Sliding window property testing 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

Tatiana Starikovskaya
DI/ENS, PSL Research University, France
tat.starikovskaya@gmail.com

Abstract

We study the problem of recognizing regular languages in a variant of the streaming model of computation, called the sliding window model. In this model, we are given a size of the sliding window n and a stream of symbols. At each time instant, we must decide whether the suffix of length n of the current stream ("the active window") belongs to a given regular language.

Recent works [14, 15] showed that the space complexity of an optimal deterministic sliding window algorithm for this problem is either constant, logarithmic or linear in the window size n and provided natural language theoretic characterizations of the space complexity classes. Subsequently, [16] extended this result to randomized algorithms to show that any such algorithm admits either constant, double logarithmic, logarithmic or linear space complexity.

In this work, we make an important step forward and combine the sliding window model with the property testing setting, which results in ultra-efficient algorithms for all regular languages. Informally, a sliding window property tester must accept the active window if it belongs to the language and reject it if it is far from the language. We show that for every regular language, there is a deterministic sliding window property tester that uses logarithmic space and a randomized sliding window property tester with two-sided error that uses constant space.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Streaming algorithms, approximation algorithms, regular languages

1 Introduction

Regular expression search constitutes an important part of many search engines for biological data or code, such as, for example, Elasticsearch Service[1]. In this paper, we consider the following formalization of this problem. We assume to be given an integer n, a regular language L, and a stream of symbols that we receive one symbol at a time. At each time instant, we have direct access only to the last arrived symbol, and must decide whether the suffix of length n of the current stream ("the active window") belongs to L.

The model described above is a variant of the streaming model and was introduced by Datar et al. [10], where the authors proved that the number of 1's in a 0/1-sliding window of

[1] https://www.elastic.co

size n can be maintained in space $\mathcal{O}(\frac{1}{\epsilon} \cdot \log^2 n)$ if one allows a multiplicative error of $1 \pm \epsilon$. The motivation for this model of computation is that in many streaming applications, data items are outdated after a certain time, and the sliding window setting is a simple way to model this. In general, we aim to avoid storing the window content explicitly, and, instead, to work in considerably smaller space, e.g. polylogarithmic space with respect to the window length. For more details on the sliding window model see [1, Chapter 8].

The study of recognizing regular languages in the sliding window model was commenced in [14, 15]. In [15], Ganardi et al. showed that for every regular language L the optimal space bound for a deterministic sliding window algorithm is either constant, logarithmic or linear in the window size n. In [14], Ganardi et al. gave characterizations for these space classes. More formally, they showed that a regular language has a deterministic sliding window algorithm with space $\mathcal{O}(\log n)$ (resp., $\mathcal{O}(1)$) if and only if it is a Boolean combination of so-called regular left-ideals and regular length languages (resp., suffix-testable languages and regular length languages). A subsequent work [16] studied the space complexity of randomized sliding window algorithms for regular languages. It was shown that for every regular language L the optimal space bound of randomized sliding window algorithm is $\mathcal{O}(1)$, $\mathcal{O}(\log \log n)$, $\mathcal{O}(\log n)$, or $\mathcal{O}(n)$. Moreover, complete characterizations of these space classes were provided.

1.1 Our results

Previous study implies that even simple languages require linear space in the sliding window model, which gives the motivation to seek for novel approaches in order to achieve efficient algorithms for all regular languages. We take our inspiration from the property testing model introduced by Goldreich et. al [21]. In this model, the task is to decide whether the input has a particular property P, or is "far" from any input satisfying it. For a function $\gamma : \mathbb{N} \to \mathbb{R}_{\geq 0}$, we say that a word w of length n is γ-far from satisfying P, if the Hamming distance between w and any word w' satisfying P is at least $\gamma(n)$. We will call the function $\gamma(n)$ the Hamming gap of the tester. We must make the decision by inspecting as few symbols of the input as possible, and the time complexity of the algorithm is defined to be equal to the number of inspected symbols. The motivation is that when working with large-scale data, accessing a data item is a very time-expensive operation. The membership problem for a regular language in the property testing model was studied by Alon et al. [2] who showed that for every regular language L and every constant $\epsilon > 0$, there is a property tester with Hamming gap $\gamma(n) = \epsilon n$ for deciding membership in L that can make the decision by inspecting a random constant-size sample of symbols of the input word.

In this work, we introduce a class of algorithms called *sliding window property testers*. Informally, at each time moment, a sliding window property tester must accept if the active window has the property P and reject if it is far from satisfying P. The space complexity of a sliding window property tester is defined to be all the space used, including the space we need to store information about the input. We consider deterministic sliding window property testers and randomized sliding window property testers with one-sided and two-sided errors (for a formal definition, see Section 2). A similar but simpler model of streaming property testers, where the whole stream is considered, was introduced by Feigenbaum et al. [11]. François et al. [12] continued the study of this model in the context of language membership problems and came up with a streaming property tester for visibly pushdown languages that uses polylogarithmic space. Note that deciding membership in a regular languages becomes trivial in this model (where the active window is the whole stream): one can simply simulate a deterministic finite automaton on the stream. What makes the sliding window model more

difficult is the fact that the oldest symbol in the active window expires in the next step.

While at first sight the only connection between property testers and sliding window property testers is that we must accept the input if it satisfies P and reject if it is far from satisfying P, there is, in fact, a deeper link. In particular, the above mentioned result of Alon et al. [2] combined with an optimal sampling algorithm for sliding windows [4], immediately yields a $\mathcal{O}(\log n)$-space, two-sided error sliding window property tester with Hamming gap $\gamma(n) = \epsilon n$ for every regular language. We will improve on this observation. Our main contribution are tight complexity bounds for each of the following classes of sliding window property testers for regular languages: deterministic sliding window property testers and randomized sliding window property testers with one-sided and two-sided error.

Deterministic sliding window property testers. We call a language L *trivial*, if for some constant $c > 0$ the following holds: For every word $w \in \Sigma^*$ such that L contains a word of length $|w|$, the Hamming distance from w to L is at most c. Every trivial regular language has a constant-space deterministic sliding window property tester with constant Hamming gap (Theorem 2.4). For generic regular languages, we show a deterministic sliding window property tester with constant Hamming gap that uses $\mathcal{O}(\log n)$ space. This is particularly surprising, because for Hamming gap zero (i.e., the exact case) [16] showed a space lower bound of $\Omega(n)$ for generic regular languages. In other words, a constant Hamming gap allows an exponential space improvement. We also show that for *non-trivial* regular languages, $\mathcal{O}(\log n)$ space is the best one can hope to achieve, even for Hamming gap $\gamma(n) = \epsilon n$ (Theorem 3.2).

Randomized sliding window property testers with two-sided error. Next, we show that for every regular language, there is a randomized sliding window property tester with Hamming gap $\gamma(n) = \epsilon n$ and two-sided error that uses constant space (Theorem 3.3). This is an optimal bound and a considerable improvement compared to the tester that can be obtained by combining the property tester of Alon et al. [2] and an optimal sampling algorithm for sliding windows [4].

Randomized sliding window property testers with one-sided error. While our randomized sliding window property tester with two-sided error is optimal, we believe that a two-sided error is a very strong relaxation and to be avoided in some applications. To this end, we study the one-sided error randomized setting. The general landscape for this setting is the most complex: In Theorems 3.4 and 3.5, we show that for every regular language L, the space complexity of an optimal randomized sliding window property tester with one-sided error is either $\mathcal{O}(1)$, $\mathcal{O}(\log \log n)$, or $\mathcal{O}(\log n)$, and we provide characterizations of these complexity classes.

In order to show our upper bound results, we demonstrate novel combinatorial properties of automata and regular languages and develop new streaming techniques, such as probabilistic counters, which can be of interest on their own. To show the lower bound results, we introduce a new methodology, which could potentially simplify further establishments of lower bounds in string processing tasks in the streaming setting: Namely, we view the testers as nondeterministic automata, and study their behaviour.

1.2 Related work

The results above assume that the regular language admits a constant-space description and we will follow the same assumption in this work. Currently, there are few studies on the dependency of the complexity of sliding window algorithms on the size of the language description. On the negative side, Ganardi et al. [14] showed that there are

regular languages such that any sliding window algorithm that achieves logarithmic space (in the window size) depends exponentially on the automata size. On the positive side, there is an extensive study of the pattern matching problem and its variants that gives sub-exponential upper bounds for a class of (very simple) regular languages. In this problem, we are given a pattern and a streaming text T, and at each moment we must decide if the active window is equal to the pattern. This problem and its generalisations have been studied in [5, 6, 7, 8, 9, 18, 19, 20, 29, 31].

Similar to regular languages, we can ask whether the current active window belongs to a given context-free language. This question was studied in [3, 23, 24, 26] for the model where the active window is the complete stream and in [13, 17] for the sliding-window model.

2 Sliding window property tester

We fix a finite alphabet Σ for the rest of the paper. We denote by Σ^* the set of all words over Σ and by Σ^n the set of words over Σ of length n. The empty word is denoted by λ. Let w be a word. We say that v is a *prefix (suffix)* of w if $w = xv$ ($w = vx$) for some word x. We say that v is a *factor* of w if $w = xvy$ for some words x, y. The *Hamming distance* between two words $u = a_1 \cdots a_n$ and $v = b_1 \cdots b_n$ of equal length is the number of positions where u and v differ, i.e. $\mathrm{dist}(u, v) = |\{i : a_i \neq b_i\}|$. The distance of a word u to a language L is defined as $\mathrm{dist}(u, L) = \inf\{\mathrm{dist}(u, v) : v \in L\} \in \mathbb{N} \cup \{\infty\}$.

A *deterministic finite automaton* (DFA) is a tuple $A = (Q, \Sigma, q_0, \delta, F)$ where Q is a finite set of states, Σ is the input alphabet, q_0 is the initial state, $\delta : Q \times \Sigma \to Q$ is the transition mapping and $F \subseteq Q$ is the set of final states. We extend δ to a mapping $\delta : Q \times \Sigma^* \to Q$ inductively in the usual way: $\delta(q, \lambda) = q$ and $\delta(q, aw) = \delta(\delta(q, a), w)$. The language accepted by A is $L(A) = \{w \in \Sigma^* : \delta(q_0, w) \in F\}$. A language is *regular* if it is accepted by a DFA. For more background in automata theory see [22].

A *stream* is a word $a_1 a_2 \cdots a_m$ over Σ. A *sliding window algorithm* is a family $\mathcal{A} = (A_n)_{n \geq 0}$ of streaming algorithms. Given a window size $n \in \mathbb{N}$ and an input stream $a_1 a_2 \cdots a_m \in \Sigma^*$ the algorithm A_n reads the stream symbol by symbol from left to right and thereby updates its memory content. After reading a prefix $a_1 \cdots a_t$ ($0 \leq t \leq m$) the algorithm is required to compute an output value that depends on the *active window* $\mathrm{last}_n(a_1 \cdots a_t) = a_{t-n+1} \cdots a_t$ at time t. For convenience, for $i < 0$ we define $a_i = \square$ where $\square \in \Sigma$ is an arbitrary fixed symbol. In other words, we assume an initial window $\square^n$ that is active at time $t = 0$. We consider *deterministic sliding window algorithms* (where every A_n can be viewed as a DFA) and *randomized sliding window algorithms* (where every A_n can be viewed as a probabilistic finite automaton in the sense of Rabin [30]). In the latter case, A_n updates in each step its memory content according to a probability distribution that depends on the current memory content and the current input symbol. Let $\gamma : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be a function such that $\gamma(n) \leq n$ for all $n \in \mathbb{N}$, let α, β be probabilities, and let $L \subseteq \Sigma^*$ be a language.

▶ **Definition 2.1.** *A* deterministic sliding window (property) tester *for L with Hamming gap $\gamma(n)$ is a deterministic sliding window algorithm $\mathcal{A} = (A_n)_{n \geq 0}$ such that for every input stream $w \in \Sigma^*$ and every window size n the following properties hold:*

- *if* $\mathrm{last}_n(w) \in L$, *then A_n accepts;*
- *if* $\mathrm{dist}(\mathrm{last}_n(w), L) > \gamma(n)$, *then A_n rejects.*

▶ **Definition 2.2.** *A* randomized sliding window (property) tester *for L with Hamming gap $\gamma(n)$ and error (α, β) is a randomized sliding window algorithm $\mathcal{A} = (A_n)_{n \geq 0}$ such that for every input stream $w \in \Sigma^*$ and every window size n the following properties hold:*

- *if* $\mathrm{last}_n(w) \in L$*, then* A_n *accepts with probability at least* $1-\alpha$*;*
- *if* $\mathrm{dist}(\mathrm{last}_n(w), L) > \gamma(n)$*, then* A_n *rejects with probability at least* $1-\beta$*.*

We say that $\mathcal{A}$ *has* one-sided error *if* $\mathcal{A}$ *has error* $(0, 1/2)$ *and* two-sided error *if* $\mathcal{A}$ *has error* $(1/3, 1/3)$*.*

Notice that our definition is non-uniform since we allow an arbitrary algorithm A_n for each window size n. If the window size is not specified, then it is implicitly universally quantified. The space consumption of $\mathcal{A}$ is the mapping $s(n)$, where $s(n)$ is the space consumption of A_n, i.e., the maximal number of bits stored by A_n while reading any input stream. We can assume that $s(n) \in \mathcal{O}(n)$ since A_n can store the active window in $\mathcal{O}(n)$ bits. The goal is to devise algorithms which only use $o(n)$ space. Using probability amplification (similar to [16]) one can replace the error probability $1/3$ in the two-sided error setting (resp. $1/2$ in the one-sided error setting) by any probability $p < 1/2$ (resp. $p < 1$). This influences the space complexity only by a constant factor. The case of Hamming gap $\gamma(n) = 0$ corresponds to exact membership testing to L which was studied in [14, 15, 16]. In this paper, we focus on the two cases $\gamma(n) = c$ for some constant $c > 0$ and $\gamma(n) = \epsilon n$ for some $\epsilon > 0$.

Before we come to the main results of the paper we state two simple facts about the sliding window testers.

▶ **Lemma 2.3.** *Assume that* $L = \bigcup_{i=1}^{k} L_i$ *and that for every* $1 \le i \le k$ *there exists a randomized sliding window tester for* L_i *with Hamming gap* $\gamma(n)$ *and error* (α, β) *that uses space* $s_i(n)$*. Then there exists a sliding window tester for* L *with Hamming gap* $\gamma(n)$ *and error* (α, β) *that uses space* $\mathcal{O}(\sum_{i=1}^{k} s_i(n))$*.*

The second fact concerns so-called trivial languages. Let $\gamma : \mathbb{N} \to \mathbb{R}_{\ge 0}$ be a mapping with $\gamma(n) \le n$ for all $n \ge 0$. A language is $L \subseteq \Sigma^*$ is γ*-trivial* if there exists $n_0 \in \mathbb{N}$ such that for all $n \ge n_0$ with $L \cap \Sigma^n \neq \emptyset$ and all $w \in \Sigma^n$ we have $\mathrm{dist}(w, L) \le \gamma(n)$. If $\gamma(n) \in \mathcal{O}(1)$, we say that L is *trivial*. Note that Alon et al. [2] call a language L trivial if L is (ϵn)-trivial for all $\epsilon > 0$ according to our definition. In the appendix we show that both definitions coincide for regular languages (see Corollary B.5), but we will not make use of this fact.

▶ **Theorem 2.4.** *For every trivial (but not necessarily regular) language there is a deterministic sliding window tester with constant Hamming gap that uses constant space. The converse is also true: If for a language* L *there is a deterministic constant-space sliding window tester with Hamming gap* $\gamma(n)$*, then there exists a constant* c *such that* L *is* $(\gamma + c)$*-trivial.*

3 Main results

Our first main contribution is a deterministic logspace sliding window tester for every regular language, together with a matching lower bound for so-called *nontrivial* regular languages (defined above).

▶ **Theorem 3.1** (deterministic setting, upper bound)**.** *For every regular language* L*, there exists a deterministic sliding window tester for* L *with constant Hamming gap which uses* $\mathcal{O}(\log n)$ *space.*

▶ **Theorem 3.2** (deterministic setting, lower bound)**.** *For every non-trivial regular language* L*, there exist* $\epsilon > 0$ *and infinitely many window sizes* $n \in \mathbb{N}$ *on which every deterministic sliding window tester for* L *with Hamming gap* ϵn *uses space* $\Omega(\log n)$*.*

Our second main contribution is a constant-space randomized sliding window property tester with two-sided error for any regular language:

▶ **Theorem 3.3** (two-sided error randomized setting, upper bound). *For every regular language L and every $\epsilon > 0$, there exists a randomized sliding window tester for L with two-sided error and Hamming gap $\gamma(n) = \epsilon n$ that uses space $\mathcal{O}(1/\epsilon)$.*

While the randomized setting with two-sided error allows ultra-efficient testers, we find that allowing a two-sided error is a very strong relaxation. To this end, we study the randomized setting with one-sided error. In this setting, only a small class of regular languages admits sliding window testers working in space $o(\log n)$. A language $L \subseteq \Sigma^*$ is *suffix-free* if $xy \in L$ and $x \neq \lambda$ imply $y \notin L$.

▶ **Theorem 3.4** (one-sided error randomized setting, upper bound). *If L is a finite union of trivial regular languages and suffix-free regular languages, then there exists a randomized sliding window tester for L with one-sided error and constant Hamming gap which uses $\mathcal{O}(\log \log n)$ space.*

▶ **Theorem 3.5** (one-sided error randomized setting, lower bound). *Let L be a regular language.*

- *If L is not a finite union of trivial regular languages and suffix-free regular languages, there exist $\epsilon > 0$ and infinitely many window sizes n on which every randomized sliding window tester for L with one-sided error and Hamming gap ϵn uses space $\Omega(\log n)$.*
- *If L is non-trivial, then there exist $\epsilon > 0$ and infinitely many window sizes n on which every sliding window tester for L with one-sided error and Hamming gap ϵn uses space $\Omega(\log \log n)$.*

We sketch the proofs of Theorem 3.1, 3.3, and 3.4 in Sections 4.1, 4.2, and 4.3, respectively. The proofs of the lower bounds (Theorems 3.2 and 3.5) can be found in Appendix B. We would like to emphasize that the lower bounds shown in the appendix are stronger than those stated in Theorems 3.2 and 3.5. More precisely, we show space lower bounds for nondeterministic and co-nondeterministic sliding window testers; see Appendix B for definitions.

4 Proofs of the upper bounds

In this section we sketch proofs of Theorems 3.1, 3.3, and 3.4 that give upper bounds for deterministic and (one-sided and two-sided error) randomized sliding window testers. All algorithms in this section satisfy the stronger property that words with large prefix distance are rejected by the algorithm with high probability (probability one in the deterministic setting). The *prefix distance* between words $u = a_1 \cdots a_n$ and $v = b_1 \cdots b_n$ is $\operatorname{pdist}(u, v) = \min\{i \in \{0, \ldots, n\} : a_{i+1} \cdots a_n = b_{i+1} \cdots b_n\}$. Clearly, we have $\operatorname{dist}(u, v) \leq \operatorname{pdist}(u, v)$. We extend the definition to languages: for a language L, let $\operatorname{pdist}(u, L) = \min\{\operatorname{pdist}(u, v) : v \in L\}$. The prefix distance between two runs $\pi = (q_0, a_1, \ldots, q_{n-1}, , a_n, q_n)$ and $\rho = (p_0, b_1, \ldots, p_{n-1}, b_n, p_n)$ is defined as $\operatorname{pdist}(\pi, \rho) = \min\{i \in \{0, \ldots, n\} : (q_i, a_{i+1}, \ldots, q_{n-1}, a_n, q_n) = (p_i, b_{i+1}, \ldots, p_{n-1}, b_n, p_n)\}$.

For our upper bound proofs it is convenient to work with DFAs which read the input word from right to left. A *right-deterministic finite automaton (rDFA)* is a tuple $B = (Q, \Sigma, F, \delta, q_0)$, where Q, Σ, q_0 and F are as in a DFA, and $\delta\colon \Sigma \times Q \to Q$ is the transition function. We extend δ to a mapping $\delta : Q \times \Sigma^* \to Q$ analogously to DFAs: $\delta(q, \lambda) = q$ and $\delta(q, wa) = \delta(\delta(q, a), w)$. The regular language recognized by the rDFA B is $L(B) = \{w \in \Sigma^* : \delta(w, q_0) \in F\}$. A run from $p_0 \in Q$ to $p_n \in Q$ on a word $x = a_n \cdots a_2 a_1 \in \Sigma^*$ is a sequence $\pi = (p_n, a_n, p_{n-1}, \ldots, p_2, a_2, p_1, a_1, p_0)$ such that $p_i = \delta(a_i, p_{i-1})$ for all $1 \leq i \leq n$. The *length* of π is $|\pi| = n$. We visualize π in the form

$$\pi\colon p_n \xleftarrow{a_n} p_{n-1} \xleftarrow{a_{n-1}} \cdots \xleftarrow{a_2} p_1 \xleftarrow{a_1} p_0.$$

If $p_n \in F$, then π is an *accepting run*. A run of length 1 is a *transition*. If π is a run from p to q on a word v, and ρ is a run from q to r on a word u, then $\rho\pi$ denotes the unique run from p to r on uv. We denote by $\pi_{w,q}$ the unique run on w from q.

Strongly connected graphs. With a DFA $A = (Q, \Sigma, q_0, \delta, F)$ we associate the directed graph (Q, E) with edge set $E = \{(p, \delta(p, a)) \mid p \in Q, a \in \Sigma\}$. Similarly, with an rDFA $A = (Q, \Sigma, F, \delta, q_0)$ we associate the directed graph (Q, E) with edge set $E = \{(p, \delta(a, p)) \mid p \in Q, a \in \Sigma\}$. Let A be a DFA or an rDFA. Two states p, q in A are *strongly connected* if there exists a path in (Q, E) from p to q, and vice versa. The *strongly connected components (SCCs)* of A with state set Q are the maximal subsets $C \subseteq Q$ in which all states $p, q \in C$ are strongly connected. A state $q \in Q$ is *transient* if there exists no nonempty path from q to q. An SCC C is *transient* if it only contains a single transient state. There is a natural partial order on the SCCs, called the *SCC-ordering*, where the SCC C_1 is smaller than the SCC C_2 if there exists a path in (Q, E) from a state in C_1 to a state in C_2.

The following combinatorial result from [2] will be used in this paper. Consider a directed graph $G = (V, E)$. The period of G is the greatest common divisor of all cycle lengths in G. If G is acyclic we define the period to be ∞.

▶ **Lemma 4.1** (c.f.[2]). *Let $G = (V, E)$ be a strongly connected directed graph with $E \neq \emptyset$ and finite period g. Then there exist a partition $V = \bigcup_{i=0}^{g-1} V_i$ and a constant $m(G) \leq 3|V|^2$ with the following properties:*

- *For every $0 \leq i, j \leq g-1$ and for every $u \in V_i$, $v \in V_j$ the length of every directed path from u to v in G is congruent to $j - i$ modulo g.*
- *For every $0 \leq i, j \leq g-1$, for every $u \in V_i$, $v \in V_j$ and every integer $r \geq m(G)$, if r is congruent to $j - i$ modulo g, then there exists a directed path from u to v in G of length r.*

If $G = (V, E)$ is strongly connected with $E \neq \emptyset$ and finite period g, and $V_0, \ldots, V_{g-1}$ satisfy the properties from Lemma 4.1, then we define the *shift* from $u \in V_i$ to $v \in V_j$ by

$$\operatorname{shift}(u, v) = j - i \pmod g \in \{0, \ldots, g-1\}. \tag{1}$$

Notice that this definition is independent of the partition $\bigcup_{i=0}^{g-1} V_i$ since any path from u to v has length $\ell \equiv \operatorname{shift}(u, v) \pmod g$ by Lemma 4.1. Also note that $\operatorname{shift}(u, v) + \operatorname{shift}(v, u) \equiv 0 \pmod g$. In the following let $g(C)$ denote the period of the SCC C.

▶ **Lemma 4.2** (Uniform period). *For every regular language L there exists an rDFA A for L and a number g such that every non-transient SCC C in A has period $g(C) = g$.*

Path summaries. We start by recalling the notion of a path summary from [14], where it was used in order to prove a logspace upper bound for regular left-ideals (in the exact setting where the Hamming gap is zero). For the rest of Section 4 we fix a regular language $L \subseteq \Sigma^*$ and an rDFA $B = (Q, \Sigma, F, \delta, q_0)$ which recognizes L. By Lemma 4.2, we can assume that every non-transient SCC C of B has period $g(C) = g$. Consider a run $\pi = (p_n, a_n, \ldots, a_1, p_0)$ on $x = a_n \cdots a_1$. If all states $p_n, \ldots, p_0$ are contained in a single SCC we call π *internal*. We can decompose $\pi = \pi_m \tau_{m-1} \pi_{m-1} \cdots \tau_1 \pi_1$, where each π_i is a possibly empty internal run and each τ_i is a single transition connecting two distinct SCCs. We call this unique factorization the *SCC-factorization* of π, which is illustrated in Figure 1. The *path summary* of π is

$$\operatorname{ps}(\pi) = (|\pi_m|, q_m)(|\tau_{m-1}\pi_{m-1}|, q_{m-1}) \cdots (|\tau_2\pi_2|, q_2)(|\tau_1\pi_1|, q_1),$$

where q_i is the first state in π_i $(1 \leq i \leq m)$. Note that m is bounded by the constant number of states of B. Hence, a path summary can be stored with $\mathcal{O}(\log |\pi|)$ bits.

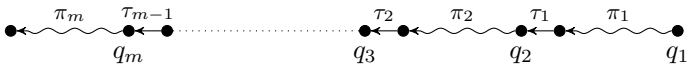

Figure 1 The SCC-factorization of a run

Periodic acceptance sets. For $a \in \mathbb{N}$ and $X \subseteq \mathbb{N}$ we use the standard notation $X + a = \{a + x : x \in X\}$. For a state $q \in Q$ we define $\mathrm{Acc}(q) = \{n \in \mathbb{N} : \exists w \in \Sigma^n : \delta(w, q) \in F\}$. A set $X \subseteq \mathbb{N}$ is *eventually d-periodic*, where $d \geq 1$ is an integer, if there exists a *threshold* $t \in \mathbb{N}$ such that for all $x \geq t$ we have $x \in X$ if and only if $x + d \in X$. If X is eventually d-periodic for some $d \geq 1$, then X is *eventually periodic*.

▶ **Lemma 4.3.** *For every $q \in Q$ the set $\mathrm{Acc}(q)$ is eventually g-periodic.*

Two sets $X, Y \subseteq \mathbb{N}$ are *equal up to a threshold* $t \in \mathbb{N}$, in symbol $X =_t Y$, if for all $x \geq t$: $x \in X$ iff $x \in Y$. Sets $X, Y \subseteq \mathbb{N}$ are *almost equal* if $X =_t Y$ for some threshold $t \in \mathbb{N}$.

▶ **Lemma 4.4.** *Let C be a non-transient SCC in B, $p, q \in C$ and $s = \mathrm{shift}(p, q)$. Then $\mathrm{Acc}(p)$ and $\mathrm{Acc}(q) + s$ are almost equal.*

▶ **Corollary 4.5.** *There exists a threshold $t \in \mathbb{N}$ such that*
1. *$\mathrm{Acc}(q) =_t \mathrm{Acc}(q) + g$ for all $q \in Q$, and*
2. *$\mathrm{Acc}(p) =_t \mathrm{Acc}(q) + \mathrm{shift}(p, q)$ for all non-transient SCCs C and all $p, q \in C$.*

We fix the threshold t from Corollary 4.5 for the rest of Section 4. The following lemma is the main tool to prove the correctness of our sliding window testers. It states that if a word of length n is accepted from p and ρ is any internal run from p of length at most n, then, up to a bounded length prefix, ρ can be extended to an accepting run of length n. Formally, a run π *k-simulates* a run ρ if one can factorize $\rho = \rho_1\rho_2$ and $\pi = \pi'\rho_2$ where $|\rho_1| \leq k$.

▶ **Lemma 4.6.** *If ρ is an internal run starting from p of length at most n and $n \in \mathrm{Acc}(p)$, then there exists an accepting run π from p of length n which t-simulates ρ.*

4.1 Deterministic logspace tester

Proof of Theorem 3.1. Let $n \in \mathbb{N}$ such that $n \geq |Q|$ (for $n < |Q|$ we use a trivial streaming algorithm which stores the window explicitly). The algorithm maintains the set $\{\mathrm{ps}(\pi_{w,q}) \mid q \in Q\}$ where $w \in \Sigma^n$ is the active window. Initially this set is $\{\mathrm{ps}(\pi_{w,q}) \mid q \in Q\}$ for $w = \square^n$. Now suppose $w = av$ for some $a \in \Sigma$ and the next symbol of the stream is $b \in \Sigma$, i.e. the new active window is vb. For each transition $q \xleftarrow{b} p$ in B we can compute $\mathrm{ps}(\pi_{vb,p})$ from $\mathrm{ps}(\pi_{av,q})$ as follows. Suppose that $\mathrm{ps}(\pi_{av,q}) = (\ell_m, q_m) \cdots (\ell_1, q_1)$ where $q = q_1$.

- If p and q belong to the same SCC, then we increment ℓ_1 by one, else we append a new pair $(1, p)$.
- If $\ell_m > 0$ we decrement ℓ_m by one. If $\ell_m = 0$ we remove the pair (ℓ_m, q_m) and we decrement ℓ_{m-1} by one (in this case we must have $m > 1$ and $\ell_{m-1} > 0$).

The obtained path summary is $\mathrm{ps}(\pi_{vb,p})$. This data structure can be stored with $\mathcal{O}(\log n)$ bits since it contains $|Q|$ path summaries, each of which can be stored in $\mathcal{O}(\log n)$ bits.

It remains to define a proper acceptance condition. Consider the run $\pi = \pi_{w,q_0}$, its SCC-factorization $\pi_m\tau_{m-1}\pi_{m-1} \cdots \tau_1\pi_1$ and its path summary $(\ell_m, q_m) \cdots (\ell_1, q_1)$. The algorithm accepts if and only if $\ell_m = |\pi_m| \in \mathrm{Acc}(q_m)$. If $w \in L$, then clearly $|\pi_m| \in \mathrm{Acc}(q_m)$. If $|\pi_m| \in \mathrm{Acc}(q_m)$, then the internal run π_m can be t-simulated by an accepting run π'_m of equal length by Lemma 4.6. The run $\pi'_m\tau_{m-1}\pi_{m-1} \cdots \tau_1\pi_1$ is accepting and witnesses that $\mathrm{pdist}(w, L) \leq t$. ◀

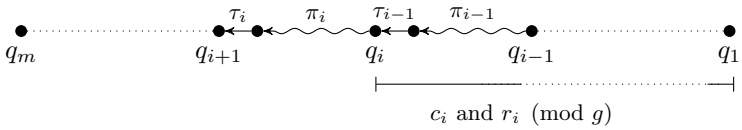

Figure 2 A compact summary of a run π.

4.2 Randomized constant-space tester with two-sided error

Let us first define a probabilistic counter, similar to the approximate counter by Morris [28], which uses $O(\log\log n)$ bits. For our purposes it suffices to distinguish high and low counters states. Consider a probabilistic data structure Z representing a counter. Its operations are incrementing the counter (using random coins) and querying whether the state of the counter is *low* or *high*. Initially Z is in a low state. The random state reached after k increments is denoted by $Z(k)$. Given numbers $0 \leq \ell < h$ (they will depend on our window size n) we say that Z is an (h,ℓ)-*counter with error probability* $\delta < \frac{1}{2}$ if for all $k \in \mathbb{N}$ we have:

- If $k \leq \ell$, then $\text{Prob}[Z(k) \text{ is high}] \leq \delta$.
- If $k \geq h$, then $\text{Prob}[Z(k) \text{ is low}] \leq \delta$.

▶ **Lemma 4.7.** *For all $h, \ell, \xi > 0$ with $\ell \leq (1-\epsilon)h + \mathcal{O}(1)$ there exists an (h,ℓ)-counter Z with error probability $1/3|Q|$ which internally stores $\mathcal{O}(\log(1/\epsilon))$ bits.*

Fix a parameter $0 < \epsilon < 1$ and a window length $n \in \mathbb{N}$. Based on the previous concepts, we are now able to describe a randomized sliding window tester for a regular language L with Hamming gap ϵn that uses $\mathcal{O}(\log(1/\epsilon))$ bits. Let Z be the (h,ℓ)-counter with error probability $1/(3|Q|)$ from Lemma 4.7 where $h = n - t$ and $\ell = (1-\epsilon)n + t + 1$. The counter is used to define so-called compact summaries of runs.

▶ **Definition 4.8.** *A* compact summary $cs = (q_m, r_m, c_m)\cdots(q_2, r_2, c_2)(q_1, r_1, c_1)$ *is a sequence of triples, where each triple (q_i, r_i, c_i) consists of a state $q_i \in Q$, a remainder $0 \leq r_i \leq g-1$, and a state c_i of the (h,ℓ)-counter Z. The state c_1 must be low and $r_1 = 0$.*

A compact summary $(q_m, r_m, c_m)\cdots(q_1, r_1, c_1)$ represents *a run π if the SCC-factorization of π has the form $\pi_m\tau_{m-1}\pi_{m-1}\cdots\tau_1\pi_1$, and the following properties hold:*

1. *for all $1 \leq i \leq m$, π_i starts in q_i;*
2. *for all $2 \leq i \leq m$, if $|\tau_{i-1}\pi_{i-1}\cdots\tau_1\pi_1| \leq (1-\epsilon)n + t + 1$, then c_i is the low state; and if $|\tau_{i-1}\pi_{i-1}\cdots\tau_1\pi_1| \geq n - t$, then c_i is the high state;*
3. *for all $2 \leq i \leq m$, $r_i = |\tau_{i-1}\pi_{i-1}\cdots\tau_1\pi_1| \pmod g$.*

The idea of a compact summary is visualized in Figure 2. If $m > |Q|$ then the above compact summary cannot represent a run. Therefore, we can assume that $m \leq |Q|$. For every triple (q_i, r_i, c_i), the entries q_i and r_i only depend on the rDFA B, and hence can be stored with $\mathcal{O}(1)$ bits. Every state c_i of the probabilistic counter needs $\mathcal{O}(\log(1/\epsilon))$ bits. Hence, a compact summary can be stored in $\mathcal{O}(\log(1/\epsilon))$ bits. In contrast to Theorem 3.1, we maintain a set of compact summaries which represent all runs of B on the *complete* stream read so far (not only on the active window) with high probability.

▶ **Proposition 4.9.** *For a given input stream $w \in \Sigma^*$, we can maintain a set of compact summaries S containing for each $q \in Q$ a compact summary $cs_q \in S$ starting in q such that cs_q represents the unique run $\pi_{w,q}$ with probability at least $2/3$.*

It remains to define an acceptance condition on compact summaries. For every $q \in Q$ we define $\text{Acc}_{mod}(q) = \{\ell \pmod g : \ell \in \text{Acc}(q) \text{ and } \ell \geq t\}$, which is intuitively speaking

the set of accepting remainders. Let $\mathrm{cs} = (q_m, r_m, c_m) \cdots (q_1, r_1, c_1)$ be a compact summary. Since c_1 is the low initial state of the probabilistic counter, there exists a maximal index $i \in \{1, \ldots, m\}$ such that c_i is low. We say that cs is *accepting* if $n - r_i \pmod g \in \mathrm{Acc}_{mod}(q_i)$.

▶ **Proposition 4.10.** *Assume that $\epsilon n \geq t$. Let $w \in \Sigma^*$ with $|w| \geq n$ and let cs be a compact summary which represents π_{w,q_0}.*

1. *If $\mathrm{last}_n(w) \in L$, then cs is accepting.*
2. *If cs is accepting, then $\mathrm{pdist}(\mathrm{last}_n(w), L) \leq \epsilon n$.*

Proof of Theorem 3.3. Assume that $\epsilon n \geq t$, otherwise we use a trivial streaming algorithm that stores the window explicitly with $\mathcal{O}(1/\epsilon)$ bits. We use the algorithm from Proposition 4.9 for each incoming symbol from the stream. To initialize, we run the algorithm on $\square^n$. The algorithm accepts if the computed compact summary starting in q_0 is accepting. From Proposition 4.9 and 4.10 we get:

- If $\mathrm{pdist}(\mathrm{last}_n(w), L) > \epsilon n$, then the algorithm rejects with probability at least 2/3.
- If $\mathrm{last}_n(w) \in L$, then the algorithm accepts with probability at least 2/3.

This concludes the proof of the theorem. ◀

Comparing Theorems 3.1 and 3.3 leads to the question whether one can replace the Hamming gap $\gamma(n) = \epsilon n$ in Theorem 3.3 by $\gamma(n) = o(n)$ while retaining constant space at the same time. We show that this is not the case:

▶ **Lemma 4.11.** *Every randomized sliding window tester with two-sided error for $a^* \subseteq \{a, b\}^*$ with Hamming gap $\gamma(n)$ needs space $\Omega(\log n - \log \gamma(n))$ for infinitely many n.*

4.3 Randomized loglogspace tester with one-sided error

Let L be a finite union of trivial regular languages and suffix-free regular languages. In this section, we present a randomized sliding window tester for L with one-sided error and Hamming gap $\gamma(n) = \epsilon n$ that uses space $\mathcal{O}(\log \log n)$. By Lemma 2.3 and Theorem 2.4, it suffices to consider the case when L is a suffix-free regular language. As in Section 4 we fix an rDFA $B = (Q, \Sigma, F, \delta, q_0)$ for L such that $g(C) = g$ for all SCCs of A. Since L is suffix-free, B has the property that no final state can be reached from a final state by a non-empty run. We decompose B into a finite union of *partial automata*, similar to [14].

▶ **Definition 4.12.** *A sequence $(q_k, a_k, p_{k-1}), C_{k-1}, \ldots, (q_2, a_2, p_1), C_1, (q_1, a_1, p_0), C_0, q_0$ is a* path description *if $C_{k-1}, \ldots, C_0$ is a chain (read from right to left) in the SCC-ordering of B, $p_i, q_i \in C_i$, $q_{i+1} \xleftarrow{a_{i+1}} p_i$ is a transition in B for all $0 \leq i \leq k-1$, and $q_k \in F$.*

Each path description defines a *partial rDFA* $B_P = (Q_P, \Sigma, \{q_k\}, \delta_P, q_0)$ by restricting B to the state set $Q_P = \bigcup_{i=0}^{k-1} C_i \cup \{q_k\}$, restricting the transitions of B to internal transitions from the SCCs C_i and the transitions $q_{i+1} \xleftarrow{a_{i+1}} p_i$, and declaring q_k to be the only final state. The rDFA is partial since for every state p_i and every symbol $a \in \Sigma$ there exists at most one transition $q \xleftarrow{a} p_i$. Since the number of path descriptions P is finite and $L(B) = \bigcup_P L(B_P)$, it suffices to provide a sliding window tester for $L(B_P)$ (we again use Lemma 2.3 here).

From now on, we fix a path description P from Definition 4.12 and the partial automaton $B_P = (Q_P, \Sigma, \{q_k\}, \delta_P, q_0)$ corresponding to it. The acceptance sets $\mathrm{Acc}(q)$ are defined with respect to B_P. If all C_i are transient, then $L(B_P)$ is a singleton and we can use a trivial sliding window tester with space complexity $\mathcal{O}(1)$. Now assume the contrary and let $0 \leq e \leq k-1$ be maximal such that C_e is nontransient.

▶ **Lemma 4.13.** *There exist $r_0, \ldots, r_{k-1}, s_0, \ldots, s_e \in \mathbb{N}$ such that the following holds:*

1. *For all* $e+1 \leq i \leq k$*, the set* $\mathrm{Acc}(q_i)$ *is a singleton.*
2. *Every run from* q_i *to* q_{i+1} *has length* $r_i \pmod g$*.*
3. *For all* $0 \leq i \leq e$*,* $\mathrm{Acc}(q_i) =_{s_i} \sum_{j=i}^{k-1} r_j + g\mathbb{N}$*.*

Let $s = \max\{k, \sum_{j=0}^{k-1} r_j, s_0, \ldots, s_e\}$ and for a word $w \in \Sigma^*$ define the function $\ell_w \colon Q \to \mathbb{N} \cup \{\infty\}$ where $\ell_w(q) = \inf\{\ell \in \mathbb{N} \mid \delta_P(\mathrm{last}_\ell(w), q) = q_k\}$ (we set $\inf \emptyset = \infty$).

Let p be a random prime with $\Theta(\log\log n)$ bits. We now define an acceptance condition on $\ell_w(q)$. If $n \notin \mathrm{Acc}(q_0)$, we always reject. Otherwise, we accept w iff $\ell_w(q_0) \equiv n$ modulo our randomly chosen prime p.

▶ **Lemma 4.14.** *Let* $n \in \mathrm{Acc}(q_0)$ *be a window size with* $n \geq s + |Q_P|$ *and* $w \in \Sigma^*$ *with* $|w| \geq n$*. There exists a constant* $c > 0$ *such that:*

1. *if* $\mathrm{last}_n(w) \in L(B_P)$*, then* w *is accepted with probability* 1*;*
2. *if* $\mathrm{pdist}(\mathrm{last}_n(w), L(B_P)) > c$*, then* w *is rejected with probability at least* $2/3$*.*

Proof of Theorem 3.4. Let $n \in \mathbb{N}$ be the window size. From the discussion above, it suffices to show a tester for a fixed partial automaton B_P. Assume $n \geq s + |Q|$, otherwise a trivial tester can be used. If $n \notin \mathrm{Acc}(q_0)$, the tester always rejects. Otherwise, the tester picks a random prime p with $\Theta(\log\log n)$ bits and maintains $\ell_w(q) \pmod p$ for all $q \in Q_P$, where w is the stream read so far, which requires $\mathcal{O}(\log\log n)$ bits. When a symbol $a \in \Sigma$ is read, we can update ℓ_{wa} using ℓ_w: If $q = q_k$, then $\ell_{wa}(q) = 0$, otherwise $\ell_{wa}(q) = 1 + \ell_w(\delta_P(a, q)) \pmod p$ where $1+\infty = \infty$. The tester accepts if $\ell_w(q_0) \equiv n \pmod p$. Lemma 4.14 guarantees correctness of the tester in the one-sided error setting. ◀

References

1 Charu C. Aggarwal. *Data Streams — Models and Algorithms*. Springer, 2007.
2 Noga Alon, Michael Krivelevich, Ilan Newman, and Mario Szegedy. Regular languages are testable with a constant number of queries. *SIAM J. Comput.*, 30(6):1842–1862, 2000.
3 Ajesh Babu, Nutan Limaye, Jaikumar Radhakrishnan, and Girish Varma. Streaming algorithms for language recognition problems. *Theor. Comput. Sci.*, 494:13–23, 2013.
4 Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. Optimal sampling from sliding windows. *J. Comput. Syst. Sci.*, 78(1):260–272, 2012.
5 Dany Breslauer and Zvi Galil. Real-time streaming string-matching. *ACM Trans. Algorithms*, 10(4):22:1–22:12, 2014.
6 Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana Starikovskaya. Dictionary matching in a stream. In *Proceedings of ESA 2015*, pages 361–372, 2015.
7 Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana Starikovskaya. The k-mismatch problem revisited. In *Proceedings of SODA 2016*, pages 2039–2052, 2016.
8 Raphaël Clifford, Tomasz Kociumaka, and Ely Porat. The streaming k-mismatch problem. In *Proceedings of SODA 2019*, pages 1106–1125, 2019.
9 Raphaël Clifford and Tatiana Starikovskaya. Approximate Hamming distance in a stream. In *Proceedings of ICALP 2016*, pages 20:1–20:14, 2016.
10 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.
11 Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. Testing and spot-checking of data streams. *Algorithmica*, 34(1):67–80, 2002.
12 Nathanaël François, Frédéric Magniez, Michel de Rougemont, and Olivier Serre. Streaming property testing of visibly pushdown languages. In *Proceedings of ESA 2016*, volume 57 of *LIPIcs*, pages 43:1–43:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
13 Moses Ganardi. Visibly pushdown languages over sliding windows. In *Proceedings of STACS 2019*, volume 126 of *LIPIcs*, pages 29:1–29:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019.

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

15 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.

16 Moses Ganardi, Danny Hucke, and Markus Lohrey. Randomized sliding window algorithms for regular languages. In *Proceedings of ICALP 2018*, volume 107 of *LIPIcs*, pages 127:1–127:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

17 Moses Ganardi, Artur Jeż, and Markus Lohrey. Sliding windows over context-free languages. In *Proceedings of MFCS 2018*, volume 117 of *LIPIcs*, pages 15:1–15:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

18 Shay Golan, Tsvi Kopelowitz, and Ely Porat. Streaming pattern matching with d wildcards. In *Proceedings of ESA 2016*, pages 44:1–44:16, 2016.

19 Shay Golan, Tsvi Kopelowitz, and Ely Porat. Towards optimal approximate streaming pattern matching by matching multiple patterns in multiple streams. In *Proceedings of ICALP 2018*, pages 65:1–65:16, 2018.

20 Shay Golan and Ely Porat. Real-time streaming multi-pattern search for constant alphabet. In *Proceedings of ESA 2017*, pages 41:1–41:15, 2017.

21 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

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

23 Rahul Jain and Ashwin Nayak. The space complexity of recognizing well-parenthesized expressions in the streaming model: The index function revisited. *IEEE Trans. Inf. Theory*, 60(10):6646–6668, Oct 2014.

24 Andreas Krebs, Nutan Limaye, and Srikanth Srinivasan. Streaming algorithms for recognizing nearly well-parenthesized expressions. In *Proceedings of MFCS 2011*, volume 6907 of *Lecture Notes in Computer Science*, pages 412–423. Springer, 2011.

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

26 Frédéric Magniez, Claire Mathieu, and Ashwin Nayak. Recognizing well-parenthesized expressions in the streaming model. *SIAM J. Comput.*, 43(6):1880–1905, 2014.

27 Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis, 2nd edition*. Cambridge University Press, 2017.

28 Robert H. Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, 1978.

29 Benny Porat and Ely Porat. Exact and approximate pattern matching in the streaming model. In *Proceedings of FOCS 2009*, pages 315–323, 2009.

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

31 Tatiana Starikovskaya. Communication and streaming complexity of approximate pattern matching. In *Proceedings of CPM 2017*, LIPIcs, pages 13:1–13:11. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017.

A Appendix

A.1 Additional proof for Section 2

▶ **Lemma 2.3.** *Assume that* $L = \bigcup_{i=1}^{k} L_i$ *and that for every* $1 \leq i \leq k$ *there exists a randomized sliding window tester for* L_i *with Hamming gap* $\gamma(n)$ *and error* (α, β) *that uses*

space $s_i(n)$. Then there exists a sliding window tester for L with Hamming gap $\gamma(n)$ and error (α, β) that uses space $\mathcal{O}(\sum_{i=1}^{k} s_i(n))$.

Proof. We can combine these testers for the languages L_i into a tester for L as follows: First, using probability amplification, we reduce the error of each given sliding window tester to $(\alpha/k, \beta/k)$. Then we run the sliding window testers for L_i in parallel and accept if and only if one of them accepts. ◀

▶ **Theorem 2.4.** *For every trivial (but not necessarily regular) language there is a deterministic sliding window tester with constant Hamming gap that uses constant space. The converse is also true: If for a language L there is a deterministic constant-space sliding window tester with Hamming gap $\gamma(n)$, then there exists a constant c such that L is $(\gamma + c)$-trivial.*

Proof. Assume first that L is trivial. Let $n \in \mathbb{N}$ be a window size. If $L \cap \Sigma^n = \emptyset$, then the algorithm always rejects, which is obviously correct since any active window of length n has infinite Hamming distance to L. Otherwise, the algorithm always accepts. In this case, we use the fact that L is trivial, i.e., there is a constant c such that the Hamming distance between an arbitrary active window of length n and L is at most c.

We now show the converse statement. Let $\mathcal{A} = (A_n)$ be a deterministic sliding window tester for L with Hamming gap $\gamma(n)$ which uses constant space. Assume that every A_n works on at most s bits for a constant s. Let $N \subseteq \mathbb{N}$ be the set of all n such that $L \cap \Sigma^n \neq \emptyset$. Note that every A_n with $n \in N$ can be viewed as a DFA with at most 2^{s+1} states that accepts a non-empty language. The number of DFAs of size at most 2^{s+1} over the input alphabet Σ is bounded by a fixed constant d (up to isomorphism). Hence, at most d different DFAs can appear in the list $(A_n)_{n \in N}$. We therefore can choose numbers $n_1 < n_2 < \cdots < n_e$ from N with $e \leq d$ such that for every $n \in N$ there exists a unique $n_i \leq n$ with $A_n = A_{n_i}$ (here and in the following we do not distinguish between isomorphic DFAs). Let us choose for every $1 \leq i \leq e$ a word $u_i \in L$ of length n_i. Now take any $n \in N$. Assume that $A_n = A_{n_i}$ where $n_i \leq n$. Consider any word $u \in \Sigma^* u_i$. Since $\mathrm{last}_{n_i}(u) = u_i \in L$, A_{n_i} has to accept u. Hence, A_n accepts all words from $\Sigma^* u_i$. In particular, for every word x of length $n - n_i$, A_n accepts xu_i. This implies that $\mathrm{dist}(xu_i, L) \leq \gamma(n)$ for all $x \in \Sigma^{n-n_i}$. Recall that this holds for all $n \in N$ and that N is the set of all lengths realized by L. Hence, if we define $c := \max\{n_1, \ldots, n_e\}$ (which is a constant that only depends on our deterministic sliding window tester), then every word w of length $n \in N$ has Hamming distance at most $\gamma(n) + c$ from a word in L. Therefore L is $(\gamma + c)$-trivial. ◀

A.2 Additional proofs for Section 4

▶ **Lemma 4.2** (Uniform period)**.** *For every regular language L there exists an rDFA A for L and a number g such that every non-transient SCC C in A has period $g(C) = g$.*

Proof. Let $B = (Q, \Sigma, F, \delta, q_0)$ be an rDFA for L. Let g be the product of all periods $g(C)$ over all non-transient SCCs C. As usual, we consider $\mathbb{Z}_g = \{0, \ldots, g-1\}$ with arithmetic operations modulo g. Then $A = B \times \mathbb{Z}_g = (Q \times \mathbb{Z}_g, \Sigma, F \times \mathbb{Z}_g, \delta', (q_0, 0))$, where for all $(p, i) \in Q \times \mathbb{Z}_g$ and $a \in \Sigma$ we set

$$\delta'(a, (p, i)) = \begin{cases} (\delta(a, p), i + 1), & \text{if } p \text{ and } \delta(a, p) \text{ are strongly connected,} \\ (\delta(a, p), 0), & \text{otherwise.} \end{cases}$$

Clearly, A is equivalent to B. We show that every non-transient SCC of A has period g. The non-transient SCCs of A are the sets $C \times \mathbb{Z}_g$, where C is a non-transient SCC of B. Let C be

a non-transient SCC of B. Clearly, every cycle length in $C \times \mathbb{Z}_g$ is a multiple of g. Moreover, by Lemma 4.1 the SCC C contains a cycle of length $k \cdot g(C)$ for every sufficiently large $k \in \mathbb{N}$ ($k \geq m(C)$ suffices). Since g is a multiple of $g(C)$, C also contains a cycle of length $k \cdot g$ for every sufficiently large k. But every such cycle induces a cycle of the same length $k \cdot g$ in $C \times \mathbb{Z}_g$. Hence, there exist primes $p_1 \neq p_2$ such that p_1 and p_2 are not divisors of g and $C \times \mathbb{Z}_g$ contains cycles of length $p_1 \cdot g$ and $p_2 \cdot g$. It follows that the period of $C \times \mathbb{Z}_g$ divides $\gcd(p_1 \cdot g, p_2 \cdot g) = g$. This proves that the period of $C \times \mathbb{Z}_g$ is exactly g. ◀

▶ **Lemma 4.3.** *For every $q \in Q$ the set $\mathrm{Acc}(q)$ is eventually g-periodic.*

Proof. It suffices to show that for all $0 \leq r \leq g-1$ the set $S_r = \{i \in \mathbb{N} : r + i \cdot g \in \mathrm{Acc}(q)\}$ is either finite or co-finite. Consider a remainder $0 \leq r \leq g-1$ where S_r is infinite. We need to show that S_r is indeed co-finite. Let $i \in S_r$ with $i \geq |Q|$, i.e. there exists an accepting run π from q of length $r + i \cdot g$. Since π has length at least $|Q|$ is must traverse a state q in a non-transient SCC C. Choose j_0 such that $j_0 \cdot g \geq m(C)$ where $m(C)$ is the reachability constant from Lemma 4.1. By Lemma 4.1 for all $j \geq j_0$ there exists a cycle from q to q of length $j \cdot g$. Therefore we can prolong π to a longer accepting run by $j \cdot g$ symbols for any $j \geq j_0$. This proves that $x \in S_r$ for every $x \geq i + j_0$ and that S_r is co-finite. ◀

▶ **Lemma A.1.** *A set $X \subseteq \mathbb{N}$ is eventually d-periodic iff X and $X + d$ are almost equal.*

Proof. Let $t \in \mathbb{N}$ be such that for all $x \geq t$ we have $x \in X$ if and only if $x + d \in X$. Then X and $X + d$ are equal up to threshold $t + d$. Conversely, if $X =_t X + d$, then for all $x \geq t$ we have $x + d \in X$ if and only if $x + d \in X + d$, which is true if and only if $x \in X$. ◀

▶ **Lemma 4.4.** *Let C be a non-transient SCC in B, $p, q \in C$ and $s = \mathrm{shift}(p, q)$. Then $\mathrm{Acc}(p)$ and $\mathrm{Acc}(q) + s$ are almost equal.*

Proof. Let $k \in \mathbb{N}$ such that $k \cdot g \geq m(C)$ where $m(C)$ is the large enough constant from Lemma 4.1. By Lemma 4.1 there exists a run from p to q of length $s + k \cdot g$, and a run from q to p of length $(k+1) \cdot g - s$ (the latter number is congruent to $\mathrm{shift}(q, p)$ modulo g). By prolonging accepting runs we obtain

$$\mathrm{Acc}(q) + s + k \cdot g \subseteq \mathrm{Acc}(p) \text{ and } \mathrm{Acc}(p) + (k+1) \cdot g - s \subseteq \mathrm{Acc}(q).$$

Adding $s + k \cdot g$ to both sides of the last inclusion yields

$$\mathrm{Acc}(p) + (2k+1) \cdot g \subseteq \mathrm{Acc}(q) + s + k \cdot g \subseteq \mathrm{Acc}(p).$$

By Lemmas 4.3 and A.1 the three sets above are almost equal. Also $\mathrm{Acc}(q) + s + k \cdot g$ is almost equal to $\mathrm{Acc}(q) + s$ by Lemmas 4.3 and A.1. Since almost equality is a transitive relation, this proves the statement. ◀

▶ **Lemma 4.6.** *If ρ is an internal run starting from p of length at most n and $n \in \mathrm{Acc}(p)$, then there exists an accepting run π from p of length n which t-simulates ρ.*

Proof. If $|\rho| \leq t$, then we choose any accepting run π from p of length $n \in \mathrm{Acc}(p)$. Otherwise, if $|\rho| > t$, then the SCC C containing p is non-transient and we can factor $\rho = \rho_1 \rho_2$ such that $|\rho_1| = t$ where ρ_2 leads from p to q. Set $s := \mathrm{shift}(q, p)$, which satisfies $s + |\rho_2| \equiv 0 \pmod{g}$ by the properties in Lemma 4.1. Since $\mathrm{Acc}(q) =_t \mathrm{Acc}(p) + s$ by Corollary 4.5, $n > t$ and $n \in \mathrm{Acc}(p)$, we have $n + s \in \mathrm{Acc}(q)$. Finally since $n + s \equiv n - |\rho_2| \pmod{g}$ and $n - |\rho_2| = n - |\rho| + t \geq t$ we know $n - |\rho_2| \in \mathrm{Acc}(q)$. This yields an accepting run π' from q of length $n - |\rho_2|$. Then ρ is t-simulated by $\pi = \pi' \rho_2$. ◀

A.3 Additional proofs for Section 4.2

▶ **Lemma 4.7.** *For all $h, \ell, \xi > 0$ with $\ell \leq (1-\epsilon)h + \mathcal{O}(1)$ there exists an (h,ℓ)-counter Z with error probability $1/3|Q|$ which internally stores $\mathcal{O}(\log(1/\epsilon))$ bits.*

Proof. Since $\ell \leq (1-\epsilon)h + \mathcal{O}(1)$, we can choose $\xi = \epsilon - \mathcal{O}(1)$ such that $\ell \leq (1-\xi)h$.

We use the following probabilistic data structure from [16]: A *Bernoulli counter* Z_p is parameterized by a probability $0 < p < 1$ and stores a single bit x. Initially we set $x = 0$, representing the low state. On every increment the bit x is set to 1 (representing the high state) with probability p, and is unchanged with probability $1-p$. After i increments the bit has value 0 with probability $(1-p)^i$, and value 1 with probability $1-(1-p)^i$. Let us first show the following claim:

▷ Claim 1. For all $h, \ell, \xi > 0$ with $\xi < 1$ and $\ell \leq (1-\xi)h$ there exists $0 < p < 1$ such that Z_p is an (h,ℓ)-counter with error probability $1/2 - \xi/8$.

We need to choose p such that (i) $1-(1-p)^{(1-\xi)h} \leq 1/2 - \xi/8$, or equivalently, $1/2 + \xi/8 \leq (1-p)^{(1-\xi)h}$, and (ii) $(1-p)^h \leq 1/2 - \xi/8$, or equivalently, $(1-p)^{(1-\xi)h} \leq (1/2-\xi/8)^{1-\xi}$. It suffices to show

$$\frac{1}{2} + \frac{\xi}{8} \leq \left(\frac{1}{2} - \frac{\xi}{8}\right)^{1-\xi}, \tag{2}$$

then one can pick $p = 1-(1/2-\xi/8)^{1/h}$. Note that (ii) holds automatically for this value of p. Taking logarithms shows that (2) is equivalent to $\ln(4+\xi) - \ln 8 \leq (1-\xi)\cdot(\ln(4-\xi) - \ln 8)$, and by rearranging we obtain $\ln(4+\xi) \leq \ln(4-\xi) + \xi(\ln 8 - \ln(4-\xi))$. Since $\ln 8 - \ln(4-\xi) \geq \ln 8 - \ln 4 = \ln 2$, it suffices to prove

$$\ln(4+\xi) \leq \ln(4-\xi) + \xi \ln 2. \tag{3}$$

One can verify $3\ln 2 \approx 2.0794 \geq 2$. We have:

$$\begin{aligned} 4+\xi &\leq 4 + (3\ln 2 - 1)\xi = 4 + (4\ln 2 - 1)\xi - \xi\ln 2 \leq \\ &\leq 4 + (4\ln 2 - 1)\xi - \xi^2 \ln 2 = (4-\xi)(\xi \ln 2 + 1) \end{aligned}$$

By taking logarithms and plugging in $\ln x \leq x - 1$ for all $x > 0$, we obtain

$$\ln(4+\xi) \leq \ln(4-\xi) + \ln(\xi\ln 2 + 1) \leq \ln(4-\xi) + \xi \ln 2$$

This proves (3) and hence (2), and hence Claim 1.

We now show the main claim of the lemma by probability amplification. Let Z be the counter which uses m copies of Z_p in parallel with independent random bits and returns the majority vote of the m outputs. Notice that it suffices to store the sum of all bits, which takes $\mathcal{O}(\log m)$ bits of space.

Let us now estimate the error probability and choose m suitably. Let $X_1, \ldots, X_m$ be independent Bernoulli variables with $\mathrm{Prob}[X_i = 1] = 1/2 - \xi/8$. By Claim 1, $\mathrm{Prob}[X_i = 1]$ is an upper bound on the error probability of the i-th copy of Z_p. Let $X = \sum_{i=1}^m X_i$. Then $\mathrm{Prob}[X \geq m/2]$ is an upper bound on the error probability of the probabilistic counter Z. We have $\mu = \mathbf{E}[X] = m(1/2 - \xi/8) = \frac{m(4-\xi)}{8}$. Choosing $\delta = \frac{\xi}{4-\xi} \geq \frac{\xi}{4}$ we have $(1+\delta)\mu = m/2$ and $\mu\delta^2 = \xi m\delta/8 \geq \xi^2 m/32$. The Chernoff bound [27, Theorem 4.4] states that

$$\mathrm{Prob}[X \geq m/2] = \mathrm{Prob}\left[X \geq (1+\delta)\mu\right] \leq \exp(-\mu\delta^2/3) \leq \exp(-\xi^2 m/96).$$

To enforce $\mathrm{Prob}[X \geq m/2] \leq 1/(3|Q|)$ we choose $m = \lceil 96\ln(3|Q|)/\xi^2 \rceil$. Hence the algorithm has space complexity $\mathcal{O}(\log m) = \mathcal{O}(\log(1/\xi)) = \mathcal{O}(\log(1/\epsilon))$. ◀

▶ **Proposition 4.9.** *For a given input stream $w \in \Sigma^*$, we can maintain a set of compact summaries S containing for each $q \in Q$ a compact summary $cs_q \in S$ starting in q such that cs_q represents the unique run $\pi_{w,q}$ with probability at least $2/3$.*

Proof. For each state in Q, we initialize the compact summary so that it represents the run $\pi_{\lambda,q}$ (recall that λ is the empty word). Consider a compact summary $\mathrm{cs} = (q_m, r_m, c_m) \cdots (q_1, r_1, c_1)$, which represents a run π_{x,q_1}. We prolong cs by a transition $q_1 \xleftarrow{a} p$ in B as follows:

- if p and q are not in the same SCC, then we increment all counter states c_i, increment all remainders r_i mod g, and append a new triple $(p, 0, c_1)$;
- if p and q belong to the same SCC, then we increment all counter states c_i for $2 \leq i \leq m$, increment the remainder r_i mod g for $2 \leq i \leq m$, and replace q_1 by p.

If $a \in \Sigma$ is the next input symbol of the stream, then S is updated to the new set S' of compact summaries by iterating over all transition $q \xleftarrow{a} p$ in B and prolonging the compact summary starting in q by the transition.

To verify correctness, consider a compact summary $\mathrm{cs} = (q_m, r_m, c_m) \cdots (q_1, r_1, c_1)$ computed by the algorithm. Properties (1) and (3) from Definition 4.8 are satisfied by construction. Furthermore, since $m \leq |Q|$ the probability that Property (2) or (4) is violated is at most $m/(3|Q|) \leq 1/3$ by the union bound. ◀

▶ **Proposition 4.10.** *Assume that $\epsilon n \geq t$. Let $w \in \Sigma^*$ with $|w| \geq n$ and let cs be a compact summary which represents π_{w,q_0}.*

1. *If $\mathrm{last}_n(w) \in L$, then cs is accepting.*
2. *If cs is accepting, then $\mathrm{pdist}(\mathrm{last}_n(w), L) \leq \epsilon n$.*

Proof. Consider the SCC-factorization of $\pi = \pi_{w,q_0} = \pi_m \tau_{m-1} \cdots \tau_1 \pi_1$. Let

$$\mathrm{cs} = (q_m, c_m, r_m) \cdots (q_1, c_1, r_1)$$

be a compact summary representing π. Thus, $q_1 = q_0$. Consider the maximal index $1 \leq i \leq m$ where c_i is low, which means that $|\tau_{i-1}\pi_{i-1} \cdots \tau_1\pi_1| < n - t$ by Definition 4.8(4). The run of B on $\mathrm{last}_n(w)$ has the form $\pi'_k \tau_{k-1}\pi_{k-1} \cdots \tau_1\pi_1$ for some suffix π'_k of π_k. We have $|\pi'_k \tau_{k-1} \cdots \pi_i| = n - |\tau_{i-1}\pi_{i-1} \cdots \tau_1\pi_1| > t$. By Definition 4.8(2) we know that

$$r_i = |\tau_{i-1}\pi_{i-1} \cdots \tau_1\pi_1| \pmod g = n - |\pi'_k \tau_{k-1} \cdots \pi_i| \pmod g.$$

For point 1 assume that $\mathrm{last}_n(w) \in L$. Thus, $\pi'_k \tau_{k-1}\pi_{k-1} \cdots \tau_1\pi_1$ is an accepting run starting in q_0. By Definition 4.8(1), the run $\pi'_k \tau_{k-1} \cdots \pi_i$ starts in q_i. Hence, $\pi'_k \tau_{k-1} \cdots \pi_i$ is an accepting run from q_i of length at least t. By definition of $\mathrm{Acc}_{mod}(q_i)$ we have $|\pi'_k \tau_{k-1} \cdots \pi_i| \pmod g = n - r_i \pmod g \in \mathrm{Acc}_{mod}(q_i)$, and therefore cs is accepting.

For point 2 assume that cs is accepting, i.e.

$$n - r_i \pmod g = |\pi'_k \tau_{k-1} \cdots \pi_i| \pmod g \in \mathrm{Acc}_{mod}(q_i).$$

Recall that $|\pi'_k \tau_{k-1} \cdots \pi_i| > t$. By definition of $\mathrm{Acc}_{mod}(q_i)$ there exists an accepting run from q_i whose length is congruent to $|\pi'_k \tau_{k-1} \cdots \pi_i|$ mod g and at least t. By Corollary 4.5(1) we derive that $|\pi'_k \tau_{k-1} \cdots \pi_i| \in \mathrm{Acc}(q_i)$. We claim that $|\pi_i \tau_{i-1}\pi_{i-1} \cdots \tau_1\pi_1| \geq (1-\epsilon)n + t$ by a case distinction. If $i = m$, then clearly $|\pi_i \tau_{i-1}\pi_{i-1} \cdots \tau_1\pi_1| \geq n \geq (1-\epsilon)n + t$. If $i < m$, then c_{i+1} is high by maximality of i, which implies $|\tau_i\pi_i \cdots \tau_1\pi_1| > (1-\epsilon)n + t + 1$ by Definition 4.8(3). Since τ_i has length one, we have $|\pi_i \tau_{i-1}\pi_{i-1} \cdots \tau_1\pi_1| > (1-\epsilon)n + t$.

Since $|\pi'_k\tau_{k-1}\cdots\pi_i| \in \mathrm{Acc}(q_i)$, we can apply Lemma 4.6 and obtain an accepting run ρ of length $|\pi'_k\tau_{k-1}\cdots\pi_i| \in \mathrm{Acc}(q_i)$ starting in q_i which t-simulates the internal run π_i. The prefix distance from ρ to $\pi'_k\tau_{k-1}\cdots\pi_i$ is at most

$$|\pi'_k\tau_{k-1}\cdots\tau_i| + t = n - |\pi_i\tau_{i-1}\pi_{i-1}\cdots\tau_1\pi_1| + t \leq n - (1-\epsilon)n = \epsilon n.$$

Therefore the prefix distance from the accepting run $\rho\tau_{i-1}\pi_{i-1}\cdots\tau_1\pi_1$ to $\pi'_k\tau_{k-1}\pi_{k-1}\cdots\tau_1\pi_1$ is also at most ϵn. This implies $\mathrm{pdist}(\mathrm{last}_n(w), L) \leq \epsilon n$. ◀

▶ **Lemma 4.11.** *Every randomized sliding window tester with two-sided error for $a^* \subseteq \{a,b\}^*$ with Hamming gap $\gamma(n)$ needs space $\Omega(\log n - \log\gamma(n))$ for infinitely many n.*

Proof. We prove the lemma by a reduction from the randomized one-way communication complexity of the greater-than-function.[2] The setting is the following: Alice (resp. Bob) holds a number $i \in \{1,\ldots,m\}$ (resp., $j \in \{1,\ldots,m\}$). Moreover, both parties receive a random string. Then Alice sends a message to Bob (depending on her input i and her random string), and Bob has to decide whether $i > j$ or $i \leq j$ holds. It is known that in every such one-way protocol, where Bob gives a correct answer with probability at least $2/3$, Alice has to send $\Omega(\log m)$ bits to Bob [25, Theorem 3.8].

Consider a randomized sliding window tester for a^* with Hamming gap $\gamma(n)$ that uses space $s(n)$. Fix a window size n, which is divisible by $k := \gamma(n)+1$. Let $m = n/k$. We divide the window into m blocks of length k. We then obtain a randomized one-way protocol for the greater-than-function on the interval $\{1,\ldots,m\}$: Alice produces from her input $i \in \{1,\ldots,m\}$ the word $w_i = a^{(i-1)k}b^k a^{(m-i)k}$. She then runs the randomized sliding window tester on w_i (using her random bits) and sends the final memory content ($s(n)$ bits) to Bob. Bob continues the run of the randomized sliding window tester (starting from the transferred memory content) with the input stream a^{jk}. He obtains the memory content reached after the input $a^{(i-1)k}b^k a^{(m-i+j)k}$. Finally, Bob outputs the answer given by the randomized sliding window tester. If $i \leq j$, then the window content at the end is a^n and hence belongs to a^*. On the other hand, if $i > j$, then the window content at the end contains the block b^k, hence, the Hamming distance between the window content and a^* is at least $\gamma(n)+1$. This implies that Bob will give a correct answer with probability at least $2/3$. It follows that $s(n) \in \Omega(\log m) = \Omega(\log n - \log\gamma(n))$. Note that for the case $\gamma(n) \leq n^\epsilon$ for a constant $\epsilon > 0$ we obtain $s(n) \in \Omega(\log n)$. ◀

A.4 Additional proofs for Section 4.3

▶ **Lemma 4.13.** *There exist $r_0,\ldots,r_{k-1},s_0,\ldots,s_e \in \mathbb{N}$ such that the following holds:*

1. *For all $e+1 \leq i \leq k$, the set $\mathrm{Acc}(q_i)$ is a singleton.*
2. *Every run from q_i to q_{i+1} has length $r_i \pmod g$.*
3. *For all $0 \leq i \leq e$, $\mathrm{Acc}(q_i) =_{s_i} \sum_{j=i}^{k-1} r_j + g\mathbb{N}$.*

Proof. Point 1 follows immediately from the definition of transient SCCs. Let us now show the second part of the claim.

Let $0 \leq i \leq k-1$ and let N_i be the set of lengths of runs of the form $q_{i+1} \xleftarrow{a_{i+1}} p_i \xleftarrow{w} q_i$ in B_P. If C_i is transient, then $N_i = \{1\}$. Otherwise, by Lemma 4.1 there exist a number $r_i \in \mathbb{N}$ and a cofinite set $D_i \subseteq \mathbb{N}$ such that $N_i = r_i + gD_i$. We can summarize both cases by

[2] A similar reduction was used in [16].

saying that there exist a number $r_i \in \mathbb{N}$ and a set $D_i \subseteq \mathbb{N}$ which is either cofinite or $D_i = \{0\}$ such that $N_i = r_i + gD_i$. This implies Point 2. Then the acceptance sets in B_P satisfy

$$\mathrm{Acc}(q_i) = \sum_{j=i}^{k-1} N_j = \sum_{j=i}^{k-1} (r_j + gD_j) = \sum_{j=i}^{k-1} r_j + g \sum_{j=i}^{k-1} D_j.$$

For all $0 \leq i \leq e$ we get $\mathrm{Acc}(q_i) =_{s_i} \sum_{j=i}^{k-1} r_j + g\mathbb{N}$ for some threshold $s_i \in \mathbb{N}$ (note that a nonempty sum of cofinite subsets of $\mathbb{N}$ is again cofinite). ◀

▶ **Lemma 4.14.** *Let $n \in \mathrm{Acc}(q_0)$ be a window size with $n \geq s + |Q_P|$ and $w \in \Sigma^*$ with $|w| \geq n$. There exists a constant $c > 0$ such that:*

1. *if $\mathrm{last}_n(w) \in L(B_P)$, then w is accepted with probability 1;*
2. *if $\mathrm{pdist}(\mathrm{last}_n(w), L(B_P)) > c$, then w is rejected with probability at least $2/3$.*

Proof. Assume first that $\mathrm{last}_n(w) \in L(B_P)$. Since $L(B_P) \subseteq L$ is suffix-free, $\ell_w(q_0) = n \pmod p$ and w is accepted with probability 1.

Consider now the case when $\mathrm{last}_n(w) \notin L(B_P)$. By definition, in this case $\ell_w(q_0) \neq n$. In other words, only two cases are possible: either $\ell_w(q_0) < n$, or $\ell_w(q_0) > n$. If $\ell_w(q_0) < n$, then by the choice of p $\ell_w(q_0) \not\equiv n \pmod p$ with probability at least $2/3$.

We finally consider the case $\ell_w(q_0) > n$. We will show that in this case the prefix distance between $\mathrm{last}_n(w)$ and $L(B_P)$ is bounded by a constant c, which means that we can either accept or reject. Let π be the run of B_P on $\mathrm{last}_n(w)$ starting from the initial state q_0, and let $\pi = \pi_m \tau_{m-1} \pi_{m-1} \cdots \tau_0 \pi_0$ be its SCC-factorization. We have $|\pi| = n$. Since $\ell_w(q_0) > n$, the run π can be strictly prolonged to a run to q_k and hence we must have $m < k$. For all $0 \leq i \leq m$, the run π_i is an internal run in the SCC C_i from q_i to p_i. For all $0 \leq i \leq m-1$ we have $\tau_i = (q_{i+1} \xleftarrow{a_{i+1}} p_i)$ and $|\tau_i \pi_i| \equiv r_i \pmod g$, by Point 2 from Lemma 4.13. We claim that there exists an index $0 \leq i_0 \leq m$ such that the following three properties hold:

1. q_{i_0} is nontransient,
2. $|\pi_m \tau_{m-1} \pi_{m-1} \cdots \tau_{i_0} \pi_{i_0}| \geq s$,
3. $|\pi_m \tau_{m-1} \pi_{m-1} \cdots \tau_{i_0+1} \pi_{i_0+1}| \leq s + |Q_P|$.

Indeed, let $0 \leq i \leq m$ be the smallest integer such that q_i is nontransient (recall that $n \geq |Q_P|$ and hence π must traverse a nontransient SCC). Then $\tau_{i-1} \pi_{i-1} \cdots \tau_0 \pi_0$ only passes transient states and hence its length is bounded by $|Q_P|$. Therefore, $|\pi_m \tau_{m-1} \pi_{m-1} \cdots \tau_i \pi_i| = n - |\tau_{i-1} \pi_{i-1} \cdots \tau_0 \pi_0| \geq n - |Q_P| \geq s$. Now let $0 \leq i_0 \leq m$ be the largest integer satisfying Properties 1 and 2. If $\pi_m \tau_{m-1} \pi_{m-1} \cdots \tau_{i_0+1} \pi_{i_0+1}$ only passes transient states, then its length is bounded by $m - i_0 \leq s + m$, and we are done. Otherwise, let $i_0 + 1 \leq j \leq m$ be the smallest integer such that q_j is nontransient. The run $\tau_{j-1} \pi_{j-1} \cdots \tau_{i_0+1} \pi_{i_0+1}$ only passes transient states and therefore it has length $j - i_0 - 1$. By maximality of i_0, we have $|\pi_m \tau_{m-1} \pi_{m-1} \cdots \tau_j \pi_j| < s$ and hence Property 3 holds:

$$|\pi_m \tau_{m-1} \pi_{m-1} \cdots \tau_{i_0+1} \pi_{i_0+1}| = |\pi_m \cdots \tau_j \pi_j| + |\tau_{j-1} \pi_{j-1} \cdots \tau_{i_0+1} \pi_{i_0+1}| < s + j - i_0 \leq s + m.$$

Let $0 \leq i_0 \leq m$ be the index satisfying Properties 1-3. Since q_{i_0} is nontransient, we have $i_0 \leq e$ and therefore $\mathrm{Acc}(q_{i_0}) =_s \sum_{j=i_0}^{k-1} r_j + g\mathbb{N}$. We have $|\pi_m \tau_{m-1} \pi_{m-1} \cdots \tau_{i_0} \pi_{i_0}| \in \mathrm{Acc}(q_{i_0})$ because it is larger than s (by Property 2) and

$$|\pi_m \tau_{m-1} \pi_{m-1} \cdots \tau_{i_0} \pi_{i_0}| = n - |\tau_{i_0-1} \pi_{i_0-1} \cdots \tau_0 \pi_0| \equiv n - \sum_{j=0}^{i_0-1} r_j \equiv \sum_{j=i_0}^{k-1} r_j \pmod g,$$

where the last congruence follows from the fact that $n \in \mathrm{Acc}(q_0) =_s \sum_{j=0}^{k-1} r_j + g\mathbb{N}$. By Lemma 4.6 there exists an accepting run π' of length $|\pi_m\tau_{m-1}\pi_{m-1}\cdots\tau_{i_0}\pi_{i_0}|$ which t-simulates π_{i_0}. The prefix distance between $\pi'\tau_{i-1}\pi_{i_0-1}\cdots\tau_0\pi_0$ and $\pi = \pi_m\tau_{m-1}\pi_{m-1}\cdots\tau_0\pi_0$ is at most

$$|\pi_m\tau_{m-1}\pi_{m-1}\cdots\tau_{i_0}| + t = |\pi_m\tau_{m-1}\pi_{m-1}\cdots\tau_{i_0+1}\pi_{i_0+1}| + 1 + t \leq 1 + s + m + t$$

by Property 3. ◀

B Proofs of the lower bounds (Theorems 3.2 and 3.5)

A sliding window algorithm can be naturally seen as a family of finite automata (see [14, 16]). We make use of this viewpoint in order to prove the lower bounds of Theorem 3.2 and Theorem 3.5. To get the strongest possible statements, we prove those lower bounds for so-called nondeterministic and co-nondeterministic sliding window testers.

A *nondeterministic finite automaton* (NFA) is a tuple $A = (Q, \Sigma, I, \delta, F)$ consisting of a finite set of states Q, a finite alphabet Σ, a set of initial states $I \subseteq Q$, a transition relation $\delta \subseteq Q \times \Sigma \times Q$ and a set of final states $F \subseteq Q$. Runs in NFAs are defined similarly to DFAs and rDFAs. Formally, a run in the NFA A is a sequence $(q_0, a_1, q_1, a_2, q_2, \ldots, a_n, q_n)$ such that $(q_{i-1}, a_i, q_i) \in \delta$ for all $1 \leq i \leq n$. A word w is accepted by A ($w \in L(A)$ for short) if it labels a run from an initial state to a final state.

▶ **Definition B.1.** *A* nondeterministic sliding window tester $\mathcal{A} = (A_n)_{n\geq 0}$ *for the language* L *with Hamming gap* $\gamma(n)$ *is a family of NFAs* A_n *such that for each window size* $n \geq 0$ *and each stream* $w \in \Sigma^*$ *the following holds:*

1. *if* $\mathrm{last}_n(w) \in L$*, then* $w \in L(A_n)$*;*
2. *if* $\mathrm{dist}(\mathrm{last}_n(w), L) > \gamma(n)$*, then* $w \notin L(A_n)$*.*

One can view every A_n as a nondeterministic streaming algorithm that updates its memory state nondeterministically depending on the current input symbol. Note that in order to have $\mathrm{last}_n(w) \in L$, it is enough to have at least one run of A_n on $w \in \Sigma^*$ from an initial state to an accepting state. This is equivalent to require that the active window is accepted by the algorithm with some probability greater than 0 (if we assign to every state q and every symbol a a probability distribution on the outgoing a-transitions of q). On the other hand, if $\mathrm{dist}(\mathrm{last}_n(w), L) > \gamma(n)$, then all runs of A_n on $w \in \Sigma^*$ from an initial state end in non-accepting states, i.e. the active window is rejected with probability 1.

A second concept we use in this section are *coNFA*s. The only difference to NFAs is that a word w is accepted by a coNFA A if all runs on w that begin in an initial state have to end in an accepting state. In other words, a word w is rejected by A if and only if there is at least one run on w from an initial state to a non-accepting state. A co-nondeterministic sliding window tester $\mathcal{A} = (A_n)_{n\geq 0}$ for L with Hamming gap $\gamma(n)$ is a family of coNFAs A_n such that for each window size $n \geq 0$ and each stream $w \in \Sigma^*$ the properties 1 and 2 in Definition B.1 hold. So if $\mathrm{last}_n(w) \in L$, then all runs of A_n on $w \in \Sigma^*$ that start in an initial state end in an accepting state. In other words, the algorithm accepts with probability 1. If $\mathrm{dist}(\mathrm{last}_n(w), L) > \gamma(n)$, then there is at least one run of A_n on $w \in \Sigma^*$ that starts in an initial state and ends in a non-accepting state, i.e. the algorithm rejects with probability strictly greater than 0.

Let $\mathcal{A} = (A_n)_{n\geq 0}$ be a (co-)nondeterministic sliding window tester and let Q_n be the state set of A_n. Then the *space consumption* of $\mathcal{A}$ is defined as $s_{\mathcal{A}}(n) = \lceil \log |Q_n| \rceil$. This reflects the fact that states from Q_n can be encoded with $s_{\mathcal{A}}(n)$ many bits.

In order to prove the Theorem 3.2 we will show a logarithmic space lower bound for every nondeterministic sliding window tester for a regular nontrivial language (see Theorem B.6 below). To do this, we will use so-called cut languages.

Given $i, j \geq 0$ and a word w of length at least $i + j$ we define $\mathrm{cut}_{i,j}(w) = y$ such that $w = xyz$, $|x| = i$ and $|z| = j$. If $|w| < i + j$, then $\mathrm{cut}_{i,j}(w)$ is undefined. For a language L we define the *cut-language* $\mathrm{cut}_{i,j}(L) = \{\mathrm{cut}_{i,j}(w) \mid w \in L\}$.

▶ **Lemma B.2.** *If L is regular, then there are finitely many languages $\mathrm{cut}_{i,j}(L)$.*

Proof. Let $A = (Q, \Sigma, q_0, \delta, F)$ be a DFA for L. Given $i, j \geq 0$, let I be the set of states reachable from q_0 via i symbols and let F' be the set of states from which F can be reached via j symbols. Then the nondeterministic finite automaton $(Q, \Sigma, I, \delta, F')$ recognizes $\mathrm{cut}_{i,j}(L)$ (see Section B for the definition of nondeterministic finite automata). Since there are at most $2^{2|Q|}$ such choices for I and F', the number of languages of the form $\mathrm{cut}_{i,j}(L)$ must be finite. ◀

A language L is a *length language* if for all $n \in \mathbb{N}$ either $\Sigma^n \subseteq L$ or $\Sigma^n \cap L = \emptyset$.

▶ **Lemma B.3.** *If $\mathrm{cut}_{i,j}(L)$ is a length language for some $i, j \geq 0$, then L is trivial.*

Proof. Assume that $\mathrm{cut}_{i,j}(L)$ is a length language. Let $n \in \mathbb{N}$ such that $L \cap \Sigma^n \neq \emptyset$ and $n \geq i + j$. We claim that $\mathrm{dist}(w, L) \leq i + j$ for all $w \in \Sigma^n$. Let $w \in \Sigma^n$ and $w' \in L \cap \Sigma^n$. Then $\mathrm{cut}_{i,j}(w') \in \mathrm{cut}_{i,j}(L)$ and hence also $\mathrm{cut}_{i,j}(w) \in \mathrm{cut}_{i,j}(L)$. Therefore there exist $x \in \Sigma^i$ and $z \in \Sigma^j$ such that $x\,\mathrm{cut}_{i,j}(w)\,z \in L$ satisfying $\mathrm{dist}(w, x\,\mathrm{cut}_{i,j}(w)\,z) \leq i + j$. ◀

The *restriction* of a language L to a set of lengths $N \subseteq \mathbb{N}$ is $L|_N = \{w \in L : |w| \in N\}$. A language L *excludes a word w as a factor* if w is not a factor of any word in L. A simple but important observation is that if L excludes w as a factor and v contains k disjoint occurrences of w, then $\mathrm{dist}(v, L) \geq k$: If we change at most $k - 1$ many symbols in v, then the resulting word v' must still contain w as a factor and hence $v' \notin L$.

▶ **Proposition B.4.** *Let L be regular. If $\mathrm{cut}_{i,j}(L)$ is not a length language for all $i, j \geq 0$, then L has an infinite restriction $L|_N$ to an arithmetic progression $N = \{a + bn \mid n \in \mathbb{N}\}$ which excludes a factor.*

Proof. First notice that $\mathrm{cut}_{i,j}(L)$ determines $\mathrm{cut}_{i+1,j}(L)$ and $\mathrm{cut}_{i,j+1}(L)$: we have $\mathrm{cut}_{i+1,j}(L) = \{w \mid \exists a \in \Sigma : aw \in \mathrm{cut}_{i,j}(L)\}$ and similarly for $\mathrm{cut}_{i,j+1}(L)$. Since the number of cut-languages $\mathrm{cut}_{i,j}(L)$ is finite there exist numbers $i \geq 0$ and $d > 0$ such that $\mathrm{cut}_{i,0}(L) = \mathrm{cut}_{i+d,0}(L)$. Hence, we have $\mathrm{cut}_{i,j}(L) = \mathrm{cut}_{i+d,j}(L)$ for all $j \geq 0$. By the same argument, there exist numbers $j \geq 0$ and $e > 0$ such that $\mathrm{cut}_{i,j}(L) = \mathrm{cut}_{i,j+e}(L) = \mathrm{cut}_{i+d,j}(L) = \mathrm{cut}_{i+d,j+e}(L)$, which implies $\mathrm{cut}_{i,j}(L) = \mathrm{cut}_{i,j+h}(L) = \mathrm{cut}_{i+h,j}(L) = \mathrm{cut}_{i+h,j+h}(L)$ for some $h > 0$ (we can take $h = ed$). This implies that $\mathrm{cut}_{i,j}(L)$ is closed under removing prefixes and suffixes of length h.

By assumption $\mathrm{cut}_{i,j}(L)$ is not a length language, i.e. there exist words $y' \in \mathrm{cut}_{i,j}(L)$ and $y \notin \mathrm{cut}_{i,j}(L)$ of the same length k. Let $N = \{k + i + j + hn \mid n \in \mathbb{N}\}$. For any $n \in \mathbb{N}$ the restriction $L|_N$ contains a word of length $k + i + j + hn$ because $y' \in \mathrm{cut}_{i,j}(L) = \mathrm{cut}_{i+hn,j}(L)$. This proves that $L|_N$ is infinite.

Let u be an arbitrary word which contains for every remainder $0 \leq r \leq h - 1$ an occurrence of y as a factor starting at a position which is congruent to r mod h. We claim that $L|_N$ excludes $a^i u a^j$ as a factor where a is an arbitrary symbol. Assume that there exists a word $w \in L|_N$ which contains $a^i u a^j$ as a factor. Then $\mathrm{cut}_{i,j}(w)$ contains u as a factor, has length $k + hn$ for some $n \geq 0$, and belongs to $\mathrm{cut}_{i,j}(L)$. Therefore $\mathrm{cut}_{i,j}(w)$ also contains h many

occurrences of y, one per remainder $0 \leq r \leq h-1$. Consider the occurrence of y in $\mathrm{cut}_{i,j}(w)$ which starts at a position which is divisible by h, i.e. we can factorize $\mathrm{cut}_{i,j}(w) = xyz$ such that $|x|$ is a multiple of h. Since $\mathrm{cut}_{i,j}(w)$ has length $k + hn$ also $|z|$ is a multiple of h. Therefore $y \in \mathrm{cut}_{i+|x|,j+|z|}(L) = \mathrm{cut}_{i,j}(L)$, which is a contradiction. ◀

Before we continue with our lower bound proof, let us prove the following result for nontrivial regular languages that is of independent interest:

▶ **Corollary B.5.** *If L is a nontrivial regular language, then there exists $\epsilon > 0$ such that L is not ϵn-trivial.*

Proof. Let L be nontrivial and regular. By Lemma B.3 and Proposition B.4 there exists an infinite restriction $L|_N$ of L which excludes a factor w. Hence if $n \in N$ and v is any word of length n, which contains at least $\lfloor n/|w| \rfloor$ many disjoint occurrences of w, then $\mathrm{dist}(v, L) \geq \lfloor n/|w| \rfloor$, which proves the claim. ◀

We can now state and prove our general lower bounds.

▶ **Theorem B.6.** *Let L be regular and nontrivial. Then there is a constant ϵ_0, $0 < \epsilon_0 \leq 1$, such that for every $0 \leq \epsilon < \epsilon_0$, every nondeterministic sliding window tester for L with Hamming gap ϵn uses space at least $\log_2 n + \log_2(1 - \epsilon/\epsilon_0) - \mathcal{O}(1)$ on an infinite set of window sizes n (that only depends on L).*

Proof. By Lemma B.3, $\mathrm{cut}_{i,j}(L)$ is not a length language for all $i, j \geq 0$. Let N be the set of lengths from Proposition B.4 such that $L|_N$ is infinite and excludes some factor w_f. Let $c = |w_f| > 0$ and $\epsilon_0 = 1/c$. Since N is an arithmetic progression, $L|_N$ is regular. Recall that every word v that contains k disjoint occurrences of w_f has Hamming distance at least k from any word in $L|_N$. Let $A = (Q, \Sigma, q_0, \delta, F)$ be a DFA for $L|_N$. Since $L(A)$ is infinite, there must exist words x, y, z such that $y \neq \lambda$ and for $\delta(q_0, x) = q$ we have $\delta(q, y) = q$ and $\delta(q, z) \in F$. Let $d = |xz|$ and $e = |y| > 0$.

Consider a nondeterministic sliding window tester $\mathcal{A} = (A_n)_{n \geq 0}$ for L with Hamming gap ϵn for some $\epsilon < \epsilon_0$. Fix a window length $n \in N$ and define for $k \geq 0$ the input streams $u_k = w_f^n x y^k$ and $v_k = u_k z = w_f^n x y^k z$. Let $\alpha = c\epsilon < 1$. If $0 \leq k \leq \lfloor \frac{(1-\alpha)n-c-d}{e} \rfloor$, then the suffix of v_k of length n contains at least

$$\left\lfloor \frac{n-d-ek}{c} \right\rfloor \geq \left\lfloor \frac{n-d-(1-\alpha)n+c+d}{c} \right\rfloor = \left\lfloor \frac{\alpha n + c}{c} \right\rfloor = \lfloor \epsilon n + 1 \rfloor > \epsilon n$$

many disjoint occurrences of w_f. Hence, after reading any of the input streams v_k for $0 \leq k \leq \lfloor \frac{(1-\alpha)n-c-d}{e} \rfloor$, the NFA A_n has to reject with probability one, i.e., every run of A_n on v_k that starts in an initial state has to end in a rejecting state.

Assume now that the window size n satisfies $n \geq d$ and $n \equiv d \pmod{e}$. Write $n = d + le$ for some $l \geq 0$. Note that each n with this property satisfies $n \in N$ since $xy^l z \in L|_N$. We have $l > \lfloor \frac{(1-\alpha)n-c-d}{e} \rfloor$. The suffix of $v_l = w_f^n x y^l z$ of length n is $xy^l z \in L|_N$. Therefore A_n accepts v_l, i.e., there exists a run π of A_n on v_l that starts in an initial state and ends in an accepting state. Let m be the number of states of A_n. For $0 \leq i \leq l$ let p_i be the state on the run π that is reached after the prefix $w_f^n x y^i$ of v_l.

Assume now that $m \leq \lfloor \frac{(1-\alpha)n-c-d}{e} \rfloor$. Then there must exist numbers i and j with $0 \leq i < j \leq \lfloor \frac{(1-\alpha)n-c-d}{e} \rfloor$ such that $p_i = p_j =: p$. By cutting off cycles at p from the run π and repeating this, we finally obtain a run of A_n on an input stream $v_k = w_f^n x y^k z$ with $k \leq \lfloor \frac{(1-\alpha)n-c-d}{e} \rfloor$. This run still goes from an initial state to an accepting state. Hence, A_n

accepts with probability > 0 an input stream v_k with $k \leq \lfloor \frac{(1-\alpha)n-c-d}{e} \rfloor$. This contradicts our previous observation. Hence, for every $n \geq d$ with $n \equiv d \pmod{e}$, A_n must have more than $\lfloor \frac{(1-\alpha)n-c-d}{e} \rfloor$ states. This implies

$$s_{\mathcal{A}}(n) \geq \log_2 \left(\frac{(1-\alpha)n - c - d}{e} \right) \geq \log_2 n + \log_2(1-\alpha) - \mathcal{O}(1),$$

which proves the theorem. ◀

Theorem 3.2 is a direct corollary of Theorem B.6 since every deterministic sliding window tester is also a nondeterministic sliding window tester.

▶ **Example B.7.** For the lower bound $\log_2 n + \log_2(1 - \epsilon/\epsilon_0) - \mathcal{O}(1)$ in Theorem B.6 the Hamming gap has to be strictly below $\epsilon_0 n$, where ϵ_0 is a constant that depends on L. This is in general not avoidable. Consider for instance the language $L_c = (\{a, b\}^{c-1}a)^*$. It is nontrivial, since for any k, the word $w_k = b^{c \cdot k}$ has Hamming distance $\mathrm{dist}(w_k, L_c) = k$ from L_c. On the other hand this is also the worst-case, i.e., any word w of length $n = ck$ has Hamming distance $\mathrm{dist}(w, L_c) \leq k = n/c$ from L_c. Hence, with constant space one can achieve a Hamming gap of n/c using the algorithm that always accepts.

We next want to transfer the lower bound from Theorem B.6 to co-nondeterministic sliding window testers. For this, we make use of a power set construction.

▶ **Lemma B.8.** *If there is a co-nondeterministic sliding window tester $\mathcal{A} = (A_n)_{n \geq 0}$ for L with Hamming gap $\gamma(n)$ that uses space $s(n)$, then there is a deterministic sliding window tester for L with Hamming gap $\gamma(n)$ that uses space $2^{s(n)}$.*

Proof. Let $A_n = (Q_n, \Sigma, I_n, \delta, F_n)$. We apply the powerset construction and transform every coNFA A_n into a DFA A'_n with state set $\mathcal{P}(Q_n)$ (the power set of Q_n). The only difference to the powerset construction for NFAs is the following: a state $Q \subseteq Q_n$ of A'_n is final if and only if $Q \subseteq F_n$ (for NFAs it is only required that $Q \cap F_n \neq \emptyset$). It is straightforward to see that $L(A_n) = L(A'_n)$. Moreover, A'_n has $2^{|Q_n|}$ many states. ◀

Lemma B.8 and Theorem B.6 immediately yield the following result:

▶ **Theorem B.9.** *For every non-trivial regular language L there is a constant ϵ_0, $0 < \epsilon_0 \leq 1$, such that for every $0 \leq \epsilon < \epsilon_0$, every co-nondeterministic sliding window tester for L with Hamming gap ϵn uses space at least $\log_2 \log_2 n - \mathcal{O}(1)$ on an infinite set of window sizes n (that only depends on L).*

Note that a randomized sliding window tester for L with one-sided error is also a co-nondeterministic sliding window tester for L. Hence, the doubly logarithmic space lower bound for non-trivial regular languages from Theorem 3.5 is a direct corollary of Theorem B.9.

It remains to prove the logarithmic space lower bound in Theorem 3.5. For this, we start with two lemmas.

▶ **Lemma B.10.** *Every regular suffix-free language excludes a factor.*

Proof. Let $B = (Q, \Sigma, F, \delta, q_0)$ be an rDFA for L. Since L is suffix-free, we can assume that there is a single maximal SCC that consists of a single state $q_{fail} \notin F$ (if a maximal SCC would contain a final state, then L would not be suffix-free). We have $\delta(a, q_{fail}) = q_{fail}$ for all $a \in \Sigma$. We construct a word $w_f \in \Sigma^*$ such that $\delta(p, w_f) = q_{fail}$ for all $p \in Q$. Let $p_1, \ldots, p_m$ be an enumeration of all states in $Q \setminus \{q_{fail}\}$. We then construct inductively words

$w_0, w_1, \ldots, w_m \in \Sigma^*$ such that for all $0 \leq i \leq m$: $\delta(w_i, p) = q_{fail}$ for all $p \in \{p_1, \ldots, p_i\}$. We start with $w_0 = \lambda$. Assume that w_i has been constructed for some $i < m$. There is a word x such that that $\delta(x, \delta(w_i, p_{i+1})) = q_{fail}$. We set $w_{i+1} = xw_i$. Then $\delta(w_{i+1}, p_{i+1}) = \delta(xw_i, p_{i+1}) = q_{fail}$ and $\delta(w_{i+1}, p_j) = \delta(w_i x, p_j) = \delta(x, q_{fail}) = q_{fail}$ for $1 \leq j \leq i$. We finally define $w_f = w_m$. ◀

▶ **Lemma B.11.** *Every regular language L satisfies one of the following properties:*

- *L is a finite union of regular trivial languages and regular suffix-free languages.*
- *L has a restriction $L|_N$ which excludes some factor and contains y^*z for some $y, z \in \Sigma^*$, $|y| > 0$.*

Proof. Let $B = (Q, \Sigma, F, \delta, q_0)$ be an rDFA for L. Let $B_r = (Q, \Sigma, F_r, \delta, q_0)$ where F_r is the set of non-transient final states and $B_q = (Q, \Sigma, \{q\}, \delta, q_0)$ for $q \in Q$. We can decompose L as a union of $L_r = L(B_r)$ and all languages $L(B_q)$ over all transient states $q \in F$. Notice that $L(B_q)$ is suffix-free for all transient $q \in F$ since any run to q cannot be prolonged to another run to q. If L_r is trivial, then L satisfies the first property. If L_r is nontrivial, then by Lemma B.3 and Proposition B.4 there exists an arithmetic progression $N = \{a + bn \mid n \in \mathbb{N}\}$ such that $L_r|_N$ is infinite and excludes some word $w \in \Sigma^*$ as a factor. Let $z \in L_r|_N$ be any word. Since B_r reaches some non-transient final state p on input z there exists a word y which leads from p back to p. We can ensure that $|y|$ is a multiple of b by replacing y by a suitable power y^i. Then $y^*z \subseteq L_r|_N \subseteq L|_N$. Furthermore since each language $L(B_q)$ excludes some factor w_q by Lemma B.10 the language $L|_N \subseteq L_r|_N \cup \bigcup_q L(B_q)$ excludes any concatenation of w and all words w_q as a factor. ◀

▶ **Theorem B.12.** *Let L be a regular language that is not a finite union of regular trivial languages and regular suffix-free languages. Then there is a constant ϵ_0, $0 < \epsilon_0 \leq 1$, such that for every $0 \leq \epsilon < \epsilon_0$, every co-nondeterministic sliding window tester for L with Hamming gap ϵn uses space at least $\log_2 n + \log_2(1 - \epsilon/\epsilon_0) - \mathcal{O}(1)$ on an infinite set of window sizes n (that only depends on L).*

Proof. By Lemma B.11, L has a restriction $L|_N$ which excludes some factor w_f and contains y^*z for some $y, z \in \Sigma^*$, $|y| > 0$. Let $c = |w_f| \geq 1$. We set $\epsilon_0 = 1/c$. Let $d = |z|$ and $e = |y|$. Fix a window length $n \in N$ and define for $k \geq 0$ the input streams $u_k = w_f^n y^k$ and $v_k = u_k z = w_f^n y^k z$. Consider a co-nondeterministic sliding window tester $\mathcal{A} = (A_n)_{n \geq 0}$ for L with Hamming gap ϵn for some $\epsilon < \epsilon_0$. Let $\alpha = c\epsilon < 1$ and $r = \lfloor \frac{(1-\alpha)n - c - d}{e} \rfloor$. If $0 \leq k \leq r$, then the suffix of v_k of length n contains at least

$$\left\lfloor \frac{n - d - ek}{c} \right\rfloor \geq \left\lfloor \frac{n - d - (1-\alpha)n + c + d}{c} \right\rfloor = \left\lfloor \frac{\alpha n + c}{c} \right\rfloor = \lfloor \epsilon n + 1 \rfloor > \epsilon n$$

many disjoint occurrences of w_f. Hence, after reading any of the input streams v_k for $0 \leq k \leq r$, the coNFA A_n has to reject, i.e., there is an A_n-run on v_k that starts in an initial state and ends in a non-accepting state. Consider an A_n-run π on v_r that goes from an initial state to a non-accepting state. For $0 \leq i \leq r$ let p_i be the state in π that is reached after the prefix $w_f^n y^i$ of v_r. Let now m be the number of states of A_n and assume $m \leq r$. There must exist numbers i and j with $0 \leq i < j \leq r$ such that $p_i = p_j =: p$. It follows that there is an A_n-run on y^{j-i} that starts and ends in state p. Using that cycle we can now prolong the run π, i.e., for all $t \geq 0$ there is an A_n-run on $v_{r+(j-i)\cdot t} = w_f^n y^{r+(j-i)\cdot t} z$ that starts in an initial state and ends in a non-accepting state.

Assume now that the window size satisfies $n \geq d$ and $n \equiv d \pmod{e}$. Write $n = d + le$ for some $l \geq 0$. Note again that each n with this property satisfies $n \in N$ since the word $y^l z$

belongs to $L|_N$. We have $l > \lfloor \frac{(1-\alpha)n-c-d}{e} \rfloor = r$. For every $k \geq l$, the suffix of $v_k = w_f^n y^k z$ of length n is $y^l z \in L$. Therefore A_n accepts v_k, i.e., for all $k \geq l$, every A_n-run on v_k that starts in an initial state has to end in an accepting state. This contradicts our observation that for all $t \geq 0$ there is an A_n-run on $v_{r+(j-i)\cdot t}$ that goes from an initial state to a non-accepting state. Hence, A_n has at least $r + 1 \geq \frac{(1-\alpha)n-c-d}{e}$ states. It follows that

$$s_{\mathcal{A}}(n) \geq \log_2 \left(\frac{(1-\alpha)n - c - d}{e} \right) \geq \log_2 n + \log_2(1 - \epsilon/\epsilon_0) - \mathcal{O}(1).$$

This proves the theorem. ◀

The logarithmic space lower bound from Theorem 3.5 is an immediate consequence of Theorem B.12.