# STREAMING WORD PROBLEMS

MARKUS LOHREY, LUKAS LÜCK, AND JULIO XOCHITEMOL

Abstract. We study deterministic and randomized streaming algorithms for word problems of finitely generated groups. For finitely generated groups that can be obtained from linear groups using the following operations we show the existence of randomized streaming algorithms with logarithmic space complexity for their word problems: finite extensions, taking a finitely generated subgroups, graph products and wreath products by finitely generated abelian groups. We contrast these results with several lower bounds. An example of a finitely presented group, where the word problem has only a linear space randomized streaming algorithm, is Thompson's group $F$. Finally, randomized streaming algorithms for subgroup membership problems in free groups and direct products of free groups are studied.

## 1. Introduction

The word problem for a finitely generated group $G$ is the following computational problem: Fix a finite set of generators $\Sigma$ for $G$, which means that every element of $G$ can be written as a finite product of elements from $\Sigma$. The input for the word problem is a finite word $a_1 a_2 \cdots a_n$ over the alphabet $\Sigma$ and the question is whether this word evaluates to the group identity of $G$. The word problem was introduced by Dehn in 1911 [18]. It is arguably the most important computational problem in group theory and has been studied by group theorists as well as computer scientists; see [54] for a survey. In recent years, complexity theoretic investigations of word problems moved into the focus. For many important classes of groups it turned out that the word problem belongs to low-level complexity classes. The first result in this direction was proved by Lipton and Zalcstein [43] (if the field $F$ has characteristic zero) and Simon [66] (if the field $F$ has prime characteristic): if $G$ is a finitely generated linear group over an arbitrary field $F$ (i.e., a finitely generated group of invertible matrices over $F$), then the word problem for $G$ can be solved in deterministic logarithmic space. Related results can be found in [39, 70].

The word problem of a group $G$ with a finite generating set $\Sigma$ can be identified with a formal language $\mathsf{WP}(G, \Sigma)$ consisting of all words over the alphabet $\Sigma$ that evaluate to the group identity of $G$. Language theoretic aspects of the word problem have been studied intensively in the past. For instance, Anissimov and Seifert [2] showed that $\mathsf{WP}(G, \Sigma)$ is regular if and only if $G$ is finite, and Muller and Schupp [57] showed that $\mathsf{WP}(G, \Sigma)$ is context-free if and only if $G$ is virtually free,[1] see [31] for an overview.

In this paper we initiate the study of streaming algorithms for word problems. These are algorithms that do not have random access on the whole input. Instead, the $k$-th input symbol is only available at time $k$ [1]. Quite often, streaming

[1]If $\mathcal{C}$ is a property or class of groups, then a group $G$ is called virtually $\mathcal{C}$, if $G$ is a finite extension of a $\mathcal{C}$-group.

algorithms are randomized and have a bounded error probability. Usually, one is interested in the space used by a streaming algorithm, but also update times (i.e., the worst case time spend to process a new input symbol) have been studied. Clearly, every regular language $L$ has a deterministic streaming algorithm with constant space; it is a deterministic finite automaton for $L$. Randomized streaming algorithms for context-free languages have been studied in [5, 9, 22, 48].

Let us now explain the main results of this paper. For a finitely generated group $G$ with finite generating set $\Sigma$, the *deterministic* (resp., *randomized) streaming space complexity* of $\mathsf{WP}(G, \Sigma)$ is the space complexity of the best deterministic (resp., randomized) streaming algorithm for $\mathsf{WP}(G, \Sigma)$. The concrete choice of the generating set has only a minor influence on the deterministic (resp., randomized) streaming space complexity of $\mathsf{WP}(G, \Sigma)$; see Lemma 5.1 for a precise statement. In statements where the influence of the generating set on the streaming space complexity is blurred by the Landau notation, we speak of the deterministic/randomized streaming space complexity of the word problem of $G$ or simply the deterministic/randomized streaming space complexity of $G$.

The deterministic streaming space complexity of $\mathsf{WP}(G, \Sigma)$ is directly linked to the growth function $\gamma_{G,\Sigma}(n)$ of the group $G$. The latter is the number of different group elements of $G$ that can be represented by words over the finite generating set $\Sigma$ of length at most $n$ (also here the generating set $\Sigma$ only has a minor influence). The deterministic streaming space complexity of the word problem for $G$ turns out to be $\log_2 \gamma_{G,\Sigma}(n/2)$ up to a small additive constant (Theorem 6.1). The growth of finitely generated groups is a well investigated topic in geometric group theory. A famous theorem of Gromov says that a finitely generated group has polynomial growth if and only if it is virtually nilpotent; see [17, 51] for a discussion. Theorem 6.1 reduces all questions about the deterministic streaming space complexity of word problems to questions about growth functions. Due to this, we mainly study randomized streaming algorithms for word problems in this paper.

In the randomized setting, the growth of $G$ still yields a lower bound: The randomized streaming space complexity of the word problem of $G$ is lower bounded by $\Omega(\log\log \gamma_{G,\Sigma}(n/2))$ (Theorem 6.2). A large class of groups, where this lower bound can be exactly matched by an upper bound, is the class of finitely generated linear groups. Recall that Lipton and Zalcstein [43] and Simon [66] showed that the word problem of a finitely generated linear group can be solved in logarithmic space. Their algorithm can be turned into a randomized streaming algorithm with logarithmic space complexity. In order to plug these streaming algorithms into closure results for randomized streaming space complexity (that are discussed below) we need the notion of a so-called $\epsilon$-distinguisher for $0 \leq \epsilon < 1$. Roughly speaking, a randomized streaming algorithm for a finitely generated group $G$ with finite generating set $\Sigma$ is an *$\epsilon$-distinguisher* if for all words $u, v \in \Sigma^*$ of length at most $n$ the following hold: (i) if $u$ and $v$ evaluate to the same element of $G$ then with probability at least $1 - \epsilon$, $u$ and $v$ lead to the same memory state of the streaming algorithm, and (ii) if $u$ and $v$ evaluate to different elements of $G$ then with probability at least $1-\epsilon$, $u$ and $v$ lead to different memory states of the streaming algorithm; see Section 8. The error probability $\epsilon$ many depend on the input length $n$. It is easy to obtain from an $\epsilon$-distinguisher $\mathcal{R}$ for the group $G$ a randomized streaming algorithm $\mathcal{S}$ for the word problem of $G$ with error probability $\epsilon$. Moreover, the space complexity of $\mathcal{S}$ is only twice the space complexity of $\mathcal{R}$; see Lemma 8.1.

We then show that for every finitely generated linear group $G$ there is an $\epsilon(n)$-distinguisher with space complexity $\mathcal{O}(\log n)$ (Theorem 9.2) and inverse polynomial error probability $\epsilon(n) = 1/n^c$ for any constant $c \geq 1$. If $G$ is moreover virtually nilpotent, then the space complexity can be further reduced to $\mathcal{O}(\log \log n)$ at the cost of an inverse polylogarithmic error probability $1/\log^c n$ (for any constant $c \geq 1$); see Theorem 9.3. In fact, using a known gap theorem for the growth of linear groups [55, 71], it turns out that the randomized streaming space complexity of the word problem for a finitely generated linear group $G$ is either $\Theta(\log \log n)$ (if $G$ is virtually nilpotent) or $\Theta(\log n)$ (if $G$ is not virtually nilpotent), see Theorem 10.3.

For non-linear groups the situation turns out to be more difficult. We show that the existence of low-error distinguishers with logarithmic space complexity is preserved by certain group constructions including finite extensions (Theorem 10.2), graph products (Theorem 10.7) and wreath products by finitely generated abelian groups (Corollary 10.13). Using these transfer results we obtain also non-linear groups with a logarithmic randomized streaming space complexity, e.g., metabelian groups (Corollary 10.5) and free solvable groups (Corollary 10.14).

In Section 12 we prove lower bounds for the randomized streaming space complexity of word problems. For wreath products of the form $H \wr G$ such that $H$ is non-abelian and $G$ is infinite, we can show that the randomized streaming space complexity is $\Theta(n)$ by a reduction from the randomized communication complexity of disjointness (Theorem 11.1). A concrete finitely presented group with randomized streaming space complexity $\Theta(n)$ is Thompson's group $F$ (Corollary 11.3). Thompson's group $F$ (introduced by Richard Thompson in 1965) belongs due to its unusual properties to the most intensively studied infinite groups; see e.g. [12]. From a computational perspective it is interesting to note that $F$ is co-context-free (i.e., the set of all words over any set of generators that do not evaluate to the group identity is a context-free language) [42]. This implies that the word problem for Thompson's group is in $\mathsf{DSPACE}(\log^2 n)$. Finally, we consider the famous Grigorchuk group $G$ [26], which was the first example of a group with intermediate word growth as well as the first example of a group that is amenable but not elementary amenable. We show that the deterministic streaming space complexity of $G$ is $\mathcal{O}(n^{0.768})$, whereas the randomized streaming space complexity of $G$ is $\Omega(n^{1/3})$ (Theorem 11.6).

In the last section of the paper we consider randomized streaming algorithms for subgroup membership problems. In a subgroup membership problem one has a subgroup $H$ of a finitely generated group $G$ and for a given input word $w \in \Sigma^*$ ($\Sigma$ is again a finite set of generators for $G$) one has to determine whether $w$ represents an element of $H$. The word problem is the special case where $H = 1$. We present a randomized streaming algorithm with logarithmic space complexity for the case where $G$ is a finitely generated free group and $H$ is a finitely generated subgroup of $G$ (Theorem 12.4). Moreover, we show that this result extends neither to the case where $H$ is not finitely generated (Theorem 12.5) nor the case where $H$ is a finitely generated subgroup of a direct product of two free groups of rank two (Theorem 12.6).

**Related results.** In this paper, we are only interested in streaming algorithms for a fixed infinite group. Implicitly, streaming algorithms for finite groups are studied in [24]. Obviously, every finite group $G$ has deterministic streaming space complexity

$\mathcal{O}(\log |G|)$.[2] In [24], it is shown that for the group $G = \mathsf{SL}(2, \mathbb{F}_p)$ this upper bound is matched by a lower bound, which even holds for the randomized streaming space complexity. More precisely, Gowers and Viola study the communication cost of the following problem: Alice receives a sequence of elements $a_1, \ldots, a_n \in G$, Bob receives a sequence of elements $b_1, \ldots, b_n \in G$ and they are promised that the interleaved product $a_1 b_1 \cdots a_n b_n$ is either 1 or some fixed element $g \in G \setminus \{1\}$ and their job is to determine which of these two cases holds. For $G = \mathsf{SL}(2, \mathbb{F}_p)$ it is shown that the randomized communication complexity of this problem is $\Theta(\log |G| \cdot n)$ (the upper bound is trivial). From this it follows easily that the randomized streaming space complexity of $\mathsf{SL}(2, \mathbb{F}_p)$ is $\Omega(\log |G|)$.

Our transfer theorems for graph products (Theorem 10.7) and wreath products (Corollary 10.13) have similar counterparts in classical complexity theory: For graph products, the following result is shown in [19]: If the word problem for every group $G_i$ ($1 \leq i \leq k$) can be solved in deterministic logspace on a Turing machine then the same is true for every graph product of the groups $G_1, \ldots, G_k$. Kausch in his thesis [38] strengthened this result by showing that the word problem of the graph product is $\mathsf{AC}_0$-Turing-reducible to the word problems of the $G_i$ and the free group of rank two. A similar result holds for the wreath product: The word problem for the wreath product $G \wr H$ is $\mathsf{AC}_0$-Turing-reducible to the word problems for $G$ and $H$ [52].

The results of this paper where presented at the conferences MFCS 2022 and MFCS 2024; extended abstracts appeared in [46, 47].

## 2. Preliminaries

For integers $a < b$ let $[a, b]$ be the integer interval $\{a, a+1, \ldots, b\}$. We write $[0,1]_{\mathbb{R}}$ for the set $\{r \in \mathbb{R} \colon 0 \leq r \leq 1\}$ of all probabilities. We write $\exp(x)$ for $e^x$, where $e$ is Euler's number.

Let $\Sigma$ be a finite alphabet. As usual we write $\Sigma^*$ for the set of all finite words over the alphabet $w$. The empty word is denoted with $\varepsilon$. For a word $w = a_1 a_2 \cdots a_n$ ($a_1, a_2, \ldots, a_n \in \Sigma$) let $|w| = n$ be its length and $w[i] = a_i$ (for $1 \leq i \leq n$) the symbol at position $i$. A prefix of a word $w$ is a word $u$ such that $w = uv$ for some word $v$. We denote with $\mathcal{P}(w)$ the set of all prefixes of $w$. Let $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ be the set of non-empty words and $\Sigma^{\leq n} = \{w \in \Sigma^* \colon |w| \leq n\}$ be the set of all words of length at most $n$. For a subalphabet $\Theta \subseteq \Sigma$ we denote with $\pi_\Theta : \Sigma^* \to \Theta^*$ the projection homomorphism that deletes all symbols from $\Sigma \setminus \Theta$ in a word: $\pi_\Theta(a) = a$ for $a \in \Theta$ and $\pi_\Theta(a) = \varepsilon$ for $a \in \Sigma \setminus \Theta$.

Several times we will make use of the Chernoff bound. There are many variations of the Chernoff bound, the following form can be found for instance in [20, equation (1)]:

**Theorem 2.1.** *Let $\delta > 0$, $p \in [0,1]_{\mathbb{R}}$, and $X_1, X_2, \ldots X_k$ be independent identically distributed Bernoulli random variables with $\mathsf{Prob}[X_i = 1] = \epsilon$ and $\mathsf{Prob}[X_i = 0] = 1 - \epsilon$ for all $i$. Then we have:*

$$\mathsf{Prob}\left[\sum_{i=1}^{k} X_i > (1+\delta)\epsilon k\right] < \exp\left(-\frac{\delta^2 \epsilon k}{\delta + 2}\right) \overset{\text{if } \delta \geq 1}{\leq} \exp\left(-\frac{\delta \epsilon k}{3}\right). \tag{1}$$

[2] In our setting, $|G|$ would be a constant, but for the moment let us make the dependence on the finite group $G$ explicit.

2.1. **Communication complexity.** Our lower bounds for randomized streaming space complexity will be based on randomized communication complexity. We present the necessary background from communication complexity; see [41] for a detailed introduction. Consider a function $f\colon X \times Y \to \{0,1\}$ for some finite sets $X$ and $Y$. A *randomized (communication) protocol* $P$ for $f$ consists of two parties called Alice and Bob. The input for Alice (resp., Bob) is an element $x \in X$ and a random choice $r \in R$ (resp., $y \in Y$ and a random choice $s \in S$). Here, $R$ and $S$ are finite sets. The goal of Alice and Bob is to compute $f(x,y)$. For this, they communicate in a finite number of rounds, where in each round either Alice sends a bit to Bob or Bob sends a bit to Alice. The protocol determines which of the two communication directions is chosen. At the end, Bob outputs a bit $P(x,r,y,s)$. In a *one-way protocol*, only Alice sends bits to Bob. We assume a probability distribution on the set $R$ (resp., $S$) of Alice's (resp., Bob's) random choices. The protocol $P$ *computes* $f$ if for all $(x,y) \in X \times Y$ we have

$$\mathop{\mathsf{Prob}}_{r\in R, s\in S}[P(x,r,y,s) \neq f(x,y)] \leq \frac{1}{3}. \tag{2}$$

The cost of the protocol is the maximum of the number of transmitted bits, where the maximum is taken over all $(x,r,y,s) \in X \times R \times Y \times S$. The *randomized (one-way) communication complexity* of $f$ is the minimal cost among all (one-way) randomized protocols that compute $f$. Here, the size of the finite sets $R$ and $S$ is not restricted. The choice of the constant $1/3$ in (2) is arbitrary in the sense that changing the constant to any $\lambda < 1/2$ only changes the communication complexity by a constant (depending on $\lambda$), see [41, p. 30]. Also note that we only use the private version of randomized communication protocols, where Alice and Bob make private random choices from the sets $R$ and $S$, respectively, and their choices are not known to the other party (in contrast to the public version of randomized communication protocols).

2.2. **Probabilistic finite automata.** In the following we introduce probabilistic finite automata [60, 61], which will be used as our model for randomized streaming algorithms. A *probabilistic finite automaton* (PFA) $\mathcal{A} = (Q, \Sigma, \iota, \rho, F)$ consists of a finite set of states $Q$, a finite alphabet $\Sigma$, an *initial state distribution* $\iota\colon Q \to [0,1]_{\mathbb{R}}$, a *transition probability function* $\rho\colon Q \times \Sigma \times Q \to [0,1]_{\mathbb{R}}$ and a set of final states $F \subseteq Q$ such that $\sum_{q\in Q} \iota(q) = 1$ and $\sum_{q\in Q} \rho(p,a,q) = 1$ for all $p \in Q$, $a \in \Sigma$. If $\iota$ and $\rho$ map into $\{0,1\}$, then $\mathcal{A}$ is a *deterministic finite automaton* (DFA). If only $\rho$ is required to map into $\{0,1\}$, then $\mathcal{A}$ is called a *semi-probabilitistic finite automaton* (semiPFA). This means that after choosing the initial state according to the distribution $\iota$, $\mathcal{A}$ proceeds deterministically.

Let $\mathcal{A} = (Q, \Sigma, \iota, \rho, F)$ be a PFA. For a random variable $X$ with values from $Q$ and $a \in \Sigma$ we define the random variable $X \cdot a$ (which also takes values from $Q$) by

$$\mathsf{Prob}[X \cdot a = q] = \sum_{p\in Q} \mathsf{Prob}[X = p] \cdot \rho(p,a,q).$$

For a word $w \in \Sigma^*$ we define a random variable $\mathcal{A}(w)$ with values from $Q$ inductively as follows: the random variable $\mathcal{A}(\varepsilon)$ is defined such that $\mathsf{Prob}[\mathcal{A}(\varepsilon) = q] = \iota(q)$ for all $q \in Q$. Moreover, $\mathcal{A}(wa) = \mathcal{A}(w) \cdot a$ for all $w \in \Sigma^*$ and $a \in \Sigma$. Thus, $\mathsf{Prob}[\mathcal{A}(w) = q]$ is the probability that $\mathcal{A}$ is in state $q$ after reading $w$.

We can define $\mathsf{Prob}[\mathcal{A}(w) = q]$ also via runs: A *run* on a word $a_1 \cdots a_m \in \Sigma^*$ in the PFA $\mathcal{A}$ is a sequence $\pi = (q_0, a_1, q_1, a_2, \ldots, a_m, q_m)$ where $q_0, \ldots, q_m \in$

$Q$. We say that $\pi$ ends in $q_m$. Given a run $\pi$ in $\mathcal{A}$ we define $\rho_\iota(\pi) = \iota(q_0) \cdot \prod_{i=1}^{n} \rho(q_{i-1}, a_i, q_i)$. For each $w \in \Sigma^*$ the function $\rho_\iota$ is a probability distribution on the set $\mathrm{Runs}(w)$ of all runs of $\mathcal{A}$ on $w$. Then, $\mathsf{Prob}[\mathcal{A}(w) = q]$ is the sum of all probabilities $\rho_\iota(\pi)$, where $\pi \in \mathrm{Runs}(w)$ ends in $q$.

If $\mathcal{A}$ is a semiPFA then we can identify $\rho$ with a mapping $\rho : Q \times \Sigma \to Q$, where $\rho(p, a)$ is the unique state $q$ with $\rho(p, a, q) = 1$. This mapping $\rho$ is extended to a mapping $\rho : Q \times \Sigma^* \to Q$ in the usual way: $\rho(p, \varepsilon) = p$ and $\rho(p, aw) = \rho(\rho(p, a), w)$ for all $a \in \Sigma$ and $w \in \Sigma^*$. We then obtain

$$\mathsf{Prob}[\mathcal{A}(w) = q] = \sum \{\iota(p) : p \in Q, \rho(p, w) = q\}.$$

For a semiPFA $\mathcal{A} = (Q, \Sigma, \iota, \rho, F)$ and a boolean condition $\mathcal{E}(q)$ that depends on the state $q \in Q$, we define the probability

$$\mathop{\mathsf{Prob}}_{q \in Q}[\mathcal{E}(q)] = \sum_{q \in Q, \mathcal{E}(q)=1} \iota(q).$$

For a language $L \subseteq \Sigma^*$, a PFA $\mathcal{A}$ and a word $w \in \Sigma^*$ we define the *error probability* of $\mathcal{A}$ on $w \in \Sigma^*$ for $L$ as

$$\epsilon(\mathcal{A}, w, L) = \begin{cases} \mathsf{Prob}[\mathcal{A}(w) \notin F] & \text{if } w \in L, \\ \mathsf{Prob}[\mathcal{A}(w) \in F] & \text{if } w \notin L. \end{cases}$$

2.3. **Sequential transducer.** In Section 10.2 we make use of (left-)sequential transducers, see e.g. [10] for more details. A sequential transducer is a tuple $\mathcal{T} = (Q, \Sigma, \Gamma, q_0, \delta)$, where $Q$ is a finite set of states, $\Sigma$ is the input alphabet, $\Gamma$ is the output alphabet, $q_0 \in Q$ is the initial state, and $\delta : Q \times \Sigma \to Q \times \Gamma^*$ is the transition function. If $\delta(q, a) = (p, u)$ then this should be read as follows: if the transducer is in state $q$ and the next input symbol is $a$ then it moves to state $p$ and outputs the word $u$. We extend $\delta$ to a mapping $\delta : Q \times \Sigma^* \to Q \times \Gamma^*$ as follows, where $q \in Q$, $a \in \Sigma$ and $w \in \Sigma^*$:

- $\delta(q, \varepsilon) = (q, \varepsilon)$ for all $q \in Q$, and
- if $\delta(q, a) = (p, u)$ and $\delta(p, w) = (r, v)$ then $\delta(q, aw) = (r, uv)$.

Finally, we define the function $f_{\mathcal{T}} : \Sigma^* \to \Gamma^*$ computed by $\mathcal{T}$ as follows (where $w \in \Sigma^*$ and $x \in \Gamma^*$): $f_{\mathcal{T}}(w) = x$ if and only if $\delta(q_0, w) = (q, x)$ for some $q \in Q$. Intuitively, in order compute $f_{\mathcal{T}}(w)$, $\mathcal{T}$ reads the word $w$ starting in the initial state $q_0$ and thereby concatenates all the outputs produced in the transitions.

## 3. Streaming algorithms: definitions

In this section we define our model of randomized streaming algorithms. It is a non-uniform model in the sense that for every input length $n$ we have a separate algorithm that handles inputs of length at most $n$. Formally, a (non-uniform) *randomized streaming algorithm* is a sequence $\mathcal{R} = (\mathcal{A}_n)_{n \geq 0}$ of PFA $\mathcal{A}_n$ over the same input alphabet $\Sigma$. If every $\mathcal{A}_n$ is deterministic (resp., semi-probabilitistic), we speak of a *deterministic (resp., semi-randomized)* streaming algorithm.

Let $\epsilon_0, \epsilon_1 : \mathbb{N} \to [0, 1]_{\mathbb{R}}$ be monotonically decreasing functions. A randomized streaming algorithm $\mathcal{R} = (\mathcal{A}_n)_{n \geq 0}$ is $(\epsilon_0, \epsilon_1)$-*correct* for a language $L \subseteq \Sigma^*$ if for every large enough $n \geq 0$ and every word $w \in \Sigma^{\leq n}$ we have the following:

- if $w \in L$ then $\epsilon(\mathcal{A}_n, w, L) \leq \epsilon_1(n)$ and
- if $w \notin L$ then $\epsilon(\mathcal{A}_n, w, L) \leq \epsilon_0(n)$.

If $\epsilon_0 = \epsilon_1 =: \epsilon$ then we also say that $\mathcal{R}$ is $\epsilon$-correct for $L$.

We say that $\mathcal{R}$ is a

- *randomized streaming algorithm for* $L$ if it is 1/3-correct for $L$;
- 0*-sided randomized streaming algorithm for* $L$ if it is $(1/3, 0)$-correct for $L$;
- 1*-sided randomized streaming algorithm for* $L$ if it is $(0, 1/3)$-correct for $L$;
- *deterministic streaming algorithm for* $L$ if it is deterministic and 0-correct for $L$;
- *nondeterministic streaming algorithm for* $L$ if it is $(0, \epsilon)$-correct for $L$ for any monotonically decreasing function $\epsilon$ with $0 \leq \epsilon(n) < 1$;
- *co-nondeterministic streaming algorithm for* $L$ if it is $(\epsilon, 0)$-correct for $L$ for any monotonically decreasing function $\epsilon$ with $0 \leq \epsilon(n) < 1$.

The choice of 1/3 for the error probability is not important. Using a standard application of the Chernoff bound, one can make the error probability an arbitrarily small constant; see Theorem 4.1 below.

The *space complexity* of the randomized streaming algorithm $\mathcal{R} = (\mathcal{A}_n)_{n \geq 0}$ is the function $s(\mathcal{R}, n) = \lceil \log_2 |Q_n| \rceil$, where $Q_n$ is the state set of $\mathcal{A}_n$. The motivation for this definition is that states of $Q_n$ can be encoded by bit strings of length at most $\lceil \log_2 |Q_n| \rceil$. The *randomized streaming space complexity of the language* $L$ is the smallest possible function $s(\mathcal{R}, n)$, where $\mathcal{R}$ is a randomized streaming algorithm for $L$. In an analogous way we define the 0-sided (resp., 1-sided) randomized streaming space complexity, the deterministic streaming space complexity, and the (co-)nondeterministic streaming space complexity of a language $L$.

The (non)deterministic streaming space complexity of a language $L$ is directly linked to the *automaticity of* $L$. The automaticity of $L \subseteq \Sigma^*$ is the function $A_L(n)$ that maps $n$ to the number of states of a smallest DFA $\mathcal{A}_n$ such that for all words $w \in \Sigma^{\leq n}$ we have: $w \in L$ if and only if $w$ is accepted by $\mathcal{A}_n$. If we allow the automata $\mathcal{A}_n$ to be nondeterministic then we obtain the *nondeterministic automaticity* $N_L(n)$ *of* $L$. Hence, the deterministic (resp., nondeterministic) streaming space complexity of $L$ is exactly $\lceil \log_2 A_L(n) \rceil$ (resp., $\lceil \log_2 N_L(n) \rceil$). The (nondeterministic) automaticity of languages was studied in [23, 65]. Interesting in our context is the following result of Karp [37]: if $L$ is a non-regular language then $A_L(n) \geq (n+3)/2$ for infinitely many $n$. Hence, for every non-regular language the deterministic streaming space complexity of $L$ is lower bounded by $\log_2(n) - c$ for a constant $c$ and infinitely many $n$.

As remarked before, our model of streaming algorithms is non-uniform in the sense that for every input length $n$ we have a separate streaming algorithm $\mathcal{A}_n$.[3] This makes lower bounds of course stronger. On the other hand, the streaming algorithms that we construct for concrete groups will be mostly uniform in the sense that there is an efficient algorithm that constructs from a given $n$ the PFA $\mathcal{A}_n$.

## 4. Streaming algorithms: general results

Before we investigate streaming algorithms for word problems we prove a few general results that are of independent interest. Let us first prove that (as stated above) the error probability of a randomized streaming algorithm can be pushed

[3]This is analogous to circuit complexity, where for every input length $n$ one has a separate boolean circuits with $n$ input gates.

down to any constant $\epsilon > 0$ at the cost of an additional constant factor in the space complexity:

**Theorem 4.1.** *Let $r : \mathbb{N} \to \mathbb{N}$ a monotonic function and $\mathcal{R}$ a randomized streaming algorithm such that $\mathcal{R}$ is $\frac{1}{3}$-correct for the language $L$. Then there exists a randomized streaming algorithm $\mathcal{S}$ such that $s(\mathcal{S}, n) = r(n) \cdot s(\mathcal{R}, n)$ and $\mathcal{S}$ is $\exp(-r(n)/30)$-correct for the language $L$.*

*Proof.* Let $\mathcal{R} = (\mathcal{A}_n)_{n\geq 0}$ with $\mathcal{A}_n = (Q_n, \Sigma, \iota_n, \rho_n, F_n)$. We use the standard idea of running (in parallel) $r(n)$ copies of $\mathcal{A}_n$ and making a majority vote at the end. Formally, for an $n \geq 0$ and $k \geq 1$ we define the semiPFA $\mathcal{A}_n^k$ as follows:

- $\mathcal{A}_n^k = (Q_n^k, \Sigma, \iota_n^k, \rho_n^k, F_{n,k})$,
- $\iota_n^k(q_1, \ldots, q_k) = \prod_{1\leq i\leq k} \iota_n(q_i)$,
- $\rho_n^k((p_1, \ldots, p_k), a, (q_1, \ldots, q_k)) = \prod_{1\leq i\leq k} \rho_n(p_i, a, q_i)$, and
- $F_{n,k} = \{(q_1, \ldots, q_k) \colon q_i \in F_n \text{ for more than } k/2 \text{ many } i \in [1, k]\}$.

We then define the new randomized streaming algorithm $\mathcal{R}^r = (\mathcal{A}_n^{r(n)})_{n\geq 0}$. In order to bound the error probability of $\mathcal{R}^r$ by $\exp(-r(n)/30)$ we have to show that $\epsilon(\mathcal{A}_n^k, w, L) \leq \exp(-k/30)$ for every input word $w \in \Sigma^{\leq n}$. For this we introduce identically distributed independent Bernoulli random variables $X_1, \ldots, X_k$ with $\mathsf{Prob}[X_i = 1] = \frac{1}{3}$. Then, for every $w \in \Sigma^{\leq n}$ we have:

$$\epsilon(\mathcal{A}_n^k, w, L) \leq \mathsf{Prob}\left[\sum_{i=1}^{k} X_i > \frac{k}{2}\right].$$

Let $\delta = \frac{1}{2}$. With $\epsilon = \frac{1}{3}$ we obtain with the Chernoff bound (1):

$$\mathsf{Prob}\left[\sum_{i=1}^{k} X_i > \frac{k}{2}\right] = \mathsf{Prob}\left[\sum_{i=1}^{k} X_i > (1+\delta)k\epsilon\right] < \exp\left(-\frac{\delta^2\epsilon k}{\delta+2}\right) = \exp\left(-\frac{k}{30}\right).$$

The space complexity of $\mathcal{R}^r$ is clearly $r(n)$ times the space complexity of $\mathcal{R}$. □

Let $\mathcal{A} = (Q, \Sigma, \iota, \rho, F)$ be a PFA and $0 < \delta < 1$, $\epsilon > 0$. We say that $\delta$ is an $\epsilon$-isolated cutpoint for $\mathcal{A}$ if for all words $w \in \Sigma^*$ we have

$$|\,\mathsf{Prob}[\mathcal{A}(w) \in F] - \delta| \geq \epsilon. \tag{3}$$

The language $L(\mathcal{A}, \delta)$ accepted by $\mathcal{A}$ with cut-point $\delta$ is the set of all words $w$ with $\mathsf{Prob}[\mathcal{A}(w) \in F] > \delta$. Paz stated in [59, Theorem 30'] that in this situation there exists a DFA for $L(\mathcal{A}, \delta)$ with $(1 + 1/2\epsilon)^{|Q|-1}$ states. A proof can be found in [60, p. 160]; it uses the proof technique for a slightly weaker result of Rabin [61, Theorem 3]. Paz's proof easily yields the following result:

**Theorem 4.2.** *Let $L$ be a language with randomized streaming space complexity $S(n)$. Then the deterministic streaming space complexity of $L$ is bounded by $2^{S(n)+1}$.*

*Proof.* Let $\mathcal{R} = (\mathcal{A}_n)$ be a randomized streaming algorithm for $L$ such that $S(n) = \lceil \log_2 |Q_n| \rceil$. Fix an $n$ and set $\delta = 1/2$ and $\epsilon = 1/6$, so that $\delta - \epsilon = 1/3$ and $\delta + \epsilon = 2/3$. We cannot directly apply the above mentioned result of Paz since $1/2$ is not necessarily a $1/6$-isolated cut-point for $\mathcal{A}_n$: (3) only has to hold for words $w$ of length at most $n$. But we can argue as follows: Recall the automaticity function $A_L(n)$ of the language $L$ from Section 3. Then the deterministic streaming space complexity of $L$ is $\lceil \log_2 A_L(n) \rceil$.

It is shown in [34] (see also [65]) that $A_L(n)$ is the maximal number $k$ for which there exist words $v_1, \ldots, v_k \in \Sigma^{\leq n}$ such that for all $i, j \in [1, k]$ with $i < j$ there exists a word $w_{i,j} \in \Sigma^*$ such that $v_i w_{i,j}, v_j w_{i,j} \in \Sigma^{\leq n}$ and $v_i w_{i,j} \in L$ if and only if $v_j w_{i,j} \notin L$.

Assume now that $k = A_L(n)$ and fix the above words $v_i$ and $w_{i,j}$. Consider $i, j \in [1, k]$ with $i < j$. Since $v_i w_{i,j}, v_j w_{i,j} \in \Sigma^{\leq n}$ and $v_i w_{i,j} \in L$ if and only if $v_j w_{i,j} \notin L$ we get

$$|\,\mathsf{Prob}[\mathcal{A}_n(v_i w_{i,j}) \in F] - \mathsf{Prob}[\mathcal{A}_n(v_i w_{i,j}) \in F]| \geq 2\epsilon = \frac{1}{3}.$$

whenever $i < j$. In the proof of [59, Theorem 30'] (see [60, p. 160]) it is shown that this implies

$$k \leq (1 + 1/2\epsilon)^{|Q_n|-1} = 4^{|Q_n|-1}.$$

We obtain $A_L(n) \leq 4^{|Q_n|} \leq 4^{2^{S(n)}}$ and hence $\lceil \log_2 A_L(n) \rceil \leq 2^{S(n)+1}$. □

We now turn to the connection between randomized and semi-randomized streaming algorithms. Our next result states that a randomized streaming algorithm can be transformed into an equivalent semi-randomized streaming algorithms with a moderate blow-up in the space complexity.

**Theorem 4.3.** *Let $0 < \epsilon(n) < 1/4$ for all $n \geq 0$ and let $\mathcal{R}$ be a randomized streaming algorithm which is $\epsilon(n)$-correct for the language $L$. Then there is a semi-randomized streaming algorithm $\mathcal{S}$ with $s(\mathcal{S}, n) = s(\mathcal{R}, n) + \Theta(\log n + \log(1/\epsilon(n)))$ and $\mathcal{S}$ is $2\epsilon(n)$-correct for the language $L$.*

*Proof.* Let $\mathcal{R} = (\mathcal{A}_n)_{n \geq 0}$. Let us fix an $n$ and consider the PFA $\mathcal{A}_n = (Q, \Sigma, \iota, \rho, F)$. We first transform $\mathcal{A}_n$ into an acyclic PFA $\mathcal{A}'_n = (Q', \Sigma, \iota', \rho', F')$, where acyclic means that for every run $\pi = (q_0, a_1, \ldots, a_m, q_m)$ of $\mathcal{A}'_n$ such that $m \leq n$ and $q_i = q_j$ for some $i < j$ we have $\rho_\iota(\pi) = 0$. We define the components of $\mathcal{A}'_n$ as follows:

- $Q' = Q \times [0, n]$
- $F' = F \times [0, n]$
- For all states $(p, i), (q, i+1) \in Q'$ and all $a \in \Sigma$ we set $\rho'((p, i), a, (q, i+1)) = \rho(p, a, q)$. Moreover, $\rho'((p, i), a, (q, j)) = 0$ if $i < n$ and $j \neq i + 1$.
- For states $(p, n)$ we define $\rho'$ arbitrarily. Let us set $\rho'((p, n), a, (p, n)) = 1$ for all $a \in \Sigma$ and $\rho'((p, n), a, (q, i)) = 0$ whenever $(q, i) \neq (p, n)$.
- For all states $(q, 0) \in Q'$ we set $\iota'(q, 0) = \iota(q)$. Moreover, $\iota'(q, i) = 0$ if $i > 0$.

The randomized streaming algorithm $\mathcal{R}' = (\mathcal{A}'_n)_{n \geq 0}$ is also $\epsilon(n)$-*correct* for the language $L$. Moreover the space complexity of $\mathcal{R}'$ is $s(\mathcal{R}, n) + \lceil \log_2(n + 1) \rceil$.

We now define a random variable $\mathsf{D}_n$, whose value is a DFA $(Q', \Sigma, (q_0, 0), \delta, F')$, as follows:

- For every state $(p, i)$ of $\mathcal{A}'_n$ and every $a \in \Sigma$ we choose a state $(q, j)$ with probability $\rho'((p, i), a, (q, j))$ and define $\delta((p, i), a) = (q, j)$.
- The initial state $(q_0, 0) \in Q'$ is chosen with probability $\iota'(q_0, 0)$.

The above choices are made independently. Let $\mathsf{sup}(\mathsf{D}_n)$ be the support of $\mathsf{D}_n$ (the set of DFAs that have non-zero probability).

For every fixed word $w \in \Sigma^*$ with $|w| \leq n$ and $\mathcal{D} \in \mathsf{sup}(\mathsf{D}_n)$ define $Z[w, \mathcal{D}] \in \{0, 1\}$ by $Z[w, \mathcal{D}] = 1$ if and only if $w \in L \setminus L(\mathcal{D}) \cup L(\mathcal{D}) \setminus L$. In other words:

$Z[w, \mathcal{D}] = 1$ if and only if $\mathcal{D}$ makes an error (with respect to the language $L$) on the word $w$. For the expected value of $Z[w, \mathcal{D}]$ we obtain

$$\mathsf{E}[w] := \sum_{\mathcal{D} \in \mathsf{sup}(\mathsf{D}_n)} \mathsf{Prob}[\mathsf{D}_n = \mathcal{D}] \cdot Z[w, \mathcal{D}] \leq \epsilon(n),$$

because the left-hand side of the inequality is exactly the error probability of $\mathcal{A}'_n$ on $w$. For this, it is important that we construct the DFAs from the acyclic PFA $\mathcal{A}'_n$: in our original PFA $\mathcal{A}_n$, there could be a run of the form $\pi = (\ldots, p, a, q, \ldots, p, a, q', \ldots)$ with $q \neq q'$ and $\rho_\iota(\pi) > 0$. But runs of this form cannot occur in a DFA.

The rest of the proof follows the arguments from the proof of Newman's theorem from communication complexity, see e.g. [41]. Fix a number $t$ that will be suitably chosen later. For a $t$-tuple of DFAs $\overline{\mathcal{D}} = (\mathcal{D}_{n,1}, \ldots, \mathcal{D}_{n,t})$ from $\mathsf{sup}(\mathsf{D}_n)$ we construct a semi-probabilistic automaton $\mathcal{S}(\overline{\mathcal{D}})$ by taking the disjoint union of the $\mathcal{D}_{n,i}$. To define the initial state distribution $\bar{\iota}$ of $\mathcal{S}(\overline{\mathcal{D}})$, let $q_{0,i}$ be the initial state of $\mathcal{D}_{n,i}$. Then we set $\bar{\iota}(q_{0,i}) = 1/t$. Thus, the starting state of a run in $\mathcal{S}(\overline{\mathcal{D}})$ is chosen uniformly among the initial states of the $\mathcal{D}_{n,i}$.

We show that there exists a $t$-tuple $\overline{\mathcal{D}}$ of the above form such that for every input word $w \in \Sigma^{\leq n}$ the error probability of $\mathcal{S}_n := \mathcal{S}(\overline{\mathcal{D}})$ on $w$ is at most $2\epsilon(n)$. Then $(\mathcal{S}_n)_{n \geq 0}$ is the desired semi-randomized streaming algorithm from the theorem.

Fix again an input word $w \in \Sigma^{\leq n}$ and a $t$-tuple $\overline{\mathcal{D}} = (\mathcal{D}_{n,1}, \ldots, \mathcal{D}_{n,t})$. Then the error probability of $\mathcal{S}(\overline{\mathcal{D}})$ on $w$ is

$$\epsilon(\mathcal{S}(\overline{\mathcal{D}}), w, L) = \frac{1}{t} \cdot \sum_{i=1}^{t} Z[w, \mathcal{D}_{n,i}].$$

We now choose the tuple $\overline{\mathcal{D}} = (\mathcal{D}_{n,1}, \ldots, \mathcal{D}_{n,t})$ randomly by taking $t$ independent copies of the random variable $\mathsf{D}_n$. With the Chernoff bound (1) and $\mathsf{E}[w] \leq \epsilon(n)$ (i.e., $\frac{2\epsilon(n)}{\mathsf{E}[w]} - 1 \geq 1$) we obtain

$$\begin{aligned}
& \mathsf{Prob}\left[\frac{1}{t} \cdot \sum_{i=1}^{t} Z[w, \mathcal{D}_{n,i}] > 2\epsilon(n)\right] \\
= \; & \mathsf{Prob}\left[\sum_{i=1}^{t} Z[w, \mathcal{D}_{n,i}] > \left(1 + \frac{2\epsilon(n)}{\mathsf{E}[w]} - 1\right) \cdot \mathsf{E}[w] \cdot t\right] \\
\leq \; & \exp\left(-\frac{2\epsilon(n)/\mathsf{E}[w] - 1}{3} \cdot \mathsf{E}[w] \cdot t\right) \\
= \; & \exp\left(\frac{-2\epsilon(n) + \mathsf{E}[w]}{3} \cdot t\right) \\
\leq \; & \exp\left(-\frac{\epsilon(n) \cdot t}{3}\right).
\end{aligned}$$

By the union bound, the probability that $\epsilon(\mathcal{S}(\overline{\mathcal{D}}), w, L) > 2\epsilon(n)$ for some word $w \in \Sigma^*$ of length at most $n$ (where $\overline{\mathcal{D}}$ is randomly chosen using $t$ independent copies of the random variable $\mathsf{D}_n$) is bounded by

$$|\Sigma|^{n+1} \cdot \exp(-\epsilon(n) \cdot t/3) = \exp(\ln |\Sigma| \cdot (n+1) - \epsilon(n) \cdot t/3).$$

If we choose $t = 3(n+1)\ln|\Sigma|/\epsilon(n) + \mathcal{O}(1)$ then this probability is strictly below 1. With such a $t$ the space complexity of $\mathcal{S}(\overline{\mathcal{D}})$ becomes $s(\mathcal{R}', n) + \lceil \log_2 t \rceil = s(\mathcal{R}, n) + \Theta(\log n + \log(1/\epsilon(n)))$. □

Note that if $s(\mathcal{R}, n) \geq \Omega(\log n)$ and $\epsilon(n) \geq \Omega(1/n^c)$ for some constant $c \geq 1$ then $s(\mathcal{S}, n) = \Theta(s(\mathcal{R}, n))$ in Theorem 4.3. Also notice that the proof of Theorem 4.3 uses non-uniformity in a crucial way.

Our final result in this section is a trade-off between the space complexity and the error probability for semi-randomized streaming algorithms:

**Theorem 4.4.** *Let $s(n)$ be the deterministic streaming space complexity of the language $L \subseteq \Sigma^*$ and let $\mathcal{R} = (\mathcal{A}_n)_{n\geq 0}$ be a semi-randomized streaming algorithm that is $\epsilon(n)$-correct for the language $L$. Then for every large enough $n \geq 0$ we have*

$$s(\mathcal{R}, n) \geq \min\{s(n), \log_2(1/\epsilon(n))\}.$$

*Proof.* Fix an $n \geq 0$ large enough such that for every word $w \in \Sigma^{\leq n}$ the error probability $\epsilon(\mathcal{A}_n, w, L)$ is bounded by $\epsilon(n)$. Let $\mathcal{A}_n = (Q_n, \Sigma, \iota_n, \rho_n, F_n)$. Hence, we have $s(\mathcal{R}, n) = \lceil \log_2 |Q_n| \rceil$. If $s(\mathcal{R}, n) \geq \log_2(1/\epsilon(n))$ then we are done. Therefore, assume that $1/\epsilon(n) > 2^{s(\mathcal{R},n)} \geq |Q_n|$, i.e., $\epsilon(n) < 1/|Q_n|$. There must exist a state $q_n \in Q_n$ with $\iota_n(q_n) \geq 1/|Q_n| > \epsilon(n)$. Consider the DFA $\mathcal{B}_n = (Q_n, \Sigma, q_n, \rho_n, F_n)$. If there is a word $w \in \Sigma^{\leq n}$ such that $w \in L(\mathcal{B}_n) \Leftrightarrow w \notin L$ then we would have $\epsilon(n) \geq \iota_n(q_n)$, which yields a contradition. Therefore we have $L(\mathcal{B}_n) \cap \Sigma^{\leq n} = L \cap \Sigma^{\leq n}$. Since $\mathcal{B}_n$ is a DFA with state set $Q_n$, we get $s(\mathcal{R}, n) \geq s(n)$. □

## 5. Groups and word problems

Let $G$ be a group. The identity element will be always denoted with 1. For a subset $\Sigma \subseteq G$, we denote with $\langle\Sigma\rangle$ the subgroup of $G$ generated by $\Sigma$. It is the set of all products of elements from $\Sigma \cup \Sigma^{-1}$. It can be also defined as the smallest (w.r.t. inclusion) subgroup of $G$ that contains $\Sigma$. Similarly, the *normal closure* $N(\Sigma)$ of $\Sigma$ is smallest normal subgroup of $G$ that contains $\Sigma$. It can be also defined as the subgroup $\langle\{g^{-1}ag : a \in \Sigma, g \in G\}\rangle$. We can then construct the quotient group $G/N(\Sigma)$. The *commutator* of $g, h \in G$ is the element $[g, h] = ghg^{-1}h^{-1}$ and for subsets $A, B \subseteq G$ we write $[A, B]$ for the subgroup $\langle\{[a, b] : a \in A, b \in B\}\rangle$.

In this paper, we only consider *finitely generated (f.g.) groups*. The group $G$ is finitely generated if there is a finite set $\Sigma \subseteq G$ such that $G = \langle\Sigma\rangle$. In this situation, $\Sigma$ is called a *finite generating set* for $G$. If $\Sigma = \Sigma^{-1}$ then we say that $\Sigma$ is a *finite symmetric generating set* for $G$. In the following we assume that all finite generating sets are symmetric. Every word $w \in \Sigma^*$ evaluates to a group element $\pi_G(w)$ in the natural way. Here $\pi_G : \Sigma^* \to G$ is the unique morphism from the free monoid $\Sigma^*$ to $G$ such that $\pi_G(a) = a$ for all $a \in \Sigma$. Instead of $\pi_G(u) = \pi_G(v)$ we also write $u \equiv_G v$. For a word $u = a_1 a_2 \cdots a_n \in \Sigma^*$ with $a_i \in \Sigma$ we define the word $u^{-1} = a_n^{-1} \cdots a_2^{-1} a_1^{-1} \in \Sigma^*$. Clearly, we have $\pi_G(u^{-1}) = \pi_G(u)^{-1}$.

Let $C(G, \Sigma)$ be the *Cayley graph* of $G$ with respect to the finite symmetric generating set $\Sigma$. It is the edge-labelled graph whose vertex set is $G$ and that has an $a$-labelled edge from $\pi_G(u)$ to $\pi_G(ua)$ for all $u \in \Sigma^*$ and $a \in \Sigma$. Let $\mathsf{WP}(G, \Sigma) = \{w \in \Sigma^* : \pi_G(w) = 1\}$ be the *word problem for $G$* with respect to the generating set $\Sigma$.

Next we introduce free groups and some related concepts. Fix a finite alphabet $\Gamma$ and take a copy $\Gamma^{-1} = \{a^{-1} : a \in \Gamma\}$ of formal inverses. Let $\Sigma = \Gamma \cup \Gamma^{-1}$.

We extend the mapping $a \mapsto a^{-1}$ ($a \in \Gamma$) to the whole alphabet $\Sigma$ by setting $(a^{-1})^{-1} = a$. For a word $u \in \Sigma^*$ the word $u^{-1}$ is defined as above. A word $u \in \Sigma^*$ is called reduced if it contains no factor of the form $aa^{-1}$ for $a \in \Sigma$. Let $\mathsf{Red}(\Sigma) \subseteq \Sigma^*$ be the set of reduced words. The *free group* $F(\Gamma)$ can be defined as the set $\mathsf{Red}(\Sigma)$ of reduced words together with the following multiplication operation: Let $u, v \in \mathsf{Red}(\Sigma)$. Then one can uniquely write $u$ and $v$ as $u = xy$ and $v = y^{-1}z$ such that $xz \in \mathsf{Red}(\Sigma)$ and define the product of $u$ and $v$ in the free group $F(\Gamma)$ as $xz$. For every word $w \in \Sigma^*$ we can define a unique reduced word $\mathsf{red}(w)$ as follows: if $w \in \mathsf{Red}(\Sigma)$ then $\mathsf{red}(w) = w$ and if $w = uaa^{-1}v$ for $u, v \in \Sigma^*$ and $a \in \Sigma$ then $\mathsf{red}(w) = \mathsf{red}(uv)$. It is important that this definition does not depend on which factor $aa^{-1}$ is deleted in $w$. The reduction relation $uaa^{-1}v \to uv$ for all $u, v \in \Sigma^*$ and $a \in \Sigma$ is a so-called confluent relation. The reduction mapping $w \mapsto \mathsf{red}(w)$ then becomes the unique morphism mapping a word $w \in \Sigma^*$ to the element of the free group represented by $w$.

Group presentations are a common way to describe groups. Let $\Gamma$ and $\Sigma$ as in the previous paragraph and let $R \subseteq F(\Gamma)$. Then the quotient group $F(\Gamma)/N(R)$ is also denoted by $\langle \Gamma \mid R \rangle$ and the pair $(\Gamma, R)$ is called a *group presentation*. The group $\langle \Gamma \mid R \rangle$ is finitely generated (since we assume $\Gamma$ to be finite) and every f.g. group can be written in this form. If also $R$ is finite then the group $\langle \Gamma \mid R \rangle$ is called *finitely presented*.

We are interested in streaming algorithms for word problems $\mathsf{WP}(G, \Sigma)$. The following lemma is simple but important:

**Lemma 5.1.** *Let $\Sigma_1$ and $\Sigma_2$ be finite symmetric generating sets for the group $G$ and let $s_i(n)$ be the deterministic/randomized streaming space complexity of $\mathsf{WP}(G, \Sigma_i)$. Then there exists a constant $c$ that depends on $G$, $\Sigma_1$ and $\Sigma_2$ such that $s_1(n) \leq s_2(c \cdot n)$.*

*Proof.* For every generator $a \in \Sigma_1$ there is a word $w_a \in \Sigma_2^*$ such that $\pi_G(a) = \pi_G(w_a)$. Let $c = \max\{|w_a| : a \in \Sigma_1\}$ and let $\phi : \Sigma_1^* \to \Sigma_2^*$ be the homomorphism with $\phi(a) = w_a$ for $a \in \Sigma_1$. Let $\mathcal{R}_2 = (\mathcal{A}_{2,n})_{n \geq 0}$ be a deterministic/randomized streaming algorithm for the language $\mathsf{WP}(G, \Sigma_2)$ with space complexity $s_2(n)$. We obtain a deterministic/randomized streaming algorithm $\mathcal{R}_1 = (\mathcal{A}_{1,n})_{n \geq 0}$ for $\mathsf{WP}(G, \Sigma_1)$ as follows: on input $w \in \Sigma_1^{\leq n}$, the PFA $\mathcal{A}_{1,n}$ simulates the PFA $\mathcal{A}_{2,c \cdot n}$ on the input word $\phi(w) \in \Sigma_2^{\leq c \cdot n}$. This yields a deterministic/randomized streaming algorithm for $\mathsf{WP}(G, \Sigma_1)$ with space complexity $s_2(c \cdot n)$. □

By Lemma 5.1, the dependence of the streaming space complexity from the generating set is often blurred by the use of the Landau notation. In such situations we will speak of the deterministic/randomized streaming space complexity for the group $G$ (instead of the deterministic/randomized streaming space complexity of the language $\mathsf{WP}(G, \Sigma)$).

## 6. Streaming algorithms for word problems and growth

Let $G$ be a finitely generated group and let $\Sigma$ be a finite symmetric generating set for $G$. For $n \in \mathbb{N}$ let $B_{G,\Sigma}(n) = \pi_G(\Sigma^{\leq n}) \subseteq G$ be the ball of radius $n$ in the Cayley graph $C(G, \Sigma)$ with center 1. The growth function $\gamma_{G,\Sigma} : \mathbb{N} \to \mathbb{N}$ is defined by

$$\gamma_{G,\Sigma}(n) = |B_{G,\Sigma}(n)|$$

for all $n \geq 0$. For different finite generating sets $\Sigma_1, \Sigma_2$ of $G$ the functions $\gamma_{G,\Sigma_1}$ and $\gamma_{G,\Sigma_2}$ are different, but their asymptotic behavior is the same; see e.g. [51, Proposition 1.3] for a precise statement.

The (non)deterministic streaming space complexity of $G$ is directly linked to the growth of $G$ by the following theorem.

**Theorem 6.1.** *Let $G$ be a finitely generated infinite group and let $\Sigma$ be a finite symmetric generating set for $G$. Define the function $S(n)$ by*

$$S(n) = \begin{cases} \gamma_{G,\Sigma}(n/2) & \text{if } n \text{ is even,} \\ \gamma_{G,\Sigma}(\lfloor n/2 \rfloor) + 1 & \text{if } n \text{ is odd.} \end{cases} \tag{4}$$

*Then, the deterministic streaming space complexity of $\mathsf{WP}(G,\Sigma)$ is $\lceil \log_2 S(n) \rceil$ and the nondeterministic streaming space complexity of $\mathsf{WP}(G,\Sigma)$ is $\lceil \log_2 \gamma_{G,\Sigma}(\lfloor n/2 \rfloor) \rceil$.*

*Proof.* We start with the upper bound for the deterministic streaming space complexity in case $n$ is even. In the following we identify the ball $B_{G,\Sigma}(n/2)$ with its induced subgraph of the Cayley graph $C(G,\Sigma)$. We define a deterministic finite automaton $\mathcal{A}_n$ by taking the edge-labelled graph $B_{G,\Sigma}(n/2)$ with the initial and unique final state 1. It can be viewed as a partial DFA in the sense that for every $g \in B_{G,\Sigma}(n/2)$ and every $a \in \Sigma$, $g$ has at most one outgoing edge labelled with $a$ (that leads to $ga$ if $ga \in B_{G,\Sigma}(n/2)$). In order to add the missing transitions we choose an element $g_f \in B_{G,\Sigma}(n/2) \setminus B_{G,\Sigma}(n/2-1)$ (here, we set $B_{G,\Sigma}(-1) = \emptyset$). Such an element exists because $G$ is infinite. If $g \in B_{G,\Sigma}(n/2)$ has not outgoing $a$-labelled edge in $B_{G,\Sigma}(n/2)$ then we add an $a$-labelled edge from $g$ to $g_f$. We call those edges *spurious*. The resulting DFA is $\mathcal{A}_n$.

We claim that for every word $w \in \Sigma^{\leq n}$, $w$ is accepted by $\mathcal{A}_n$ if and only if $w \in \mathsf{WP}(G,\Sigma)$. This is clear, if no spurious edge is traversed while reading $w$ into $\mathcal{A}_n$. In this case, after reading $w$, we end up in state $\pi_G(w)$. Now assume that a spurious edge is traversed while reading $w$ into $\mathcal{A}_n$ and let $x$ be the shortest prefix of $w$ such that a spurious edge is traversed while reading the last symbol of $x$. Let us write $w = xy$. We must have $|x| > n/2$ and $\pi_G(x) \notin B_{G,\Sigma}(n/2)$. Moreover, $|y| < n - n/2 = n/2$. Since $\pi_G(x) \notin B_{G,\Sigma}(n/2)$, we have $w = xy \notin \mathsf{WP}(G,\Sigma)$. Moreover, $w$ is rejected by $\mathcal{A}_n$, because $x$ leads in $\mathcal{A}_n$ from the initial state 1 to state $g_f$ and there is no path of length at most $n/2-1$ from $g_f$ back to the final state 1.

For the case that $n$ is odd, we take the ball $B_{G,\Sigma}(\lfloor n/2 \rfloor)$. Instead of adding spurious edges we add a failure state $f$. If $g \in B_{G,\Sigma}(\lfloor n/2 \rfloor)$ has no outgoing $a$-labelled edge in $B_{G,\Sigma}(\lfloor n/2 \rfloor)$, then we add an $a$-labelled edge from $g$ to $f$. Moreover, for every $a \in \Sigma$ we add an $a$-labelled loop at state $f$. As for the case $n$ even, one can show that the resulting DFA accepts a word $w \in \Sigma^{\leq n}$ if and only if $w \in \mathsf{WP}(G,\Sigma)$.

The upper bound for the nondeterministic streaming space complexity follows with the same arguments. Notice that the failure state $f$ in case $n$ is odd is not needed in a nondeterministic automaton.

For the lower bound we start with the nondeterministic streaming space complexity. Let $k = \gamma_{G,\Sigma}(\lfloor n/2 \rfloor)$ and choose words $w_1, \ldots, w_k$ such that $|w_i| \leq \lfloor n/2 \rfloor$ for all $i \in [1,k]$ and $w_i \not\equiv_G w_j$ whenever $i \neq j$. Then for every $i \in [1,k]$ we have $w_i w_i^{-1} \in \mathsf{WP}(G,\Sigma)$ and $w_j w_i^{-1} \notin \mathsf{WP}(G,\Sigma)$ for all $j \in [1,k] \setminus \{i\}$. Moreover, $|w_j w_i^{-1}| \leq n$ for all $i,j \in [1,k]$. In the language of [23], $\{w_1, \ldots, w_k\}$ is a set

of uniformly $n$-dissimilar words. By [23, Lemma 3.1] this implies that the nondeterministic automaticity of $\mathsf{WP}(G,\Sigma)$ satisfies $N_{\mathsf{WP}(G,\Sigma)}(n) \geq \gamma_{G,\Sigma}(\lfloor n/2 \rfloor)$. This shows the lower bound on the nondeterministic streaming space complexity.

For the lower bound on the deterministic streaming space complexity, let $\mathcal{A} = (Q,\Sigma,q_0,\delta,F)$ be a smallest DFA such that $L(\mathcal{A}) \cap \Sigma^{\leq n} = \mathsf{WP}(G,\Sigma) \cap \Sigma^{\leq n}$. We have to show that $|Q| \geq S(n)$ for $S(n)$ from (4). Let us consider two words $u, v \in \Sigma^*$ of length at most $\lfloor n/2 \rfloor$ such that $u \not\equiv_G v$ and $\delta(q_0,u) = \delta(q_0,v)$. We then have $uv^{-1} \notin \mathsf{WP}(G,\Sigma)$ and $vv^{-1} \in \mathsf{WP}(G,\Sigma)$. On the other hand, we have $\delta(q_0,uv^{-1}) = \delta(q_0,vv^{-1})$, which is a contradiction (note that $|uv^{-1}|, |vv^{-1}| \leq n$). Hence, if $\delta(q_0,u) = \delta(q_0,v)$ for two words $u,v \in \Sigma^*$ of length at most $\lfloor n/2 \rfloor$, then $u \equiv_G v$.

Let $Q' = \{\delta(q_0,w) \colon w \in \Sigma^*, |w| \leq \lfloor n/2 \rfloor\} \subseteq Q$. The previous paragraph shows that $|Q'| \geq \gamma_{G,\Sigma}(\lfloor n/2 \rfloor)$. If $n$ is even then $\lfloor n/2 \rfloor = n/2$ and we are done. So, let us assume that $n$ is odd.

If $|Q| > \gamma_{G,\Sigma}(\lfloor n/2 \rfloor)$ then we are again done. So, let us assume that $Q = Q'$ and $|Q| = \gamma_{G,\Sigma}(\lfloor n/2 \rfloor)$. Then, to every state $q \in Q$ we can assign a unique group element $g_q \in B_{G,\Sigma}(\lfloor n/2 \rfloor)$ such that for every word $w \in \Sigma^*$ with $|w| \leq \lfloor n/2 \rfloor$ we have $\delta(q_0,w) = q$ if and only if $\pi_G(w) = g_q$. The mapping $q \mapsto g_q$ is a bijection between $Q$ and $B_{G,\Sigma}(\lfloor n/2 \rfloor)$.

Let us now take a state $q \in Q$ and a generator $a \in \Sigma$ such that $g_q \cdot a \notin B_{G,\Sigma}(\lfloor n/2 \rfloor)$. Such a state and generator must exist since $G$ is infinite. Let $u,v \in \Sigma^*$ be words of length at most $\lfloor n/2 \rfloor$ such that $\delta(q_0,u) = q$ and $\delta(q_0,v) = \delta(q,a) = \delta(q_0,ua)$. We obtain $\delta(q_0,vv^{-1}) = \delta(q_0,uav^{-1})$. But $vv^{-1} \in \mathsf{WP}(G,\Sigma)$ and $uav^{-1} \notin \mathsf{WP}(G,\Sigma)$ since $\pi_G(uav^{-1}) = g_q \cdot a \cdot \pi_G(v^{-1})$ and $g_q \cdot a \notin B_{G,\Sigma}(\lfloor n/2 \rfloor)$, $\pi_G(v^{-1}) \in B_{G,\Sigma}(\lfloor n/2 \rfloor)$. This is a contradiction since $vv^{-1}$ and $uav^{-1}$ both have length at most $n$. Hence, we must have $|Q| > \gamma_{G,\Sigma}(\lfloor n/2 \rfloor)$. □

The growth of f.g. groups is well-studied and Theorem 6.1 basically closes the chapter on (non)deterministic streaming algorithms for word problems. Hence, in the rest of the paper we focus on randomized streaming algorithms. Here, we can still prove a lower bound (that will turn out to be sharp in some cases but not always) using the randomized one-way communication complexity of the equality problem:

**Theorem 6.2.** *Let $G$ be a finitely generated group and let $\Sigma$ be a finite symmetric generating set for $G$. The randomized streaming space complexity of $\mathsf{WP}(G,\Sigma)$ is $\Omega(\log\log \gamma_{G,\Sigma}(\lfloor n/2 \rfloor))$.*

*Proof.* We make a reduction from the equality problem in communication complexity. In this problem, Alice and Bob each have a private number (say $i \in [1,n]$ for Alice and $j \in [1,n]$ for Bob) and their goal is to check whether $i = j$. It is known that the randomized one-way communication complexity (where Alice can send information to Bob in one round) of the equality problem is $\Theta(\log\log n)$ when Alice and Bob make private random choices [41].

Fix an arbitrary bijection

$$\alpha : [1, \gamma_{G,\Sigma}(\lfloor n/2 \rfloor)] \to B_{G,\Sigma}(\lfloor n/2 \rfloor)$$

and let

$$\beta : B_{G,\Sigma}(\lfloor n/2 \rfloor) \to \Sigma^{\leq \lfloor n/2 \rfloor}$$

be an injective mapping that maps every group element $g \in B_{G,\Sigma}(\lfloor n/2 \rfloor)$ to a word $w \in \Sigma^{\leq \lfloor n/2 \rfloor}$ such that $\pi_G(w) = g$. Assume now that $\mathcal{R} = (\mathcal{A}_n)_{n \geq 0}$ is a randomized

streaming algorithm for $\mathsf{WP}(G,\Sigma)$ and assume that its space complexity is $s(n)$. Then we obtain a randomized one-way communication protocol for equality on numbers from $[1,\gamma_{G,\Sigma}(\lfloor n/2\rfloor)]$ with communication cost $s(n)$, which implies that $s(n) \geq \Omega(\log\log\gamma_{G,\Sigma}(\lfloor n/2\rfloor))$: If Alice holds the number $i \in [1,\gamma_{G,\Sigma}(\lfloor n/2\rfloor)]$, then she runs (using her random choices) the PFA $\mathcal{A}_n$ on input $\beta(\alpha(i))$. The state $q$ reached at the end (which can be encoded by a bit string of length at most $s(n)$) is communicated to Bob. Assume that Bob holds the number $j \in [1,\gamma_{G,\Sigma}(\lfloor n/2\rfloor)]$. Bob then simulates (using his random choices) the PFA $\mathcal{A}_n$ on input $\beta(\alpha(j)^{-1})$ starting from state $q$ and accepts if and only if a final state of $\mathcal{A}_n$ is reached. We have $i = j$ if and only if $\alpha(i)\cdot\alpha(j)^{-1} = 1$ in $G$ if and only if $\beta(\alpha(i))\beta(\alpha(j)^{-1}) \equiv_G 1$. This shows that we obtain indeed a randomized one-way protocol for equality. □

**Remark 6.3.** *Since every f.g. infinite group has growth at least $n$, Theorem 6.2 has the following consequence: If $G$ is a f.g. infinite group, then the randomized streaming space complexity of $G$ is $\Omega(\log\log n)$.*

**Remark 6.4.** *Later in this paper, we will make use of the following two famous results on the growth of groups, see also* [17, 51]*:*

- *Gromov's theorem* [28]*: A f.g. group $G$ has polynomial growth if and only if $G$ is virtually nilpotent (i.e., $G$ has a nilpotent subgroup of finite index).*
- *Wolf-Milnor theorem* [55, 71]*; see also* [17, p. 202]*: A f.g. solvable group $G$ is either virtually nilpotent (and hence has polynomial growth) or there is a constant $c > 1$ such that $G$ has growth $c^n$ (i.e., $G$ has exponential growth). It is well known that the same dichotomy also holds for f.g. linear groups. This is a consequence of Tits alternative* [67]*: A f.g. linear group $G$ is either virtually solvable or contains a free group of rank at least two (in which case $G$ has exponential growth).*

The dichotomy theorem of Milnor and Wolf does not generalize to all f.g. groups. Grigorchuk [26] constructed a group whose growth is lower bounded by $\exp(n^{0.515})$ [7] and upper bounded by $\exp(n^{0.768})$ [6]. The streaming space complexity of this remarkable group will be studied in Theorem 11.6.

## 7. Comparison to sofic groups

In this section we will discuss a relationship between randomized streaming space complexity and sofic groups. There are many equivalent definitions of sofic groups. The following definition is from [4, 13]:

With $\mathsf{Sym}(k)$ we denote the symmetric group on $[1,k]$ (the set of all permutations on $[1,k]$ together with the operation of function composition). For $\sigma \in \mathsf{Sym}(k)$ the normalized Hamming weight $w_H(\sigma)$ is defined by

$$w_H(\sigma) = \frac{1}{k}\cdot|\{i \in [1,k] : \sigma(i) \neq i\}|.$$

Let $G$ be a f.g. group and $\Sigma$ be a finite symmetric generating set for $G$. Let $\pi_G : \Sigma^* \to G$ be the canonical morphism that evaluates words in the group $G$. Then $G$ is called *sofic* if for every $n \geq 0$ there exists a $k \geq 1$ and a monoid morphism $\sigma : \Sigma^* \to \mathsf{Sym}(k)$ (with $\sigma(a^{-1}) = \sigma(a)^{-1}$) such that for every word $w \in \Sigma^{\leq n}$ the following holds:

- if $\pi_G(w) = 1$ then $w_H(\sigma(w)) \leq 1/n$, and
- if $\pi_G(w) \neq 1$ then $w_H(\sigma(w)) \geq 1 - 1/n$.

In case $G$ is sofic, we define the *sofic dimension growth* of $G$ (with respect to $\Sigma$) as the function $\kappa_{G,\Sigma} : \mathbb{N} \to \mathbb{N}$ such that $\kappa_{G,\Sigma}(n)$ is the minimal value $k$ for which the above conditions hold. For different finite generating sets $\Sigma_1, \Sigma_2$ of $G$ the functions $\kappa_{G,\Sigma_1}$ and $\kappa_{G,\Sigma_2}$ are different, but their asymptotic behavior is the same (analogously to the growth functions $\gamma_{G,\Sigma_1}$ and $\gamma_{G,\Sigma_2}$); see [13, Proposition 3.3.2] for a precise statement.

It is a famous open problem whether every group $G$ is sofic.[4] The connection to randomized streaming complexity can be seen as follows: Assume that $G$ is sofic and consider its sofic dimension growth $\kappa_{G,\Sigma}$. For every $n \geq 0$ let $k_n = \kappa_{G,\Sigma}(n)$ and let $\pi_n : \Sigma^* \to \mathsf{Sym}(k_n)$ be the monoid morphism satisfying the above conditions for soficity. Then we obtain a semi-randomized streaming algorithm $\mathcal{R} = (\mathcal{A}_n)_{n\geq 0}$ that is $1/n$-correct for $\mathsf{WP}(G, \Sigma)$ as follows: define $\mathcal{A}_n = (Q_n, \Sigma, \iota_n, \rho_n, F_n)$ with

- $Q_n = [1, k_n] \times [1, k_n]$,
- $\iota_n(i, i) = 1/k_n$ for all $i \in [1, k_n]$ and $\iota_n(i, j) = 0$ for $i \neq j$,
- $\rho_n((i, j), a) = (i, \pi_n(j))$ for all $i, j \in [1, k_n]$ and $a \in \Sigma$, and
- $F_n = \{(i, i) : i \in [1, k_n]\}$.

The space complexity of this algorithm is $s(\mathcal{R}, n) = \lceil 2 \log_2 k_n \rceil$.

The above semi-randomized streaming algorithm $\mathcal{R} = (\mathcal{A}_n)_{n\geq 0}$ has some particular properties:

- for every $a \in \Sigma$, the transition function $q \mapsto \rho_n(q, a)$ (for $q \in Q_n$) is a permutation on $Q_n$, and
- the initial state distribution $\iota_n$ is a uniform distribution on a subset of $Q_n$.

The second property is not a real restriction. With an additional constant factor in the space complexity one can easily ensure that $\iota_n$ is the uniform distribution on a subset of $Q_n$. The first property is a severe restriction that makes the existence of non-sofic groups possible.

## 8. Distinguishers for groups

Let $G$ be a f.g. group $G$ with the finite generating set $\Sigma$. Moreover, let $\epsilon_0, \epsilon_1 : \mathbb{N} \to [0, 1]_{\mathbb{R}}$ be monotonically decreasing functions. A semi-randomized streaming algorithm $(\mathcal{A}_n)_{n\geq 0}$ with $\mathcal{A}_n = (Q_n, \Sigma, \iota_n, \rho_n, F_n)$ (a semiPFA) is called an $(\epsilon_0, \epsilon_1)$-*distinguisher* for $G$ (with respect to $\Sigma$), if the following properties hold for all large enough $n \geq 0$ and all words $u, v \in \Sigma^{\leq n}$:

- If $u \equiv_G v$ then $\mathsf{Prob}_{q\in Q_n}[\rho_n(q, u) = \rho_n(q, v)] \geq 1 - \epsilon_1(n)$. In other words: for a randomly chosen initial state, the semiPFA $\mathcal{A}_n$ arrives with probability at least $1 - \epsilon_1(n)$ in the same state after reading $u$ and $v$.
- If $u \not\equiv_G v$ then $\mathsf{Prob}_{q\in Q_n}[\rho_n(q, u) \neq \rho_n(q, v)] \geq 1 - \epsilon_0(n)$. In other words: for a randomly chosen initial state, the semiPFA $\mathcal{A}_n$ arrives with probability at least $1 - \epsilon_0(n)$ in different states after reading $u$ and $v$.

Note that the set $F_n$ of final states of $\mathcal{A}_n$ is not important and we will just write $\mathcal{A}_n = (Q_n, \Sigma, \iota_n, \rho_n)$ in the following if we talk about an $(\epsilon_0, \epsilon_1)$-distinguisher $(\mathcal{A}_n)_{n\geq 0}$.

[4] One can define the concept of sofic groups also for non-finitely generated groups, but here we only talk about finitely generated groups.

**Lemma 8.1.** *Let $\mathcal{R}$ be an $(\epsilon_0, \epsilon_1)$-distinguisher for $G$ with respect to $\Sigma$. Then $\mathsf{WP}(G, \Sigma)$ has an $(\epsilon_0, \epsilon_1)$-correct semi-randomized streaming algorithm with space complexity $2 \cdot s(\mathcal{R}, n)$.*

*Proof.* Let $\mathcal{R} = (\mathcal{A}_n)_{n \geq 0}$ with $\mathcal{A}_n = (Q_n, \Sigma, \iota_n, \rho_n)$. Using the above definition of an $(\epsilon_0, \epsilon_1)$-distinguisher with the empty string $v = \varepsilon$ we get for every word $u \in \Sigma^{\leq n}$:

- If $u \in \mathsf{WP}(G, \Sigma)$ then $\mathsf{Prob}_{q \in Q_n}[\rho_n(q, u) = q] \geq 1 - \epsilon_1(n)$.
- If $u \notin \mathsf{WP}(G, \Sigma)$ then $\mathsf{Prob}_{q \in Q_n}[\rho_n(q, u) \neq q] \geq 1 - \epsilon_0(n)$.

This allows to construct an $(\epsilon_0, \epsilon_1)$-correct randomized streaming algorithm $(\mathcal{B}_n)_{n \geq 0}$ for $\mathsf{WP}(G, \Sigma)$. Thereby the space complexity of the algorithm only doubles: We define $\mathcal{B}_n = (Q_n \times Q_n, \Sigma, \iota'_n, \rho'_n, F_n)$ where

- $\iota'_n(p, p) = \iota_n(p)$ for all $p \in Q_n$ and $\iota'_n(p, q) = 0$ if $p \neq q$,
- $\rho'_n((p, q), a) = (p, \rho_n(q, a))$ for $p, q \in Q_n$ and $a \in \Sigma$, and
- $F_n = \{(p, p) \colon p \in Q_n\}$.

It is easy to check that this semi-randomized streaming algorithm is indeed $(\epsilon_0, \epsilon_1)$-correct for $\mathsf{WP}(G, \Sigma)$. □

Due to Lemma 8.1, our goal in the rest of the paper will be the construction of space efficient $(\epsilon_0, \epsilon_1)$-distinguishers for groups.

We will need $(\epsilon_0, \epsilon_1)$-distinguishers in order to get transfer results for graph products and wreath products. For this, we need some further observations on $(\epsilon_0, \epsilon_1)$-distinguishers that we discuss in the rest of the section.

For equivalence relations $\equiv_1$ and $\equiv_2$ on a set $A$ and a subset $S \subseteq A$ we say that:

- $\equiv_1$ refines $\equiv_2$ on $S$ if for all $a, b \in S$ we have: if $a \equiv_1 b$ then $a \equiv_2 b$;
- $\equiv_1$ equals $\equiv_2$ on $S$ if for all $a, b \in S$ we have: $a \equiv_1 b$ if and only if $a \equiv_2 b$.

For a semiPFA $\mathcal{A} = (Q, \Sigma, \iota, \rho)$ and a state $q \in Q$ we define the equivalence relation $\equiv_{\mathcal{A},q}$ on $\Sigma^*$ as follows: $u \equiv_{\mathcal{A},q} v$ if and only if $\rho(q, u) = \rho(q, v)$. Whenever $\mathcal{A}$ is clear from the context, we just write $\equiv_q$ instead of $\equiv_{\mathcal{A},q}$.

**Lemma 8.2.** *Let $(\mathcal{A}_n)_{n \geq 0}$ be an $(\epsilon_0, \epsilon_1)$-distinguisher for the finitely generated group $G$ with respect to the finite generating set $\Sigma$. Let $\mathcal{A}_n = (Q_n, \Sigma, \iota_n, \rho_n)$. Consider a set $S \subseteq \Sigma^{\leq n}$. Then, the following statements hold, where $\equiv_q$ refers to $\mathcal{A}_n$:*

- $\mathsf{Prob}_{q \in Q_n}[\equiv_G \textit{ equals } \equiv_q \textit{ on } S] \geq 1 - \max\{\epsilon_0(n), \epsilon_1(n)\}\binom{|S|}{2}$,
- $\mathsf{Prob}_{q \in Q_n}[\equiv_G \textit{ refines } \equiv_q \textit{ on } S] \geq 1 - \epsilon_1(n)\binom{|S|}{2}$,
- $\mathsf{Prob}_{q \in Q_n}[\equiv_q \textit{ refines } \equiv_G \textit{ on } S] \geq 1 - \epsilon_0(n)\binom{|S|}{2}$.

*Proof.* All three statements follow from the union bound and the fact that there are $\binom{|S|}{2}$ unordered pairs of different elements from $S$. For the first statement note that $\mathsf{Prob}_{q \in Q_n}[(u \equiv_G v \text{ and } u \not\equiv_q v) \text{ or } (u \not\equiv_G v \text{ and } u \equiv_q v)] \leq \max\{\epsilon_0(n), \epsilon_1(n)\}$ for all $u, v \in S$. For the second statement, note that $\mathsf{Prob}_{q \in Q_n}[u \equiv_G v \text{ and } u \not\equiv_q v] \leq \epsilon_1(n)$, and similarly for the third statement. □

Recall that for a word $w$ we write $\mathcal{P}(w)$ for the set of all prefixes of $w$.

**Lemma 8.3.** *Let $G$ be a finitely generated group with the finite generating set $\Sigma$ and let $\mathcal{A} = (Q, \Sigma, \iota, \rho)$ be a semiPFA with $q \in Q$. Consider words $u, v \in \Sigma^*$ such that $\equiv_G$ refines $\equiv_q$ on $\mathcal{P}(u) \cup \mathcal{P}(v)$ and let $u = xyz$ with $y \equiv_G 1$. Then $\equiv_G$ refines $\equiv_q$ on $\mathcal{P}(xz) \cup \mathcal{P}(v)$.*

*Proof.* Assume that $s, t \in \mathcal{P}(xz) \cup \mathcal{P}(v)$ are such that $s \equiv_G t$. We have to show that $\rho(q, s) = \rho(q, t)$. If $s \notin \mathcal{P}(u) \cup \mathcal{P}(v)$ we must have $s = xz'$ for a prefix $z'$ of $z$. Since $x \equiv_G xy$, $x, xy \in \mathcal{P}(u)$ and $\equiv_G$ refines $\equiv_q$ on $\mathcal{P}(u) \cup \mathcal{P}(v)$, we have $\rho(q, x) = \rho(q, xy)$. This implies that $\rho(q, s) = \rho(q, xz') = \rho(q, xyz')$. In addition we have $s \equiv_G xyz'$ and $xyz' \in \mathcal{P}(u)$. In this way we obtain from $s$ a word $\tilde{s} \in \mathcal{P}(u) \cup \mathcal{P}(v)$ such that $\rho(q, s) = \rho(q, \tilde{s})$ and $s \equiv_G \tilde{s}$ (we might have $s = \tilde{s}$). In the same way, we can obtain from $t$ a word $\tilde{t} \in \mathcal{P}(u) \cup \mathcal{P}(v)$ such that $\rho(q, t) = \rho(q, \tilde{t})$ and $t \equiv_G \tilde{t}$. Since $s \equiv_G t$ we have $\tilde{s} \equiv_G \tilde{t}$. Since $\equiv_G$ refines $\equiv_q$ on $\mathcal{P}(u) \cup \mathcal{P}(v)$ and $\tilde{s}, \tilde{t} \in \mathcal{P}(u) \cup \mathcal{P}(v)$ we get $\rho(q, s) = \rho(q, \tilde{s}) = \rho(q, \tilde{t}) = \rho(q, t)$. □

**Lemma 8.4.** *Let $G$, $\mathcal{A}$, and $q$ be as in Lemma 8.3. Consider words $u, v \in \Sigma^*$ such that $\equiv_q$ refines $\equiv_G$ on $\mathcal{P}(u) \cup \mathcal{P}(v)$ and let $u = xyz$ with $\rho(q, x) = \rho(q, xy)$. Then $\equiv_q$ refines $\equiv_G$ on $\mathcal{P}(xz) \cup \mathcal{P}(v)$.*

*Proof.* Assume that $s, t \in \mathcal{P}(xz) \cup \mathcal{P}(v)$ are such that $\rho(q, s) = \rho(q, t)$. We have to show that $s \equiv_G t$. Since $\rho(q, x) = \rho(q, xy)$ and $x, xy \in \mathcal{P}(u)$, we have $x \equiv_G xy$, i.e., $y \equiv_G 1$. We can then define the words $\tilde{s}, \tilde{t} \in \mathcal{P}(u) \cup \mathcal{P}(v)$ in the same way as in the proof of Lemma 8.3. We obtain $\rho(q, \tilde{s}) = \rho(q, s) = \rho(q, t) = \rho(q, \tilde{t})$. Since $\equiv_q$ refines $\equiv_G$ on $\mathcal{P}(u) \cup \mathcal{P}(v)$ we get $s \equiv_G \tilde{s} \equiv_G \tilde{t} \equiv_G t$. □

## 9. Randomized streaming algorithms for linear groups

Recall that a group is linear if it is isomorphic to a group of invertible matrices over a field $K$. The group of all invertible $(r \times r)$-matrices with entries from $F$ is denoted with $\mathsf{GL}_r(K)$. For every f.g. linear group, the word problem can be solved in logarithmic space. This was shown by Lipton and Zalcstein [43] (if the underlying field has characteristic zero) and Simon [66] (if the underlying field has prime characteristic). In this section, we show that with some care, one can turn the algorithms from [43, 66] into $(\epsilon_0(n), 0)$-distinguishers with $\epsilon_0(n) = 1/n^c$ for a constant $c$ and space complexity $\mathcal{O}(\log n)$. We will make use of the following well-known result of DeMillo, Lipton, Schwartz and Zippel [72, 64, 16]. The degree of a multivariate polynomial $p(x_1, \ldots, x_n) \in K[x_1, \ldots, x_n]$ with coefficients from the field $K$ is the maximal sum $k_1 + k_2 + \cdots + k_n$ where $x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$ is a monomial of $p$.

**Theorem 9.1.** *Let $p(x_1, \ldots, x_n) \in K[x_1, \ldots, x_n]$ be a non-zero multivariate polynomial of degree $d$, and let $S \subseteq K$ be finite. If $(s_1, \ldots, s_n) \in S^n$ is randomly chosen according to the uniform distribution, then $\mathsf{Prob}[p(s_1, \ldots, s_n) = 0] \leq \frac{d}{|S|}$.*

We now come to the main result of this section.

**Theorem 9.2.** *For every f.g. linear group $G$ and every $c > 0$ there exists a $(1/n^c, 0)$-distinguisher with space complexity $\mathcal{O}(\log n)$.*

*Proof.* By [43], $G$ is a finitely generated subgroup of $\mathsf{GL}_r(K)$, where the field $K$ is of the form $K = F(x_1, \ldots, x_m)$ for a prime field $F$. Thus, $F$ is either $\mathbb{Q}$ or a finite field $\mathbb{F}_p$ for a prime $p$ and $F(x_1, \ldots, x_m)$ is the field of all fractions $q_1/q_2$ for polynomials $q_1, q_2 \in F[x_1, \ldots, x_m]$ with $q_2 \neq 0$.

Let us first assume that $F = \mathbb{Q}$. Let $\Sigma$ be a generating set for $G$. Then every generator $M \in \Sigma$ is a matrix, whose entries are quotients of polynomials from $\mathbb{Z}[x_1, \ldots, x_m]$. Therefore there exists a fixed non-zero polynomial $t \in \mathbb{Z}[x_1, \ldots, x_m]$ such that every matrix $\hat{M} := t \cdot M$ for $M \in \Sigma$ has entries from $\mathbb{Z}[x_1, \ldots, x_m]$.

Let $r$ be the dimension of the matrices. Let $d$ be the maximal degree of $t$ and all polynomials that appear in matrices $\hat{M}$ with $M \in \Sigma$. The parameters $m$, $r$, and $d$ are constants in the further considerations.

Fix an input length $n$. Clearly, for all matrices $M_1, \ldots, M_k, N_1, \ldots, N_l \in \Sigma$ with $k, l \leq n$ we have $\prod_{i=1}^{k} M_i = \prod_{i=1}^{l} N_i$ if and only if $t^{n+1-k} \prod_{i=1}^{k} \hat{M}_i = t^{n+1-l} \prod_{i=1}^{l} \hat{N}_i$.

Consider two input words $M_1 M_2 \cdots M_k, N_1 N_2 \cdots N_l \in \Sigma^*$ with $k, l \leq n$ and assume that

$$t^{n+1-k} \prod_{i=1}^{k} \hat{M}_i \neq t^{n+1-l} \prod_{i=1}^{l} \hat{N}_i.$$

Define the matrix

$$A := t^{n+1-k} \prod_{i=1}^{k} \hat{M}_i - t^{n+1-l} \prod_{i=1}^{l} \hat{N}_i \in \mathbb{Z}[x_1, \ldots, x_m]^{r \times r}.$$

Note that all entries of the matrix $A$ are polynomials of degree at most $d(n+1)$ and at least one of them is not the zero polynomial.

Let $S = [1, 2d(n+1)^{c+1}]$, where $c$ is the value from the theorem. For a tuple $\bar{\phi} = (s_1, \ldots, s_m) \in S^m$ and a matrix $M \in \mathbb{Z}[x_1, \ldots, x_m]^{r \times r}$ let $\hat{M}(\bar{\phi})$ be the integer matrix obtained from $\hat{M}$ by replacing every variable $x_i$ by $s_i$. For a randomly chosen tuple $\bar{\phi} \in S^m$, Theorem 9.1 implies that

$$\operatorname*{Prob}_{\bar{\phi} \in S^m}[A(\bar{\phi}) \neq 0] \geq 1 - \frac{1}{2(n+1)^c} \geq 1 - \frac{1}{2n^c}. \tag{5}$$

Let us now consider a tuple $\bar{\phi} \in S^m$ such that $A(\bar{\phi}) \neq 0$. Every entry in a matrix $\hat{M}(\bar{\phi})$ ($M \in \Sigma$) has an absolute value of order $\mathcal{O}((2d(n+1)^{c+1})^d) = \mathcal{O}(n^{d(c+1)})$ ($d$ is a constant) and also $t(\bar{\phi}) \leq \mathcal{O}(n^{d(c+1)})$. Therefore, all entries in the matrix $t(\bar{\phi})^{n+1-k} \prod_{i=1}^{k} \hat{M}_i(\bar{\phi})$ are of absolute value $\mathcal{O}(r^n n^{d(c+1)n})$, and similarly for $t(\bar{\phi})^{n+1-l} \prod_{i=1}^{l} \hat{N}_i(\bar{\phi})$. Hence, $A(\bar{\phi})$ is a non-zero matrix with all entries of absolute value at most $\mathcal{O}(r^n n^{d(c+1)n})$.

The number of different prime factors of a number $D \leq \mathcal{O}(r^n n^{d(c+1)n})$ is bounded by

$$\frac{\ln D}{\ln \ln D} \cdot (1 + o(1)) \leq \mathcal{O}\left(\frac{n \log n}{\log n}\right) = \mathcal{O}(n);$$

see [62, Theorem 16]. By a weak form of the prime number theorem, the number of primes of size at most $n^{c+2} \ln n$ is $\Theta(n^{c+2})$. Hence, by randomly choosing a prime of size at most $n^{c+2} \ln n$ we can obtain the bound

$$\operatorname*{Prob}_{\bar{\phi}, p}[A(\bar{\phi}) \bmod p = 0 \mid A(\bar{\phi}) \neq 0] \leq \mathcal{O}\left(\frac{1}{n^{c+1}}\right) \leq \frac{1}{2n^c}$$

for $n$ large enough. Hence, we obtain

$$\begin{aligned}
& \mathop{\mathsf{Prob}}_{\bar\phi,p}\left[t(\bar\phi)^{n+1-k}\prod_{i=1}^{k}\hat M_i(\bar\phi) \not\equiv t(\bar\phi)^{n+1-l}\prod_{i=1}^{l}\hat N_i(\bar\phi) \bmod p\right] \\
= \;& \mathop{\mathsf{Prob}}_{\bar\phi,p}[A(\bar\phi) \bmod p \neq 0 \mid A(\bar\phi)\neq 0]\cdot \mathop{\mathsf{Prob}}_{\bar\phi}[A(\bar\phi)\neq 0] \\
\geq \;& \left(1-\frac{1}{2n^c}\right)^2 \\
\geq \;& 1-\frac{1}{n^c}
\end{aligned}$$

for $n$ large enough. The streaming algorithm for inputs of length at most $n$ is now clear: Initially, the algorithm guesses $\bar\phi \in S^m$ (for $S = [1, 2d(n+1)^{c+1}]$) and a prime $p \leq n^{c+2}\ln n$. All these numbers need $\mathcal{O}(\log n)$ bits in total. If $t(\bar\phi) \bmod p = 0$ then the algorithm ignores the input word. Otherwise, the algorithm initializes a matrix $B := t(\bar\phi)^{n+1}\cdot \mathsf{Id}_r \bmod p$, where $\mathsf{Id}_r$ is the $r$-dimensional identity matrix. Then, for every new generator matrix $M \in \Sigma$ the algorithm updates $B$ by

$$B := t(\bar\phi)^{-1}\cdot B\cdot \hat M(\bar\phi) \bmod p.$$

All computation are carried out in the field $\mathbb{F}_p$. If $\prod_{i=1}^{k} M_i = \prod_{i=1}^{l} N_i$ $(k,l \leq n)$ then after reading the input words $M_1\cdots M_k$ and $N_1\cdots N_l$, the algorithm arrives with probability one in the same state. On the other hand, if $\prod_{i=1}^{k} M_i \neq \prod_{i=1}^{l} N_i$ then the reached states differ with probability at least $1-1/n^c$ by the above error analysis.

Let us now briefly discuss the case where the underlying prime field is $F = \mathbb{F}_p$ for a prime $p$. Then we have to work in a finite extension $\mathbb{F}_{p^e}$ for some $e$ such that $p^e \geq 2d(n+1)^{c+1}$, which can be achieved by taking $e$ of size $\Theta(\log n)$. By fixing a subset $S \subseteq \mathbb{F}_{p^e}$ of size $2d(n+1)^{c+1}$ and choosing a tuple $\bar\phi = (s_1,\ldots,s_m) \in S^m$ randomly, we obtain the bound (5). Since an $r$-dimensional matrix over the field $\mathbb{F}_{p^e}$ can be stored in space $\mathcal{O}(\log n)$ ($r$ and $p$ are constants and $e = \Theta(\log n)$), this yields the desired algorithm in the same way as for the case $F = \mathbb{Q}$. □

A group is nilpotent if its lower central series terminates after finitely many steps in the trivial group 1. The lower central series of a group $G$ is the series $G = G_1 \trianglerighteq G_2 \trianglerighteq G_3 \trianglerighteq \cdots$ where $G_{i+1} = [G, G_i]$. Every nilpotent group is linear. For nilpotent groups we can improve the algorithm from the proof of Theorem 9.2, at least if we sacrifice the inverse polynomial error probability:

**Theorem 9.3.** *For every f.g. nilpotent group $G$ and every constant $c > 0$ there exists a $(1/\log^c n, 0)$-distinguisher with space complexity $\mathcal{O}(\log\log n)$.*

*Proof.* We can assume that $G$ is infinite. With $\mathsf{UT}_d(\mathbb{Z})$ we denote the set of all upper triangular $(d\times d)$-matrices over $\mathbb{Z}$ with all diagonal entries equal to 1 (so-called unitriangular matrices). These matrices form a f.g. nilpotent group. Let $G$ be a f.g. nilpotent group. Then $G$ has a f.g. torsion-free nilpotent subgroup $H$ such that the index $[G:H]$ is finite [36, Theorem 17.2.2]. Moreover, there exists $d \geq 1$ such that the finitely generated torsion-free nilpotent group $H$ can be embedded into the group $\mathsf{UT}_d(\mathbb{Z})$ [36, Theorem 17.2.5]. By Theorem 10.2 below it suffices to show that every $\mathsf{UT}_d(\mathbb{Z})$ has an $\epsilon(n)$-distinguisher with $\epsilon(n) = 1/\log^c n$ and space complexity $\mathcal{O}(\log\log n)$.

Fix a finite generating set $\Sigma$ for $\mathsf{UT}_d(\mathbb{Z})$ and an input length $n$. Consider a product $M := \prod_{i=1}^{m} M_i$ with $m \leq n$ and $M_i \in \Sigma$. From [45, Proposition 4.18] it follows that the absolute value of every entry of the matrix $M$ has size at most $\mathcal{O}(m^{d-1}) \leq \mathcal{O}(n^{d-1})$. The randomized streaming algorithm for $\mathsf{UT}_d(\mathbb{Z})$ will guess a prime number of size $\Theta(\log^{c+1} n \cdot \log\log(n))$ and computes the product matrix $M$ modulo $p$. For this, $\mathcal{O}(\log\log n)$ bits are sufficient.

Consider two input words $u = M_1 M_2 \cdots M_l \in \Sigma^*$ and $v = N_1 N_2 \cdots N_m \in \Sigma^*$ with $l, m \leq n$. If $\prod_{i=1}^{l} M_i = \prod_{i=1}^{m} N_i$ then our randomized streaming algorithm will reach with probability one the same state after reading the input words $u$ and $v$, respectively. On the other hand, if $\prod_{i=1}^{l} M_i \neq \prod_{i=1}^{m} N_i$, then consider a non-zero matrix entry $a \in \mathbb{Z}$ of the matrix $\prod_{i=1}^{l} M_i - \prod_{i=1}^{m} N_i$. We have $|a| \leq \mathcal{O}(n^{d-1})$. The number of different prime factors of $a$ is therefore bounded by $\mathcal{O}(\log n / \log\log n)$. Hence, by randomly choosing a prime number $p$ of size at most $\mathcal{O}(\log^{c+1} n \cdot \log\log(n))$ we can obtain a probability of at most $1/\log^c n$ for $a \bmod p = 0$. Hence, with probability $1 - 1/\log^c n$ we reach different states after reading $u$ and $v$, respectively. □

Note that if $G$ is infinite, the space bound from Theorem 9.3 is sharp up to constant factors even if we allow a constant error probability; see Remark 6.3.

By Theorem 4.4 the inverse polylogarithmic error in Theorem 9.3 cannot be improved if $G$ is infinite: Consider an $\epsilon(n)$-distinguisher with space complexity $\mathcal{O}(\log\log n)$ for the infinite group $G$. Lemma 8.1 yields an $\epsilon(n)$-correct semi-randomized streaming algorithm for the word problem of $G$ with space complexity $r(n) \leq \mathcal{O}(\log\log n)$. By Theorem 6.1, the deterministic streaming space complexity of the word problem for $G$ is lower bounded by $\Omega(\log n)$. Hence, if $n$ is large enough, we must have $r(n) \geq \log_2(1/\epsilon(n))$ by Theorem 4.4. We get $\log_2(1/\epsilon(n)) \leq c \cdot \log_2 \log_2 n$ for some constant $c > 0$, i.e., $\epsilon(n) \geq 1/\log_2^c n$.

## 10. Closure properties for the space complexity of distinguishers

In this section, we will show that many group theoretical constructions preserve the space complexity of distinguishers.

### 10.1. Finitely generated subgroups, finite extensions, direct products.

For many algorithmic problems in group theory, the complexity is preserved when (i) going down to a finitely generated subgroup and (ii) going up to a finite extension. This is also true for our model of distinguishers:

**Theorem 10.1.** *Let $G$ be a f.g. group and $H$ a f.g. subgroup of $H$. If $\mathcal{R}$ is an $(\epsilon_0(n), \epsilon_1(n))$-distinguisher for $G$ then $H$ has an $(\epsilon_0(cn), \epsilon_1(cn))$-distinguisher with space complexity $s(\mathcal{R}, c \cdot n)$ for some constant $c$.*

*Proof.* Fix the generating sets $\Sigma$ and $\Gamma$ of $G$ and $H$, respectively. Then for every generator $a \in \Gamma$ there is a word $w_a \in \Sigma^*$ such that $a$ and $w_a$ represent the same element of $H$. We can then argue as in the proof of Lemma 5.1. □

**Theorem 10.2.** *Assume that $H$ is a f.g. group and $G$ is a subgroup of $H$ of finite index (hence, also $G$ must be finitely generated). Assume that $\mathcal{R}$ is an $(\epsilon_0(n), \epsilon_1(n))$-distinguisher for $G$. Then $H$ has an $(\epsilon_0(cn), \epsilon_1(cn))$-distinguisher with space complexity $s(\mathcal{R}, c \cdot n) + \mathcal{O}(1)$ for some constant $c$.*

*Proof.* We can assume that $G$ is a normal subgroup of $H$: It is well known that there exists a normal subgroup $N$ of $H$ (the so-called normal core of $G$) such that $N \leq G$ and $N$ has finite index in $H$, see e.g. [63, Excercise 1.6.9]. Since $N$ has finite index in $H$, also $N$ must be finitely generated. Since $N \leq G$, Theorem 10.1 implies that $N$ has an $(\epsilon_0(dn), \epsilon_1(dn))$-distinguisher with space complexity $s(\mathcal{R}, dn)$ for some constant $d$. This shows that we can replace $G$ by $N$.

For the rest of the proof we assume that $G$ is normal in $H$. Fix a generating set $\Sigma$ for $G$. Let $h_1, \ldots, h_k \in H$ be a set of coset representatives for $G$ where $h_1 = 1$. Then $\Gamma = \Sigma \cup \{h_2, \ldots, h_k\}$ generates $H$. Since $G$ is normal, for every $a \in \Sigma$ and every $i \in [1, k]$ there exists an element $g(a, i) \in G$ such that $h_i a = g(a, i)h_i$ in $H$ (with $g(a, 1) = a$). Moreover, for all $i, j \in [1, k]$ there are $g(i, j) \in G$ and $1 \leq \alpha(i, j) \leq k$ such that $h_i h_j = g(i, j)h_{\alpha(i,j)}$ in $H$ (with $\alpha(1, i) = \alpha(i, 1) = i$ and $g(i, 1) = g(1, i) = 1$). We identify the group elements $g(a, i), g(i, j)$ with words over the alphabet $\Sigma$. Let $c$ be the maximal length of these words.

Let $\mathcal{R} = (\mathcal{A}_n)_{n\geq 0}$ be the $(\epsilon_0(n), \epsilon_1(n))$-distinguisher for $G$ with respect to $\Sigma$. An $(\epsilon_0(cn), \epsilon_1(cn))$-distinguisher $\mathcal{S} = (\mathcal{B}_n)_{n\geq 0}$ for $H$ with respect to $\Gamma$ works as follows: Fix $n \geq 0$ and an input word $w \in \Gamma^{\leq n}$. The automaton $\mathcal{B}_n$ will store a coset representative $h \in \{h_1, \ldots, h_k\}$ (using space $\mathcal{O}(1)$) and a state of $\mathcal{A}_{cn}$. Let $\mathcal{A}_{cn} = (Q, \Sigma, \iota, \rho, F)$. Initially, we set $h = h_1 = 1$ and a state from $Q$ is guessed according to the initial state distribution $\iota$. Assume that $q \in Q$ is the current state of $\mathcal{A}_{cn}$, $h = h_i$ is the current coset and we read a generator from $\Gamma$. If this generator is $a \in \Sigma$ then (recall the identity $h_i a = g(a, i)h_i$ in $H$) we proceed to the state $\rho(q, g(a, i)) \in Q$ and the coset representative $h_i$ is not changed. If we read a generator $h_j \in \{h_2, \ldots, h_k\}$ then (recall the identity $h_i h_j = g(i, j)h_{\alpha(i,j)}$ in $H$) we proceed to the state $\rho(q, g(i, j)) \in Q$ and the coset representative $h_{\alpha(i,j)}$. It is easy to observe that $\mathcal{S}$ is an $(\epsilon_0(cn), \epsilon_1(cn))$-distinguisher for $H$. □

Recall that Gromov [28] proved that a finitely generated group has polynomial growth if and only if it is virtually nilpotent.

**Corollary 10.3.** *Let $G$ be an infinite finitely generated linear group.*

- *If $G$ is virtually nilpotent then the (0-sided) randomized streaming space complexity of $G$ is $\Theta(\log \log n)$.*
- *If $G$ is not virtually nilpotent then (0-sided) the randomized streaming space complexity of $G$ is $\Theta(\log n)$.*

*Proof.* The upper bounds follow from Theorems 9.2, 9.3 and 10.2. Since $G$ is infinite, the randomized streaming space complexity of the word problem of $G$ is $\Omega(\log \log n)$ (see Remark 6.3), which yields the lower bound for the virtually nilpotent case. If $G$ is not virtually nilpotent, then $G$ has growth $c^n$ for some constant $c > 1$ (see Remark 6.4), which yields the lower bound $\Theta(\log n)$ by Theorem 6.2. □

It is conjectured that for every f.g. group $G$ that is not virtually nilpotent the growth is lower bounded by $\exp(n^{0.5})$. This is known as the *gap conjecture* [27]. It would imply that for every f.g. group that is not virtually nilpotent the randomized streaming space complexity is lower bounded by $\Omega(\log n)$.

Also direct products preserve the space complexity of distinguishers (simply run the distinguishers for the two factor groups in parallel):

**Lemma 10.4.** *Let $G$ (resp., $H$) be a finitely generated group for which there exists an $(\epsilon_0, \epsilon_1)$-distinguisher (resp., $(\zeta_0, \zeta_1)$-distinguisher) $\mathcal{R}$ (resp., $\mathcal{S}$). Then*

*there exists an* $(\max\{\epsilon_0, \zeta_0\}, \epsilon_1 + \zeta_1)$*-distinguisher for* $G \times H$ *with space complexity* $s(\mathcal{R}, n) + s(\mathcal{S}, n)$.[5]

Recall that a group $G$ is metabelian if it has an abelian normal subgroup $A \leq G$ such that the quotient $G/A$ is abelian as well. Every finitely generated metabelian group can be embedded into a direct product of finitely generated linear groups (over fields of different characteristics) [69]. Hence, with Lemma 10.4 and Theorem 9.2 we obtain:

**Corollary 10.5.** *For every finitely generated metabelian group and every* $c > 0$ *there exists a* $(1/n^c, 0)$*-distinguisher with space complexity* $\mathcal{O}(\log n)$.

10.2. **Randomized streaming algorithms for graph products.** In this section we investigate a common generalization of the free product and direct product, which is known as the graph product of groups.

A graph product is specified by a list of groups $G_1, \ldots, G_c$ (here, we only consider the case where all $G_i$ are f.g.) and a symmetric and irreflexive relation $I \subseteq [1, c] \times [1, c]$. To define the corresponding graph product, write every $G_i$ as $G_i = \langle \Gamma_i \mid R_i \rangle$ for a finite set $\Gamma_i$ and a possible infinite set $R_i \subseteq F(\Gamma_i)$. W.l.o.g. we can assume that the $\Gamma_i$ are pairwise disjoint. Let

$$\Gamma = \bigcup_{i=1}^{c} \Gamma_i \quad \text{and} \quad R = \bigcup_{i=1}^{c} R_i.$$

Then, the *graph product* $G = \mathsf{GP}(G_1, \ldots, G_c, I)$ is the group

$$\langle \Gamma \mid R \cup \bigcup_{(i,j) \in I} \{[a, b] : a \in \Gamma_i, b \in \Gamma_j\} \rangle.^{6} \tag{6}$$

Graph products interpolate in a natural way between free products ($I = \emptyset$) and direct products ($I = \{(i, j) : i, j \in [1, c], i \neq j\}$). The graph product $\mathsf{GP}(G_1, \ldots, G_c, I)$ is obtained from the free product $*_{i \in [1,c]} G_i$ by allowing elements from groups $G_i$ and $G_j$ with $(i, j) \in I$ to commute. Graph products were introduced by Green in her thesis [25].

Graph products $\mathsf{GP}(G_1, \ldots, G_c, I)$, where every $G_i$ is isomorphic to $\mathbb{Z}$, are also known as *graph groups* (or right-angled Artin groups). We will make use of the fact that every graph group is linear [33].

Let $\Sigma_i$ be a finite symmetric generating set for $G_i$, where w.l.o.g. $1 \notin \Sigma_i$ and $\Sigma_i \cap \Sigma_j = \emptyset$ for $i \neq j$. Then, $\Sigma = \bigcup_{i=1}^{c} \Sigma_i$ generates $G$. For a word $u \in \Sigma^*$, the *block factorization* of $u$ is the unique factorization $u = u_1 u_2 \cdots u_l$ such that $l \geq 0$, $u_1, \ldots, u_l \in \bigcup_{i \in [1,c]} \Sigma_i^+$ and $u_j u_{j+1} \notin \bigcup_{i \in [1,c]} \Sigma_i^+$ for all $j \in [1, l-1]$. The factors $u_1, u_2, \ldots, u_l$ are also called the *blocks* of $u$.

We define several rewrite relations on words from $\Sigma^*$ as follows: take $u, v \in \Sigma^*$ and let $u = u_1 u_2 \cdots u_l$ be the block factorization of $u$.

- We write $u \leftrightarrow_s v$ ($s$ for swap) if there is $i \in [1, l-1]$ and $(j, k) \in I$ such that $u_i \in \Sigma_j^+$, $u_{i+1} \in \Sigma_k^+$ and $v = u_1 u_2 \cdots u_{i-1} u_{i+1} u_i u_{i+2} \cdots u_l$. In other words, we swap consecutive commuting blocks. Note that $\leftrightarrow_s$ is a symmetric relation.

[5] Here, $\max\{\epsilon_0, \zeta_0\}$ denotes the pointwise maximum of the two functions $\epsilon_0$ and $\zeta_0$.

[6] Up to isomorphism, this definition does not depend on the presentations $(\Gamma_i, R_i)$ for the groups $G_i$.

- We write $u \to_d v$ ($d$ for delete) if there is $i \in [1,l]$ and $j \in [1,c]$ such that $u_i \in \Sigma_j^+$, $u_i \equiv_{G_j} 1$ and $v = u_1 u_2 \cdots u_{i-1} u_{i+1} u_{i+2} \cdots u_l$. In other words, we delete a block that is trivial in its group.
- We write $u \leftrightarrow_r v$ ($r$ for replace) if there is $i \in [1,l]$ and $j \in [1,c]$ such that $u_i, u_i' \in \Sigma_j^+$, $u_i \equiv_{G_j} u_i'$ and $v = u_1 u_2 \cdots u_{i-1} u_i' u_{i+1} u_{i+2} \cdots u_l$. In other words, we replace a block by an equivalent non-empty word. Note that $\leftrightarrow_r$ is a symmetric relation.

Clearly, in all three cases we have $u \equiv_G v$. If $u \to_d v$, then the number of blocks of $v$ is smaller than the number of blocks of $u$ and if $u \leftrightarrow_s v$ then the number of blocks of $v$ can be smaller than the number of blocks of $u$ (since two blocks can be merged into a single block). We write $u \leftrightarrow_{sr} v$ if $u \leftrightarrow_s v$ or $u \leftrightarrow_r v$ and we write $u \to_{sd} v$ if $u \leftrightarrow_s v$ or $u \to_d v$.

Let us say that a word $u \in \Sigma^*$ with $l$ blocks is *reduced*, if there is no $v \in \Sigma^*$ such that $u \to_{sd}^* v$ and $v$ has at most $l-1$ blocks. Clearly, for every word $u \in \Sigma^*$ there is a reduced word $u' \in \Sigma^*$ such that $u \to_{sd}^* u'$. The following result can be found in [25, Theorem 3.9] and [32] in slightly different notations.

**Lemma 10.6.** *Let $G$ be a graph product as above and $u, v \in \Sigma^*$. Then the following are equivalent:*

- $u \equiv_G v$
- *There are reduced words $u', v'$ such that $u \to_{sd}^* u'$, $v \to_{sd}^* v'$, and $u' \leftrightarrow_r^* v'$.*

Consider a word $u \in \Sigma^*$ and its block factorization $u = u_1 u_2 \dots u_l$. A *pure prefix* of $u$ is a word $u_{k_1} u_{k_2} \cdots u_{k_m}$ such that for some $i \in [1,c]$ we have

- $1 \le k_1 < k_2 < \cdots < k_m \le l$,
- $u_{k_1}, u_{k_2}, \dots, u_{k_m} \in \Sigma_i^+$ and
- if $k_j < p < k_{j+1}$ for some $j \in [1, m-1]$ or $1 \le p < k_1$ then $u_p \notin \Sigma_i^+$.

**Theorem 10.7.** *Let $G = \mathsf{GP}(G_1, \dots, G_c, I)$ be a graph product as above and let $\mathcal{R}_i = (\mathcal{A}_{i,n})_{n \ge 0}$ be an $(\epsilon_0, \epsilon_1)$-distinguisher for $G_i$. Let $d \ge 1$ and define*

$$\begin{aligned} \zeta_0(n) &= 2\epsilon_0(n) c n^2 + 1/n^d, \\ \zeta_1(n) &= 2\epsilon_1(n) c n^2. \end{aligned}$$

*Then, there is a $(\zeta_0, \zeta_1)$-distinguisher for the graph product $G$ with space complexity $\mathcal{O}(\sum_{i=1}^c s(\mathcal{R}_i, n) + \log n)$.*[7]

*Proof.* Let us fix an input length $n$ and let $\mathcal{A}_{i,n} = (Q_{i,n}, \Sigma_i, \iota_{i,n}, \rho_{i,n})$, where w.l.o.g. $Q_{i,n} = [0, |Q_{i,n}| - 1]$ consists of the first $|Q_n|$ integers. To simplify the notation, we will omit the second subscript $n$ in the following, i.e., we write $\mathcal{A}_i = (Q_i, \Sigma_i, \iota_i, \rho_i)$ with $Q_i = [0, |Q_i| - 1]$ for the semiPFA $\mathcal{A}_{i,n}$. For a state $q \in Q_i$, we will use in the following the equivalence relation $\equiv_q = \equiv_{\mathcal{A}_i, q}$ defined in Section 8. For a word $w \in \Sigma^*$, we will write $\pi_i(w)$ for the projection $\pi_{\Sigma_i}(w)$.

For every $i \in [1,c]$ we choose a new symbol $a_i$ and consider the infinite cyclic group $\langle a_i \rangle \cong \mathbb{Z}$. Let $\Delta = \{a_1, a_1^{-1}, \dots, a_c, a_c^{-1}\}$ and consider the graph group $H = \mathsf{GP}(\langle a_1 \rangle, \dots, \langle a_c \rangle, I)$. Since every graph group is linear, there is a $(1/m^d, 0)$-distinguisher $(\mathcal{B}_m)_{m \ge 0}$ with space complexity $\mathcal{O}(\log m)$ for $H$ by Theorem 9.2. Let $\mathcal{B}_m = (R_m, \Delta, \lambda_m, \sigma_m)$.

[7] Note that Theorem 10.7 only makes sense if $\epsilon_0(n), \epsilon_1(n) < 1/2cn^2$.

**Algorithm 1:** $(\zeta_0, \zeta_1)$-distinguisher for $G = \mathsf{GP}(G_1, \ldots, G_c, I)$

**global variables:** $q_i \in Q_i$ for all $i \in [1, c]$, $r \in R_m$

**initialization:**

guess $q_i \in Q_i = [0, |Q_i| - 1]$ according to the input distribution $\iota_i$ of $\mathcal{A}_i$ ;

guess $r \in R_m$ according to the input distribution $\lambda_m$ of $\mathcal{B}_m$ ;

**next input letter:** $a \in \Sigma$

**if** $a \in \Sigma_i$ **then**

  $r := \sigma_m(r, a_i^{-q_i} a_i^{\rho_i(q_i,a)})$ ;

  $q_i := \rho_i(q_i, a)$

**end**

We build from the semiPFA $\mathcal{A}_i$ and a state $q \in Q_i$ a sequential transducer $\mathcal{T}_{i,q} = (Q_i, \Sigma_i, \{a_i, a_i^{-1}\}, q, \delta_i)$, where

$$\delta_i(p, a) = (\rho_i(p, a), a_i^{-p} a_i^{\rho_i(p,a)})$$

for all $a \in \Sigma_i$ and $p \in Q_i$. Let $f_{i,q} = f_{\mathcal{T}_{i,q}} : \Sigma_i^* \to \{a_i, a_i^{-1}\}^*$ be the function computed by the sequential transducer $\mathcal{T}_{i,q}$.

For a tuple $\bar{q} = (q_1, \ldots, q_c) \in \prod_{i\in[1,c]} Q_i$ of states from the semiPFAs $\mathcal{A}_i$ we define the sequential transducer $\mathcal{T}_{\bar{q}}$ by taking the direct product of the sequential transducers $\mathcal{T}_{i,q_i}$ $(i \in [1, c])$. Formally, it is defined as follows:

$$\mathcal{T}_{\bar{q}} = \Big( \prod_{i\in[1,c]} Q_i, \Sigma, \Delta, \bar{q}, \delta \Big)$$

where for every $i \in [1, c]$, $a \in \Sigma_i$, and $(p_1, \ldots, p_c) \in \prod_{i\in[1,c]} Q_i$ we have

$$\delta((p_1, \ldots, p_c), a) = \big((p_1, \ldots, p_{i-1}, \rho_i(p_i, a), p_{i+1}, \ldots, p_c), a_i^{-p_i} a_i^{\rho_i(p_i,a)}\big).$$

Let $f_{\bar{q}} = f_{\mathcal{T}_{\bar{q}}} : \Sigma^* \to \Delta^*$ be the function computed by $\mathcal{T}_{\bar{q}}$; note that it is not a homomorphism. Let

$$m = 2 \cdot n \cdot \max\{|Q_i| \colon i \in [1, c]\} \leq n \cdot 2^{1+\max\{s(\mathcal{R}_i,n)\colon i\in[1,c]\}}.$$

Note that $|f_{\bar{q}}(w)| \leq m$ if $|w| \leq n$.

Our randomized streaming algorithm for $G$ and input length $n$ will use the semiPFA $\mathcal{B}_m$ for the graph group $H$. States of $\mathcal{B}_m$ can be stored with $\mathcal{O}(\log m) \leq \mathcal{O}(\max\{s(\mathcal{R}_i, n) \colon i \in [1, c]\} + \log n)$ bits. Basically, for an input word $w \in \Sigma^{\leq n}$ the algorithm simulates the automata $\mathcal{A}_i$ $(i \in [1, c])$ on the projections $w_i = \pi_i(w)$ and feeds the word $f_{\bar{q}}(w)$ into the semiPFA $\mathcal{B}_m$. Here, the state tuple $\bar{q}$ is randomly guessed in the beginning according to the distributions $\iota_i$. The complete streaming algorithm is Algorithm 1. It stores in total at most $\sum_{i=1}^{c} s(\mathcal{R}_i, n) + \mathcal{O}(\max\{s(\mathcal{R}_i, n) \colon i \in [1, c]\} + \log n)$ bits.

Before we analyze the error probability of the algorithm we need some preparations. For $i \in [1, c]$ and a word $w \in \Sigma^*$ let $\mathcal{P}_i(w) = \mathcal{P}(\pi_i(w))$ be the set of all prefixes of the projection $\pi_i(w)$. Assume that $y \in \Sigma_i^+$ is a block of $w$ and write $w = xyz$. We then have $f_{\bar{q}} = f_{\bar{q}}(x) f_{\bar{r}}(y) f_{\bar{s}}(z)$, where $\delta(\bar{q}, x) = (\bar{r}, f_{\bar{q}}(x))$ and $\delta(\bar{r}, y) = (\bar{s}, f_{\bar{r}}(y))$. The word $f_{\bar{r}}(y)$ is also a block of $f_{\bar{q}}$ (for this it is important that every $\mathcal{T}_{j,q}$ translates non-empty words into non-empty words). Since $y \in \Sigma_i^+$

we have $r_j = s_j$ for all $j \in [1,c]\setminus\{i\}$ and $f_{\bar r}(y) = f_{i,r_i}(y)$. In addition, the definition of the sequential transducer $\mathcal{T}_{i,r_i}$ implies that $f_{\bar r}(y) \equiv_{\langle a_i\rangle} a_i^{-r_i}a_i^{s_i}$.

Consider now two input words $u, v \in \Sigma^{\leq n}$ and let $\mathcal{S}_i = \mathcal{P}_i(u) \cup \mathcal{P}_i(v)$ for $i \in [1,c]$. By Lemma 8.2 we have for all $i \in [1,c]$:

$$\begin{aligned}
\mathop{\mathsf{Prob}}_{q\in Q_i}[\equiv_q \text{ refines } \equiv_{G_i} \text{ on } \mathcal{S}_i] &\geq 1 - \epsilon_0(n)\binom{|\mathcal{S}_i|}{2} \geq 1 - 2\epsilon_0(n)n^2,\\
\mathop{\mathsf{Prob}}_{q\in Q_i}[\equiv_{G_i} \text{ refines } \equiv_q \text{ on } \mathcal{S}_i] &\geq 1 - \epsilon_1(n)\binom{|\mathcal{S}_i|}{2} \geq 1 - 2\epsilon_1(n)n^2.
\end{aligned}$$

*Claim 1.* Assume that $\bar q = (q_1, \ldots, q_c)$ is such that $\equiv_{G_i}$ refines $\equiv_{q_i}$ on $\mathcal{S}_i$ for every $i \in [1,c]$. If $u \to_{sd}^* u'$ and $v \to_{sd}^* v'$, then $f_{\bar q}(u) \to_{sd}^* f_{\bar q}(u')$, $f_{\bar q}(v) \to_{sd}^* f_{\bar q}(v')$ and $\equiv_{G_i}$ refines $\equiv_{q_i}$ on $\mathcal{P}_i(u') \cup \mathcal{P}_i(v')$ for every $i \in [1,c]$.

*Proof of Claim 1.* It suffices to show the following: If $u \to_{sd} u'$ holds, then $f_{\bar q}(u) \to_{sd} f_{\bar q}(u')$ and $\equiv_{G_i}$ refines $\equiv_{q_i}$ on $\mathcal{P}_i(u') \cup \mathcal{P}_i(v)$ for every $i \in [1,c]$. From this (and the symmetric statement where $v \to_{sd} v'$ and $u = u'$) we obtain the general statement by induction on the number of $\to_{sd}$-steps. We distinguish two cases.

*Case 1.* $u \leftrightarrow_s u'$. We must have $u = xy_1y_2z$ and $u' = xy_2y_1z$ for blocks $y_1, y_2$ such that $y_1 \in \Sigma_i^+$, $y_2 \in \Sigma_j^+$ and $(i,j) \in I$ (in particular $i \neq j$). We obtain

$$\begin{aligned}
f_{\bar q}(u) &= f_{\bar q}(x)f_{\bar p}(y_1)f_{\bar r}(y_2)f_{\bar s}(z) \text{ and}\\
f_{\bar q}(u') &= f_{\bar q}(x)f_{\bar p}(y_2)f_{\bar r'}(y_1)f_{\bar s}(z),
\end{aligned}$$

where $\delta(\bar q, x) = (\bar p, f_{\bar q}(x))$, $\delta(\bar p, y_1) = (\bar r, f_{\bar p}(y_1))$, $\delta(\bar r, y_2) = (\bar s, f_{\bar r}(y_2))$, $\delta(\bar p, y_2) = (\bar r', f_{\bar p}(y_2))$, and $\delta(\bar r', y_1) = (\bar s, f_{\bar r'}(y_1))$. If we write $\bar p = (p_1, \ldots, p_c)$, then there are states $r_i \in Q_i$ and $r_j \in Q_j$ such that

$$\bar r = (p_1, \ldots, p_{i-1}, r_i, p_{i+1}, \ldots, p_c), \tag{7}$$

$$\bar r' = (p_1, \ldots, p_{j-1}, r_j, p_{j+1}, \ldots, p_c), \text{ and} \tag{8}$$

$$\bar s = (p_1, \ldots, p_{i-1}, r_i, p_{i+1}, \ldots, p_{j-1}, r_j, p_{j+1}, \ldots, p_c) \tag{9}$$

(we assume w.l.o.g. that $i < j$). Moreover, $f_{\bar p}(y_1) = f_{i,p_i}(y_1) = f_{\bar r'}(y_1) \in \{a_i, a_i^{-1}\}^+$ and $f_{\bar r}(y_2) = f_{j,p_j}(y_2) = f_{\bar p}(y_2) \in \{a_j, a_j^{-1}\}^+$. Thus, we have

$$\begin{aligned}
f_{\bar q}(u) &= f_{\bar q}(x)f_{\bar p}(y_1)f_{\bar r}(y_2)f_{\bar s}(z)\\
&= f_{\bar q}(x)f_{i,p_i}(y_1)f_{j,p_j}(y_2)f_{\bar s}(z)\\
&\leftrightarrow_s f_{\bar q}(x)f_{j,p_j}(y_2)f_{i,p_i}(y_1)f_{\bar s}(z)\\
&= f_{\bar q}(x)f_{\bar p}(y_2)f_{\bar r'}(y_1)f_{\bar s}(z)\\
&= f_{\bar q}(u').
\end{aligned}$$

Moreover, since $\mathcal{P}_i(u') = \mathcal{P}_i(u)$ and $\equiv_{G_i}$ refines $\equiv_{q_i}$ on $\mathcal{S}_i$ for all $i \in [1,c]$, it follows that $\equiv_{G_i}$ refines $\equiv_{q_i}$ on $\mathcal{P}_i(u') \cup \mathcal{P}_i(v)$ for all $i \in [1,c]$.

*Case 2.* $u \to_d u'$. Then we obtain a factorization $u = xyz$, where $y \in \Sigma_i^+$ is a block, $y \equiv_{G_i} 1$, and $u' = xz$. We obtain a factorization

$$f_{\bar q}(u) = f_{\bar q}(x)f_{\bar r}(y)f_{\bar s}(z),$$

where $\delta(\bar q, x) = (\bar r, f_{\bar q}(x))$ and $\delta(\bar r, y) = (\bar s, f_{\bar r}(y))$. The word $f_{\bar r}(y)$ is a block of $f_{\bar q}(u)$. For the projection $\pi_i(u)$ we have $\pi_i(u) = \pi_i(x)y\pi_i(z)$. Since $\equiv_{G_i}$ refines $\equiv_{q_i}$ on $\mathcal{S}_i$ and $\pi_i(x) \equiv_{G_i} \pi_i(x)y$, we obtain $\pi_i(x) \equiv_{q_i} \pi_i(x)y$. Since $r_i$ (resp., $s_i$) is the

state reached from $q_i$ by the automaton $\mathcal{A}_i$ after reading $\pi_i(x)$ (resp., $\pi_i(x)y$), we obtain $r_i = s_i$ and hence $\bar{r} = \bar{s}$. This implies

$$f_{\bar{r}}(y) \equiv_{\langle a_i \rangle} a_i^{-r_i} a_i^{s_i} \equiv_{\langle a_i \rangle} 1.$$

Moreover, we have

$$f_{\bar{q}}(u') = f_{\bar{q}}(xz) = f_{\bar{q}}(x) f_{\bar{r}}(z) = f_{\bar{q}}(x) f_{\bar{s}}(z).$$

We therefore get $f_{\bar{q}}(u) \to_d f_{\bar{q}}(u')$.

It remains to show that $\equiv_{G_j}$ refines $\equiv_{q_j}$ on $\mathcal{P}_j(u') \cup \mathcal{P}_j(v)$ for every $j \in [1, c]$. For $j \neq i$ this is clear since $\mathcal{P}_j(u') \cup \mathcal{P}_j(v) = \mathcal{S}_j$. For $j = i$ we can use Lemma 8.3 for the words $\pi_i(u) = \pi_i(x) y \pi_i(z)$ and $\pi_i(v)$. This concludes the proof of Claim 1.

*Claim 2.* Assume that $\bar{q} = (q_1, \ldots, q_c)$ is such that $\equiv_{q_i}$ refines $\equiv_{G_i}$ on $\mathcal{S}_i$ for every $i \in [1, c]$. If $f_{\bar{q}}(u) \to_{sd}^* \tilde{u}$ and $f_{\bar{q}}(v) \to_{sd}^* \tilde{v}$, then there are $u', v' \in \Sigma^*$ such that $u \to_{sd}^* u'$, $v \to_{sd}^* v'$, $f_{\bar{q}}(u') = \tilde{u}$, $f_{\bar{q}}(v') = \tilde{v}$ and $\equiv_{q_i}$ refines $\equiv_{G_i}$ on $\mathcal{P}_i(u') \cup \mathcal{P}_i(v')$ for every $i \in [1, c]$.

*Proof of Claim 2.* The proof is very similar to the proof of Claim 1. As in the proof of Claim 1, it suffices to consider the case where $f_{\bar{q}}(u) \to_{sd} \tilde{u}$ and $\tilde{v} = f_{\bar{q}}(v)$.

*Case 1.* $f_{\bar{q}}(u) \leftrightarrow_s \tilde{u}$. Since the blocks of $u$ are translated into the blocks of $f_{\bar{q}}(u)$ by the sequential transducer $\mathcal{T}_{\bar{q}}$, we obtain a factorization $u = x y_1 y_2 z$ for blocks $y_1 \in \Sigma_i^+, y_2 \in \Sigma_j^+$ of $u$ such that $(i, j) \in I$ (in particular $i \neq j$) and

$$\begin{aligned} f_{\bar{q}}(u) &= f_{\bar{q}}(x) f_{\bar{p}}(y_1) f_{\bar{r}}(y_2) f_{\bar{s}}(z), \\ \tilde{u} &= f_{\bar{q}}(x) f_{\bar{r}}(y_2) f_{\bar{p}}(y_1) f_{\bar{s}}(z). \end{aligned}$$

Here, the state tuples $\bar{p} = (p_1, \ldots, p_c)$, $\bar{r}$, and $\bar{s}$ are as in the proof of Claim 1, see in particular (7) and (9). We can then define the tuple $\bar{r}'$ as in (8) and get

$$\begin{aligned} f_{\bar{p}}(y_1) = f_{i,p_i}(y_1) &= f_{\bar{r}'}(y_1) \in \{a_i, a_i^{-1}\}^+ \text{ and} \\ f_{\bar{r}}(y_2) = f_{j,p_j}(y_2) &= f_{\bar{p}}(y_2) \in \{a_j, a_j^{-1}\}^+. \end{aligned}$$

We thus have

$$\tilde{u} = f_{\bar{q}}(x) f_{\bar{r}}(y_2) f_{\bar{p}}(y_1) f_{\bar{s}}(z) = f_{\bar{q}}(x) f_{\bar{p}}(y_2) f_{\bar{r}'}(y_1) f_{\bar{s}}(z) = f_{\bar{q}}(x y_2 y_1 z).$$

Clearly, we also have $u = x y_1 y_2 z \to_s x y_2 y_1 z$. So, we can set $u' = x y_2 y_1 z$. Since $\mathcal{P}_i(u') = \mathcal{P}_i(u)$ for all $i \in [1, c]$, it follows that $\equiv_{q_i}$ refines $\equiv_{G_i}$ on $\mathcal{P}_i(u') \cup \mathcal{P}_i(v)$ for all $i \in [1, c]$.

*Case 2.* $f_{\bar{q}}(u) \to_d \tilde{u}$. Then we obtain a factorization $u = xyz$, where $y \in \Sigma_i^+$ is a block of $u$,

$$\begin{aligned} f_{\bar{q}}(u) &= f_{\bar{q}}(x) f_{\bar{r}}(y) f_{\bar{s}}(z), \text{ and} \\ \tilde{u} &= f_{\bar{q}}(x) f_{\bar{s}}(z). \end{aligned}$$

The state tuples $\bar{r}$ and $\bar{s}$ are such that $\delta(\bar{q}, x) = (\bar{r}, f_{\bar{q}}(x))$ and $\delta(\bar{r}, y) = (\bar{s}, f_{\bar{r}}(y))$. Moreover, the word $f_{\bar{r}}(y)$ is a block of $f_{\bar{q}}(u)$ with

$$a_i^{-r_i} a_i^{s_i} \equiv_{\langle a_i \rangle} f_{\bar{r}}(y) \equiv_{\langle a_i \rangle} 1.$$

This implies that $r_i = s_i$ and hence $\bar{r} = \bar{s}$. We therefore have

$$\rho_i(q_i, \pi_i(x)) = r_i = s_i = \rho_i(q_i, \pi_i(x) y).$$

Since $\equiv_{q_i}$ refines $\equiv_{G_i}$ on $\mathcal{S}_i$ and $\pi_i(x), \pi_i(x)y \in \mathcal{S}_i$, we get $\pi_i(x) \equiv_{G_i} \pi_i(x)y$, i.e., $y \equiv_{G_i} 1$. If we set $u' = xz$ we get $u \to_d u'$ and

$$\tilde{u} = f_{\bar{q}}(x)f_{\bar{s}}(z) = f_{\bar{q}}(x)f_{\bar{r}}(z) = f_{\bar{q}}(xz) = f_{\bar{q}}(u').$$

It remains to show that $\equiv_{q_j}$ refines $\equiv_{G_j}$ on $\mathcal{P}_j(u') \cup \mathcal{P}_j(v)$ for every $j \in [1, c]$. For $j \neq i$ this is clear since $\mathcal{P}_j(u') \cup \mathcal{P}_j(v) = \mathcal{S}_j$. For $j = i$ we can use Lemma 8.4 for the words $\pi_i(u) = \pi_i(x)y\pi_i(z)$ and $\pi_i(v)$. This concludes the proof of Claim 2.

We now estimate the error for the input words $u$ and $v$. There are two cases to consider:

*Case 1.* $u \equiv_G v$. We will show that Algorithm 1 reaches with probability at least $1 - 2\epsilon_1(n)cn^2$ the same state when running on $u$ and $v$, respectively. For this, assume that the randomly selected initial states $q_i \in Q_i$ are such that $\equiv_{G_i}$ refines $\equiv_{q_i}$ on $\mathcal{S}_i$ for all $i \in [1, c]$. This happens with probability at least $1 - 2\epsilon_1(n)cn^2$.

First note that $u \equiv_G v$ implies $\pi_i(u) \equiv_{G_i} \pi_i(v)$ for all $i \in [1, c]$. Since $\equiv_{G_i}$ refines $\equiv_{q_i}$ on $\mathcal{S}_i$, we obtain $\rho_i(q_i, \pi_i(u)) = \rho_i(q_i, \pi_i(v))$. It remains to show that after reading $u$ and $v$, also the states of $\mathcal{B}_m$ are the same. For this we show that $f_{\bar{q}}(u) \equiv_H f_{\bar{q}}(v)$ in the graph group $H$.

From Lemma 10.6 it follows that there are reduced words $u', v' \in \Sigma^*$ such that $u \to_{sd}^* u'$, $v \to_{sd}^* v'$, and $u' \leftrightarrow_r^* v'$. Claim 1 implies $f_{\bar{q}}(u) \to_{sd}^* f_{\bar{q}}(u')$, $f_{\bar{q}}(v) \to_{sd}^* f_{\bar{q}}(v')$, and $\equiv_{G_i}$ refines $\equiv_{q_i}$ on $\mathcal{P}_i(u') \cup \mathcal{P}_i(v')$ for every $i \in [1, c]$. Since $u' \leftrightarrow_r^* v'$ we can write the block factorizations of $u'$ and $v'$ as $u' = u_1u_2 \cdots u_l$ and $v' = v_1v_2 \cdots v_l$ with $u_i, v_i \in \Sigma_{j_i}^+$ for some $j_i \in [1, c]$ and $u_i \equiv_{G_{j_i}} v_i$ for all $i \in [1, l]$. The block factorizations of $f_{\bar{q}}(u')$ and $f_{\bar{q}}(v')$ can be written as $f_{\bar{q}}(u') = \tilde{u}_1\tilde{u}_2 \cdots \tilde{u}_l$ and $f_{\bar{q}}(v') = \tilde{v}_1\tilde{v}_2 \cdots \tilde{v}_l$ with $\tilde{u}_i, \tilde{v}_i \in \{a_{j_i}, a_{j_i}^{-1}\}^+$.

We claim that $\tilde{u}_i \equiv_{\langle a_{j_i} \rangle} \tilde{v}_i$ for all $i \in [1, l]$, which implies $f_{\bar{q}}(u') \equiv_H f_{\bar{q}}(v')$. Since $u_i \equiv_{G_{j_i}} v_i$ for all $i \in [1, l]$, we get the following: if $u'' = u_{k_1}u_{k_2} \cdots u_{k_e} \in \Sigma_j^*$ is a pure prefix of $u'$ for some $j \in [1, c]$ then $v'' = v_{k_1}v_{k_2} \cdots v_{k_e} \in \Sigma_j^*$ is a pure prefix of $v'$ such that $u'' \equiv_{G_j} v''$. Since $\equiv_{G_j}$ refines $\equiv_{q_j}$ on $\mathcal{P}_j(u') \cup \mathcal{P}_j(v')$ and $u'', v'' \in \mathcal{P}_j(u') \cup \mathcal{P}_j(v')$, we obtain $\rho_j(q_j, u'') = \rho_j(q_j, v'')$. This implies $\tilde{u}_i \equiv_{\langle a_{j_i} \rangle} \tilde{v}_i$ for all $i \in [1, l]$ and hence $f_{\bar{q}}(u') \equiv_H f_{\bar{q}}(v')$. From this, we finally get $f_{\bar{q}}(u) \equiv_H f_{\bar{q}}(u') \equiv_H f_{\bar{q}}(v') \equiv_H f_{\bar{q}}(v)$.

Recall that $f_{\bar{q}}(u)$ (resp., $f_{\bar{q}}(v)$ is the word fed into the semiPFA $\mathcal{B}_m$ on input $u$ (resp., $v$). Since $(\mathcal{B}_n)_{n \geq 0}$ is a $(1/n^d, 0)$-distinguisher for the graph group $H$, it follows that $f_{\bar{q}}(u)$ and $f_{\bar{q}}(v)$ lead in $\mathcal{B}_m$ with probability one to the same state. Hence, Algorithm 1 reaches with probability at least $1 - 2\epsilon_1(n)cn^2$ the same state when running on $u$ and $v$, respectively.

*Case 2.* $u \not\equiv_G v$. We will show that Algorithm 1 reaches with probability at least $1 - (2\epsilon_0(n)cn^2 + 1/n^d)$ different states when running on $u$ and $v$, respectively. To show this, assume that the randomly selected initial states $q_i \in Q_i$ are such that $\equiv_{q_i}$ refines $\equiv_{G_i}$ on $\mathcal{S}_i$ for all $i \in [1, c]$. This happens with probability at least $1 - 2\epsilon_0(n)cn^2$.

We claim that $f_{\bar{q}}(u) \not\equiv_H f_{\bar{q}}(v)$ holds, where $\bar{q} = (q_1, \ldots, q_c)$. In order to get a contradiction, assume that $f_{\bar{q}}(u) \equiv_H f_{\bar{q}}(v)$. From Lemma 10.6 it follows that there are reduced words $\tilde{u}, \tilde{v} \in \Delta^*$ such that $f_{\bar{q}}(u) \to_{sd}^* \tilde{u}$, $f_{\bar{q}}(v) \to_{sd} \tilde{v}$ and $\tilde{u} \leftrightarrow_r^* \tilde{v}$. Claim 2 implies that there exist $u', v' \in \Sigma^*$ such that $u \to_{sd}^* u'$, $v \to_{sd}^* v'$, $f_{\bar{q}}(u') = \tilde{u}$, $f_{\bar{q}}(v') = \tilde{v}$ and $\equiv_{q_i}$ refines $\equiv_{G_i}$ on $\mathcal{P}_i(u') \cup \mathcal{P}_i(v')$ for every $i \in [1, c]$.

Since $f_{\bar{q}}(u') = \tilde{u} \leftrightarrow_r^* \tilde{v} = f_{\bar{q}}(v')$ we can write the block factorizations of $f_{\bar{q}}(u')$ and $f_{\bar{q}}(v')$ as $f_{\bar{q}}(u') = \tilde{u}_1\tilde{u}_2\cdots\tilde{u}_l$ and $f_{\bar{q}}(v') = \tilde{v}_1\tilde{v}_2\cdots\tilde{v}_l$ with $\tilde{u}_i, \tilde{v}_i \in \{a_{j_i}, a_{j_i}^{-1}\}^+$ for some $j_i \in [1,c]$ and $\tilde{u}_i \equiv_{\langle a_{j_i}\rangle} \tilde{v}_i$ for all $i \in [1,l]$. Clearly, the block factorizations of $u'$ and $v'$ can then be written as $u' = u_1u_2\cdots u_l$ and $v' = v_1v_2\cdots v_l$, where the block $u_i \in \Sigma_{j_i}^+$ (resp., $v_i \in \Sigma_{j_i}^+$) is translated into the block $\tilde{u}_i$ (resp., $\tilde{v}_i$) by the sequential transducer $\mathcal{T}_{\bar{q}}$.

We claim that $u_i \equiv_{G_{j_i}} v_i$ for all $i \in [1,l]$. Since $\tilde{u}_i \equiv_{\langle a_{j_i}\rangle} \tilde{v}_i$ for all $i \in [1,l]$ we have the following: if $\tilde{u}'' = \tilde{u}_{k_1}\tilde{u}_{k_2}\cdots\tilde{u}_{k_e} \in \{a_j, a_j^{-1}\}^*$ is a pure prefix of $f_{\bar{q}}(u')$ for some $j \in [1,c]$ then $\tilde{v}'' = \tilde{v}_{k_1}\tilde{v}_{k_2}\cdots\tilde{v}_{k_e} \in \{a_j, a_j^{-1}\}^*$ is a pure prefix of $f_{\bar{q}}(v')$ such that $\tilde{u}'' \equiv_{\langle a_j\rangle} \tilde{v}''$. Let $p_j = \rho_j(q_j, u_{k_1}u_{k_2}\cdots u_{k_e})$ and $r_j = \rho_j(q_j, v_{k_1}v_{k_2}\cdots v_{k_e})$. We then have

$$a_j^{-q_j}a_j^{p_j} \equiv_{\langle a_j\rangle} \tilde{u}'' \equiv_{\langle a_j\rangle} \tilde{v}'' \equiv_{\langle a_j\rangle} a_j^{-q_j}a_j^{r_j},$$

i.e., $p_j = r_j$. Since $\equiv_{q_j}$ refines $\equiv_{G_j}$ on $\mathcal{P}_j(u') \cup \mathcal{P}_j(v')$ and $u_{k_1}u_{k_2}\cdots u_{k_e}$ as well as $v_{k_1}v_{k_2}\cdots v_{k_e}$ belong to $\mathcal{P}_j(u') \cup \mathcal{P}_j(v')$, we obtain $u_{k_1}u_{k_2}\cdots u_{k_e} \equiv_{G_j} v_{k_1}v_{k_2}\cdots v_{k_e}$. This holds for all pure prefixes of $u'$. We therefore have $u_i \equiv_{G_{j_i}} v_i$ for all $i \in [1,l]$, which implies $u' \equiv_G v'$. Finally, we get $u \equiv_G u' \equiv_G v' \equiv_G v$, which is a contradiction. Hence, we must have $f_{\bar{q}}(u) \not\equiv_H f_{\bar{q}}(v)$.

Since the algorithm feeds $f_{\bar{q}}(u)$ (resp., $f_{\bar{q}}(v)$) into the semiPFA $\mathcal{B}_m$, the latter reaches different states with probability at least $1 - 1/m^d \geq 1 - 1/n^d$ (under the assumption that $\equiv_{q_i}$ refines $\equiv_{G_i}$ on $\mathcal{S}_i$ for all $i \in [1,c]$). Hence, the probability that Algorithm 1 reaches different states when running on $u$ and $v$ is at least $(1 - 2\epsilon_0(n)cn^2)(1 - 1/n^d) \geq 1 - (2\epsilon_0(n)cn^2 + 1/n^d)$. □

10.3. **Randomized streaming algorithms for wreath products.** In this section we will investigate distinguishers for wreath products. We start with the definition of the wreath product of two group.

Let $G$ and $H$ be groups. Consider the direct sum $K = \bigoplus_{g\in G} H_g$, where $H_g$ is a copy of $H$. We view $K$ as the set $H^{(G)}$ of all mappings $f\colon G \to H$ such that $\mathsf{supp}(f) := \{g \in G\colon f(g) \neq 1\}$ is finite, together with pointwise multiplication in $H$ as the group operation. The set $\mathsf{supp}(f) \subseteq G$ is called the *support* of $f$. The group $G$ has a natural left action on $H^{(G)}$ given by $gf(a) = f(g^{-1}a)$, where $f \in H^{(G)}$ and $g, a \in G$. The corresponding semidirect product $H^{(G)} \rtimes G$ is the *wreath product* $H \wr G$. More concretely:

- Elements of $H \wr G$ are pairs $(f, g)$, where $g \in G$ and $f \in H^{(G)}$.
- The multiplication in $H \wr G$ is defined as follows: Let $(f_1, g_1), (f_2, g_2) \in H \wr G$. Then $(f_1, g_1)(f_2, g_2) = (f, g_1g_2)$, where $f(a) = f_1(a)f_2(g_1^{-1}a)$ for all $a \in G$.

The following intuition might be helpful: An element $(f, g) \in H \wr G$ can be seen as a finite multiset of elements of $H \setminus \{1_H\}$ that are sitting at certain elements of $G$ (the mapping $f$) together with the distinguished element $g \in G$, which can be thought of as a cursor moving in $G$. If we want to compute the product $(f_1, g_1)(f_2, g_2)$, we do this as follows: First, we shift the finite collection of $H$-elements that corresponds to the mapping $f_2$ by $g_1$: If the element $h \in H \setminus \{1_H\}$ is sitting at $a \in G$ (i.e., $f_2(a) = h$), then we remove $h$ from $a$ and put it to the new location $g_1a \in G$. This new collection corresponds to the mapping $f_2'\colon a \mapsto f_2(g_1^{-1}a)$. After this shift, we multiply the two collections of $H$-elements pointwise: If in $a \in G$ the elements $h_1$ and $h_2$ are sitting (i.e., $f_1(a) = h_1$ and $f_2'(a) = h_2$), then we put the product

$h_1h_2$ into the location $a$. Finally, the new distinguished $G$-element (the new cursor position) becomes $g_1g_2$.

Clearly, $G$ is a subgroup of $H \wr G$. We also regard $H$ as a subgroup of $H \wr G$ by identifying $H$ with the set of all $f \in H^{(G)}$ with $\mathsf{supp}(f) \subseteq \{1\}$. This copy of $H$ together with $G$ generates $H \wr G$. In particular, if $G = \langle \Sigma \rangle$ and $H = \langle \Gamma \rangle$ with $\Sigma \cap \Gamma = \emptyset$ then $H \wr G$ is generated by $\Sigma \cup \Gamma$. In [58] it was shown that the word problem of a wreath product $H \wr G$ is $\mathsf{TC}^0$-reducible to the word problems for $G$ and $H$.

The above wreath product $H \wr G$ is also called the restricted wreath product. In the unrestricted wreath product one takes all mappings $f\colon G \to H$ and not only those with finite support. We will only consider the restricted wreath product and just call it the wreath product. The reason for this is that the unrestricted wreath product of two finitely generated groups is in general not finitely generated.

In the rest of this section, we construct distingushers for wreath products. The case of a wreath product $H \wr G$ with $G$ finite is easy:

**Theorem 10.8.** *Let $H$ be a finitely generated group for which there exists an $(\epsilon_0, \epsilon_1)$-distinguisher $\mathcal{R} = (\mathcal{A}_n)_{n\geq 0}$ and let $G$ be a finite group of size $c$. Then, there exists an $(c \cdot \epsilon_0, c \cdot \epsilon_1)$-distinguisher for $H \wr G$ with space complexity $\mathcal{O}(s(\mathcal{R}, n))$.*

*Proof.* We run $c$ independent copies of $\mathcal{A}_n$ (for the direct product of $c$ copies of $H$). In addition we have to store an element of $G$ (the cursor in the above intuition). □

The case of a wreath product $H \wr G$ with $G$ infinite turns out to be more interesting. We will start with wreath products $Z \wr G$, where $G$ is infinite and $Z$ is a cyclic group and split this case into three subcases:

- $Z = \mathbb{Z}$; see Theorem 10.9,
- $Z = \mathbb{Z}_p$ for a prime $p$; see Theorem 10.10,
- $Z = \mathbb{Z}_{p^k}$ for a prime $p$ and $k \geq 2$; see Theorem 10.12.

The case $Z = \mathbb{Z}_m$ with $m$ not a prime power (and more generally, the case of a wreath product $A \wr G$ with $A$ finitely generated abelian) will follow easily from those cases; see Corollary 10.13. In the following, $\Sigma$ denotes a finite symmetric generating set for the group $G$ and $a$ denotes a generator for the cyclic group $Z$. Then, $\Gamma = \Sigma \cup \{a, a^{-1}\}$ is a symmetric generating set for the wreath product $Z \wr G$.

**Theorem 10.9.** *Let $G$ be a f.g. infinite group and $\mathcal{R} = (\mathcal{A}_n)_{n\geq 0}$ an $(\epsilon_0, \epsilon_1)$-distinguisher for $G$. Let $d$ be a fixed constant and define*

$$\begin{aligned} \zeta_0(n) &= 2\max\{\epsilon_0(n), \epsilon_1(n)\}n^2 + 1/n^d, \\ \zeta_1(n) &= 2\epsilon_1(n)n^2 + \epsilon_1(n). \end{aligned}$$

*Then there is a $(\zeta_0, \zeta_1)$-distinguisher for $\mathbb{Z} \wr G$ with space complexity $3 \cdot s(\mathcal{R}, n) + \Theta(\log n)$.*

*Proof.* Fix an input length $n$ and let $\mathcal{A}_n = (Q_n, \Sigma, \iota_n, \rho_n)$. W.l.o.g. we can assume that $Q_n = [0, |Q_n| - 1]$ consists of the first $|Q_n|$ integers.

For a word $u = va^\beta$ with $v \in \Gamma^*$ and $\beta \in \{-1, 1\}$ we define $\sigma(u) = \beta$.

Consider a potential input word $w \in \Gamma^{\leq n}$ and a state $q \in Q_n$ (later, it will be randomly guessed according to the initial state distribution $\iota_n$). Define the mapping $\rho_q : \Sigma^* \to Q$ by $\rho_q(w) = \rho_n(q, w)$. With $w$ and $q$ we associate a polynomial $P_{q,w}(x) \in \mathbb{Z}[x]$ as follows: Let $R_w$ be the set of all prefixes of $w$ that end with a

letter $a^\gamma$ for $\gamma \in \{-1, 1\}$. For every $v \in R_w$ consider the $\mathcal{A}_n$-state $q_v = \rho_q(\pi_\Sigma(v)) \in [0, |Q_n| - 1]$. We then define the polynomial

$$P_{q,w}(x) := \sum_{v \in R_w} \sigma(v) \cdot x^{q_v}. \tag{10}$$

Note that this polynomial has degree at most $|Q_n| - 1$, all its coefficients have absolute value at most $n$, and there are at most $n$ monomials.

Let us write $(f_w, g_w) \in \mathbb{Z} \wr G$ for the group element represented by the word $w$. The element $g_w \in G$ is obtained by evaluating the projection $\pi_\Sigma(w)$ in the group $G$. For this, the streaming algorithm for $\mathbb{Z} \wr G$ that we aim for (and that we describe in detail later) will simply use the semiPFA $\mathcal{A}_n$. The difficult part is the mapping $f_w$. For this we make use of the polynomial $P_{q,w}(x)$.

*Claim 1.* Let $u, v \in \Gamma^{\leq n}$ be two input words such that $f_u = f_v$. Then we have

$$\mathop{\mathsf{Prob}}_{q \in Q_n}[P_{q,u}(x) = P_{q,v}(x)] \geq 1 - 2\epsilon_1(n)n^2.$$

*Proof of Claim 1.* Define the set of words $S = \pi_\Sigma(R_u \cup R_v) \subseteq \Sigma^*$ and let $G_S = \pi_G(S) \subseteq G$ be the finite set of group elements represented by the words in $S$. Clearly, $|S| \leq 2n$. We will use the equivalence relations $\equiv_q$ ($q \in Q_n$) from Lemma 8.2. It suffices to show for every state $q \in Q_n$ the following: if $\equiv_G$ refines $\equiv_q$ on $S$, then $P_{q,u}(x) = P_{q,v}(x)$. If this is shown then the second statement of Lemma 8.2 implies

$$\begin{aligned}
\mathop{\mathsf{Prob}}_{q \in Q_n}[P_{q,u}(x) = P_{q,v}(x)] &\geq \mathop{\mathsf{Prob}}_{q \in Q_n}[\equiv_G \text{ refines } \equiv_q \text{ on } S] \\
&\geq 1 - \epsilon_1(n)\binom{|S|}{2} \\
&\geq 1 - 2\epsilon_1(n)n^2.
\end{aligned}$$

So, consider a state $q \in Q_n$ and assume that $\equiv_G$ refines $\equiv_q$ on $S$. Let $Q_S = \rho_q(S)$ be the image of $S$ under the mapping $\rho_q$. It is the set of states of $\mathcal{A}_n$ that can be reached from $q$ via words from $S$. Let $[S]_{\equiv_G}$ and $[S]_{\equiv_q}$ be the set of equivalence classes of $\equiv_G$ and $\equiv_q$, respectively, on $S$. With every state $r \in Q_S$ we can associate the equivalence class $A_r = \rho_q^{-1}(r) \cap S \in [S]_{\equiv_q}$. Similarly, with every group element $g \in G_S$ we associate the equivalence class $B_g = \pi_G^{-1}(g) \cap S \in [S]_{\equiv_g}$. Moreover, the $A_r$ ($r \in Q_S$) and $B_g$ ($g \in G_S$) are all equivalence classes of $\equiv_q$ and $\equiv_G$ on $S$, respectively. Since $\equiv_G$ refines $\equiv_q$ on $S$, there is a surjective mapping $h : G_S \to Q_S$ such that

$$A_r = \biguplus_{g \in h^{-1}(r)} B_g$$

for every $r \in Q_S$. The definition of the mappings $f_u$ and $f_v$ yields for every $g \in G_S$:

$$\sum_{y \in \pi_\Sigma^{-1}(B_g) \cap R_u} \sigma(y) \;=\; f_u(g) \;=\; f_v(g) \;=\; \sum_{y \in \pi_\Sigma^{-1}(B_g) \cap R_v} \sigma(y).$$

Note that for every $r \in Q_S$ we have

$$\pi_\Sigma^{-1}(A_r) \cap R_u = \pi_\Sigma^{-1}\left(\biguplus_{g \in h^{-1}(r)} B_g\right) \cap R_u = \biguplus_{g \in h^{-1}(r)} (\pi_\Sigma^{-1}(B_g) \cap R_u)$$

and similarly for $R_v$. If we write $P_{q,u} * x^r$ for the coefficient of the monomial $x^r$ (where $r \in Q_S$) in the polynomial $P_{q,u}(x)$ and analogously for $P_{q,v}(x)$, then we obtain for every $r \in Q_S$:

$$\begin{aligned} P_{q,u} * x^r = \sum_{y \in \pi_\Sigma^{-1}(A_r) \cap R_u} \sigma(y) &= \sum_{g \in h^{-1}(r)} \sum_{y \in \pi_\Sigma^{-1}(B_g) \cap R_u} \sigma(y) \\ &= \sum_{g \in h^{-1}(r)} \sum_{y \in \pi_\Sigma^{-1}(B_g) \cap R_v} \sigma(y) \\ &= \sum_{y \in \pi_\Sigma^{-1}(A_r) \cap R_v} \sigma(y) \;=\; P_{q,v} * x^r. \end{aligned}$$

Hence, we finally get $P_{q,u}(x) = P_{q,v}(x)$, which proves Claim 1.

*Claim 2.* Let $u, v \in \Gamma^{\leq n}$ be two input words such that $f_u \neq f_v$. Then we have

$$\mathop{\mathsf{Prob}}_{q \in Q_n}[P_{q,u}(x) \neq P_{q,v}(x)] \geq 1 - 2\max\{\epsilon_0(n), \epsilon_1(n)\}n^2.$$

*Proof of Claim 2.* The proof is very similar to the proof of Claim 1. Assume that $f_u \neq f_v$. We use the same notations as in the proof of Claim 1. By the first statement of Lemma 8.2 it suffices to show for every state $q \in Q_n$: if $\equiv_G$ equals $\equiv_q$ on $S$ then $P_{q,u}(x) \neq P_{q,v}(x)$.

Assume that $\equiv_G$ equals $\equiv_q$ on $S$ for a state $q$. There is a bijection $h : G_S \to Q_S$ such that $B_g = A_{h(g)}$ for every $g \in G_S$. Since $f_u \neq f_v$ there is a $g \in G_S$ with

$$\sum_{y \in \pi_\Sigma^{-1}(B_g) \cap R_u} \sigma(y) \;=\; f_u(g) \;\neq\; f_v(g) \;=\; \sum_{y \in \pi_\Sigma^{-1}(B_g) \cap R_v} \sigma(y).$$

For $r = h(g)$ we obtain

$$\begin{aligned} P_{q,u} * x^r \;&=\; \sum_{y \in \pi_\Sigma^{-1}(A_r) \cap R_u} \sigma(y) \;=\; \sum_{y \in \pi_\Sigma^{-1}(B_g) \cap R_u} \sigma(y) \\ &\neq \sum_{y \in \pi_\Sigma^{-1}(B_g) \cap R_v} \sigma(y) \;=\; \sum_{y \in \pi_\Sigma^{-1}(A_r) \cap R_v} \sigma(y) \;=\; P_{q,v} * x^r. \end{aligned}$$

Thus, we have $P_{q,u}(x) \neq P_{q,v}(x)$, which proves Claim 2.

In order to verify $f_u = f_v$, a randomized streaming algorithm could compute the polynomial $P_{q,w}(x)$ for an input word $w \in \Gamma^{\leq n}$ and a random initial state $q$. The problem is that the polynomial $P_{q,w}(x)$ does not fit into the space bound we are aiming for. Therefore, we only can afford to compute a fingerprint of $P_{q,w}(x)$. This fingerprint is obtained in two steps.

First, observe that the Cauchy bound[8] implies for all words $u, v \in \Gamma^{\leq n}$ that $P_{q,u}(x) = P_{q,v}(x)$ if and only if $P_{q,u}(2n+1) = P_{q,v}(2n+1)$. To see this, note that all coefficients of the polynomial $P_{q,u}(x) - P_{q,v}(x) \in \mathbb{Z}[x]$ are bounded in absolute value by $2n$. Hence, $2n+1$ is not a root of the polynomial $P_{q,u}(x) - P_{q,v}(x)$.

The value $P_{q,w}(2n+1)$ still needs too many bits. Therefore, our streaming algorithm will compute it only modulo a sufficiently large prime number $p$. Note that for every word $w \in \Gamma^{\leq n}$ we have

$$|P_{q,w}(2n+1)| \leq n \cdot (2n+1)^{|Q_n|-1}.$$

[8]The Cauchy bound says that for a polynomial $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots a_1 x + a_0$ every root $\alpha$ of $p(x)$ satisfies $|\alpha| < 1 + \max\{|a_0|, |a_1|, \ldots |a_{n-1}|\}/|a_n|$.

**Algorithm 2:** $(\zeta_0, \zeta_1)$-distinguisher for $\mathbb{Z} \wr G$

**global variables:** prime number $p$, integer $z \in [0, p-1]$, state $q \in Q_n$

**initialization:**

1 guess $p \in \mathbb{P}_{\alpha(n)}$ according to the uniform distribution ;
2 guess $q \in Q_n$ according to the input distribution $\iota_n$ of $\mathcal{A}_n$ ;
3 $z := 0$ ;

**next input letter:** $b \in \Gamma$

4 **if** $b \in \Sigma$ **then**
5 $\quad q := \rho_n(q, a)$
6 **end**
7 **if** $b = a^\beta$ *for* $\beta \in \{-1, 1\}$ **then**
8 $\quad z := (z + \beta \cdot (2n+1)^q) \bmod p$
9 **end**

Hence, for two input words $u, v \in \Gamma^{\leq n}$ we have

$$D_{u,v} := |P_{q,u}(2n+1) - P_{q,v}(2n+1)| \leq 2n(2n+1)^{|Q_n|-1} \leq (2n+1)^{|Q_n|}.$$

Let $\mathbb{P}_\alpha$ be the set of prime numbers in $[2, \alpha]$. We want to choose $\alpha = \alpha(n)$ large enough such that for a prime $p$ uniformly chosen from $\mathbb{P}_\alpha$ we obtain

$$\mathop{\mathsf{Prob}}_{p \in \mathbb{P}_\alpha}[p \text{ divides } D_{u,v}] \leq 1/n^d \tag{11}$$

(in case $D_{u,v} \neq 0$), where $d$ is the parameter from the theorem.

The number of different prime factors of $D_{u,v}$ is bounded by

$$\begin{aligned} \frac{\ln D_{u,v}}{\ln \ln D_{u,v}} \cdot (1 + o(1)) &\leq \frac{\ln\left((2n+1)^{|Q_n|}\right)}{\ln \ln \left((2n+1)^{|Q_n|}\right)} \cdot (1 + o(1)) \\ &\leq \mathcal{O}\left(\frac{|Q_n| \cdot \log n}{\log |Q_n| + \log \log n}\right) \leq \mathcal{O}\left(\frac{|Q_n| \cdot \log n}{\log \log n}\right) \end{aligned}$$

(recall that $\log |Q_n| \geq s(\mathcal{R}, n) - 1 \geq \Omega(\log \log n)$ since $G$ is infinite, see Remark 6.3). Moreover, there are $\Theta(x)$ many primes of size at most $x \log x$. Hence, by fixing the number $\alpha = \alpha(n)$ such that

$$\begin{aligned} \alpha(n) &= \Theta\left(\frac{|Q_n| \cdot \log n \cdot n^d}{\log \log n} \cdot \log\left(\frac{|Q_n| \cdot \log n \cdot n^d}{\log \log n}\right)\right) \\ &= \Theta\left(\frac{|Q_n| \cdot \log n \cdot n^d}{\log \log n} \cdot (\log |Q_n| + d \log n)\right) \leq \Theta(|Q_n| \cdot n^{d+2}), \end{aligned}$$

we can obtain (11).

We can now finally explain our streaming algorithm for the wreath product $\mathbb{Z} \wr G$; see Algorithm 2. For an input word $w \in \Gamma^{\leq n}$ we simulate in lines 2, 4 and 5 the semiPFA $\mathcal{A}_n$ on the projection $\pi_\Sigma(w)$. In the integer variable $z$ we compute the number $P_{q,w}(2n+1) \bmod p$, where $p$ is the prime guessed in line 1 and $q$ is the state guessed in line 2. If we want to describe the algorithm by a semiPFA, then the state of the algorithm would consist of the prime number $p$ and the current values of $q$ and $z$. The prime number $p$ is not changed when a new symbol $b$ arrives.

The algorithm stores $3 \cdot s(\mathcal{R}, n) + \Theta(\log n)$ bits:

- $s(\mathcal{R}, n) + \Theta(\log n) = \lceil \log |Q_n| \rceil + \Theta(\log n)$ bits for the prime $p \leq \alpha(n)$,
- $s(\mathcal{R}, n) + \Theta(\log n)$ bits for the number $z < p$.
- $s(\mathcal{R}, n)$ bits for the state $q$.

It remains to compute the error probabilities of the algorithm. For this, consider two words $u, v \in \Gamma^{\leq n}$. Let $(f_u, g_u) \in \mathbb{Z} \wr G$ (resp., $(f_v, g_v) \in \mathbb{Z} \wr G$) be the group element represented by the word $u$ (resp., $v$). Let $z(p, q, u)$ (resp., $z(p, q, v)$) be the value of the variable $z$ that Algorithm 2 computes on input $u$ (resp., $v$) when $p$ and $q$ are the random choices in lines 1 and 2, respectively.

*Claim 3.* If $u \equiv_{\mathbb{Z} \wr G} v$ then

$$\mathop{\mathsf{Prob}}_{p \in \mathbb{P}_\alpha, q \in Q_n} [\rho_n(q, \pi_\Sigma(u)) = \rho_n(q, \pi_\Sigma(v)) \wedge z(p, q, u) = z(p, q, v)] \geq 1 - \zeta_1(n).$$

*Proof of Claim 3.* Assume that $u \equiv_{\mathbb{Z} \wr G} v$, i.e., $f_u = f_v$ and $g_u = g_v$. Since we run the algorithm $\mathcal{A}_n$ on $\pi_\Sigma(u)$ and $\pi_\Sigma(v)$, respectively, and $\pi_\Sigma(u) \equiv_G \pi_\Sigma(v)$, we get

$$\mathop{\mathsf{Prob}}_{q \in Q_n} [\rho_n(q, \pi_\Sigma(u)) = \rho_n(q, \pi_\Sigma(v))] \geq 1 - \epsilon_1(n).$$

Moreover, $f_u = f_v$ implies

$$\mathop{\mathsf{Prob}}_{p \in \mathbb{P}_\alpha, q \in Q_n} [z(p, q, u) = z(p, q, v)] \geq \mathop{\mathsf{Prob}}_{q \in Q_n} [P_{q,u}(x) = P_{q,v}(x)] \geq 1 - 2\epsilon_1(n)n^2,$$

where the second inequality follows from Claim 1. In total, we obtain

$$\begin{aligned} & \mathop{\mathsf{Prob}}_{p \in \mathbb{P}_\alpha, q \in Q_n} [\rho_n(q, \pi_\Sigma(u)) \neq \rho_n(q, \pi_\Sigma(v)) \vee z(p, q, u) \neq z(p, q, v)] \\ \leq & \mathop{\mathsf{Prob}}_{p \in \mathbb{P}_\alpha, q \in Q_n} [\rho_n(q, \pi_\Sigma(u)) \neq \rho_n(q, \pi_\Sigma(v))] + \mathop{\mathsf{Prob}}_{p \in \mathbb{P}_\alpha, q \in Q_n} [z(p, q, u) \neq z(p, q, v)] \\ \leq & \epsilon_1(n) + 2\epsilon_1(n)n^2 = \zeta_1(n). \end{aligned}$$

*Claim 4.* If $u \not\equiv_{\mathbb{Z} \wr G} v$ then

$$\mathop{\mathsf{Prob}}_{p \in \mathbb{P}_\alpha, q \in Q_n} [\rho_n(q, \pi_\Sigma(u)) \neq \rho_n(q, \pi_\Sigma(v)) \vee z(p, q, u) \neq z(p, q, v)] \geq 1 - \zeta_0(n).$$

*Proof of Claim 4.* Assume that $u \not\equiv_{\mathbb{Z} \wr G} v$. If $g_u \neq g_v$, i.e., $\pi_\Sigma(u) \not\equiv_G \pi_\Sigma(v)$, then we get

$$\mathop{\mathsf{Prob}}_{q \in Q_n} [\rho_n(q, \pi_\Sigma(u)) \neq \rho_n(q, \pi_\Sigma(v))] \geq 1 - \epsilon_0(n) \geq 1 - \zeta_0(n).$$

On the other hand, if the mappings $f_u$ and $f_v$ are different, we obtain

$$\mathop{\mathsf{Prob}}_{q \in Q_n} [P_{q,u}(x) \neq P_{q,v}(x)] \geq 1 - 2\max\{\epsilon_0(n), \epsilon_1(n)\}n^2.$$

from Claim 2. Consider a state $q$ with $P_{q,u}(x) \neq P_{q,v}(x)$. With inequality (11) for $D_{u,v} := |P_{q,u}(2n+1) - P_{q,v}(2n+1)|$ we obtain the bound

$$\mathop{\mathsf{Prob}}_{p \in \mathbb{P}_\alpha} [z(p, q, u) = z(p, q, v)] = \mathop{\mathsf{Prob}}_{p \in \mathbb{P}_\alpha} [p \text{ divides } D_{u,v}] \leq 1/n^d.$$

for every fixed state $q$ with $P_{q,u}(x) \neq P_{q,v}(x)$. Therefore we have

$$\mathop{\mathsf{Prob}}_{p \in \mathbb{P}_\alpha, q \in Q_n} [z(p, q, u) \neq z(p, q, v) \mid P_{q,u}(x) \neq P_{q,v}(x)] \geq 1 - 1/n^d.$$

Finally, we get

$$\begin{aligned}
& \underset{p\in\mathbb{P}_\alpha, q\in Q_n}{\mathsf{Prob}}[z(p,q,u)\neq z(p,q,v)] \\
= & \underset{p\in\mathbb{P}_\alpha, q\in Q_n}{\mathsf{Prob}}[z(p,q,u)\neq z(p,q,v) \mid P_{q,u}(x)\neq P_{q,v}(x)]\cdot \underset{q\in Q_n}{\mathsf{Prob}}[P_{q,u}(x)\neq P_{q,v}(x)] \\
\geq & (1-1/n^d)\cdot(1-2\max\{\epsilon_0(n),\epsilon_1(n)\}n^2) \\
\geq & 1-(2\max\{\epsilon_0(n),\epsilon_1(n)\}n^2+1/n^d) = 1-\zeta_0(n)
\end{aligned}$$

This proves the theorem. □

It is easy to extend Theorem 10.9 to a wreath product $\mathbb{Z}_p \wr G$ with $p$ prime.

**Theorem 10.10.** *Let $G$ be a f.g. infinite group and $\mathcal{R} = (\mathcal{A}_n)_{n\geq 0}$ an $(\epsilon_0,\epsilon_1)$-distinguisher for $G$. Let $d$ be a fixed constant and define*

$$\begin{aligned}
\zeta_0(n) &= 2\max\{\epsilon_0(n),\epsilon_1(n)\}n^2 + 1/n^d, \\
\zeta_1(n) &= 2\epsilon_1(n)n^2 + \epsilon_1(n).
\end{aligned}$$

*Then there is a $(\zeta_0,\zeta_1)$-distinguisher for $\mathbb{Z}_p \wr G$ with space complexity $3\cdot s(\mathcal{R},n) + \mathcal{O}(\log n)$.*

*Proof.* The proof is mostly identical to the proof of Theorem 10.9 and we reuse most of the notation. We consider the polynomial $P_{q,w}(x)$ from (10) as a polynomial over $\mathbb{F}_p[x]$. Instead of computing the number $z(p,q,w) = P_{q,w}(2n+1) \bmod p$ for a randomly guessed prime $p$, we compute

$$z(r,q,w) = P_{q,w}(r) \in \mathbb{F}_{p^e},$$

where $e \geq 1$ is chosen such that $e \geq \log_p |Q_n| + d\cdot\log_p(n)$ (and hence $|Q_n|/p^e \leq 1/n^d$) and $r \in \mathbb{F}_{p^e}$ is randomly chosen in the initialization phase of the algorithm. Since $P_{q,w}$ has degree at most $|Q_n|-1$, we obtain for all words $u,v \in \Gamma^{\leq n}$:

$$\underset{r\in\mathbb{F}_{p^e}, q\in Q_n}{\mathsf{Prob}}[z(r,q,u)=z(r,q,v) \mid P_{q,u}(x)\neq P_{q,v}(x)] \leq \frac{|Q_n|-1}{p^e} \leq \frac{1}{n^d}.$$

Then, the error bounds of the algorithm can be computed as in the proof of Theorem 10.9. The algorithm has to store $s(\mathcal{R},n) + 2\cdot e\cdot\log_2(p) = 3\cdot s(\mathcal{R},n) + \mathcal{O}(\log n)$ bits (the prime $p$ is a constant). □

Finally, we deal with a wreath product $\mathbb{Z}_{p^k} \wr G$ with $p$ a prime and $k \geq 2$. For this, we need the following strengthening of the famous isolation lemma. Let $S$ be a finite set and $\nu : S \to [1,k]$ for some $k$. We view $\nu(a)$ as the weight associated to $a \in S$. For $A \subseteq S$ we define the weight $\nu(A)$ as $\nu(A) = \sum_{a\in A}\nu(a)$. For a set of subsets $\mathcal{P} \subseteq 2^S$ we say that $\mathcal{P}$ has a *unique minimum weight set with respect to $\nu$* if there is a set $A \in \mathcal{P}$ such that $\nu(A) < \nu(B)$ for all $B \in \mathcal{P}\setminus\{A\}$.

**Theorem 10.11** ([15])**.** *Let $S$ be a finite set of size $n$ and $\mathcal{P} \subseteq 2^S$ with $\mathcal{P} \neq \emptyset$ and $|\mathcal{P}| \leq m$. Let $\mathcal{F}$ be the set of all mappings $\nu : S \to [1,n^7]$. Using $\Theta(\log m + \log n)$ random bits (chosen uniformly and independently) one can construct a random function $\nu \in \mathcal{F}$ such that*

$$\mathsf{Prob}[\mathcal{P} \text{ has a unique minimum weight set with respect to } \nu] \geq 1/4.$$

**Theorem 10.12.** *Let $G$ be a f.g. infinite group, $\mathcal{R} = (\mathcal{A}_n)_{n\geq 0}$ an $(\epsilon_0, \epsilon_1)$-distinguisher for $G$, $p$ a prime and $k \geq 2$. Let $\epsilon' > 0$ be a fixed constant and define*

$$\begin{aligned} \zeta_0(n) &= 2\max\{\epsilon_0(n), \epsilon_1(n)\}n^2 + \epsilon', \\ \zeta_1(n) &= 2\epsilon_1(n)n^2 + \epsilon_1(n). \end{aligned}$$

*Then there is a $(\zeta_0, \zeta_1)$-distinguisher for $\mathbb{Z}_{p^k} \wr G$ with space complexity $s(\mathcal{R}, n) + \Theta(\log^2 s(\mathcal{R}, n) + \log n)$.*

*Proof.* The proof follows again the strategy from the proof of Theorem 10.9. Let $\mathcal{A}_n = (Q_n, \Sigma, \iota_n, \rho_n)$ and $s(n) = s(\mathcal{R}, n)$. Note that $|Q_n| \leq 2^{s(n)}$. We assume that $Q_n = [0, |Q_n| - 1]$. For $d \geq 1$ let $\Phi_d$ be the set of all monic polynomials $\phi(x) \in \mathbb{Z}_{p^k}[x]$ of degree $d$. Below, we will apply Theorem 10.11 with $n$ replaced by $s(n)$ and $m$ replaced by $2n$.

Initially, the streaming algorithm for $\mathbb{Z}_{p^k} \wr G$ guesses the following data independently from each other:

(i) a state $q \in Q_n$ according to the initial state distribution $\iota_n$,
(ii) a tuple $\bar{\nu} = (\nu_1, \ldots, \nu_{c_1})$ of independently chosen functions $\nu_i : [1, s(n)] \to [1, s(n)^7]$ for some constant $c_1$ (that we fix later), where each $\nu_i$ is specified by a uniformly distributed bit string of length $\Theta(\log n + \log s(n))$ according to Theorem 10.11, and
(iii) a $t$-tuple $\bar{\phi} = (\phi_1, \ldots, \phi_t)$ of monic polynomials $\phi_i \in \Phi_d$ (uniformly distributed), where

$$d = \left\lceil \log_2 \left(12 s(n)^8\right) \right\rceil = \Theta(\log s(n)) \text{ and } t = c_2 d = \Theta(\log s(n)) \tag{12}$$

for some constant $c_2$.

In total, $s(n) + \Theta(\log s(n) + \log n) + \Theta(d^2) = s(n) + \Theta(\log^2 s(n) + \log n)$ bits are needed to store these data. Note that each coefficient in a polynomial $\phi_i$ is from $\mathbb{Z}_{p^k}$ and fits into constant space.

For the tuple $\bar{\nu} = (\nu_1, \ldots, \nu_{c_1})$, Theorem 10.11 implies the following for every $i \in [1, c_1]$: Whenever $\mathcal{P} \subseteq 2^{[1,s(n)]}$ contains at most $2n$ sets then with probability at least $1/4$, $\mathcal{P}$ has a unique minimum weight set with respect to $\nu_i$. Since the $\nu_i$ are chosen independently, we can choose the constant $c_1$ such that

$$\mathop{\mathsf{Prob}}_{\bar{\nu}}\left[\bigvee_{i=1}^{c_1} \mathcal{P} \text{ has a unique minimum weight set with respect to } \nu_i\right] \geq 1 - \epsilon'/2,$$

where $\epsilon'$ is the constant from Theorem 10.12. We want to apply the random mappings $\nu_i$ to states of $\mathcal{A}_n$. For this, we fix an arbitrary injective mapping $\kappa : [0, |Q_n| - 1] \to 2^{[1,s(n)]}$. If $P(x) \in \mathbb{Z}_{p^k}[x]$ is a polynomial of degree at most $|Q_n| - 1$ then we can define the polynomial $\nu_i(P)$ of degree at most $s(n)^8$ by replacing every monomial $x^k$ in $P$ by $x^{\nu_i(\kappa(k))}$ (recall that $\nu_i(\kappa(k))$ is the total weight of the set $\kappa(k) \subseteq [1, s(n)]$ under $\nu_i$ and every individual weight is bounded by $s(n)^7$). If $P(x)$ has at most $2n$ monomials and $P(x) \neq 0$, then with probability at least $1/4$, $P(x)$ has a unique monomial $x^k$ such that $\kappa(k) \subseteq [1, s(n)]$ has minimum weight (with respect to $\nu_i$) among all sets $\kappa(k')$ with $x^{k'}$ a monomial in $P(x)$. In particular, $\nu_i(P(x)) \neq 0$ with probability at least $1/4$. Hence, we have

$$\mathop{\mathsf{Prob}}_{\bar{\nu}}\left[\bigvee_{i=1}^{c_1} \nu_i(P(x)) \neq 0\right] \geq 1 - \epsilon'/2.$$

Recall the definition of the polynomial $P_{q,w}(x)$ from (10). We consider $P_{q,w}$ as a polynomial from $\mathbb{Z}_{p^k}[x]$. Its degree is at most $|Q_n| - 1$ and it contains at most $n$ monomials. The exponents of $x$ in $P_{q,w}(x)$ are numbers in the range $[0, |Q_n| - 1]$.

The ring $\mathbb{Z}_{p^k}$ is a finite local commutative ring.[9] We use a randomized polynomial identity test for finite local commutative rings from [3]. By [3, Lemma 6.5], if $P(x) \in \mathbb{Z}_{p^k}[x]$ is a non-zero polynomial of degree at most $s(n)^8$ then

$$\mathop{\mathsf{Prob}}_{\phi \in \Phi_d}\big[\phi(x) \text{ does not divide } P(x) \text{ in } \mathbb{Z}_{p^k}[x]\big] \geq \frac{1}{4d}, \tag{13}$$

where $d$ is from (12). In [3, Lemma 6.5] this is stated for an arbitrary finite local commutative ring $R$, but the right-hand side of the inequality is $(1-\epsilon)/4d$ for a constant $\epsilon$. Let us explain, where this $\epsilon$ comes from: For an arbitrary finite local commutative ring $R$ one can find a copy of a field $\mathbb{F}_p$ inside a certain quotient of $R$ [3, Claim 5.6]. In [3], the authors show how to sample an element from this copy of $\mathbb{F}_p$ such that the probability to sample a certain element from $\mathbb{F}_p$ is between $(1-\epsilon/2)/p$ and $(1+\epsilon/2)/p$ for a constant $\epsilon$; see [3, proof of Lemma 5.4]. For the case $R = \mathbb{Z}_{p^k}$ we have the canonical epimorphism $h : R \to \mathbb{F}_p$ with $h(b) = b \bmod p$ for $b \in R$. Moreover, for every $a \in \mathbb{F}_p$ we have $\mathsf{Prob}_{b \in R}[h(b) = a] = 1/p$ if $b$ is chosen uniformly from $R$. Hence, we get $\epsilon = 0$.

For the input word $w \in \Gamma^{\leq n}$ and the initial random guesses from (i), (ii) and (iii), our streaming algorithm computes and stores the data from (ii), (iii), as well as

- the state $\rho_n(q, \pi_\Sigma(w))$ of the semiPFA $\mathcal{A}_n$ (the algorithm does not have to store the initial state $q$ from (i)) and
- the polynomials $p_{i,j}(x) = \nu_j(P_{q,w}(x)) \bmod \phi_i(x)$ of degree $d-1$ for all $1 \leq i \leq t$ and $1 \leq j \leq c_1$.

This yields the space bound $s(n) + \Theta(\log^2 s(n) + \log n)$ of the algorithm.

The polynomials $p_{i,j}(x)$ can be easily computed in a streaming fashion: Assume that at some time instant the algorithm reads the letter $a^\gamma$, where $\gamma \in \{-1, 1\}$. If $p_{i,j}(x)$ is the current polynomial, and $q$ is the current $\mathcal{A}_n$-state, which is a number in the range $[0, |Q_n| - 1]$, then the algorithm updates $p_{i,j}(x)$ as follows:

$$p_{i,j}(x) := (p_{i,j}(x) + \gamma \cdot x^{\nu_j(\kappa(q))}) \bmod \phi_i(x)$$

Recall that all computations are done in $\mathbb{Z}_{p^k}[x]$ and that the polynomials $\phi_i(x)$ are monic so that polynomial division by $\phi_i(x)$ can be done.

Let us now compute the error probabilities. In the following, we write

$$\mathop{\mathsf{Prob}}_{q,\bar{\nu},\bar{\phi}}[\mathcal{E}]$$

for the probability of the event $\mathcal{E}$ when $q \in Q_n$ is chosen according to the initial state distribution $\iota_n$ of $\mathcal{A}_n$, $\bar{\nu} = (\nu_1, \ldots, \nu_{c_1})$ is the mapping determined by the uniformly chosen random bit string of length $\Theta(\log s(n) + \log n)$ (see point (ii) above), and $\bar{\phi} = (\phi_1, \ldots, \phi_t)$ is the uniformly chosen $t$-tuple of polynomials from $\Phi_d$ (see point (iii) above). We omit $q$, (resp., $\bar{\nu}$, $\bar{\phi}$) if the event $\mathcal{E}$ does not depend on $q$, (resp., $\bar{\nu}$, $\bar{\phi}$).

[9] A commutative ring $R$ is local if it has a unique maximal ideal. The unique maximal ideal of the ring $\mathbb{Z}_{p^k}$ is $p\mathbb{Z}_{p^k}$.

Consider two words $u, v \in \Gamma^{\leq n}$ and let $(f_u, g_u) \in \mathbb{Z}_{p^k} \wr G$ (resp., $(f_v, g_v) \in \mathbb{Z}_{p^k} \wr G$) be the group element represented by the word $u$ (resp., $v$). If $f_u = f_v$ and $g_u = g_v$, then we obtain

$$\operatorname*{\mathsf{Prob}}_{q,\bar{\nu},\bar{\phi}}\Bigg[\ \rho_n(q,\pi_\Sigma(u)) \neq \rho_n(q,\pi_\Sigma(v)) \ \vee \bigvee_{i=1}^{t}\bigvee_{j=1}^{c_1} \nu_j(P_{q,u}(x)) \not\equiv \nu_j(P_{q,v}(x)) \bmod \phi_i(x)\Bigg] \leq \zeta_1(n)$$

in the same way as in the proof of Theorem 10.9 (proof of Claim 3).

Now assume that $(f_u, g_u) \neq (f_v, g_v)$. If $g_u \neq g_v$, i.e., $\pi_\Sigma(u) \not\equiv_G \pi_\Sigma(v)$, then we get

$$\operatorname*{\mathsf{Prob}}_{q}[\rho_n(q,\pi_\Sigma(u)) \neq \rho_n(q,\pi_\Sigma(v))] \geq 1 - \epsilon_0(n) \geq 1 - \zeta_0(n)$$

and we are done. Let us therefore assume that the mappings $f_u$ and $f_v$ differ. With the same argument as in the proof of Theorem 10.9 (Claim 2) we obtain

$$\operatorname*{\mathsf{Prob}}_{q}[P_{q,u}(x) \neq P_{q,v}(x)] \geq 1 - 2\max\{\epsilon_0(n), \epsilon_1(n)\}n^2.$$

Consider a fixed state $q \in Q_n$ with $P_{q,u}(x) \neq P_{q,v}(x)$. Then $P_{q,u}(x) - P_{q,v}(x)$ is a non-zero polynomial of degree at most $|Q_n| - 1$, which contains at most $2n$ monomials. Therefore, for our tuple $\bar{\nu} = (\nu_1, \ldots, \nu_{c_1})$ of random mappings $\nu_j : [1, s(n)] \to [1, s(n)^7]$ we have

$$\operatorname*{\mathsf{Prob}}_{q,\bar{\nu}}\Bigg[\bigvee_{j=1}^{c_1} \nu_j(P_{q,u}(x)) \neq \nu_j(P_{q,v}(x)) \mid P_{q,u}(x) \neq P_{q,v}(x)\Bigg] \geq 1 - \epsilon'/2$$

(note that $\nu_j(P_{q,u}(x)) - \nu_j(P_{q,v}(x)) = \nu_j(P_{q,u}(x) - P_{q,v}(x))$). Moreover, under the assumption that $\nu_j(P_{q,u}(x)) - \nu_j(P_{q,v}(x)) \neq 0$ for some $j \in [1, c_1]$, the above cited result from [3] (see (13)) gives

$$\operatorname*{\mathsf{Prob}}_{\phi \in \Phi_d}\left[\nu_j(P_{q,u}(x)) \not\equiv \nu_j(P_{q,v}(x)) \bmod \phi(x)\right] \geq \frac{1}{4d}.$$

Recall that $d = \Theta(\log s(n))$ diverges when $n \to \infty$. By choosing the constant $c_2$ in $t = c_2 d$ (see (12)) large enough depending on the constant $\epsilon' > 0$ from the theorem, we obtain

$$\operatorname*{\mathsf{Prob}}_{\bar{\phi}}\Bigg[\bigvee_{i=1}^{t} \nu_j(P_{q,u}(x)) \not\equiv \nu_j(P_{q,v}(x)) \bmod \phi_i(x)\Bigg] \geq 1 - \left(1 - \frac{1}{4d}\right)^t \geq 1 - \epsilon'/2$$

if $n$ (and hence $d$) is large enough. We obtain

$$\begin{aligned} & \operatorname*{\mathsf{Prob}}_{q,\bar{\nu},\bar{\phi}}\Bigg[\bigvee_{i=1}^{t}\bigvee_{j=1}^{c_1} \nu_j(P_{q,u}(x)) \not\equiv \nu_j(P_{q,v}(x)) \bmod \phi_i(x)\Bigg] \\ \geq\ & (1 - 2\max\{\epsilon_0(n), \epsilon_1(n)\}n^2) \cdot (1 - \epsilon'/2) \cdot (1 - \epsilon'/2) \\ \geq\ & 1 - (2\max\{\epsilon_0(n), \epsilon_1(n)\}n^2 + \epsilon') = 1 - \zeta_0(n). \end{aligned}$$

if $n$ is large enough. This proves the theorem. □

Note that in Theorems 10.9, 10.10 and 10.12, $\epsilon_1 = 0$ implies $\zeta_1 = 0$.

**Corollary 10.13.** *Let $G$ be a finitely generated group for which there exists an $(\epsilon_0, \epsilon_1)$-distinguisher $\mathcal{R}$. Let $A$ be a finitely generated abelian group and $\epsilon' > 0$ a fixed constant. Then there is a constant $\alpha$ and a $(\zeta_0, \zeta_1)$-distinguisher for $A \wr G$ with*

$$\begin{aligned} \zeta_0(n) &= 2\max\{\epsilon_0(n), \epsilon_1(n)\}n^2 + \epsilon', \\ \zeta_1(n) &= \alpha(2n^2+1)\epsilon_1(n), \\ s(\mathcal{S}, n) &\leq \mathcal{O}(s(\mathcal{R}, n) + \log n). \end{aligned} \tag{14}$$

*If $A$ is free abelian, i.e., $A = \mathbb{Z}^k$ for some $k$, then we can replace $\epsilon'$ in (14) by $1/n^d$ for any fixed constant $d$.*

*Proof.* The wreath product $(H_1 \times H_2) \wr G$ is a subgroup of $(H_1 \wr G) \times (H_2 \wr G)$ [40, Lemma 6.2]. Since $A$ is a direct product of copies of $\mathbb{Z}$ and finite cyclic groups $\mathbb{Z}_{p^k}$ for $p$ a prime and $k \geq 1$, the statement of the theorem follows from Lemma 10.4 and Theorems 10.9, 10.10 and 10.12. The constant $\alpha$ is the number of factors $\mathbb{Z}$ and $\mathbb{Z}_{p^k}$ in the direct product decomposition of $A$. □

We can apply Corollary 10.13 to free solvable groups (the free objects in the variety of solvable groups).

**Corollary 10.14.** *Every free solvable group has 0-sided randomized streaming space complexity $\Theta(\log n)$.*

*Proof.* Magnus' embedding theorem [49] says that every free solvable group can be embedded into an iterated wreath product $\mathbb{Z}^m \wr (\mathbb{Z}^m \wr (\mathbb{Z}^m \wr \cdots))$. Since $\mathbb{Z}^m$ is linear, we can, using Theorem 9.2, obtain an $(\epsilon_0(n), 0)$-distinguisher for $\mathbb{Z}^m$ with space complexity $\mathcal{O}(\log n)$ for every inverse polynomial $\epsilon_0(n)$. We then apply Corollary 10.13 a constant number of times and obtain a 0-sided randomized streaming algorithm with space complexity $\mathcal{O}(\log n)$. Here, we need the second statement in Corollary 10.13, where $A$ is free abelian (with the first statement the additive constant $\epsilon'$ would result in an additive term $\Theta(n^2)$ in the error probability after two applications). The lower bound follows from Theorem 6.2 and the Milnor-Wolf theorem (see Remark 6.4). □

In [68] it is shown that the word problem of a free solvable group can be solved with a randomized algorithm running in time $\mathcal{O}(n \cdot \log^k n)$ for some constant $k$. Our algorithm achieves the same running time (because for every new input symbol, only numbers of bit length $\mathcal{O}(\log n)$ have to be manipulated). In contrast to our algorithm, the algorithm from [68] is non-streaming and does not work in logarithmic space.

## 11. Lower bounds

In this section, we will construct groups with a large randomized streaming space complexity. We will make use of the disjointness problem from communication complexity. The *disjointness problem* is defined as follows: Alice (resp., Bob) has a bit string $u \in \{0,1\}^n$ (resp., $v \in \{0,1\}^n$) and their goal is to determine whether there is no position $1 \leq i \leq n$ such that $u[i] = v[i] = 1$. It is well known that the randomized communication complexity for the disjointness problem is $\Theta(n)$, see e.g. [41, Section 4.6].

By the following result, the restriction to an abelian group $A$ in Corollary 10.13 cannot be relaxed.

**Theorem 11.1.** *Let $H$ be a f.g. non-abelian group and $G$ be a f.g. infinite group. The randomized streaming space complexity of $H \wr G$ is $\Theta(n)$.*

*Proof.* Let $\mathcal{R} = (\mathcal{A}_n)_{n\geq 0}$ be a randomized streaming algorithm for the word problem of $H \wr G$. We show that we obtain a randomized communication protocol for the disjointness problem with communication cost $3 \cdot s(\mathcal{R}, 12n-8)$.

Fix $n \geq 1$ and two elements $g, h \in H$ with $[g,h] \neq 1$. We can w.l.o.g. assume that $g$ and $h$ are generators of $H$. We also fix a finite generating set for $G$. Let $s := t_1 t_2 \cdots t_{n-1}$ be a word over the generators of $G$ such that $t_1 t_2 \cdots t_i$ and $t_1 t_2 \cdots t_j$ represent different elements of $G$ whenever $i, j \in [0, n-1]$ with $i \neq j$. Such a word exists since the Cayley graph of $G$ is an infinite locally finite graph and hence contains an infinite ray. For a word $w = a_0 a_1 \cdots a_{n-1} \in \{0,1\}^n$ and an element $x \in \{g, h, g^{-1}, h^{-1}\}$ define the word

$$w[x] = x^{a_0} t_1 x^{a_1} t_2 \cdots x^{a_{n-2}} t_{n-1} x^{a_{n-1}} s^{-1}.$$

It represents the element $(f_{w,x}, 1) \in H \wr G$ with

$$\mathsf{supp}(f_{w,x}) = \{t_1 \cdots t_i \colon i \in [0, n-1], w[i] = 1\}$$

and $f_{w,x}(t) = x$ for all $t \in \mathsf{supp}(f_{w,x})$. Therefore, for two words $u, v \in \{0,1\}^n$ we have $u[g]v[h]u[g^{-1}]v[h^{-1}] = 1$ in $H \wr G$ if and only if there is no position $i \in [0, n-1]$ with $u[i] = v[i] = 1$. Note that the length of the word $u[g]v[h]u[g^{-1}]v[h^{-1}]$ is $4(3n-2) = 12n-8$.

Our randomized communication protocol for the disjointness problem works as follows, where $u \in \{0,1\}^n$ is the input for Alice and $v \in \{0,1\}^n$ is the input for Bob.

- Alice reads the word $u[g]$ into $\mathcal{A}_{12n-8}$ and sends the resulting state to Bob.
- Bob continues the run in the state he received from Alice, reads the word $v[h]$ into the automaton and sends the resulting state back to Alice.
- Alice continues the run with the word $u[g^{-1}]$ and sends the resulting state to Bob.
- Bob continues the run with $v[h^{-1}]$ and finally accepts if the resulting state is an accepting state of $\mathcal{A}_{12n-8}$.

Both Alice and Bob use their private random choices in order to make the random decisions in the PFA $\mathcal{A}_{12n-8}$. Clearly, the protocol is correct and its communication cost is $3 \cdot s(\mathcal{R}, 12n-8)$. Hence, we must have $3 \cdot s(\mathcal{R}, 12n-8) \geq \Omega(n)$ which implies $s(\mathcal{R}, m) \geq \Omega(m)$. □

**Corollary 11.2.** *Let $H$ be a finitely generated group and assume that $G$ is an infinite group such that $H \wr G$ embeds into $H$. Then the randomized streaming space complexity of $H$ is in $\Omega(n)$.*

*Proof.* Since $H \wr G$ is non-abelian, also $H$ must be non-abelian. Hence, the corollary follows directly from Theorem 11.1. □

In 1965 Richard Thompson introduced three finitely presented groups $F < T < V$ acting on the unit-interval, the unit-circle and the Cantor set, respectively. Of these three groups, $F$ received most attention (the reader should not confuse $F$ with a free group). This is mainly due to the still open conjecture that $F$ is not amenable, which would imply that $F$ is another counterexample to a famous conjecture of von Neumann (a counterexample was found by Ol'shanskii). The group $F$ consists of

all homeomorphisms of the unit interval that are piecewise affine, with slopes a power of 2 and dyadic breakpoints. It is a finitely presented group:

$$F \quad = \quad \langle a, b \mid [ab^{-1}, a^{-1}ba], [ab^{-1}, a^{-2}ba^2] \rangle. \tag{15}$$

The group $F$ is orderable (so in particular torsion-free), its derived subgroup $[F, F]$ is simple and the center of $F$ is trivial (in particular, $F$ is non-abelian); see [12] for more details. Important for us is the fact that $F$ contains a copy of $F \wr \mathbb{Z}$ [30, Lemma 20]. Hence, Corollary 11.2 implies:

**Corollary 11.3.** *The randomized streaming space complexity of Thompson's group $F$ is $\Theta(n)$.*

For the case that $G$ is finite, we can prove the following variant of Corollary 11.2.

**Theorem 11.4.** *Let $H$ be a finitely generated group and assume there is a non-trivial finite group $G$ such that $H \wr G$ embeds into $H$. Then there is a constant $0 < c < 1$ such that the randomized streaming space complexity of $H$ is in $\Omega(n^c)$.*

*Proof.* We can assume that $G = \mathbb{Z}_k$ for some $k \geq 2$. We fix an embedding $\phi : H \wr \mathbb{Z}_k \to H$. Let $\tau$ be the generator of $\mathbb{Z}_k$ and let $\Sigma$ be a finite generating set for $G$. We use a construction from [8] that yields an embedding $\phi_m : H \wr \mathbb{Z}_{k^m} \to H$ for every $m \geq 1$. More precisely, it is shown in [8, Lemma 9.5] that the following mapping $\phi_m$ (where $\tau_m$ is the generator of $\mathbb{Z}_{k^m}$ in $H \wr \mathbb{Z}_{k^m}$) defines an embedding of $H \wr \mathbb{Z}_{k^m}$ into $H$:

$$\begin{aligned} \phi_m(\tau_m) &= \phi^m(\tau)\phi^{m-1}(\tau) \cdots \phi^2(\tau)\phi(\tau), \\ \phi_m(a) &= \phi^m(a) \text{ for } a \in \Sigma. \end{aligned}$$

Let $\lambda$ be the maximal length among the words $\phi(\tau)$ and $\phi(a)$ for $a \in \Sigma$. W.l.o.g. we assume that $\lambda \geq 2$. Then, the maximal length among the words $\phi_m(\tau_m)$ and $\phi_m(a)$ ($a \in \Sigma$) is $\sum_{i=1}^{m} \lambda^i \leq \lambda^{m+1}$.

Let us now fix an $n$. We want to get a randomized communication protocol for the disjointness problem on inputs of length $n$. For this we choose $m = \lceil \log_k n \rceil$ so that $k^m \geq n$. Now observe that the protocol from the proof of Theorem 11.1 also works if the group $G$ (the right factor of the wreath product) is $\mathbb{Z}_\ell$ for some $\ell \geq n$. In particular, we can take the copy of $H \wr \mathbb{Z}_{k^m}$ in $H$. Note that $H$ must be non-abelian since the wreath product $H \wr \mathbb{Z}_k$ is non-abelian.

In our situation, the length of the word $u[g]v[h]u[g^{-1}]v[h^{-1}]$ from the proof of Theorem 11.1 blows up to $\mathcal{O}(n\lambda^m) = \mathcal{O}(n^{1+1/\log_\lambda k})$, since every generator of $H \wr \mathbb{Z}_{k^m}$ becomes a word of length at most $\mathcal{O}(\lambda^m) = \mathcal{O}(n^{1/\log_\lambda k})$ in the group $H$. If $s_H(n)$ is the randomized streaming space complexity of $H$, we obtain

$$3 \cdot s_H(\mathcal{O}(n^{1+1/\log_\lambda k})) \geq \Omega(n),$$

which yields $s_H(n) \geq \Omega(n^c)$, where one can take $c = \log_\lambda(k)/(\log_\lambda(k) + 1)$. □

Theorem 11.4 can be applied to a large class of self-similar groups acting on regularly branching infinite trees. In particular, it is shown in [8, Lemma 9.8] that if $G$ is a weakly branched group whose branching subgroup $K$ contains elements of finite order (see [8] for definitions), then $K$ contains a copy of $K \wr \mathbb{Z}_k$ for some $k \geq 2$. Hence we get:

**Corollary 11.5.** *Let $G$ is a weakly branched group whose branching subgroup $K$ contains elements of finite order. Then there is a constant $0 < c < 1$ such that the randomized streaming space complexity of $G$ is in $\Omega(n^c)$.*

An important example of a group covered by Corollary 11.5 is Grigorchuk's group. It was introduced by Grigorchuk in [26]. It is defined as a f.g. group of automorphisms of the infinite binary tree; the generators are usually denoted $a, b, c, d$ and satisfy the identities $a^2 = b^2 = c^2 = d^2 = 1$ and $bc = cb = d, bd = db = c, dc = cd = b$ (we do not need the precise definition). Grigorchuk's group is a f.g. infinite torsion group and was the first example of a group with intermediate growth as well as the first example of a group that is amenable but not elementary amenable. For Grigorchuk's group we can provide the following lower and upper bound on the constant from Corollary 11.5:

**Theorem 11.6.** *Let $G$ be the Grigorchuk group. Then the following hold:*

- *The deterministic streaming space complexity of $G$ is $\mathcal{O}(n^{0.768})$.*
- *The randomized streaming space complexity of $G$ is $\Omega(n^{1/3})$.*

*Proof.* The first statement follows from Theorem 6.1 and the fact that the growth of the Grigorchuk group is upper bounded by $\exp(n^{0.768})$ [6]. For the second statement we use the subgroup $K \leq G$ generated by $t = (ab)^2$, $v = (bada)^2$, and $w = (abad)^2$. We show that the randomized streaming space complexity of $K$ is $\Omega(n^{1/3})$. This subgroup $K$ has the following properties that can be all found in [17]:

- $K$ is not abelian; for instance $tv \neq vt$.
- $K$ contains a copy of $K \times K$. More precisely, the mapping $\phi$ with

$$\begin{aligned} \phi(t,1) &= v & \phi(1,t) &= w \\ \phi(v,1) &= v^{-1}t^{-1}vt & \phi(1,v) &= w^{-1}twt^{-1} \\ \phi(w,1) &= vtv^{-1}t^{-1} & \phi(1,w) &= wt^{-1}w^{-1}t \end{aligned}$$

defines an injective homomorphism $\phi : K \times K \to K$ [17, p. 262].

The embedding $\phi$ can be used to define for every $k \geq 1$ an embedding $\phi_k : K^{2^k} \to K$ inductively by $\phi_1 = \phi$ and $\phi_{k+1}(\overline{x}, \overline{y}) = \phi(\phi_k(\overline{x}), \phi_k(\overline{y}))$ for all $\overline{x}, \overline{y} \in K^{2^k}$. Note that for a $2^k$-tuple $\overline{x} \in \{1, t, v, w, t^{-1}, v^{-1}, w^{-1}\}^{2^k}$ we have $|\phi_k(\overline{x})| \leq 8^k$ when $\phi_k(\overline{x})$ is viewed as a word over $\{t, v, w, t^{-1}, v^{-1}, w^{-1}\}$.

We can now prove the second statement of the theorem using arguments similar to those from the proof of Theorem 11.1. Let $\mathcal{R} = (\mathcal{A}_n)_{n \geq 0}$ be a randomized streaming algorithm for $\mathsf{WP}(K, \{t, v, w, t^{-1}, v^{-1}, w^{-1}\})$. We show that we obtain a randomized communication protocol for the disjointness problem with communication cost $3 \cdot s(\mathcal{R}, 4n^3)$. Fix $n \geq 1$ and assume that $n = 2^k$ is a power of two. For a word $x = a_0 a_1 \cdots a_{n-1} \in \{0,1\}^n$ and an element $s \in \{t, v, t^{-1}, v^{-1}\}$ define the word

$$x[s] = \phi_k(s^{a_0}, s^{a_1}, \ldots, s^{a_{n-2}}, s^{a_{n-1}}).$$

Then, for two words $x, y \in \{0,1\}^n$ we have $x[t]y[v]x[t^{-1}]y[v^{-1}] = 1$ in $K$ if and only if there is no position $i \in [0, n-1]$ with $x[i] = y[i] = 1$. Note that the length of the word $x[t]y[v]x[t^{-1}]y[v^{-1}]$ is $4 \cdot 8^k = 4n^3$.

Our randomized communication protocol for the disjointness problem works as follows, where $x \in \{0,1\}^n$ is the input for Alice and $y \in \{0,1\}^n$ is the input for Bob.

- Alice reads the word $x[t]$ into $\mathcal{A}_{4n^3}$ and sends the resulting state to Bob.
- Bob continues the run in the state he received from Alice, reads the word $y[v]$ into the automaton and sends the resulting state back to Alice.

- Alice continues the run with the word $x[t^{-1}]$ and sends the resulting state to Bob.
- Bob continues the run with $y[v^{-1}]$ and finally accepts if the resulting state is an accepting state.

This protocol is clearly correct and has communication cost $3 \cdot s(\mathcal{R}, 4n^3)$. We hence must have $3 \cdot s(\mathcal{R}, 4n^3) \geq \Omega(n)$ which implies $s(\mathcal{R}, m) \geq \Omega(m^{1/3})$. □

## 12. Randomized streaming algorithms for subgroup membership problems

Let $G$ be a f.g. group with a finite symmetric generating set $\Sigma$ and let $H$ be a subgroup of $G$. As before, $\pi_G : \Sigma^* \to G$ is the morphism that maps a word $w \in \Sigma^*$ to the group element represented by $w$. We can define the language

$$\mathsf{GWP}(G, H, \Sigma) = \{w \in \Sigma^* : \pi_G(w) \in H\}.$$

GWP stands for generalized word problem which is another common name for the subgroup membership problem. Note that $\mathsf{GWP}(G, 1, \Sigma) = \mathsf{WP}(G, \Sigma)$. In the following we are interested in randomized streaming algorithms for $\mathsf{GWP}(G, H, \Sigma)$. One can easily show a statement for $\mathsf{GWP}(G, H, \Sigma)$ analogously to Lemma 5.1, which allows us to skip the generating set $\Sigma$ in combination with the $\mathcal{O}$-notation and just write $\mathsf{GWP}(G, H)$ in the following. The main result of this section states that for every finitely generated free group $F(\Gamma)$ and every finitely generated subgroup $G \leq F(\Gamma)$ there exists a randomized streaming algorithm for $\mathsf{GWP}(F(\Gamma), G)$ with space complexity $\mathcal{O}(\log n)$. For this we first need a few more definitions concerning finite automata.

We fix the finite alphabet $\Gamma$ in this section. As usual, $\Gamma^{-1} = \{a^{-1} : a \in \Gamma\}$ is a set of formal inverses. Let $\Sigma = \Gamma \cup \Gamma^{-1}$. Recall that we identified the free group $F(\Gamma)$ with the set of all reduced words over the alphabet $\Sigma$. In the following we have to deal with a special class of finite automata over the alphabet $\Sigma$. A partial DFA is defined as an ordinary DFA except that the transition function $\delta : Q \times \Sigma \to Q$ is only partially defined. As for (total) DFAs we extend the partial transition function $\delta : Q \times \Sigma \to Q$ to a partial function $\delta : Q \times \Sigma^* \to Q$. For $q \in Q$ and $w \in \Sigma^*$ we write $\delta(q, w) = \bot$ if $\delta(q, w)$ is undefined, which means that one cannot read the word $w$ into the automaton $\mathcal{A}$ starting from state $q$. A *partial inverse automaton* $\mathcal{A} = (Q, \Sigma, q_0, \delta, q_f)$ over the alphabet $\Sigma = \Gamma \cup \Gamma^{-1}$ is a partial DFA with a single final state $q_f$ and such that for all $p, q \in Q$ and $a \in \Sigma$, $\delta(p, a) = q$ implies $\delta(q, a^{-1}) = p$.

The main technique to deal with finitely generated subgroups of a free group is Stallings folding [35]. We do not need the details of the technique. All we need is that for every finitely generated subgroup $G \leq F(\Gamma)$ there exists a partial inverse automaton $\mathcal{A}_G$ over the alphabet $\Sigma$ such that for every reduced word $w \in \Sigma^*$ we have: $w \in G$ if and only if $w \in L(\mathcal{A}_G)$. We call $\mathcal{A}_G$ the *Stallings automaton* for $G$. It can be constructed quite efficiently from a given set of generators for $G$, but we do not need this fact since $G$ will be fixed and not considered to be part of the input in our main result, Theorem 12.4 below.[10] The Stallings automaton has the additional property that its final state is also the initial state.

[10]This setting is more natural in our context, where we consider streaming algorithms for languages. In Theorem 12.4 below, we will consider the language $\mathsf{GWP}(F(\Gamma), G)$ of all words representing an element from the subgroup $G$.

Let $G$ be a fixed finitely generated subgroup of $F(\Gamma)$ and let $\mathcal{A}_G = (Q, \Sigma, q_0, \delta, q_0)$ be its Stallings automaton in the following. An important property of $\mathcal{A}_G$ is the following: If $q, q' \in Q$ and $u \in \Sigma^*$ ($u$ is not necessarily reduced) are such that $\delta(q, u) = q'$ then also $\delta(q, \mathsf{red}(u)) = q'$. This follows from the fact that $\delta(q, aa^{-1}) = q$ for every $q \in Q$ and $a \in \Sigma$. In particular, if $\delta(q_0, u) \neq \bot$, then $u \in L(\mathcal{A}_G)$ if and only if $\mathsf{red}(u) \in L(\mathcal{A}_G)$ if and only if $\mathsf{red}(u) \in G$.

**Lemma 12.1.** *Let $u, v \in \Sigma^*$ and $a \in \Sigma$ such that $\delta(q_0, u) = q_1 \in Q$, $\delta(q_1, a) = \bot$ and $v$ has no prefix $x$ with $\mathsf{red}(ax) = \varepsilon$. Then $\mathsf{red}(uav) \notin G$.*

*Proof.* Let $u' = \mathsf{red}(u)$. Then we also have $\delta(q_0, u') = q_1$ and $\mathsf{red}(uav) \notin G$ if and only if $\mathsf{red}(u'av) \notin G$. We can therefore assume for the rest the proof that $u$ is reduced. Moreover, observe that $u$ cannot end with the symbol $a^{-1}$: if $u = u'a^{-1}$, then $\bot = \delta(q_1, a) = \delta(\delta(q_0, u'a^{-1}), a) = \delta(q_0, u') \neq \bot$.

Assume now that $v$ has no prefix $x$ with $\mathsf{red}(ax) = \varepsilon$. We claim that $\mathsf{red}(av)$ begins with the symbol $a$. Assume for a moment that this is already shown. Write $\mathsf{red}(av) = ay$ for some word $y$. Then $uav$ reduces to $uay$. The latter word is reduced, since $ua$ is reduced ($u$ is reduced and $u$ does not end with $a^{-1}$) and $ay$ is redued. But $uay \notin L(\mathcal{A})$, because $\delta(q_0, uay) = \bot$. Hence, we have $uay \notin G$ and thus $\mathsf{red}(uav) \notin G$.

It therefore remains to show that $\mathsf{red}(av)$ begins with the symbol $a$. We prove by induction that for every prefix $x$ of $v$, $\mathsf{red}(ax)$ begins with the symbol $a$. For $x = \varepsilon$ this is clear. Now assume that $xb$ is a prefix of $v$ ($b \in \Sigma$) and we have already shown that $\mathsf{red}(ax) = ax'$ for some word $x'$. We obtain $\mathsf{red}(axb) = \mathsf{red}(ax'b)$.

If $x' = \varepsilon$ then $\mathsf{red}(axb) = \mathsf{red}(ab)$. If $b = a^{-1}$ then we obtain $\mathsf{red}(axb) = \varepsilon$. This leads to a contradiction since $xb$ is a prefix of $v$. Hence, we have $b \neq a^{-1}$ and thus $\mathsf{red}(axb) = ab$ starts with $a$.

Let us now assume that $x' \neq \varepsilon$ and write $x' = x''c$ for a symbol $c \in \Sigma$. Since $ax''c = ax'$ is reduced, we obtain

$$\mathsf{red}(axb) = \begin{cases} ax''cb & \text{if } c \neq b^{-1} \\ ax'' & \text{if } c = b^{-1}. \end{cases}$$

In both cases, $\mathsf{red}(axb)$ starts with the symbol $a$. This concludes the proof of the lemma. □

**Definition 12.2.** For a word $w \in \Sigma^*$ we define the $\mathcal{A}_G$-factorization of $w$ uniquely as either

(i) $w = w_0 a_1 u_1 \; w_1 a_2 u_2 \cdots w_{k-1} a_k u_k \; w_k$ or
(ii) $w = w_0 a_1 u_1 \; w_1 a_2 u_2 \cdots w_{k-1} a_k u_k \; w_k a_{k+1} v$

such that $k \geq 0$ and the following properties hold, where we set $\ell = k$ in case (i) and $\ell = k + 1$ in case (ii):

- $w_0, \ldots, w_k, u_1, \ldots, u_k, v \in \Sigma^*$, $a_1, \ldots, a_\ell \in \Sigma$,
- there are states $q_1, \ldots, q_{k+1} \in Q$ such that $\delta(q_i, w_i) = q_{i+1}$ for all $i \in [0, k]$ (recall that $q_0$ is the initial state of $\mathcal{A}_G$),
- $\delta(q_i, a_i) = \bot$ for all $i \in [1, \ell]$,
- for all $i \in [1, k]$, $\mathsf{red}(a_i u_i) = \varepsilon$ but there is no prefix $u \neq u_i$ of $u_i$ with $\mathsf{red}(a_i u) = \varepsilon$, and
- in case (ii), $v$ has no prefix $x$ with $\mathsf{red}(a_{k+1} x) = \varepsilon$.

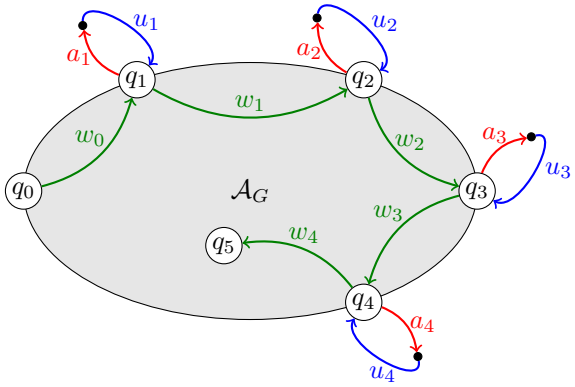


FIGURE 1. An $\mathcal{A}_G$-factorization of type (i) for $k = 4$. The red-blue loops outside of $\mathcal{A}_G$ are loops in the Cayley graph of the free group $F(\Gamma)$.

Depending on which of the two cases (i) and (ii) in Definition 12.2 holds, we say that $w$ has an $\mathcal{A}_G$-factorization of type (i) or type (ii).

Let us explain the intuition of the $\mathcal{A}_G$-factorization of $w$; see also Figures 1 and 2. We start reading the word $w$ into the automaton $\mathcal{A}_G$, beginning at state $q_0$, as long as possible. If it turns out that $\delta(q_0, w)$ is defined, then the $\mathcal{A}_G$-factorization of $w$ consists of the single factor $w_0 = w$ and we obtain type (i) with $k = 0$. Otherwise, there is a shortest prefix $w_0$ of $w$ (the first factor of the $\mathcal{A}_G$-factorization) such that after reading $w_0$ we reach the state $\delta(q_0, w_0) = q_1$ of $\mathcal{A}_G$ and $\delta(q_1, a_1) = \bot$, where $a_1$ is the symbol following $w_0$ in $w$. In other words, when trying to read $a_1$, we escape the automaton $\mathcal{A}_G$ for the first time. At this point let $w = w_0 a_1 x$. We then take the shortest prefix $u_1$ of $x$ such that $a_1 u_1$ evaluates to the identity in the free group $F(\Gamma)$ (if such a prefix does not exist, we terminate in case (ii) with $v = x$). This yields a new factorization $w = w_0 a_1 u_1 y$. We then repeat this process with the word $y$ starting from the state $q_1$ as long as possible. There are two possible terminations of the process: starting from state $q_k$ we can read the whole remaining suffix into $\mathcal{A}_G$ (and arrive in state $q_{k+1}$). This suffix then yields the last factor $w_k$ and we are in case (i). In the other case, we leave the automaton $\mathcal{A}_G$ with the symbol $a_{k+1}$ from state $q_{k+1}$ ($\delta(q_{k+1}, a_{k+1}) = \bot$) and the remaining suffix has no prefix $x$ such that $a_{k+1}x$ evaluates to the identity in the free group $F(\Gamma)$. The remaining suffix then yields the last factor $v$ and we are in case (ii).

**Lemma 12.3.** *Let $w \in \Sigma^*$ and assume that the $\mathcal{A}_G$-factorization of $w$ and the states $q_1, \ldots, q_{k+1}$ are as in Definition 12.2.*

- *If the $\mathcal{A}_G$-factorization of $w$ is of type (i) then $\mathsf{red}(w) \in G$ if and only if $q_{k+1} = q_0$ (the initial and final state of $\mathcal{A}_G$).*
- *If the $\mathcal{A}_G$-factorization of $w$ is of type (ii) then $\mathsf{red}(w) \notin G$.*

*Proof.* Let us first assume that the $\mathcal{A}_G$-factorization of $w$ is of type (i). Then the word $w$ can be reduced to $w_0 w_1 \cdots w_k$. Moreover, we have $\delta(q_0, w_0 w_1 \cdots w_k) = q_{k+1}$. Hence, we have $\mathsf{red}(w) \in G$ if and only if $\mathsf{red}(w_0 w_1 \cdots w_k) \in G$ if and only if $w_0 w_1 \cdots w_k \in L(\mathcal{A}_G)$ if and only if $q_{k+1} = q_0$.

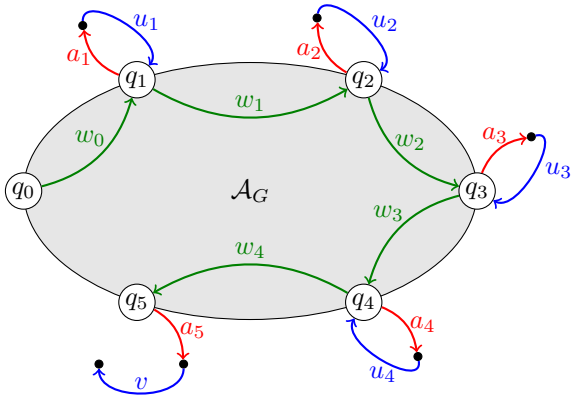


FIGURE 2. An $\mathcal{A}_G$-factorization of type (ii) for $k = 4$. The red-blue loops outside of $\mathcal{A}_G$ are loops in the Cayley graph of the free group $F(\Gamma)$.

Now assume that the $\mathcal{A}_G$-factorization of $w$ is of type (ii). The word $w$ can be reduced to $w_0w_1 \cdots w_{k-1}w_ka_{k+1}v$. Let $w' = w_0w_1 \cdots w_k$. Note that $\delta(q_0, w') = q_{k+1}$, $\delta(q_{k+1}, a_{k+1}) = \perp$ and $v$ has no prefix $x$ with $\mathsf{red}(a_{k+1}x) = \varepsilon$. Hence, Lemma 12.1 yields $\mathsf{red}(w'a_{k+1}v) \notin G$. Since $\mathsf{red}(w'a_{k+1}v) = \mathsf{red}(w)$, we obtain $\mathsf{red}(w) \notin G$. This concludes the proof of the lemma. □

We now come to the main result of this section.

**Theorem 12.4.** *Let $G$ a fixed finitely generated subgroup of $F(\Gamma)$. Then for every $c > 0$ there exists a $1/n^c$-correct randomized streaming algorithm for the language $\mathsf{GWP}(F(\Gamma), G)$ with space complexity $\mathcal{O}(\log n)$.*

*Proof.* Take the Stallings automaton $\mathcal{A}_G = (Q, \Sigma, q_0, \delta, q_0)$; note that $|Q|$ is a constant since $G$ is fixed. We would like to use $\mathcal{A}_G$ as a streaming algorithm for $\mathsf{GWP}(F(\Gamma), G)$. The problem is that we cannot assume that the input word is reduced. We solve this problem by using a $(1/n^{c+2}, 0)$-distinguisher $(\mathcal{B}_n)_{n\geq 0}$ for $F(\Gamma)$ with space complexity $\mathcal{O}(\log n)$. It exists by Theorem 9.2 since finitely generated free groups are linear.

Fix an input length $n$ and let $\mathcal{B}_n = (R_n, \Sigma, \lambda_n, \sigma_n)$. Consider an input word $w \in \Sigma^{\leq n}$. Our randomized streaming algorithm for $\mathsf{GWP}(F(\Gamma), G)$ is shown in Algorithm 3.

The space needed by Algorithm 3 is $\mathcal{O}(\log n)$. The variables $q$ and $\beta$ need constant space and $p$ and $r$ both need $\mathcal{O}(\log n)$ bits. Let us now show that the error probability of Algorithm 3 is bounded by $1/n^c$. For this let $S = \mathcal{P}(w)$ be the set of all prefixes of $w$. For the initially guessed state $r_0 \in R_n$ (line 3) we have

$$\mathop{\mathsf{Prob}}_{r_0 \in R_n}\left[\equiv_{F(\Gamma)} \text{ equals } \equiv_{r_0} \text{ on } S\right] \geq 1 - 1/n^{c+2}\binom{|S|}{2} \geq 1 - 1/n^c$$

by Lemma 8.2. Let us assume for the further consideration that the guessed state $r_0$ is such that $\equiv_{F(\Gamma)}$ and $\equiv_{r_0}$ are equal on $S$. We claim that under this assumption, Algorithm 3 accepts in line 15 after reading $w$ if and only if $\mathsf{red}(w) \in G$. For this,

**Algorithm 3:** $1/n^c$-correct randomized streaming algorithm for $\mathsf{GWP}(F(\Gamma), G)$

**global variables:** $q \in Q$, $p, r \in R_n$, $\beta \in \{0, 1\}$

**initialization:**

1 $q := q_0$ ;
2 $\beta := 1$ ;
3 guess $r \in R_n$ according to the initial state distribution $\lambda_n$ of $\mathcal{B}_n$ ;

**next input letter:** $a \in \Sigma$

4 **if** $\beta = 1$ *and* $\delta(q, a) = \bot$ **then**
5 | $\beta := 0$ ;
6 | $p := r$
7 **end**
8 **if** $\beta = 1$ *and* $\delta(q, a) \neq \bot$ **then**
9 | $q := \delta(q, a)$
10 **end**
11 $r := \sigma_n(r, a)$ ;
12 **if** $\beta = 0$ *and* $r = p$ **then**
13 | $\beta := 1$
14 **end**
15 accept if $\beta = 1$ and $q = q_0$

assume that the $\mathcal{A}_G$-factorization of $w$ and the states $q_1, \ldots, q_{k+1} \in Q$ are as in Definition 12.2. By Lemma 12.3 it suffices to show the following:

(a) If the $\mathcal{A}_G$-factorization of $w$ is of type (i) then after reading $w$ we have $\beta = 1$ and $q = q_{k+1}$ in Algorithm 3.
(b) If the $\mathcal{A}_G$-factorization of $w$ is of type (ii) then after reading $w$ we have $\beta = 0$ in Algorithm 3.

To see this, observe that Algorithm 3 simulates $\mathcal{B}_n$ on $w$ starting from $r_0$ (line 11). Moreover, initially we have $\beta = 1$ (line 2). This implies that Algorithm 3 simulates the Stallings automaton $\mathcal{A}_G$ on $w$ as long as possible (line 9). If this is possible for the whole input $w$ (i.e., $\delta(q_0, w) \neq \bot$) then $w$ has an $\mathcal{A}_G$-factorization of type (i) consisting of the single factor $w$ (i.e., $k = 0$). Moreover, after processing $w$ by Algorithm 3, we have $\beta = 1$ and the program variable $q$ holds $\delta(q_0, w) = q_1 = q_{k+1}$. We obtain the above case (a).

Assume now that $k > 0$. The $\mathcal{A}_G$-factorization of $w$ starts with $w_0 a_1$, where $\delta(q_0, w_0) = q_1$ and $\delta(q_1, a_1) = \bot$. After processing $w_0$ by Algorithm 3 we have $q = q_1$ and $r = \sigma_n(r_0, w_0)$. While processing the next letter $a_1$, Algorithm 3 sets $\beta$ to 0 (line 5) and saves the current state $r = \sigma_n(r_0, w_0)$ of $\mathcal{B}_n$ in the variable $p$ (line 6). Let us write $w = w_0 a_1 v$. Since the flag $\beta$ was set to 0, Algorithm 3 only continues the simulation of $\mathcal{B}_n$ on input $a_1 v$ starting from state $\sigma_n(r_0, w_0) = p$. Our assumption that $\equiv_{F(\Gamma)}$ and $\equiv_{r_0}$ are equal on the set $S$ implies that for every prefix $x$ of $v$ we have: $\mathsf{red}(a_1 x) = \varepsilon$ if and only if $p = \sigma_n(r_0, w_0) = \sigma_n(r_0, w_0 a_1 x)$. In line 12, the algorithm checks the latter equality in each step (as long as $\beta = 0$). If there is no prefix $x$ of $v$ with $\mathsf{red}(a_1 x) = \varepsilon$ then the $\mathcal{A}_G$-factorization of $w$ is of type (ii) (it is $w_0 a_1 v$) and the flag $\beta$ is 0 after reading $w$. We then obtain

the above case (b). Otherwise, $u_1$ is the shortest prefix of $v$ with $\mathsf{red}(a_1 u_1) = \varepsilon$. Moreover, after processing $u_1$, the if-condition in line 12 is true for the first time. The algorithm then sets the flag $\beta$ back to 1 (line 13) and resumes the simulation of the automaton $\mathcal{A}_G$ in state $q_1$ (which is still stored in the program variable $q$). This process now repeats and we see that the algorithm correctly locates the factors of the $\mathcal{A}_G$-factorization of $w$. This shows the above points (a) and (b) and concludes the proof of the theorem. □

It is not possible to generalize Theorem 12.4 to subgroups of $F(\Gamma)$ that are not finitely generated: Let $F_2 = F(\{a, b\})$ be the free group generated by two elements. In the following we make use of Thompson's group $F$. Recall that the randomized streaming space complexity of the word problem of Thompson's group $F$ is $\Theta(n)$; see Corollary 11.3. Moreover, Thompson's group $F$ is finitely presented and generated by two elements $a$ and $b$; see (15).[11] Let $R = \{[ab^{-1}, a^{-1}ba], [ab^{-1}, a^{-2}ba^2]\}$ be the set of relators from (15) and let $N = N(R)$ be the normal closure of $R$ in the free group $F(\{a, b\})$. Thus, we have $F \cong F(\{a, b\})/N$.

**Theorem 12.5.** *For the subgroup $N \leq F_2$ the randomized streaming space complexity of the language $\mathsf{GWP}(F_2, N)$ is $\Theta(n)$.*

*Proof.* A word $w \in \{a, b, a^{-1}, b^{-1}\}^*$ represents in the free group $F_2$ an element of $N$ if and only if in Thompson's group $F$, $w$ represents the group identity. The theorem follows directly from Corollary 11.3. □

We now consider the direct product of $F_2 \times F_2$ of two free groups of rank two. It is also a linear group, hence the randomized streaming space complexity of the word problem for $F_2 \times F_2$ is in $\mathcal{O}(\log n)$. For the subgroup membership problem, this fact no longer holds by the following theorem. We make use of a construction of Mihaĭlova from [53], where she constructed a finitely generated subgroup of $F_2 \times F_2$ with an undecidable subgroup membership problem.

**Theorem 12.6.** *There is a finitely generated subgroup $G$ of $F_2 \times F_2$ such that the randomized streaming space complexity of $\mathsf{GWP}(F_2 \times F_2, G)$ is in $\Theta(n)$.*

*Proof.* Take again Thompson's group $F$ and let $N \leq F(\{a, b\})$ and $R$ be as defined above. We make use of Mihaĭlova's construction from [53]. Let

$$D = \{(r, 1) : r \in R\} \cup \{(a, a), (b, b)\},$$

which is viewed as a finite subset of $F_2 \times F_2$. Mihaĭlova [53] showed that for every element $g \in F_2$:

$$g \in N \iff (g, 1) \in \langle D \rangle \leq F_2 \times F_2.$$

Hence, the theorem follows from Theorem 12.5. □

## 13. Open problems

**Hyperbolic groups.** Hyperbolic groups are one of the most important classes in geometric group theory. The word problem for a hyperbolic group belongs to the complexity class $\mathsf{LogCFL}$, which is contained in $\mathsf{DSPACE}(\log^2 n)$ [44], and it is not known whether for every hyperbolic group the word problem belongs to logspace. What is the space complexity of randomized streaming algorithms for hyperbolic

[11] For our arguments we could take any finitely presented 2-generator group whose randomized streaming space complexity is $\Omega(n)$.

groups? In particular, is the randomized streaming space complexity of the word problem for a hyperbolic group in $\mathcal{O}(\log n)$. It is known that there exist non-linear hyperbolic groups.

**Grigorchuk group.** For the randomized streaming space complexity of the Grigorchuk group we proved the lower bound $\Omega(n^{1/3})$ and the upper bound $\mathcal{O}(n^{0.768})$ (the upper bound even holds for the deterministic streaming space complexity); see Theorem 11.6. This leaves a gap that we would like to close.

**Residually finite groups.** A group $G$ is called *residually finite* if for every element $g \in G \setminus \{1\}$ there is a homomorphism $\phi : G \to H$ from $G$ to a finite group $H$ such that $\phi(g) \neq 1$. All the groups for which we have constructed randomized streaming algorithms for the word problem with space complexity $o(n)$ so far are residually finite. This follows from the following results:

- Every f.g. linear group is residually finite [50].
- Every finite extension of a residually finite groups is residually finite ; this seems to be folklore; see e.g. [14, Proposition 2.2.12].
- A graph product of f.g. residually finite group is f.g. residually finite [32].
- A wreath product $H \wr G$ of f.g. groups $G, H$ is residually finite if and only if (i) either $G$ is finite and $H$ is residually finite or (ii) $H$ is abelian and $G$ is residually finite [29].
- A f.g. group is residually finite if and only if it faithfully acts on a rooted locally finite tree; see e.g.[11] (this includes the Grigorchuk group).

Erschler [21] constructed non-residually finite groups $G$ of intermediate growth. These groups are quasiisometric to the Grigorchuk group and therefore have growth functions that are equivalent to the growth function of the Grigorchuk group. Therefore, there is a non-residually finite group $G$, whose word problem has deterministic streaming space complexity $\mathcal{O}(n^{0.768})$; see Theorem 11.6. Is there also a non-residually finite group whose word problem has randomized streaming space complexity $\mathcal{O}(\log n)$? An interesting concrete non-residually finite group is the Baumslag-Solitar group $\mathsf{BS}(2,3) = \langle a, t \mid t^{-1}a^2t = a^3 \rangle$. The word problem for every Baumslag-Solitar group $\mathsf{BS}(p,q)$ belongs to $\mathsf{DSPACE}(\log n)$ [70].

**Graph of groups.** Is it possible to prove a transfer theorem (in the style of Theorem 10.7 for graph products) for graphs of groups under certain restrictions, e.g., if all edge groups are finite?

## References


[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999. doi:10.1006/jcss.1997.1545.

[2] Anatolij W. Anissimov and Franz D. Seifert. Zur algebraischen Charakteristik der durch kontext-freie Sprachen definierten Gruppen. *Elektron. Informationsverarbeit. Kybernetik*, 11(10–12):695–702, 1975.

[3] Vikraman Arvind, Partha Mukhopadhyay, and Srikanth Srinivasan. New results on noncommutative and commutative polynomial identity testing. *Computational Complexity*, 19(4):521–558, 2010. doi:10.1007/s00037-010-0299-8.

[4] Goulnara Arzhantseva and Pierre-Alain Cherix. Quantifying metric approximations of discrete groups. arXiv:2008.12954, 2020.

[5] Ajesh Babu, Nutan Limaye, Jaikumar Radhakrishnan, and Girish Varma. Streaming algorithms for language recognition problems. *Theoretical Computer Science*, 494:13–23, 2013. doi:10.1016/j.tcs.2012.12.028.

[6] Laurent Bartholdi. The growth of Grigorchuk's torsion group. *International Mathematics Research Notices*, 20:1049–1054, 1998. doi:10.1155/S1073792898000622.

[7] Laurent Bartholdi. Lower bounds on the growth of a group acting on the binary rooted tree. *International Journal of Algebra and Computation*, 11(01):73–88, 2001. doi:10.1142/S0218196701000395.

[8] Laurent Bartholdi, Michael Figelius, Markus Lohrey, and Armin Weiß. Groups with ALOGTIME-hard word problems and PSPACE-complete compressed word problems. *ACM Transactions on Computation Theory*, 14(3–4), 2023. doi:10.1145/3569708.

[9] Gabriel Bathie and Tatiana Starikovskaya. Property testing of regular languages with applications to streaming property testing of visibly pushdown languages. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 119:1–119:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.119.

[10] J. Berstel. *Transductions and context–free languages*. Teubner Studienbücher, Stuttgart, 1979.

[11] J. Button. Groups acting faithfully on trees and properly on products of trees, 2019. arXiv:1910.04614.

[12] John W. Cannon, William J. Floyd, and Walter R. Parry. Introductory notes on Richard Thompson's groups. *L'Enseignement Mathématique*, 42(3):215–256, 1996.

[13] Matteo Cavaleri. *Algorithms and Quantifications in Amenable and Sofic Groups*. PhD thesis, Universita' Degli Studi Di Roma La Sapienza, 2016.

[14] Tullio Ceccherini-Silberstein and Michel Coornaert. *Cellular Automata and Groups, 2nd edition*. Springer, 2023.

[15] Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-optimal unique element isolation with applications to perfect matching and related problems. *SIAM Journal on Computing*, 24(5):1036–1050, 1995. doi:10.1137/S0097539793250330.

[16] Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.

[17] Pierre de la Harpe. *Topics in Geometric Group Theory*. University of Chicago Press, 2000.

[18] Max Dehn. Über unendliche diskontinuierliche Gruppen. *Mathematische Annalen*, 71:116–144, 1911. In German. doi:10.1007/BF01456932.

[19] Volker Diekert and Jonathan Kausch. Logspace computations in graph products. *Journal of Symbolic Computation*, 75:94–109, 2016.

[20] Michael Dillencourt and Michael T. Goodrich. Simplified Chernoff bounds with powers-of-two probabilities. *Information Processing Letters*, 182, 2023. 10.1016/j.ipl.2023.106397.

[21] Anna Erschler. Not residually finite groups of intermediate growth, commensurability and non-geometricity. *Journal of Algebra*, 272(1):154–172, 2004. URL: https://www.sciencedirect.com/science/article/pii/S0021869303006410, doi:https://doi.org/10.1016/j.jalgebra.2002.11.005.

[22] Nathanaël François, Frédéric Magniez, Michel de Rougemont, and Olivier Serre. Streaming property testing of visibly pushdown languages. In *Proceedings of the 24th Annual European Symposium on Algorithms, ESA 2016*, volume 57 of *LIPIcs*, pages 43:1–43:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.43.

[23] Ian Glaister and Jeffrey O. Shallit. Automaticity III: polynomial automaticity and context-free languages. *Computational Complexity*, 7(4):371–387, 1998. doi:10.1007/s000370050016.

[24] William Timothy Gowers and Emanuele Viola. Interleaved group products. *SIAM Journal on Computing*, 48(2):554–580, 2019. doi:10.1137/17M1126783.

[25] Elisabeth R. Green. *Graph Products of Groups*. PhD thesis, The University of Leeds, 1990.

[26] Rostislav I. Grigorchuk. Burnside's problem on periodic groups. *Functional Analysis and Its Applications*, 14:41–43, 1980. doi:10.1007/BF01078416.

[27] Rostislav I. Grigorchuk. On the gap conjecture concerning group growth. *Bulletin of Mathematical Sciences*, 4(1):113–128, 2014. doi:10.1007/s13373-012-0029-4.

[28] Mikhail Gromov. Groups of polynomial growth and expanding maps. *Publications Mathématiques de L'Institut des Hautes Scientifiques*, 53:53–78, 1981. doi:10.1007/BF02698687.

[29] K. W. Gruenberg. Residual properties of infinite soluble groups. *Proceedings of the London Mathematical Society*, s3-7(1):29–62, 1957. URL: https://londmathsoc.onlinelibrary.

wiley.com/doi/abs/10.1112/plms/s3-7.1.29, doi:https://doi.org/10.1112/plms/s3-7.1.29.

[30] Victor S. Guba and Mark V. Sapir. On subgroups of the R. Thompson group $F$ and other diagram groups. *Matematicheskii Sbornik*, 190(8):3–60, 1999. doi:10.1070/SM1999v190n08ABEH000419.

[31] Derek F. Holt, Sarah Rees, and Claas E. Röver. *Groups, Languages and Automata*, volume 88 of *London Mathematical Society Student Texts*. Cambridge University Press, 2017. doi:10.1017/9781316588246.

[32] Tim Hsu and Daniel T. Wise. On linear and residual properties of graph products. *Michigan Mathematical Journal*, 46(2):251–259, 1999. doi:10.1307/mmj/1030132408.

[33] Stephen P. Humphries. On representations of Artin groups and the Tits conjecture. *Journal of Algebra*, 169(3):847–862, 1994. doi:10.1006/jabr.1994.1312.

[34] Janis Kaneps and Rusins Freivalds. Running time to recognize nonregular languages by 2-way probabilistic automata. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming, ICALP91*, volume 510 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 1991. doi:10.1007/3-540-54233-7_133.

[35] Ilya Kapovich and Alexei Myasnikov. Stallings foldings and subgroups of free groups. *Journal of Algebra*, 248(2):608–668, 2002. URL: https://www.sciencedirect.com/science/article/pii/S0021869301990337, doi:https://doi.org/10.1006/jabr.2001.9033.

[36] Mikhail I. Kargapolov and Yurii I. Merzljakov. *Fundamentals of the Theory of Groups*, volume 62 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1979.

[37] Richard M. Karp. Some bounds on the storage requirements of sequential machines and turing machines. *Journal of the ACM*, 14(3):478–489, 1967. doi:10.1145/321406.321410.

[38] Jonathan Kausch. *The parallel complexity of certain algorithmic problems in group theory*. PhD thesis, University of Stuttgart, 2017. URL: http://dx.doi.org/10.18419/opus-9152.

[39] Daniel König and Markus Lohrey. Evaluation of circuits over nilpotent and polycyclic groups. *Algorithmica*, 80(5):1459–1492, 2018. doi:10.1007/s00453-017-0343-z.

[40] Daniel König and Markus Lohrey. Parallel identity testing for skew circuits with big powers and applications. *International Journal of Algebra and Computation*, 28(6):979–1004, 2018. doi:10.1142/S0218196718500431.

[41] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997. doi:10.1017/CBO9780511574948.

[42] Jörg Lehnert and Pascal Schweitzer. The co-word problem for the Higman-Thompson group is context-free. *Bulletin of the London Mathematical Society*, 39(2):235–241, 02 2007. doi:10.1112/blms/bdl043.

[43] Richard J. Lipton and Yechezkel Zalcstein. Word problems solvable in logspace. *Journal of the Association for Computing Machinery*, 24(3):522–526, 1977. doi:10.1145/322017.322031.

[44] Markus Lohrey. Decidability and complexity in automatic monoids. *International Journal of Foundations of Computer Science*, 16(4):707–722, 2005. doi:10.1142/S0129054105003248.

[45] Markus Lohrey. *The Compressed Word Problem for Groups*. SpringerBriefs in Mathematics. Springer, 2014. doi:10.1007/978-1-4939-0748-9.

[46] Markus Lohrey and Lukas Lück. Streaming word problems. In *Proceedings of the 47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022*, volume 241 of *LIPIcs*, pages 72:1–72:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.MFCS.2022.72.

[47] Markus Lohrey, Lukas Lück, and Julio Xochitemol. Streaming in graph products. to appear in *Proceedings of the 49th International Symposium on Mathematical Foundations of Computer Science, MFCS 2024*.

[48] Frédéric Magniez, Claire Mathieu, and Ashwin Nayak. Recognizing well-parenthesized expressions in the streaming model. *SIAM Journal on Computing*, 43(6):1880–1905, 2014. doi:10.1137/130926122.

[49] Wilhelm Magnus. On a theorem of Marshall Hall. *Annals of Mathematics. Second Series*, 40:764–768, 1939. doi:10.2307/1968892.

[50] A. I. Mal'cev. On isomorphic matrix representations of infinite groups. *Rec. Math. [Mat. Sbornik] N.S.*, 8(50).

[51] Avinoam Mann. *How Groups Grow*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2011. doi:10.1017/CBO9781139095129.

[52] Alexei Miasnikov, Svetla Vassileva, and Armin Weiß. The conjugacy problem in free solvable groups and wreath products of abelian groups is in $TC^0$. *Theory of Computing Systems*, 63(4):809–832, 2019.

[53] K. A. Mihaĭlova. The occurrence problem for direct products of groups. *Math. USSR Sbornik*, 70:241–251, 1966. English translation.

[54] Charles F Miller III. Decision problems for groups – survey and reflections. In G. Baumslag and Charles F Miller III, editors, *Algorithms and classification in combinatorial group theory*, pages 1–59. Springer, 1992. doi:10.1007/978-1-4613-9730-4_1.

[55] John Milnor. Growth of finitely generated solvable groups. *Journal of Differential Geometry*, 2(4):447 – 449, 1968. doi:10.4310/jdg/1214428659.

[56] Michael Mitzenmacher and Eli Upfal. *Probability and Computing, 2nd edition*. Cambridge University Press, 2017.

[57] David E. Muller and Paul E. Schupp. Groups, the theory of ends, and context-free languages. *Journal of Computer and System Sciences*, 26:295–310, 1983. doi:10.1016/0022-0000(83)90003-X.

[58] Alexei Myasnikov, Vitaly Roman'kov, Alexander Ushakov, and AnatolyVershik. The word and geodesic problems in free solvable groups. *Transactions of the American Mathematical Society*, 362(9):4655–4682, 2010. doi:10.1090/S0002-9947-10-04959-7.

[59] Azaria Paz. Some aspects of probabilistic automata. *Information and Control*, 9(1):26–60, 1966. doi:10.1016/S0019-9958(66)90092-1.

[60] Azaria Paz. *Introduction to Probabilistic Automata*. Academic Press, 1971.

[61] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963. doi:10.1016/S0019-9958(63)90290-0.

[62] Guy Robin. Estimation de la fonction de Tchebychef $\theta$ sur le k-ième nombre premier et grandes valeurs de la fonction $\omega(n)$ nombre de diviseurs premiers de $n$. *Acta Arithmetica*, 42(4):367–389, 1983. URL: http://eudml.org/doc/205883.

[63] Derek J.S. Robinson. *A Course in the Theory of Groups, 2nd edition*. Springer, 1996. doi:10.1007/978-1-4419-8594-1.

[64] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

[65] Jeffrey Shallit and Yuri Breitbart. Automaticity I: properties of a measure of descriptional complexity. *Journal of Computer and System Sciences*, 53(1):10–25, 1996. doi:10.1006/jcss.1996.0046.

[66] Hans-Ulrich Simon. Word problems for groups and contextfree recognition. In *Proceedings of Fundamentals of Computation Theory, FCT 1979*, pages 417–422. Akademie-Verlag, 1979.

[67] Jacques Tits. Free subgroups in linear groups. *Journal of Algebra*, 20:250–270, 1972. doi:10.1016/0021-8693(72)90058-0.

[68] Alexander Ushakov. Algorithmic theory of free solvable groups: Randomized computations. *Journal of Algebra*, 407:178–200, 2014. doi:https://doi.org/10.1016/j.jalgebra.2014.02.014.

[69] Bertram A. F. Wehrfritz. On finitely generated soluble linear groups. *Mathematische Zeitschrift*, 170:155–167, 1980.

[70] Armin Weiß. A logspace solution to the word and conjugacy problem of generalized Baumslag-Solitar groups. In *Algebra and Computer Science*, volume 677 of *Contemporary Mathematics*. American Mathematical Society, 2016.

[71] Joseph A. Wolf. Growth of finitely generated solvable groups and curvature of Riemannian manifolds. *Journal of Differential Geometry*, 2(4):421 – 446, 1968. doi:10.4310/jdg/1214428658.

[72] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Manipulation, EUROSAM 1979*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. doi:10.1007/3-540-09519-5_73.

*Email address*: lohrey@eti.uni-siegen.de

*Email address*: lukas.lueck@gmx.net

*Email address*: Julio.JXochitemol@uni-siegen.de

(Markus Lohrey, Lukas Lück, Julio César Juárez Xochitemol) Universität Siegen, Germany