Randomized sliding window algorithms for regular languages

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

Danny Hucke
Universität Siegen, Germany
hucke@eti.uni-siegen.de

Markus Lohrey
Universität Siegen, Germany
lohrey@eti.uni-siegen.de

Abstract

A sliding window algorithm receives a stream of symbols and has to output at each time instant a certain value which only depends on the last n symbols. If the algorithm is randomized, then at each time instant it produces an incorrect output with probability at most ϵ, which is a constant error bound. This work proposes a more relaxed definition of correctness which is parameterized by the error bound ϵ and the failure ratio ϕ: a randomized sliding window algorithm is required to err with probability at most ϵ at a portion of $1 - \phi$ of all time instants of an input stream. This work continues the investigation of sliding window algorithms for regular languages. In previous works a trichotomy theorem was shown for deterministic algorithms: the optimal space complexity is either constant, logarithmic or linear in the window size. The main results of this paper concerns three natural settings (randomized algorithms with failure ratio zero and randomized/deterministic algorithms with bounded failure ratio) and provide natural language theoretic characterizations of the space complexity classes.

2012 ACM Subject Classification Theory of computation → Regular languages, Theory of computation → Streaming models

Keywords and phrases sliding windows, regular languages, randomized complexity

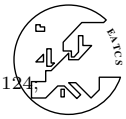
Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.124

Related Version A full version (with some additional results) can be found in [9].

1 Introduction

Sliding window algorithms process an input sequence $a_1 a_2 \cdots a_m$ from left to right and have at time t only direct access to the current symbol a_t. Moreover, at each time instant t the algorithm is required to compute a value that depends on the last n symbols. The value n is called the *window size* and the last n symbols form the *active window* at time t. In many streaming applications, data items are outdated after a certain time and the sliding window model is a simple way to model this. A general goal in the area of sliding window algorithms is to avoid the explicit storage of the window content (which requires $\Omega(n)$ bits), and, instead, to work in considerably smaller space, e.g. polylogarithmic space with respect to the window size n. A detailed introduction into the sliding window model can be found in [1, Chapter 8].

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Don Sannella; Article No. 124; pp. 124:1–124:13



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Regular languages over sliding windows. In our recent papers [7, 8] we initiated the study of sliding window algorithms for regular languages. In general, a sliding window algorithm for a language $L \subseteq \Sigma^*$ decides at every time instant whether the active window belongs to L. In [8] we proved that for every regular language L the optimal space bound for a sliding window algorithm for L is either constant, logarithmic or linear in the window size. This trichotomy result resembles the well-known fact that every regular (even context-free) language has either polynomial or exponential growth. In [7] we also gave several characterizations for these space classes: A regular language has a sliding window algorithm with space $\mathcal{O}(\log n)$ if and only if it belongs to $\langle \mathbf{LI}, \mathbf{Len} \rangle$, which denotes the Boolean closure of the class **LI** of regular left ideals and the class **Len** of regular length languages. Moreover, a regular language has a sliding window algorithm that uses space $\mathcal{O}(1)$ if and only if it belongs to $\langle \mathbf{ST}, \mathbf{Len} \rangle$, where **ST** is the class of suffix-testable languages. The formal definitions of these and other language classes can be found in Section 2. The goal of this work is to extend the results from [7, 8] to randomized algorithms.

Main results. Consider a Monte-Carlo sliding window algorithm which can produce incorrect outputs. Abstracting away from the actual computation, we will view such a randomized sliding window algorithm (SWA for short) as a family $\mathcal{R} = (R_n)_{n \geq 0}$ of probabilistic automata, where R_n is the algorithm for window size n. We denote by $f(\mathcal{R}, n)$ the number of bits stored by R_n, which is the logarithm of the number of states. There are different ways to define correctness of $\mathcal{R}$ for a certain language L. The maybe most natural choice is to require that after reading an arbitrary input word $a_1 a_2 \cdots a_m$, the algorithm R_n correctly decides whether $a_{m-n+1} \cdots a_m \in L$ with probability at least $2/3$. This ensures that for every input stream and every time instant, one can be sure to get a correct answer with probability at least $2/3$. By a standard probability amplification argument, $2/3$ can be replaced by any probability strictly between $1/2$ and 1. With this definition (formal details can be found in Section 3) our first main result is the following, where **SF** denotes the class of all regular suffix-free languages (point (1) and (5) are from [8]).

▶ **Theorem 1.1.** *Let $L \subseteq \Sigma^*$ be a regular language.*

1. *If $L \in \langle \mathbf{ST}, \mathbf{Len} \rangle$, then L has a deterministic SWA $\mathcal{R}$ with $f(\mathcal{R}, n) = \mathcal{O}(1)$.*
2. *If $L \notin \langle \mathbf{ST}, \mathbf{Len} \rangle$, then $f(\mathcal{R}, n) \notin o(\log \log n)$ for every randomized SWA $\mathcal{R}$ for L.*
3. *If $L \in \langle \mathbf{ST}, \mathbf{SF}, \mathbf{Len} \rangle$, then L has a randomized SWA $\mathcal{R}$ with $f(\mathcal{R}, n) = \mathcal{O}(\log \log n)$.*
4. *If $L \notin \langle \mathbf{ST}, \mathbf{SF}, \mathbf{Len} \rangle$, then $f(\mathcal{R}, n) \notin o(\log n)$ for every randomized SWA $\mathcal{R}$ for L.*
5. *If $L \in \langle \mathbf{LI}, \mathbf{Len} \rangle$, then L has a deterministic SWA $\mathcal{R}$ with $f(\mathcal{R}, n) = \mathcal{O}(\log n)$.*
6. *If $L \notin \langle \mathbf{LI}, \mathbf{Len} \rangle$, then $f(\mathcal{R}, n) \notin o(n)$ for every randomized SWA $\mathcal{R}$ for L.*

One may argue that an algorithm which occasionally produces a wrong answer with probability $> 1/3$ is acceptable as well. This motivates the following definition: We say that a randomized algorithm for a certain language L and a window size n has *failure ratio* ϕ if for every input stream, the portion of all time instants where the algorithm gives a wrong answer with probability $> 1/3$ is bounded by ϕ. The second main result concerns algorithms with a bounded failure ratio. If we ask for an arbitrarily small non-zero failure ratio we get the following space dichotomy, where **PF** denotes the class of regular prefix-free languages:

▶ **Theorem 1.2.** *Let $L \subseteq \Sigma^*$ be a regular language.*

1. *If $L \in \langle \mathbf{LI}, \mathbf{PF}, \mathbf{Len} \rangle$ and $0 < \phi \leq 1$, then L has a randomized SWA with $f(\mathcal{R}, n) = \mathcal{O}(1)$ and failure ratio ϕ.*
2. *If $L \notin \langle \mathbf{LI}, \mathbf{PF}, \mathbf{Len} \rangle$, then there exists a failure ratio $0 < \phi \leq 1$ such that $f(\mathcal{R}, n) \notin o(n)$ for every randomized SWA $\mathcal{R}$ for L with failure ratio ϕ.*

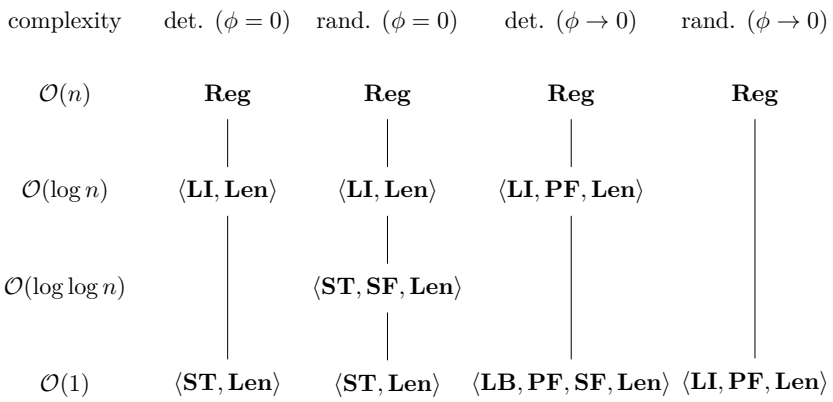


Figure 1 All language classes are defined in Section 2.

The notion of failure ratio makes sense for deterministic sliding window algorithms as well. Our third main result is a space trichotomy for deterministic sliding window algorithms having a bounded failure ratio. Let **LB** denote the class of left ideals generated by bifix-free (i.e., prefix- and suffix-free) regular languages. We then show:

▶ **Theorem 1.3.** *Let $L \subseteq \Sigma^*$ be a regular language.*

1. *If $L \in \langle \mathbf{LB}, \mathbf{PF}, \mathbf{SF}, \mathbf{Len} \rangle$ and $0 < \phi \leq 1$, then L has a deterministic SWA with $f(\mathcal{R}, n) = \mathcal{O}(1)$ and failure ratio ϕ.*
2. *If $L \notin \langle \mathbf{LB}, \mathbf{PF}, \mathbf{SF}, \mathbf{Len} \rangle$, then there exists a failure ratio $0 < \phi \leq 1$ such that $f(\mathcal{R}, n) \notin o(\log n)$ for every deterministic SWA $\mathcal{R}$ for L with failure ratio ϕ.*
3. *If $L \in \langle \mathbf{LI}, \mathbf{PF}, \mathbf{Len} \rangle$ and $0 < \phi \leq 1$, then L has a deterministic SWA with $f(\mathcal{R}, n) = \mathcal{O}(\log n)$ and failure ratio ϕ.*
4. *If $L \notin \langle \mathbf{LI}, \mathbf{PF}, \mathbf{Len} \rangle$, then there exists a failure ratio $0 < \phi \leq 1$ such that $f(\mathcal{R}, n) \notin o(n)$ for every deterministic SWA $\mathcal{R}$ for L with failure ratio ϕ.*

Note that Theorem 1.3(4) is an immediate corollary of Theorem 1.2(2), and that Theorem 1.3(3) follows from Theorem 1.3(1) and Theorem 1.1(5). Figure 1 summarizes the resulting space classes for each setting; the left column shows the three classes for the deterministic setting [7, 8]. Figure 2 shows an inclusion diagram for the language classes in Figure 1. One can show that the example languages in Figure 2 witness the strictness of the inclusions.

Technical contributions. Our randomized algorithms are based on a simulation of a counter by a Bernoulli random variable. Albeit simple, this *Bernoulli algorithm* can be utilized for sliding window algorithms with a bounded failure ratio. We present automata-theoretic descriptions of several Boolean closed language classes, to derive certain witness words for proving the lower bounds. Concerning lower bounds for sliding window algorithms with a bounded failure ratio, we consider a promise variant of the communication problem IDX_n, which has randomized one-way communication complexity $\Omega(n)$. Also we prove that a DFA which can count up to n, allowing a bounded failure ratio, needs $\Omega(n)$ states.

Related work. Let us emphasize related results which prove bounds on randomized sliding window algorithms. In the seminal paper of Datar et al. [6], where the sliding window model was introduced, the authors prove that the number of 1's in a 0/1-sliding window of size n can

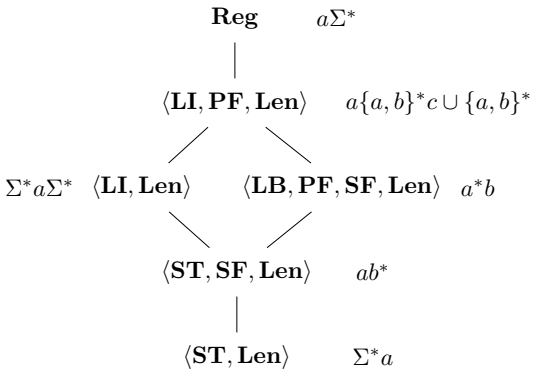


Figure 2 Examples for all occurring language classes where $\Sigma = \{a, b, c\}$.

be maintained in space $\mathcal{O}(\frac{1}{\epsilon} \cdot \log^2 n)$ if one allows a multiplicative error of $1 \pm \epsilon$. Furthermore, they proved a matching lower bound for both deterministic and randomized algorithms. Ben-Basat et al. [3] present a sliding window algorithm for the related problem of approximating counts with an additive error, and present a deterministic and randomized lower space bound. We remark that both papers [3, 6] use a stronger notion of correctness: A randomized sliding window algorithm must have the property that for every input w, the probability that the algorithm produces some incorrect output while reading w is bounded by $1/3$. The authors utilize Yao's minimax principle to lift the deterministic lower bound to a randomized lower bound [13]. Arasu et al. [2] study the problem of maintaining approximate frequency counts and quantiles over sliding windows. They present a deterministic algorithm and a randomized algorithm with improved space complexity using a simple sampling technique. Chan et al. [5] present a randomized sliding window algorithm for a certain measure of monotonicity. Furthermore, they present randomized lower bounds using communication complexity. The randomized algorithm analyzed in both papers comply with our standard definition of correctness with failure ratio 0. Further references can be found in [1, 4].

Finally let us mention the study of the communication complexity of regular languages by Tesson and Thérien [17], where a trichotomy (resp., quatrochotomy) for the deterministic (resp., randomized) communication complexity was shown. These results resemble our results, but the language classes that appear in [17] are different from the classes in our results.

2 Preliminaries

For integers $i, j \in \mathbb{N}$ let $[i, j] = \{k \in \mathbb{N} \colon i \leq k \leq j\}$. The set of all words over a finite alphabet Σ is denoted by Σ^*. The empty word is denoted by ε whereas error probabilities are denoted by the lunate epsilon ϵ. The sets of words over Σ of length exactly, at most and at least n are denoted by Σ^n, $\Sigma^{\leq n}$ and $\Sigma^{\geq n}$, respectively. Consider a word $w = a_1 a_2 \cdots a_m$. The *reversal* of w is defined as $w^{\mathsf{R}} = a_m \cdots a_2 a_1$, and for a language L we set $L^{\mathsf{R}} = \{w^{\mathsf{R}} \colon w \in L\}$. For a non-empty interval $[i, j] \subseteq [1, m]$ we define $w[i, j] = a_i a_{i+1} \cdots a_j$. If $i > j$ we set $w[i, j] = \varepsilon$. A *prefix* of w is a word of the form $w[1, i]$ for some $0 \leq i \leq m$; a *suffix* of w is a word of the form $w[i, m]$ for some $1 \leq i \leq m + 1$. A language $L \subseteq \Sigma^*$ is *prefix-free* (resp., *suffix-free*) if there are no two words $x, y \in L$ with $x \neq y$ and x is a prefix (resp., suffix) of y. A language is *bifix-free* if it is both prefix- and suffix-free.

Automata and regular languages. For general background in automata theory see [10]. A *deterministic finite automaton* (DFA) $A = (Q, \Sigma, q_0, \delta, F)$ consists of a finite set of states Q, a finite alphabet Σ, an initial state $q_0 \in Q$, a transition function $\delta\colon Q \times \Sigma \to Q$ and a set of final states $F \subseteq Q$. We inductively extend δ to a function $\delta\colon Q \times \Sigma^* \to Q$ as usual. If $P \subseteq Q$ is a set of states, then $L(A, P) = \{w \in \Sigma^*\colon \delta(q_0, w) \in P\}$. The language accepted by A is $L(A) = L(A, F)$. A language is *regular* if it is accepted by a DFA. Classes of languages are denoted by boldfaced letters. In this paper we will deal with the following language classes:

- **Reg**: the class of all *regular languages*.
- **Len**: the class of *regular length languages*, i.e., regular languages L such that for all $n \in \mathbb{N}$ we have $\Sigma^n \subseteq L$ or $\Sigma^n \cap L = \emptyset$.
- **LI** (resp., **RI**): the class of *regular left (resp., right) ideals*, i.e., languages of the form $\Sigma^* L$ (resp., $L\Sigma^*$) where L is regular.
- **ST** (resp., **PT**): the class of *suffix testable (resp., prefix testable) languages*, i.e., finite Boolean combinations of languages $\Sigma^* w$ (resp., $w\Sigma^*$) where $w \in \Sigma^*$.
- **SF** (resp., **PF**): the class of *regular suffix-free (resp., prefix-free) languages*
- **LB** (resp., **RB**): the class of *left ideals (resp., right ideals) generated by regular bifix-free languages*, i.e., languages of the form $\Sigma^* L$ (resp., $L\Sigma^*$) where $L \in \mathbf{PF} \cap \mathbf{SF}$.[1]

It is easy to see that every finite language is prefix and suffix testable. Moreover, prefix testable and suffix testable languages are regular. If $\mathbf{A}_1, \ldots, \mathbf{A}_n$ are language classes, then $\langle \mathbf{A}_1, \ldots, \mathbf{A}_n \rangle$ denotes the smallest boolean-closed class which contains $\bigcup_{i=1}^n \mathbf{A}_i$.

Probabilistic automata. In the following we introduce probabilistic automata [14, 15] as a model for randomized streaming algorithms. A *probabilistic automaton* $R = (Q, \Sigma, \iota, \rho, F)$ consists of a (possibly infinite) set of states Q, an alphabet Σ, an initial state distribution $\iota\colon Q \to \{r \in \mathbb{R}\colon 0 \leq r \leq 1\}$, a transition probability function $\rho\colon Q \times \Sigma \times Q \to \{r \in \mathbb{R}\colon 0 \leq r \leq 1\}$ and set of final states $F \subseteq Q$ such that $\sum_{q \in Q} \iota(q) = 1$ and $\sum_{q \in Q} \rho(p, a, q) = 1$ for all $p \in Q$, $a \in \Sigma$. If ι and ρ map into $\{0, 1\}$, then R is a *deterministic automaton*. A *run* on a word $a_1 \cdots a_m \in \Sigma^*$ in R is a sequence $\pi = (q_0, a_1, q_1, a_2, \ldots, a_m, q_m)$ where $q_0, \ldots, q_m \in Q$ and $\rho(q_{i-1}, a_i, q_i) > 0$ for all $1 \leq i \leq m$. Given such a run π in R we define $\rho_\iota(\pi) = \iota(q_0) \cdot \prod_{i=1}^n \rho(q_{i-1}, a_i, q_i)$. For each $w \in \Sigma^*$ the function ρ_ι is a probability distribution on the set $\mathrm{Runs}(w)$ of all runs of $\mathcal{R}$ on w.

3 Randomized streaming and sliding window algorithms

A *randomized streaming algorithm* (R, enc) consists of a probabilistic automaton $R = (Q, \Sigma, \iota, \rho, F)$ as above and an injective function $\mathrm{enc}\colon Q \to \{0, 1\}^*$. Usually, we will only refer to the underlying automaton R. If R is deterministic, we speak of a *deterministic streaming algorithm*. The maximum number of bits stored during a run $\pi = (q_0, a_1, \ldots, a_m, q_m)$ is denoted by $\mathrm{space}(R, \pi)$, i.e., $\mathrm{space}(R, \pi) = \max\{|\mathrm{enc}(q_i)| : 0 \leq i \leq m\}$. The worst case space complexity of R on w is $\mathrm{space}(R, w) = \max\{\mathrm{space}(R, \pi) : \pi \in \mathrm{Runs}(w), \rho_\iota(\pi) > 0\}$. A run $\pi = (q_0, a_1, \ldots, a_m, q_m)$ is *correct* with respect to a language $K \subseteq \Sigma^*$ if $q_m \in F \Leftrightarrow a_1 \cdots a_m \in K$ holds. The *error probability* of R on w for K is

$$\epsilon(R, w, K) = \sum \{\rho_\iota(\pi) : \pi \in \mathrm{Runs}(w) \text{ is not correct with respect to } K\}.$$

[1] or equivalently, the class of all left ideals generated by regular prefix-free languages. Since our proofs related to **LB** yield decompositions of the form $\Sigma^* L$ with L bifix-free, we decided to define **LB** as above.

Given an error bound $0 \leq \epsilon \leq 1$, the *failure ratio* of R on w is defined as

$$\phi(R, w, K, \epsilon) = \frac{1}{m+1}|\{t \in [0, m] : \epsilon(R, a_1 a_2 \cdots a_t, K) > \epsilon\}|.$$

For a window length $n \geq 0$ and a stream $x \in \Sigma^*$ we define $\text{last}_n(x)$ to be the suffix of $\square^n x$ of length n where $\square \in \Sigma$ is a fixed alphabet symbol. The word $\text{last}_n(\varepsilon) = \square^n$ is the initial window content. Given a language $L \subseteq \Sigma^*$ and a window size $n \geq 0$ we define the language

$$L_n = \{x \in \Sigma^* \colon \text{last}_n(x) \in L\}. \tag{1}$$

A *randomized sliding window algorithm* (*randomized SWA* for short) is a sequence $\mathcal{R} = (R_n)_{n \geq 0}$ of randomized streaming algorithms R_n over the same alphabet Σ. If every R_n is deterministic, we speak of a *deterministic SWA*. The *space complexity* of the randomized SWA $\mathcal{R} = (R_n)_{n \geq 0}$ is the function $f(\mathcal{R}, n) = \sup\{\text{space}(R_n, w) : w \in \Sigma^*\} \in \mathbb{N} \cup \{\infty\}$. Clearly, if R_n is finite, then one can always find a state encoding such that $f(\mathcal{R}, n) = \lfloor \log_2 |R_n| \rfloor$.

▶ **Definition 3.1.** *Let $0 \leq \epsilon \leq 1$ and $0 \leq \phi \leq 1$. A randomized SWA $\mathcal{R} = (R_n)_{n \geq 0}$ is* (ϵ, ϕ)-correct *for a language $L \subseteq \Sigma^*$ if $\phi(R_n, w, L_n, \epsilon) \leq \phi$ for all $n \geq 0$ and $w \in \Sigma^{\geq n}$. The number ϵ is the* error probability *and ϕ is the* failure ratio *of $\mathcal{R}$ (with respect to ϵ).*

Definition 3.1 also makes sense in the special case that $\mathcal{R}$ is deterministic and $\epsilon = 0$. A $(0, \phi)$-correct deterministic SWA $\mathcal{R} = (R_n)_{n \geq 0}$ for L has the property that R_n produces at most $\phi \cdot (m+1)$ many incorrect answers when running on any input word of length $m \geq n$.

We will set the error probability to $\epsilon = 1/3$, which is justified by the following lemma (that follows from a standard Chernoff bound).

▶ **Lemma 3.2.** *Let $L \subseteq \Sigma^*$, $0 < \epsilon' < \epsilon < \frac{1}{2}$ and $0 \leq \phi \leq 1$. Given a randomized SWA $\mathcal{R}$ which is (ϵ, ϕ)-correct for L, one can construct a randomized SWA $\mathcal{R}'$ which is (ϵ', ϕ)-correct for L such that $f(\mathcal{R}', n) \leq \ln(\frac{1}{\epsilon'}) \cdot \frac{1}{\text{poly}(\epsilon)} \cdot f(\mathcal{R}, n)$.*

▶ **Definition 3.3.** *Let $L \subseteq \Sigma^*$ be a language and $0 \leq \phi \leq 1$.*

- *A* randomized SWA for L *with failure ratio ϕ is a randomized SWA which is* $(1/3, \phi)$-correct for L. *If moreover $\phi = 0$, then we speak of a* randomized SWA for L.
- *A* deterministic SWA for L *with failure ratio ϕ is a deterministic SWA which is* $(0, \phi)$-correct for L. *If moreover $\phi = 0$, then we speak of a* deterministic SWA for L.

We only consider randomized SWAs $\mathcal{R} = (R_n)_{n \geq 0}$ where every R_n has a finite state set Q_n. This is justified by the fact that for every language L and every n the language L_n from (1) is regular and hence can be accepted by a DFA. The space-optimal deterministic SWA for a language L therefore consists of the minimal DFA for L_n for every $n \geq 0$. For a fixed error probability $\epsilon < 1/2$ a space-optimal randomized SWA for L consists of a minimal probabilistic finite automaton for L_n which accepts a word w with probability at least $1 - \epsilon$ if $w \in L_n$ and accepts w with probability at most ϵ if $w \notin L_n$. By a result of Rabin [15] such a probabilistic finite automaton can be transformed into an equivalent DFA with an exponential blow-up. Hence, we get:

▶ **Lemma 3.4.** *Let $\mathcal{R}$ be a randomized SWA for the language L. Then, there exists a deterministic SWA $\mathcal{D}$ for L such that $f(\mathcal{D}, n) \in \mathcal{O}(2^{f(\mathcal{R}, n)})$.*

4 Upper bounds

In this section we prove the upper bounds in Theorem 1.1, 1.2 and 1.3. We use the simple fact that space complexity classes in the sliding window model are Boolean-closed:

▶ **Lemma 4.1.** *Let L be a Boolean combination of languages $L_1, \dots, L_k$. For each $i \in [1,k]$ let $\mathcal{R}_i$ be a randomized SWA for L_i with failure ratio ϕ_i. Then L has a SWA $\mathcal{R}$ with failure ratio $\sum_{i=1}^k \phi_i$ and $f(\mathcal{R},n) = \mathcal{O}(\sum_{i=1}^k f(\mathcal{R}_i,n))$.*

4.1 The Bernoulli algorithm

In this section, we introduce a randomized SWA that will be used for the proof of the upper bounds (3) from Theorem 1.1 and (1) from Theorem 1.2. The idea is based on the algorithm from [7], which stores *path summaries* in the reversal DFA. Consider a regular language $L \subseteq \Sigma^*$ and let $A = (Q, \Sigma, q_0, \delta, F)$ be a DFA for L^{R}. Observe that if L is a left-ideal then any run in A switches from $Q \setminus F$ to F at most once; if L is suffix-free then any run in A visits F at most once. For a stream $w \in \Sigma^*$ define the function $\ell_w \colon Q \to \mathbb{N} \cup \{\infty\}$ by

$$\ell_w(q) = \inf\{k \in \mathbb{N} : \delta(q, \mathrm{last}_k(w)^{\mathsf{R}}) \in F\}, \tag{2}$$

where we set $\inf(\emptyset) = \infty$. By the observations above we know:

- If L is a left ideal, then $\mathrm{last}_n(w) \in L$ if and only if $\ell_w(q_0) \leq n$.
- If L is suffix-free, then $\mathrm{last}_n(w) \in L$ if and only if $\ell_w(q_0) = n$.

One can define a deterministic SWA which stores the function ℓ_w on input stream w. If a symbol $a \in \Sigma$ is read, we can determine ℓ_{wa} from ℓ_w: If $q \in F$, then $\ell_{wa}(q) = 0$. Otherwise $\ell_{wa}(q) = 1 + \ell_w(\delta(q,a))$ where $1 + \infty = \infty$.

Using a Bernoulli random variable, we define a randomized approximation of the above deterministic SWA to reduce the space complexity to $\mathcal{O}(1)$. Let $\beta \colon \mathbb{N} \to \mathbb{R}$ be a function such that for some n_0, $0 \leq \beta(n) \leq 1$ for all $n \geq n_0$, which controls the Bernoulli random variable and will later be instantiated by concrete functions. We define the following constant-space randomized SWA $\mathcal{B} = (B_n)_{n \geq 0}$ (which depends on the language L, the DFA A and the function β), which we call the Bernoulli algorithm. If $n < n_0$ let B_n be the trivial deterministic streaming algorithm for L_n. For $n \geq n_0$ the algorithm B_n stores a Boolean flag for each state in form of a function $b \colon Q \to \{0,1\}$. All flags $b(q)$ for $q \in F$ are fixed to 1 forever. For all other states $q \in Q \setminus F$ we define the initial value of the flag $b(q)$ as follows, where $\ell = \ell_\varepsilon(q) \in \mathbb{N} \cup \{\infty\}$:

$$b(q) := \begin{cases} 0 \text{ with probability } 1 - (1-\beta(n))^\ell \\ 1 \text{ with probability } (1-\beta(n))^\ell . \end{cases} \tag{3}$$

Here we set $x^\infty = 0$ for $0 \leq x < 1$. For all states $q \in Q \setminus F$ we do the following upon arrival of a symbol $a \in \Sigma$:

$$b(q) := \begin{cases} 0 & \text{with probability } \beta(n) \\ b(\delta(q,a)) & \text{with probability } 1 - \beta(n) \end{cases} \tag{4}$$

The algorithm accepts if and only if $b(q_0) = 1$. An induction on $|w|$ shows:

▶ **Lemma 4.2.** *For all $n \geq n_0$ and $w \in \Sigma^*$ we have $\Pr[B_n \text{ accepts } w] = (1-\beta(n))^{\ell_w(q_0)}$.*

4.2 Randomized SWAs with failure ratio zero

In this section we present a $\mathcal{O}(\log\log n)$ space algorithm for suffix-free regular languages L. Since languages in **ST** and **Len** have constant space deterministic SWAs (Theorem 1.1(1)) and the space complexity classes are closed under Boolean combinations by Lemma 4.1, this implies Theorem 1.1(3).

Let $L \in \mathbf{SF}$ and $A = (Q, \Sigma, q_0, \delta, F)$ be a DFA for $L^{\mathsf{R}} \in \mathbf{PF}$. Since the case $L = \emptyset$ is trivial, we can assume that A contains at least one final state which is reachable from q_0. Furthermore, since L^{R} is prefix-free, any run in A from q_0 contains at most one final state. Therefore, we can assume that F contains exactly one final state q_F, and all outgoing transitions from q_F lead to a sink state.

Recall the function $\ell_w \colon Q \to \mathbb{N} \cup \{\infty\}$ defined in (2). Notice that $\mathrm{last}_n(w) \in L$ if and only if $\ell_w(q_0) = n$ for all $w \in \Sigma^*$. Our randomized SWA $\mathcal{R} = (R_n)_{n \geq 0}$ consists of two parts: Firstly, we take the Bernoulli algorithm $\mathcal{B} = (B_n)_{n \geq 0}$ from Section 4.1 for the function $\beta(n) = 1/(2n)$ and $n_0 = 1$. From Lemma 4.2 we know that B_n accepts a word $w \in \Sigma^*$ with probability $(1 - 1/(2n))^{\ell_w(q_0)}$. Secondly, we simultaneously run a modulo-counting algorithm M_n. Let p_i be the i-th prime number and let $s(m)$ be the product of all prime numbers $\leq m$. It is known that $\ln(s(m)) > m \cdot (1 - 1/\ln m)$ for $m \geq 41$ [16, 3.16] and $p_i < i \cdot (\ln i + \ln \ln i)$ for $i \geq 6$ [16, 3.13]. Let k be the first natural number such that $\prod_{i=1}^{k} p_i \geq n$. By the above bounds we get $k \in \mathcal{O}(\log n)$ and $p_{3k} \in \mathcal{O}(\log n \cdot \log \log n)$. The algorithm M_n initially picks a random prime $p \in \{p_1, \ldots, p_{3k}\}$, which is stored throughout the run using $\mathcal{O}(\log \log n)$ bits. Then, after reading $w \in \Sigma^*$, M_n stores for every $q \in Q$ a bit telling whether $\ell_w(q) < \infty$ and, if the latter holds, the values $\ell_w(q) \bmod p$ using $\mathcal{O}(|Q| \cdot \log \log n)$ bits. The algorithm accepts if and only if $\ell_w(q_0) \equiv n \bmod p$.

The combined algorithm R_n accepts if and only if both B_n and T_n accept. Let us bound the error probability on an input stream $w \in \Sigma^*$ with $\ell = \ell_w(q_0)$.

Case 1. $\ell = n$, i.e., $\mathrm{last}_n(w) \in L$. Then M_n accepts w with probability 1. Moreover, B_n accepts w with probability $(1 - 1/(2n))^n \geq 0.6$ for $n \geq 12$ (note that $(1 - 1/(2n))^n$ converges to $1/\sqrt{e} \approx 0.60653$ from below). Hence, R_n accepts with probability at least 0.6.

Case 2. $\ell \geq 2n$ and hence $\mathrm{last}_n(w) \notin L$. Then B_n rejects with probability $1 - (1 - 1/(2n))^{\ell} \geq 1 - (1 - (1/2n))^{2n} \geq 1 - 1/e \geq 0.6$. Here, we use the well-known inequality $(1 - 1/y)^y \leq e^{-1}$ for all $y \geq 1$. Hence, R_n also rejects with probability at least 0.6.

Case 3. $\ell < 2n$ and $\ell \neq n$, and thus $\mathrm{last}_n(w) \notin L$. Since $\ell - n \in [-n, n]$ and any product of at least $k + 1$ pairwise distinct primes exceeds n, the number $\ell - n \neq 0$ has at most k prime factors. Therefore, M_n (and thus R_n) rejects with probability at least 2/3.

4.3 SWAs with arbitrarily small non-zero failure ratio

In this section we sketch the proofs of Theorem 1.2(1) and Theorem 1.3(1). We focus on the main base case $L \in \mathbf{LI}$ from Theorem 1.2(1).

Let $L \subseteq \Sigma^*$ be a regular left ideal. Let $A = (Q, \Sigma, q_0, \delta, F)$ be the minimal DFA for L^{R}. Since the case $L = \emptyset$ is trivial, we can assume that $L \neq \emptyset$. It is easy to see that F contains a single state q_F from which all outgoing transitions lead back to q_F. Recall the function $\ell_w \colon Q \to \mathbb{N} \cup \{\infty\}$ defined in (2). Since L is a left ideal, we have: $\mathrm{last}_n(w) \in L$ if and only if $\ell_w(q_0) \leq n$: The following lemma says that the portion of prefixes of an input stream, where $\ell_w(q_0)$ is close to n, is small:

▶ **Lemma 4.3.** *Let $0 < \xi < 1$. Let $n \geq 0$ be a window size and $w \in \Sigma^{\geq n}$ be an input stream. Then the number of prefixes v of w such that $\lceil \xi n \rceil \leq \ell_v(q_0) \leq n$ is at most*

$$\frac{(1 - \xi + \frac{1}{n}) \cdot |Q|}{\xi} \cdot (|w| + 1 + \xi n). \tag{5}$$

Proof. Let us say that a prefix v of w is a *hit*, if $\lceil \xi n \rceil \leq \ell_v(q_0) \leq n$. Consider an interval $I = [i, i']$ with $0 \leq i \leq i' \leq |w|$ and $i' - i \leq \lceil \xi n \rceil$. With each position $j \in I$ we associate the

prefix $w[1,j]$. We claim that $V := \{w[1,j] \colon j \in I\}$ contains at most $|Q| \cdot (n - \lceil \xi n \rceil + 1)$ hits. Let us assume the contrary. Since $\lceil \xi n \rceil \leq \ell_v(q_0) \leq n$ for every hit v, and the interval $[\lceil \xi n \rceil, n]$ contains $n - \lceil \xi n \rceil + 1$ many different values, there is a subset $U \subseteq V$ and some $\ell \in [\lceil \xi n \rceil, n]$ such that (i) $|U| > |Q|$ and (ii) $\ell_v(q_0) = \ell$ for all $v \in U$. Let $U = \{w[1,j_1], w[1,j_2], \ldots, w[1,j_k]\}$, where $k > |Q|$. Consider the words $u_1 = \mathrm{last}_\ell(w[1,j_1]), u_2 = \mathrm{last}_\ell(w[1,j_2]), \ldots, u_k = \mathrm{last}_\ell(w[1,j_k])$. Since $j_k - j_1 \leq \lceil \xi n \rceil$ and $\ell \geq \lceil \xi n \rceil$, we have $\ell - j_k + j_1 \geq 0$. Hence, we can consider the words $v_1 = \mathrm{last}_{\ell - j_k + j_1}(w[1,j_1]), v_2 = \mathrm{last}_{\ell - j_k + j_2}(w[1,j_2]), \ldots, v_k = \mathrm{last}_\ell(w[1,j_k])$. Clearly, v_j is a suffix of u_j. Moreover, the words v_j all start in the same position of $\square^n w$, i.e., every v_j is a prefix of $v_{j'}$ for $j \leq j'$. Consider now the state $q_j = \xi(q_0, v_j^{\mathsf{R}})$ for $1 \leq j \leq k$. Since $k > |Q|$ there exist $j < j'$ such that $q_j = q_{j'}$. But this would imply that $\ell_{v_j}(q_0) < \ell_{v_{j'}}(q_0)$, which contradicts $\ell_{v_j}(q_0) = \ell = \ell_{v_{j'}}(q_0)$. This shows the above claim.

Now we can finish the proof of the lemma: We divide the interval $[0, |w|]$ into intervals of size $\lceil \xi n \rceil + 1$ and one last interval of possibly shorter length. This yields $\lceil (|w|+1)/(\lceil \xi n \rceil + 1) \rceil$ many intervals. In each of these intervals we find at most $|Q| \cdot (n - \lceil \xi n \rceil + 1)$ many hits by the above claim. Hence, the total number of hits is bounded by

$$\begin{aligned} \left\lceil \frac{|w|+1}{\lceil \xi n \rceil + 1} \right\rceil \cdot |Q| \cdot (n - \lceil \xi n \rceil + 1) &\leq \left(\frac{|w|+1}{\xi n} + 1 \right) \cdot |Q| \cdot (n - \xi n + 1) \\ &= (|w| + 1 + \xi n) \cdot |Q| \cdot \frac{1 - \xi + \frac{1}{n}}{\xi}. \end{aligned}$$

This concludes the proof of the lemma. ◀

We now consider the Bernoulli algorithm $\mathcal{B}^\epsilon = (B_n^\epsilon)_{n \geq 0}$ for $\beta_\epsilon \colon \mathbb{N} \to \mathbb{R}$ with $\beta_\epsilon(n) = \ln(1/\epsilon)/n$. Note that $0 < \beta_\epsilon(n) \leq 1$ for $n \geq \ln(1/\epsilon)$. With Lemma 4.2 one can show:

▶ **Lemma 4.4.** *For every $0 < \xi < 1$ there exists $0 < \epsilon < 1/2$ and $n_0 \geq 1$ such that for all $n \geq n_0$ the following holds: If $w \in \Sigma^*$ and $\ell_w(q_0) \notin [\lceil \xi n \rceil, n]$, then $\epsilon(B_n^\epsilon, w, L_n) \leq \epsilon$.*

▶ **Theorem 4.5.** *Let L be a regular left ideal and $0 < \phi < 1$. Then L has a randomized SWA $\mathcal{R}$ with $f(\mathcal{R}, n) = \mathcal{O}(1)$ and failure ratio ϕ.*

Proof. Let us fix a failure ratio $0 < \phi < 1$ and let $0 < \xi < 1$, which will be defined later (depending on ϕ). Let ϵ and n_0 be the numbers from Lemma 4.4. Let $\mathcal{B}^\epsilon = (B_n^\epsilon)_{n \geq 0}$ be the randomized SWA described above. Let $n \geq n_0$ be a window size and $w \in \Sigma^{\geq n}$ be an input stream. Consider the set $P(w)$ of all prefixes v of w such that $\ell_v(q_0) \in [\lceil \xi n \rceil, n]$. By Lemma 4.4 the algorithm B_n^ϵ errs on each prefix $v \notin P(w)$ with probability at most ϵ, i.e.,

$$\begin{aligned} \phi(B_n^\epsilon, w, L_n, \epsilon) &\leq \frac{|P(w)|}{|w|+1} \overset{\text{Lemma 4.3}}{\leq} \frac{(1 - \xi + \frac{1}{n}) \cdot |Q|}{\xi} \cdot \left(1 + \frac{\xi n}{|w|+1}\right) \\ &\leq \left(1 - \xi + \frac{1}{n}\right) \cdot |Q| \cdot \left(1 + \frac{1}{\xi}\right). \end{aligned} \tag{6}$$

Note that if ξ converges to 1, then the probability (6) tends towards $2|Q|/n$. Hence we can choose numbers $n_1 \geq n_0$ and $0 < \xi < 1$ such that for all $n \geq n_1$ the probability (6) is smaller than our fixed failure ratio ϕ.

Finally for window sizes $n < n_1$ we can use the optimal deterministic sliding-window algorithms for L and window size n. The space complexity of the resulting algorithm is a constant that depends only on ϕ. ◀

The base case $L \in \mathbf{Len}$ in Theorem 1.2(1) is trivial (there is a constant-space deterministic SWA). Finally, for the case $L \in \mathbf{PF}$, one can show that the constant-space deterministic SWA that always rejects has a failure ratio of $\mathcal{O}(1/n)$ for window length n. This fact also

covers the base case $L \in \mathbf{PF}$ from Theorem 1.3(1), and a similar argument covers the case $L \in \mathbf{SF}$. For the remaining base case $L \in \mathbf{LB}$ in Theorem 1.3(1), one can basically use a DFA for L itself as a sliding window algorithm. One can prove that this algorithm gives only $\mathcal{O}(1)$ incorrect answers in n consecutive windows.

5 Lower bounds

To prove the claimed lower bounds, we apply the same proof strategy in all cases (with one exception). We first show that if a regular language does not belong to the language class under consideration then there exist certain witness words. These words can then be used to apply known lower bounds from randomized one-way communication complexity [11, 12] by deriving a randomized communication protocol from a randomized SWA. This is a standard technique for showing lower bounds for streaming algorithms.

The setting in (one-way) communication complexity is as follows: Alice holds an element $x \in X$ and Bob holds an element $y \in Y$ and they aim to compute $f(x, y)$ according to a randomized one-way protocol P. Alice computes from her input $x \in X$ and from a random string a message and sends it to Bob. Using this message, his input $y \in Y$ and a random string, Bob computes an output bit. The cost of the protocol P is the maximal number of bits sent from Alice to Bob. We say that P computes f if the protocol computes $f(x, y)$ with probability at least $2/3$ for all inputs (x, y). The minimal cost of a protocol which computes f is denoted by $C(f)$.

Due to space constraints we only prove Theorem 1.1(4) in detail and discuss the proofs of the remaining lower bounds only briefly in Section 5.2.

5.1 Proof of point 4 from Theorem 1.1

Let $A = (Q, \Sigma, q_0, \delta, F)$ be a DFA. A state q is *trivial* if $\delta(q, x) \neq q$ for all $x \in \Sigma^+$, otherwise it is *non-trivial*. A pair $(p, q) \in Q \times Q$ of states is called *synchronized* if there exist words $x, y, z \in \Sigma^*$ with $|x| = |y| = |z| \geq 1$ such that $\delta(p, x) = p$, $\delta(p, y) = q$ and $\delta(q, z) = q$. A pair (p, q) is called *reachable* from a state r if p is reachable from r. A state pair (p, q) is called *F-consistent* if either $\{p, q\} \cap F = \emptyset$ or $\{p, q\} \subseteq F$. We remark that synchronized state pairs have no connection to the notion of synchronizing words. A simple pumping argument shows:

▶ **Lemma 5.1.** *A state pair (p, q) is synchronized if and only if p and q are non-trivial and there exists $y \in \Sigma^+$ such that $|Q|!$ divides $|y|$ and $\delta(p, y) = q$.*

Let $Q = T \cup N$ be the partition of the state set into the set T of trivial states and the set N of non-trivial states. A function $\beta\colon \mathbb{N} \to \{0, 1\}$ is *k-periodic* if $\beta(i) = \beta(i + k)$ for all $i \in \mathbb{N}$.

▶ **Lemma 5.2.** *Assume that every synchronized pair in A which is reachable from q_0 is F-consistent. Then for every word $v \in \Sigma^*$ of length at least $|Q|! \cdot (|T| + 1)$ there exists a $|Q|!$-periodic function $\beta_v\colon \mathbb{N} \to \{0, 1\}$ such that the following holds: If $w \in v\Sigma^*$ and $\delta(q_0, w) \in N$, then we have $w \in L$ iff $\beta(|w|) = 1$.*

Proof. Let $v = a_1 a_2 \cdots a_k$ with $k \geq |Q|! \cdot (|T| + 1)$, and consider the run $q_0 \xrightarrow{a_1} \cdots \xrightarrow{a_k} q_k$ of A on v. Clearly, each trivial state can occur at most once in the run. First notice that for each $0 \leq i \leq |Q|! - 1$ at least one of the states in $Q_i = \{q_{i+j|Q|!} : 0 \leq j \leq |T|\}$ is non-trivial because otherwise the set would contain $|T| + 1$ pairwise distinct trivial states. Furthermore, we claim that the non-trivial states in Q_i are either all final or all non-final: Take two non-trivial states $q_{i+j_1|Q|!}$ and $q_{i+j_2|Q|!}$ with $j_1 < j_2$. Since we have a run of length

$(j_2 - j_1)|Q|!$ from $q_{i+j_1|Q|!}$ to $q_{i+j_2|Q|!}$, the states form a synchronized pair by Lemma 5.1. Hence, by assumption the two states are F-consistent. Now define $\beta_v \colon \mathbb{N} \to \{0,1\}$ by

$$\beta_v(m) = \begin{cases} 1 & \text{if the states in } Q_{m \bmod |Q|!} \cap N \text{ are final,} \\ 0 & \text{if the states in } Q_{m \bmod |Q|!} \cap N \text{ are non-final,} \end{cases}$$

which is well-defined by the remarks above. Clearly β_v is $|Q|!$-periodic.

Let $w = a_1 \cdots a_m \in v\Sigma^*$ be a word of length $m \geq k$. The run of A on w prolongs the run on v, say $q_0 \xrightarrow{a_1} \cdots \xrightarrow{a_k} q_k \xrightarrow{a_{k+1}} \cdots \xrightarrow{a_m} q_m$. Assume that $q_m \in N$. As argued above, there is a position $0 \leq i \leq k$ such that $i \equiv m \pmod{|Q|!}$ and $q_i \in N$. Hence (q_i, q_m) is a synchronized pair by Lemma 5.1 which is F-consistent by assumption. Therefore $w \in L$ iff $q_m \in F$ iff $q_i \in F$ iff $\beta_v(|w|) = 1$. ◀

▶ **Lemma 5.3.** *Assume that every synchronized pair in A which is reachable from q_0 is F-consistent. Then $L(A)$ belongs to $\langle \mathbf{PT}, \mathbf{PF}, \mathbf{Len} \rangle$.*

Proof. Let $F_N = N \cap F$ and $F_T = T \cap F$. We decompose L into

$$L = L(A, F_N) \cup \bigcup_{q \in F_T} L(A, \{q\}).$$

First observe that $L(A, \{q\}) \in \mathbf{PF}$ for all $q \in F_T$ because a trivial state q can occur at most once in a run of A. It remains to show that $L(A, F_N)$ belongs to $\langle \mathbf{PT}, \mathbf{PF}, \mathbf{Len} \rangle$. Using the threshold $k = |Q|! \cdot (|T| + 1)$, we distinguish between words of length at most $k-1$ and words of length at least k, and group the latter set by their prefix of length k, i.e.,

$$L(A, F_N) = (L(A, F_N) \cap \Sigma^{\leq k-1}) \cup \bigcup_{v \in \Sigma^k} (L(A, F_N) \cap v\Sigma^*).$$

The first part $L(A, F_N) \cap \Sigma^{\leq k-1}$ is finite and thus prefix testable. To finish the proof, we will show that $L(A, F_N) \cap v\Sigma^* \in \langle \mathbf{PT}, \mathbf{PF}, \mathbf{Len} \rangle$ for each $v \in \Sigma^k$. Let $v \in \Sigma^k$ and let $\beta_v \colon \mathbb{N} \to \{0,1\}$ be the $|Q|!$-periodic function from Lemma 5.2. We know

$$L(A, F_N) \cap v\Sigma^* = (v\Sigma^* \cap \{w \in \Sigma^* : \beta(|w|) = 1\}) \setminus L(A, T).$$

Note that $\{w \in \Sigma^* : \beta(|w|) = 1\} \in \mathbf{Len}$, $v\Sigma^* \in \mathbf{PT}$ and $L(A, T) \in \langle \mathbf{PF} \rangle$. ◀

By applying Lemma 5.3 to the language L^{R}, we obtain:

▶ **Lemma 5.4.** *If $L \in \mathbf{Reg} \setminus \langle \mathbf{ST}, \mathbf{SF}, \mathbf{Len} \rangle$, then there exist $u, x, y, z \in \Sigma^*$ with $|x| = |y| = |z| \geq 1$ such that one of the following cases holds:*

- $x^*u \subseteq L$ *and* $z^*yx^*u \cap L = \emptyset$
- $x^*u \cap L = \emptyset$ *and* $z^*yx^*u \subseteq L$.

We can now conclude the proof of Theorem 1.1(4). Let $L \in \mathbf{Reg} \setminus \langle \mathbf{ST}, \mathbf{SF}, \mathbf{Len} \rangle$. We reduce from the communication problem $\mathrm{GT}_m \colon [1,m]^2 \to \{0,1\}$ where $\mathrm{GT}_m(i,j) = 1$ iff $i > j$. It is known that $C(\mathrm{GT}_m) \in \Theta(\log m)$ [11, Theorem 3.8].

Consider the words $u, x, y, z \in \Sigma^*$ described in Lemma 5.4. Let $\mathcal{R} = (R_n)_{n \geq 0}$ be a randomized SWA for L. Let $m \geq 0$. We describe a randomized one-way protocol P_m for GT_m: Let $i \in [1,m]$ be the input of Alice and $j \in [1,m]$ be the input of Bob. Let $n = |x| \cdot m + |u| \in \Theta(m)$. Alice starts by running the probabilistic automaton R_n on $z^m y x^{m-i}$ using her random bits in order to simulate the random choices of R_n. Afterwards, she sends the encoding of the reached state to Bob. Bob then continues the run of R_n

from the transmitted state with the word $x^j u$. Hence, R_n is simulated on the word $w := z^m y x^{m-i} x^j u = z^m y x^{m-i+j} u$. We have

$$\text{last}_n(w) = \begin{cases} z^{i-1-j} y x^{m-i+j} u, & \text{if } i > j, \\ x^m u, & \text{if } i \leq j. \end{cases}$$

By Lemma 5.4, $\text{last}_n(w)$ belongs to L in exactly one of the two cases $i > j$ and $i \leq j$. Hence Bob can distinguish these two cases with probability at least $2/3$. It follows that the protocol computes GT_m and its cost is bounded by $f(\mathcal{R}, n)$. We have $f(\mathcal{R}, |x| \cdot m + |u|) = f(\mathcal{R}, n) \geq \text{cost}(P_m) \in \Omega(\log m)$ and therefore $f(\mathcal{R}, n) \notin o(\log n)$.

5.2 Remaining lower bounds

The remaining lower bounds in Theorem 1.1–1.3 are shown by reductions from communication problems as well, with one exception:

- For Theorem 1.1(2) we reduce from the communication problem EQ_m, where Alice holds a number $i \in [1, m]$, Bob holds a number $j \in [1, m]$ and Bob has to verify whether $i = j$. It is known that $C(\text{EQ}_m) \in \Theta(\log \log m)$ [12].
- For Theorem 1.1(6) we reduce from the communication problem IDX_m, where Alice holds a bitstring $a_1 \cdots a_m$, Bob holds an index $i \in [1, m]$ and Bob has to output the bit a_i. By [11, Theorem 3.7] we know that $C(\text{IDX}_m) \in \Theta(m)$.
- For Theorem 1.2(2) we reduce from a promise variant of IDX_m. Let $D \subseteq \{0,1\}^m \times [1, m]$ such that for each $x \in \{0,1\}^m$ there exist at least $7/8 \cdot m$ pairs $(x, i) \in D$. We prove that IDX_m still has communication complexity $\Omega(m)$ if the inputs for Alice and Bob are restricted to D. This extends [11, Theorem 3.7] and allows us to incorporate the notion of failure ratio into a communication protocol.
- Theorem 1.3(2) is not shown via a reduction to a communication problem. Instead, we utilize a combinatorial result on the ability of DFAs to count up to a threshold n, modulo a failure ratio ϕ.

6 Further results

The technical report [9] contains several further results that we briefly want to discuss.

In this paper, we only considered randomized SWAs with a two sided error (analogously to the complexity class BPP). Randomized SWAs with a one-sided error (analogously to the class RP) can be motivated by applications, where all "yes" outputs have to be correct, but a small probability for a false negative answer is acceptable. We prove that for every regular language the optimal space bound with respect to randomized SWAs with one-sided error coincides (up to constant factors) with the optimal space bound in the deterministic setting [7, 8] (which was discussed in the introduction).

In the introduction (related work) we remarked that some authors use a stronger correctness notion for randomized SWAs, where for every input w, the probability that the algorithm produces some incorrect output while reading w is bounded by $1/3$. We show that for every approximation problem (where approximation problems are defined by specifying for every input string a set of possible output values) every randomized SWA that fulfills this stronger notion of correctness can be completely derandomized without a space increase.

References

1 Charu C. Aggarwal. *Data Streams - Models and Algorithms*. Springer, 2007.

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 Ran Ben-Basat, Gil Einziger, Roy Friedman, and Yaron Kassner. Efficient summing over sliding windows. In *Proceedings of the 15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016*, volume 53 of *LIPIcs*, pages 11:1–11:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

4 Vladimir Braverman. Sliding window algorithms. In *Encyclopedia of Algorithms*, pages 2006–2011. Springer, 2016.

5 Ho-Leung Chan, Tak Wah Lam, Lap-Kei Lee, Jiangwei Pan, Hing-Fung Ting, and Qin Zhang. Edit distance to monotonicity in sliding windows. In *Proceedings of the 22nd International Symposium on Algorithms and Computation, ISAAC 2011*, volume 7074 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2011.

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

7 Moses Ganardi, Danny Hucke, Daniel König, Markus Lohrey, and Konstantinos Mamouras. Automata theory on sliding windows. In *Proceedings of STACS 2018*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. to appear.

8 Moses Ganardi, Danny Hucke, and Markus Lohrey. Querying regular languages over sliding windows. In *Proceedings of FSTTCS 2016*, volume 65 of *LIPIcs*, pages 18:1–18:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

9 Moses Ganardi, Danny Hucke, and Markus Lohrey. Randomized sliding window algorithms for regular languages. Technical report, arXiv.org, 2018. `https://arxiv.org/abs/1802.07600`.

10 J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison–Wesley, Reading, MA, 1979.

11 Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.

12 Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.

13 Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. In Allen B. Tucker, editor, *The Computer Science and Engineering Handbook*, pages 141–161. CRC Press, 1997.

14 Azaria Paz. *Introduction to Probabilistic Automata*. Academic Press, 1971.

15 Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

16 J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6(1):64–94, 1962.

17 Pascal Tesson and Denis Thérien. Complete classifications for the communication complexity of regular languages. *Theory Comput. Syst.*, 38(2):135–159, 2005. `doi:10.1007/s00224-004-1190-2`.