages*, volume 38 of *Teubner Studienbücher: Informatik*. Teubner, 1979.

7 Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR '97: Concurrency Theory, 8th International Conference, Warsaw, Poland, July 1-4, 1997, Proceedings*, pages 135–150, 1997. URL: `https://doi.org/10.1007/3-540-63141-0_10`, `doi:10.1007/3-540-63141-0\_10`.

8 Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science FOCS 2007*, pages 283–293. IEEE Computer Society, 2007.

9 Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. Optimal sampling from sliding windows. *J. Comput. Syst. Sci.*, 78(1):260–272, 2012.

10 Dany Breslauer and Zvi Galil. Real-time streaming string-matching. *ACM Trans. Algorithms*, 10(4):22:1–22:12, 2014.

11 J Richard Büchi. Regular canonical systems. *Archiv für mathematische Logik und Grundlagenforschung*, 6(3-4):91–111, 1964.

12 Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana A. Starikovskaya. Dictionary matching in a stream. In *Proceedings of ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 361–372. Springer, 2015.

13 Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana A. Starikovskaya. The k-mismatch problem revisited. In *Proceedings of SODA 2016*, pages 2039–2052. SIAM, 2016.

14 Raphaël Clifford and Tatiana A. Starikovskaya. Approximate hamming distance in a stream. In *Proceedings of ICALP 2016*, volume 55 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

15 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.

16 Emmanuel Filiot and Pierre-Alain Reynier. Transducers, logic and algebra for functions of finite words. *SIGLOG News*, 3(3):4–19, 2016. URL: `http://doi.acm.org/10.1145/2984450.2984453`, `doi:10.1145/2984450.2984453`.

17 Nathanaël François, Frédéric Magniez, Michel de Rougemont, and Olivier Serre. Streaming property testing of visibly pushdown languages. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 43:1–43:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. URL: `https://doi.org/10.4230/LIPIcs.ESA.2016.43`, `doi:10.4230/LIPIcs.ESA.2016.43`.

18 Moses Ganardi, Danny Hucke, Daniel König, Markus Lohrey, and Konstantinos Mamouras. Automata theory on sliding windows. In *Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science, STACS 2018*, volume 96 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

19 Moses Ganardi, Danny Hucke, Daniel König, Markus Lohrey, and Konstantinos Mamouras. Automata theory on sliding windows. Technical report, arXiv.org, 2018. `https://arxiv.org/abs/1702.04376`.

20 Moses Ganardi, Danny Hucke, and Markus Lohrey. Querying regular languages over sliding windows. In *Proceedings of the 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016*, volume 65 of *LIPIcs*, pages 18:1–18:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

21 Moses Ganardi, Artur Jez, and Markus Lohrey. Sliding windows over context-free languages. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, pages 15:1–15:15, 2018. URL: `https://doi.org/10.4230/LIPIcs.MFCS.2018.15`, `doi:10.4230/LIPIcs.MFCS.2018.15`.

22 Seymour Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, Inc., New York, NY, USA, 1966.

23 Seymour Ginsburg and Edwin H Spanier. Bounded algol-like languages. *Transactions of the American Mathematical Society*, 113(2):333–368, 1964.

24 Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. In *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994*, pages 367–392, 1994. URL: `https://doi.org/10.1007/3-540-60178-3_93`, `doi:10.1007/3-540-60178-3\_93`.

25 Frédéric Magniez, Claire Mathieu, and Ashwin Nayak. Recognizing well-parenthesized expressions in the streaming model. *SIAM J. Comput.*, 43(6):1880–1905, 2014. URL: `https://doi.org/10.1137/130926122`, `doi:10.1137/130926122`.

26 Rohit Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966. URL: `http://doi.acm.org/10.1145/321356.321364`, `doi:10.1145/321356.321364`.

27 Christophe Reutenauer and Marcel Paul Schützenberger. Minimization of rational word functions. *SIAM J. Comput.*, 20(4):669–685, 1991. URL: `https://doi.org/10.1137/0220042`, `doi:10.1137/0220042`.

28 Andreas Weber and Reinhard Klemm. Economy of description for single-valued transducers. *Inf. Comput.*, 118(2):327–340, 1995. URL: `https://doi.org/10.1006/inco.1995.1071`, `doi:10.1006/inco.1995.1071`.

A Proof of Proposition 3

Since $\overleftarrow{t}(x)$ determines $t(x)$ we have $|\overleftarrow{t}(X \cap \Sigma^{\leq n})| \geq |t(X \cap \Sigma^{\leq n})|$ for all $n \in \mathbb{N}$. It suffices to show that every non-empty preimage $t^{-1}(\{y\})$ contains at least one word of length $O(|y|)$ in X, i.e. there exists a number $c > 0$ such that $t(X) \cap \Omega^{\leq n} \subseteq t(X \cap \Sigma^{\leq cn})$ for sufficiently large $n \in \mathbb{N}$. Then, if by assumption $|t(X) \cap \Omega^{\leq n}|$ grows exponentially, then so does $|t(X \cap \Sigma^{\leq n})|$ and also $|\overleftarrow{t}(X \cap \Sigma^{\leq n})|$.

Let us now prove the claim, for which we need to define context-free grammars over arbitrary monoids. A context-free grammar over a monoid M has the form $G = (N, S, \to_G)$ where N is a finite set of nonterminals (which is disjoint from M), S is the starting nonterminal, and $\to_G \subseteq N \times (M * N^*)$ is a finite set of productions where $M * N^*$ is the free product of the monoids M and N^*. A derivation tree for $m \in M$ is a node-labelled rooted ordered tree with the following properties:

- Inner nodes are labelled by nonterminals $A \in N$.
- Leaves are labelled by monoid elements $m \in M$.
- If a node s has children $s_1, \dots, s_k$ where v is labelled by A and $s_1, \dots, s_k$ are labelled by $\alpha_1, \dots, \alpha_k$ then there exists a production $A \to_G \alpha_1 \cdots \alpha_k$.
- If $m_1, \dots, m_\ell$ are the labels of the leaves read from left to right then $m = m_1 \cdots m_\ell$.

The language $\mathsf{L}(A)$ generated by a nonterminal $A \in N$ is the set of all elements $m \in M$ such that there exists a derivation tree for m whose root is labelled by A. The language $\mathsf{L}(G)$ generated by G is the language $\mathsf{L}(S)$.

We first construct from a context-free grammar $G = (N, S, \to_G)$ for $X \subseteq \Sigma^*$ a context-free grammar $H = (N', S', \to_H)$ for $t|_X = \{(x, t(x)) \mid x \in X\}$ over the product monoid $\Sigma^* \times \Omega^*$. We can assume that $\varepsilon \notin X$ and that G is in Chomsky normal form, i.e. each rule has the form $A \to a$ where $A \in N$ and $a \in \Sigma$, or $A \to BC$ where $A, B, C \in N$. Let $R = (Q, \Sigma, \Omega, I, \Delta, F, o)$ be a real-time transducer for t. We define $N' = \{S'\} \cup \{A_{p,q} \mid A \in N,\, p, q \in Q\}$ and $\to_H$ contains the productions

- $A_{p,q} \to_H (a, y)$ for all productions $A \to_G a$ and transitions $p \xrightarrow{a|y} q$ in R,
- $A_{p,q} \to_H B_{p,r} C_{r,q}$ for all productions $A \to_G BC$ and $p, q, r \in Q$,
- $S' \to_H S_{p,q}\, (\varepsilon, o(q))$ for all $(p, q) \in I \times F$.

One can verify that for all $A \in N$ and $p, q \in Q$ the language $\mathsf{L}(A_{p,q})$ is the set of all pairs $(x, y) \in \mathsf{L}(A) \times \Omega^*$ such that $p \xrightarrow{x|y} q$ in R, and that $\mathsf{L}(H) = t|_X$.

Now let $A \in N$, $p, q \in Q$ and $(x, y) \in \mathsf{L}(A_{p,q})$ with the property that $|x| = \min\{|x'| \mid (x', y) \in \mathsf{L}(A_{p,q})\}$. Consider a derivation tree T for (x, y) whose root is labelled by $A_{p,q}$. If s is a node in T which derives (u, v) then we define the *weight* of s to be $|v|$. Clearly, the weight of an inner node is the sum of the weights of its children.

▷ Claim 26. If $(s_1, s_2, \dots, s_k)$ is a path in T such that all nodes s_i on the path have the same length then $k \le |N'|$.

Proof. Assume that $k > |N'|$. There exist two nodes $s_i \neq s_j$ with $i < j$ which are labelled by the same nonterminal from N'. The subtrees rooted in s_i and s_j are derivation trees for pairs (u, v) and (u', v) for some $u, u' \in \Sigma^*$ and $v \in \Omega^*$ where u' is a proper factor of u. We can then replace the subtree rooted in s_i by the subtree rooted in s_j and obtain a derivation tree for a pair (x', y) with $|x'| < |x|$, contradiction. ◀

Set $c = |N'|$. By the claim above every subtree whose root has weight 0 has depth at most c and hence its size is at most $C = 2^c - 1$. Define $D = c + (c-1)C$.

▷ Claim 27. The derivation tree T has $O(|y|)$ nodes.

Proof. We prove by induction on $|y|$ that, if $|y| \ge 1$ then T has at most $(2|y|-1)D$ nodes. The root of T has weight $|y|$. Let $(s_1, \dots, s_k)$ be the maximal path starting in the root whose nodes have weight $|y|$. We know that $k \le c$. If s_i' is the sibling of s_{i-1} for $2 \le i \le k$, then s_i' has weight 0 and the subtree rooted in s_i' has at most C nodes.

1. Assume that s_k is a leaf. Then T consists of at most $D = c + (c-1)C \le (2|y|-1)D$ nodes, namely $k \le c$ nodes on the path $(s_1, \dots, s_k)$ and $c - 1$ many subtrees with at most C nodes.
2. Assume that s_k has two children s_{k+1} and s_{k+1}' and let w and w' be the weights of s_{k+1} and s_{k+1}', respectively. We have $|y| = w + w'$ and $1 \le w, w' < |y|$. By induction hypothesis the subtrees rooted in s_{k+1} and s_{k+1}' have at most $(2w-1)D$ and $(2w'-1)D$ nodes, respectively. Therefore T has in total at most $D + (2w-1)D + (2w'-1)D \le (2|y|-1)D$ nodes.

This concludes the proof of the claim. ◀

Now let $y \in t(X) \cap \Omega^{\le n}$ and $x \in X$ be any word with $t(x) = y$. There exists an initial accepting run $p \xrightarrow{x|y'} q$ with $y = y'\, o(q)$. As shown above there exists a word x' with $p \xrightarrow{x'|y'} q$ and $x' \le O(|y'|) \le O(n)$, which concludes the proof.

B Proof of Lemma 7

In [5] it was observed that $\sim$ can be recognized by a synchronous 2-tape automaton. The *convolution* of two words $u = a_1 \cdots a_m, v = b_1 \cdots b_n \in \Omega^*$ is the word $u \otimes v = c_1 \cdots c_\ell$ of length $\ell = \max(m, n)$ over the alphabet $(\Omega \cup \{\Box\})^2$ where $c_i = (a_i, b_i)$ if $1 \le i \le \min(m, n)$, $c_i = (a_i, \Box)$ if $m < i \le n$ and $c_i = (\Box, b_i)$ if $n < i \le m$. Similarly, one can define an associative operation $\otimes$ on k-tuples of words. A k-ary relation $R \subseteq (\Omega^*)^k$ is *synchronous rational* if $\otimes R = \{\otimes(u_1, \dots, u_k) \mid (u_1, \dots, u_k) \in R\}$ is a regular language over $(\Omega \cup \{\Box\})^k$. The set of synchronous rational relations is known to be closed under first-order operations and, in particular, under Boolean operations, cf. [24]. Clearly, every binary synchronous rational relation is a rational transduction.

▶ **Lemma 28** ([5]). *The equivalence relation $\sim^{\mathsf{R}}$ is synchronous rational.*

Proof. We present a right automaton which recognizes the complement of $\sim^{\mathsf{R}}$. It reads two configurations αp and βq synchronously which are aligned to the right, from right to left. The automaton stores a pair of states of A, starting with the pair (p, q). It then guesses a word w by its monotonic factorization which witnesses that w belongs to exactly one of the languages $\mathsf{L}(\alpha p)$ and $\mathsf{L}(\beta q)$. Notice that it suffices to read the maximal descending prefix of w and test whether the reached state pair (p', q') belongs to some fixed set of state pairs since the remaining ascending suffix cannot access the stack contents of the reached configurations. To simulate A on a descending prefix in each step the automaton either guesses a return symbol and removes the top most stack symbol of both configurations (or leaves $\bot$ at the top), or guesses a state transformation $\tau \in \varphi(W)$ which only modifies the current state pair. ◀

It is well-known that $\leq_{llex}$ is a synchronous rational relation. By the closure properties of synchronous rational relations the function rep is rational.

C Proof of Proposition 9

Let $w = w_0 w_1 \cdots w_m \in \Sigma^*$ be a monotonic factorization and let $s = s_0 s_1 \cdots s_m \in \mathrm{Flat}$ be the associated flattening. We prove $t_f(s) \sim \delta(\bot q_0, w)$ by induction on m.

- If $m = 0$ and $s_0 = \varepsilon$ then $t_f(s) = \bot q_0 = \delta(\bot q_0, \varepsilon)$.
- If $m = 0$ and $s_0 = q_1 \cdots q_k \in Q^+$ then $t_f(s) = \bot q_1$ and $\nu_A(w) = \mathsf{rep}(\delta(\bot q_0, w)) = \bot q_1$.
- If $m \geq 1$ and $s_m \in \Sigma_c$ then $s_m = w_m$. By induction hypothesis we know that $t_f(s_0 \cdots s_{m-1}) \sim \delta(\bot q_0, w_0 \cdots w_{m-1})$. Since $\delta(\bot q_0, w) = \delta(\delta(\bot q_0, w_0 \cdots w_{m-1}), w_m)$ and $t_f(s) = \delta(t_f(s_0 \cdots s_{m-1}), s_m)$ we obtain $\delta(\bot q_0, w) \sim t_f(s)$.
- If $m \geq 1$ and $s_m = \tau q_2 \cdots q_k \in Q^Q Q^*$ then w_m is well-matched and $\varphi(w_m) = \tau$. Assume that $t_f(s_0 \cdots s_{m-1}) = \alpha p$ and $\delta(\bot q_0, w_0 \cdots w_{m-1}) = \beta q$. By induction hypothesis we know that $\alpha p \sim \beta q$. Since $t_f(s) = \alpha \tau(p) = \delta(\alpha p, w_m)$ and $\delta(\bot q_0, w) = \delta(\beta q, w_m)$ we obtain $t_f(s) \sim \delta(\bot q_0, w)$.

Since $\nu_f = \mathsf{rep} \circ t_f$ and $\nu_A(w) = \mathsf{rep}(\delta(\bot q_0, w))$ we have $\nu_f(s) = \nu_A(w)$.

▶ **Lemma 29.** *Let $w = w_0 w_1 \cdots w_m \in \Sigma^*$ be a monotonic factorization with empty initial factor $w_0 = \varepsilon$ and let $s = s_0 s_1 \cdots s_m \in \Sigma_f^*$ be the associated flattening. If $\delta(\bot p, w) = \bot \alpha q$ then $p \xrightarrow{s|\alpha} q$ in A_1 and hence $t_p(s) = \alpha q$.*

Proof. Proof by induction on m. If $m = 0$ then $w = s = \varepsilon$, $p = q$ and $\alpha = \varepsilon$. For the induction step assume $\delta(\bot p, w_1 \cdots w_{m-1}) = \bot \alpha q$ and $\delta(\bot \alpha q, w_m) = \bot \alpha \alpha_1 q_1$. By induction hypothesis the run of A_1 on s has the form $p \xrightarrow{s_1 \cdots s_{m-1} | \alpha} q \xrightarrow{s_m | \alpha_2} q_2$. We do a case distinction.

If $w_m \in \Sigma_c$ then $\delta(q, w_m) = (\alpha_1, q_1)$. Since $s_m = w_m$ and by definition of A_1 we have $\alpha_1 = \alpha_2$ and $q_1 = q_2$. Otherwise $w_m \in W \setminus \{\varepsilon\}$ and $\alpha_1 = \varepsilon$. The word $s_m = \sigma_1(w_m)$ starts with $\tau = \varphi(w_m)$ and we have $\tau(q) = q_1$. By definition of A_1 we indeed have $q_2 = \tau(q)$ and $\alpha_2 = \varepsilon$. ◀

We define the following total function $t_f \colon \Sigma_f^* \to (Q \cup \Gamma)^*$. Let $s \in \Sigma_f^*$ be an input word and let $q_1 \cdots q_k \in Q^*$ be the maximal prefix of s from Q^*, say $s = q_1 \cdots q_k s'$ for some $s' \in \Sigma_f^*$. Then we define

$$t_f(s) = \begin{cases} t_{q_0}(s), & \text{if } k = 0, \\ t_{q_1}(s'), & \text{if } k \geq 1. \end{cases}$$

It is easy to see that t_f is rational by providing a transducer for t_f. It verifies whether s starts with a state $q \in Q$. If so, it memorizes q and simulates A_1 on s' from q, and otherwise A_1 is directly simulated on s from q_0.

Now let $w = w_0 w_1 \cdots w_m$ be a monotonic factorization and $s = s_0 s_1 \cdots s_m \in \Sigma_f^*$ be the associated flattening. We claim that $\delta(\bot q_0, w) \sim \bot t_f(s)$. If $w_0 = \varepsilon$ then $s_0 = \varepsilon$ and

s does not start with a state from Q. In this case we have $\delta(\bot q_0, w) = \bot t_{q_0}(s) = \bot t_f(s)$ by Lemma 29. If $w_0 \neq \varepsilon$ then s_0 starts with some state $q_1 \in Q$. By definition of σ_0 we have $\delta(\bot q_0, w_0) \sim \bot q_1$ and thus $\delta(\bot q_0, w) \sim \delta(\bot q_1, w_1 \cdots w_m)$. By Lemma 29 we have $\delta(\bot q_1, w_1 \cdots w_m) = \bot t_{q_1}(s_1 \cdots s_m) = \bot t_f(s)$, which proves the claim. Finally, we can set $\nu_f(s) = \mathsf{rep}(\bot t_f(s))$ for all $s \in \Sigma_f^*$.

D Proof of Proposition 13

By Proposition 9 we know $\nu_f(\mathrm{Flat}) = \mathrm{Rep}$. Clearly $\nu_f(\mathrm{Flat}) \subseteq \nu_f(\mathrm{RegFlat})$ and it remains to show the other inclusion. Consider a word $r \in \mathrm{RegFlat}$ which does not have a non-empty prefix from R_0, say $r = u_1 v_1 u_2 v_2 \cdots v_m u_{m+1}$ where $u_1, \dots, u_{m+1} \in \Sigma_c^*$ and $v_1, \dots, v_m \in R_1$. Then $r' = u_1 \, \mathsf{apx}_\ell(v_1) \, u_2 \, \mathsf{apx}_\ell(v_2) \cdots \mathsf{apx}_\ell(v_m) \, u_{m+1}$ belongs to Flat and $\nu_f(r) = \nu_f(r')$.

Now assume that r has a non-empty prefix $q_1 \cdots q_k \in R_0$. We do the replacements above and the following. By definition $q_1 \cdots q_k$ is a proper suffix of some word $x = \tau p_2 \cdots p_{i-1} q_1 \cdots q_k \in R_1$. Let $y = \mathsf{apx}_i(x) \in S_1$ which has a proper suffix of the form $q_1 q_2' \cdots q_k'$ belonging to S_0. We can replace $q_1 \cdots q_k$ by $q_1 q_2' \cdots q_k'$ in r and obtain a word $r' \in \mathrm{Flat}$ with $\nu_f(r) = \nu_f(r')$.

E Proof of Proposition 14

Assume that (u_2, v_2, u, v, Z) is a linear fooling scheme for ν_f with $\{u_2, v_2\}\{u, v\}^* Z \subseteq \mathrm{RegFlat}$. We first ensure that $\{u, v\} \cup Z \subseteq (\Sigma_c \cup R_1)^*$. Assume that $u, v \in Q^*$ and hence $\{u_2, v_2\}\{u, v\}^* \subseteq Q^*$ is contained in the set of prefixes of words in R_0. Since R_0 is bounded by assumption also $\{u_2, v_2\}\{u, v\}^*$ must be bounded, which contradicts the fact that $\{u_2, v_2\}\{u, v\}^*$ has exponential growth.

Without loss of generality, assume that $u = u_3 u_4$ such that u_4 either starts with a call letter $a \in \Sigma_c$ or a transformation $\tau \in Q^Q$. We claim that $(u_2 u_3, v_2 u_3, u_4 u u_3, u_4 v u_3, u_4 Z)$ is a linear fooling scheme for ν_f. It has the following properties:

- $\{u_2 u_3, v_2 u_3\}\{u_4 u u_3, u_4 v u_3\}^* u_4 Z \subseteq \mathrm{RegFlat}$,
- $u_2 u_3$ is a suffix of $u_4 u u_3$,
- $v_2 u_3$ is a suffix of $u_4 v u_3$,
- $|u_2 u_3| = |v_2 u_3|$.

Also, we know that for every $n \in \mathbb{N}$ there exists a word $z_n \in Z$ with $|z_n| \le O(n)$ and $\nu_f(u_2 w z_n) \neq \nu_f(v_2 w z_n)$ for all $w \in \{uu, uv\}^{\le n}\{u\}$ and thus, by factoring out the first u_3- and the last u_4-factor, we have $\nu_f(u_2 u_3 w u_4 z_n) \neq \nu_f(v_2 u_3 w u_4 z_n)$ for all $w \in \{u_4 u u_3, u_4 v u_3\}^{\le n}$. Hence we have verified the conditions of a linear fooling scheme. It has the desired properties that $\{u_4 u u_3, u_4 v u_3\} \cup u_4 Z \subseteq (\Sigma_c \cup R_1)^*$ because u_4 starts with a call letter or a transformation.

Now let (u_2, v_2, u, v, Z) be a linear fooling scheme with $\{u, v\} \cup Z \subseteq (\Sigma_c \cup R_1)^*$. We replace occurring factors from R_1 by factors from S_1 while maintaining the values $\nu_f(u_2 w z)$ and $\nu_f(v_2 w z)$ for $w \in \{u, v\}^*$ and $z \in Z$.

1. First, in each word $z \in Z \subseteq (\Sigma_c \cup R_1)^*$ we replace each R_1-factor v by $\mathsf{apx}_\ell(v)$ which ensures that $Z \subseteq (\Sigma_c \cup S_1)^*$.
2. Next consider u and v, and assume that $u = u_1 u_2$ and $v = v_1 v_2$ for some $u_1, v_1 \in \Sigma_f^*$. Let us consider R_1-factors which cross the factorization $u = u_1 u_2$ or $v = v_1 v_2$, respectively. If u_2 starts with some state we can factorize u_1 and u_2 as $u_1 = u_3 \tau q_2 \cdots q_{i-1}$ and $u_2 = q_i \cdots q_k u_4$ where $u_3, u_4 \in (\Sigma_c \cup R_1)^*$ and $\tau q_2 \cdots q_k \in R_1$. Let $\mathsf{apx}_i(\tau q_2 \cdots q_k) = \tau s' q_i p_{i+1} \cdots p_k \in S_1$. We replace u_1 by $u_3 \tau s'$ and u_2 by $q_i p_{i+1} \cdots p_k u_4$. Notice that the

length of u_2 has not changed (this maintains $|u_2| = |v_2|$) and the first state of u_2 has not changed either (this maintains the values $\nu_f(u_2wz)$). If v_2 starts with some state we do the analogous replacements for v_1 and v_2.

3. Finally, each R_1-factor v in u_1, u_2, v_1 and v_2 is replaced by $\mathsf{apx}_\ell(v)$.

One can verify that the obtained tuple (u_2, v_2, u, v, Z) is again a linear fooling scheme for ν_f satisfying $\{u_2, v_2\}\{u, v\}^* Z \subseteq \text{Flat}$.

F Proof of Lemma 20

Suppose that $w = a_1 \cdots a_m$. Since $\mathcal{R}_t$ is a right congruence we know that $ua_1 \cdots a_i \; \mathcal{R}_t \; va_1 \cdots a_i$ for all $0 \le i \le m$. By definition of the look-ahead extension the words $e_{\mathcal{R}_t}(uw)$ and $e_{\mathcal{R}_t}(vw)$ have the common suffix

$$s = \binom{a_1}{[u]_{\mathcal{R}_t}} \binom{a_2}{[ua_1]_{\mathcal{R}_t}} \cdots \binom{a_m}{[ua_1 \cdots a_{m-1}]_{\mathcal{R}_t}}.$$

The initial accepting runs of B on $e_{\mathcal{R}_t}(uw)$ and $e_{\mathcal{R}_t}(vw)$ have the form

$$q \xleftarrow{e_{\mathcal{R}_t}(u)} p \xleftarrow{s} q_{in} \quad \text{and} \quad r \xleftarrow{e_{\mathcal{R}_t}(v)} p \xleftarrow{s} q_{in}$$

and thus $t(uw)$ and $t(vw)$ share the suffix $\mathsf{out}(p \xleftarrow{s} q_0)$. This implies

$$\|t(uw), t(vw)\| \le |\mathsf{out}_F(q \xleftarrow{e_{\mathcal{R}_t}(u)} p)| + |\mathsf{out}_F(r \xleftarrow{e_{\mathcal{R}_t}(v)} p)| \le \mathsf{iml}(A) \cdot (|u| + |v| + 2),$$

proving the statement.

G Proof of Lemma 21

Assume that t_1 and t_2 are not adjacent. By [27, Proof of Proposition 1.] there exist words $x, y, z \in \Sigma^*$ and $u_1, u_2, v_1, v_2, w_1, w_2 \in \Omega^*$ such that $t_1(xy^kz) = u_1v_1^kw_1$, $t_2(xy^kz) = u_2v_2^kw_2$ for all $k \in \mathbb{N}$, and $\sup\{\|u_1v_1^kw_1, u_2v_2^kw_2\| \mid k \in \mathbb{N}\} = \infty$. By the triangle inequality we have

$$\begin{aligned} \|v_1^kw_1, v_2^kw_2\| &\le \|v_1^kw_1, u_1v_1^kw_1\| + \|u_1v_1^kw_1, u_2v_2^kw_2\| + \|u_2v_2^kw_2, v_2^kw_2\| \\ &= |u_1| + \|u_1v_1^kw_1, u_2v_2^kw_2\| + |u_2| = \|t_1(xy^kz), t_2(xy^kz)\| + |u_1| + |u_2|. \end{aligned}$$

We prove that $\|v_1^kw_1, v_2^kw_2\| \ge \Omega(k)$ which implies that $\|t_1(xy^kz), t_2(xy^kz)\| \ge \Omega(k)$. If both $v_1 = v_2 = \varepsilon$ then

$$\sup_{k \in \mathbb{N}} \|v_1^kw_1, v_2^kw_2\| = \|w_1, w_2\| < \infty,$$

which contradicts $\sup\{\|u_1v_1^kw_1, u_2v_2^kw_2\| \mid k \in \mathbb{N}\} = \infty$. If $|v_1| \ne |v_2|$ then

$$\|v_1^kw_1, v_2^kw_2\| \ge \big||v_1^kw_1| - |v_2^kw_2|\big| = \Omega(k).$$

Now assume $|v_1| = |v_2| \ge 1$. Since

$$\|v_1^kw_1, v_2^kw_2\| = |v_1^kw_1| + |v_2^kw_2| - 2|v_1^kw_1 \wedge v_2^kw_2| \ge \Omega(k) - 2|v_1^kw_1 \wedge v_2^kw_2|$$

it suffices to show that $\sup_k |v_1^kw_1 \wedge v_2^kw_2| < \infty$. Towards a contradiction assume that $\sup_k |v_1^kw_1 \wedge v_2^kw_2| = \infty$. Then, for every $k \in \mathbb{N}$ there exists $K \in \mathbb{N}$ such that $|v_1^Kw_1 \wedge v_2^Kw_2| \ge \max\{|v_1^kw_1|, |v_2^kw_2|\}$. If $|v_1^kw_1| \ge |v_2^kw_2|$ then $v_1^kw_1$ is a suffix of $v_1^Kw_1 \wedge v_2^Kw_2$ and otherwise

$v_2^k w_2$ is a suffix of $v_1^K w_1 \wedge v_2^K w_2$. This shows that for all $k \in \mathbb{N}$ either $v_1^k w_1$ is a suffix of $v_2^k w_2$, or vice versa, and therefore $|v_1^k w_1 \wedge v_2^k w_2| = \min\{|v_1^k w_1|, |v_2^k w_2|\}$. Since $|v_1| = |v_2|$ we obtain

$$\|v_1^k w_1, v_2^k w_2\| = |v_1^k w_1| + |v_2^k w_2| - 2\min\{|v_1^k w_1|, |v_2^k w_2|\} = |w_1| + |w_2| - 2\min\{|w_1|, |w_2|\}$$

contradicting $\sup_k \|v_1^k w_1, v_2^k w_2\| = \infty$.