

Property and Equivalence Testing on Strings ^{*}

Eldar Fischer[†]

Frédéric Magniez[‡]

Michel de Rougemont[§]

Abstract

We investigate property testing and related questions, where instead of the usual Hamming and edit distances between input strings, we consider the more relaxed edit distance with moves. Using a statistical embedding of words which has similarities with the Parikh mapping, we first construct a tolerant tester for the equality of two words, whose complexity is independent of the string size, and we derive an approximation algorithm for the normalized edit distance with moves.

We then consider the question of testing if a string is a member of a given language. We develop a method to compute, in polynomial time in the representation, a geometric approximate description of a regular language by a finite union of polytopes. As an application, we have a new tester for regular languages given by their nondeterministic finite automaton (or regular expressions), whose complexity does not depend on the automaton, except for a polynomial time preprocessing step.

Furthermore, this method allows us to compare languages and validates the new notion of equivalent testing that we introduce. Using the geometrical embedding we can distinguish between a pair of automata that compute the same language, and a pair of automata whose languages are not ε -equivalent in an appropriate sense. Our equivalence tester is deterministic and has polynomial time complexity, whereas the non-approximated version is PSPACE-complete.

Last, we extend the geometric embedding, and hence the tester algorithms, to infinite regular languages and to context-free grammars as well. For context-free grammars the equivalence test has now exponential time complexity, but in comparison, the non-approximated version is not even recursively decidable.

1 Introduction

We consider the approximation of several classical combinatorial problems on strings in the context of property testing. Inspired by the notion of self-testing [5, 6, 21], Property testing has been initially defined and studied for graph properties [13]. It has been successfully extended for various classes of finite structures. Consider finite words over a finite alphabet Σ for which a distance function has been defined. An ε -tester for a language $L \subseteq \Sigma^*$ is a randomized algorithm which takes a word w of size n as an input, and distinguishes with high probability between the case that $w \in L$ and the case that w is ε -far from L . A language L is *testable* if for every $\varepsilon > 0$ there exists an ε -tester for L whose time complexity depends only on ε , *i.e.* is independent of the size n .

Property testing of regular languages was first considered in [1] for the *Hamming distance* and then extended to languages recognizable by bounded width read-once branching programs [17], where the Hamming distance between two words is the minimal number of character substitutions required to transform one word into the other. Then two words of size n are ε -far if they are at distance greater than εn .

The *edit distance* is a more relaxed natural distance measure for strings than the Hamming distance. The edit distance between two words is the minimal number of insertions, deletions and substitutions of a letter required to transform one word into the other. Computing the edit distance between two words is an important subproblem of many applications like text processing, genomics, web search, etc. Behind the difficulty to get a significant subquadratic time algorithm (the best known algorithm is in $O(n^2/\ln n)$ [15]), several approximation algorithms have been proposed. Nevertheless, to get a linear or sublinear time algorithm, one has to consider more drastic relaxations. One can distinguish two possible approaches. The first one is a weak version of approximation based on property testing, and

^{*}Work supported in part by *ACI Sécurité Informatique: VERA* of the French Ministry of research.

[†]Faculty of Computer Science, Technion – Israel institute of technology, Technion City, Haifa 32000, Israel, eldar@cs.technion.ac.il
Research supported in part by an Israel Science Foundation grant number 55/03, and by a grant from the Matilde Barnett Revocable Trust.

[‡]CNRS–LRI, UMR 8623 Université Paris–Sud, France, magniez@lri.fr

[§]LRI & Université Paris II, France, mdr@lri.fr

the second one considers an extension of the distance: the *edit distance with moves*, where arbitrary substrings can be moved in one step.

Concerning the property testing approach, in [3] a sublinear tester is constructed such that it accepts with high probability pairs of words of size n that are at edit distance $O(n^\alpha)$, for some $0 < \alpha < 1$, and rejects with high probability pairs of words at distance $\Omega(n)$. The running time is in $\tilde{O}(n^{\max(\alpha/2, 2\alpha-1)})$. These testers can be understood as a *weak approximation* of the edit distance, since it leads to efficient approximation algorithms whenever the distance is large [19]. Nonetheless no sublinear time testers exist for the case $\alpha = 1$. Moreover, there is no hope to get a tester whose running time is independent of the input size (even for $0 < \alpha < 1$) since a lower bound of $\Omega(n^{\alpha/2})$ on the query complexity has been proven [3]. Results of the same kind were proven in the sketching model [2], where a sketch (or a fingerprint) is associated to each string, which is a succinct yet rich enough description to approximate the edit distance. Nonetheless the complexity of computing a sketch is usually not sublinear. For instance, a near linear time algorithm is constructed to distinguish between strings at distance $O(k)$ and strings at distance $\Omega(k^2)$, for any $k \geq 1$.

The edit distance with moves has been considered and approximated efficiently in [11, 10]. This is one of many interesting variations of the edit distance that has many applications, for instance in genomics. Whereas computing the edit distance with moves is NP-hard [22], it can be approximated within an $\tilde{O}(\ln n)$ factor under near linear time [11]. If one allows other operations in the distance such as copying subwords, then there is a linear time and constant approximation algorithm [12, 23]. In the context of property testing, the edit distance with moves has been used in [14] for testing regular languages, where the tester is more efficient and simpler than the one of [1], and can be generalized to tree regular languages.

In this paper, we develop a new statistical embedding of words which has similarities with the Parikh mapping [18]. Based on this embedding, we develop a tester (**Theorem 3.8**) for the equality between two words whose complexity is $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$, where $|\Sigma|$ is the alphabet size, which is also *tolerant*, that is it is not only an ε -tester, but it also accepts with high probability words that are ε^2 -close. This notion of tolerance, initially present in self-testing, was firstly not considered in property testing. Recently, coming back to this notion, a relation between tolerant property testing and approximation was pointed out in [19]. Based on this observation and our tolerant tester, we directly get an approximation algorithm for the normalized distance ε between two words (**Corollary 3.9**), whose complexity is $O(\frac{(\ln(|\Sigma|/\varepsilon))|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$. To our knowledge this is the first such approximation algorithm whose complexity is independent of the size n . It is interesting to note that the edit distance without moves, that lies between the Hamming distance (for which there is a trivial tolerant tester) and the edit distance with moves (for which we prove the existence of a tolerant tester), is in itself hard for tolerant testing (recall the lower bound above from [3] with $\alpha = 1$).

Then we extend our embedding to languages. This leads us to an approximate geometrical description of regular languages by finite unions of polytopes, which is robust (**Theorem 4.8**). Discretizing this representation gives us a new tester (**Theorem 4.11**) for regular languages whose query complexity is $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ and time complexity is $2^{|\Sigma|^{O(1/\varepsilon)}}$. Whereas the complexity of previous testers for regular languages depended (exponentially) on the number of states of the corresponding automaton, here the automaton is only used in a preprocessing step to build the tester. The tester construction requires time $m^{|\Sigma|^{O(1/\varepsilon)}}$, where m is the number of states of the automaton.

Last, we go beyond the testing of strings to the question of comparing (in an approximate manner) whole languages. We introduce the notion of an equivalence tester between classes of structures. Intuitively, two finite representations R_1, R_2 of languages L_1, L_2 are ε -equivalent if every but finitely words of L_1 is ε -close to L_2 and conversely. An ε -*equivalence tester* accepts equivalent representations and rejects representations which are not ε -equivalent with high probability. Using again discretization, we construct an ε -equivalence tester (**Theorem 4.12**) for nondeterministic finite automata in deterministic time $m^{|\Sigma|^{O(1/\varepsilon)}}$, where m is the number of states in the larger of the two corresponding automata (the exact decision version of this problem is PSPACE-complete by [24]). We then extend this result to the ε -equivalence testing of Büchi automata (**Theorem 4.14**) (after generalizing our definitions to deal also with languages of infinite words), and an exponential algorithm for the ε -equivalence testing of context-free grammars (**Theorem 4.19**) (for which the exact decision version is not even recursively computable).

2 Preliminaries

We fix a finite alphabet Σ , a positive integer k and $\varepsilon = \frac{1}{k}$. We call Σ^k the *block alphabet* and its elements the *block letters*. Any word w of size n over Σ is also a word over Σ^k where the last $(n - k\lfloor \frac{n}{k} \rfloor)$ letters are deleted. To simplify, we will assume that k divides n . Therefore the words and the languages will be considered over both Σ and Σ^k . For a word w over Σ we denote by $|w|$ its size on Σ and by $|w|_b = \varepsilon|w|$ its size on Σ^k . We also denote by $w[i]$ the i -th letter of w , and by $w[j]_b$ the j -th block letter of w , i.e. the subword $w[(j-1)k+1]w[(j-1)k+2] \dots w[jk]$ of w .

A *subword* of a word w is a sequence of consecutive letters of w . An *elementary operation* on a word w is either an insertion, a deletion or a substitution of a letter, or the move of a subword of w into another position. The *edit distance with moves* $\text{dist}(w, w')$ between w and w' is the minimal number of elementary operations on w to obtain w' .

Define the *block Parikh equivalence* from the Parikh mapping [18] of a word on the block alphabet. Namely, if w and w' are two words of same size, then $w \equiv_k w'$ iff w' is obtained by a permutation of the block letters of w .

Proposition 2.1. *Let w, w' be two words of size n (such that k divides n). If $w \equiv_k w'$ then $\text{dist}(w, w') \leq \varepsilon n$.*

For two real vectors V, V' of dimension d , we denote by $|V - V'|$ the ℓ_1 -distance between V and V' , that is $|V - V'| = \sum_{i=1}^d |V[i] - V'[i]|$, where $V[i]$ (resp. $V'[i]$) denotes the i -th coordinate of V (resp. of V'). If the vectors V, V' denote probability distributions, then the ℓ_1 -distance coincides with twice the *total variation distance*.

Recall now the notion of property testing [13] on strings for any distance, although we will only consider the distance dist between strings in the rest of the paper. We say that two words w, w' , of respective sizes n and m , are ε -close if their distance is at most $\varepsilon \times \max(n, m)$. They are ε -far if they are not ε -close. For simplicity, we will often only consider words of the same size. We say that w is ε -far from a language L , if w is ε -far from any word of L .

Definition 2.2 (Tester). *Let $\varepsilon > 0$ be a real number. An ε -tester for a language L is a randomized algorithm A such that, for any word w as input:*

- (1) *If $w \in L$, then A accepts with probability at least $2/3$;*
- (2) *If w is ε -far from L , then A rejects with probability at least $2/3$.*

If in addition the algorithm is guaranteed to always accept if $w \in L$, then we call it a one-sided error ε -tester.

Definition 2.3 (Tolerant tester [19]). *Let $0 < \varepsilon_1 < \varepsilon_2$ be reals. A tolerant $(\varepsilon_1, \varepsilon_2)$ -tester for a language L is a randomized algorithm A such that, for any word w as input:*

- (1) *If w is ε_1 -close to L , then A accepts with probability at least $2/3$;*
- (2) *If w is ε_2 -far from L , then A rejects with probability at least $2/3$.*

We will also consider approximation algorithms which are related to tolerant testers [19].

Definition 2.4 (Approximation). *Let $\alpha, \beta : \mathbb{R} \rightarrow \mathbb{R}$. An (α, β) -approximation of a real function f is a randomized algorithm that, for any word w as input, outputs a value z such that $\Pr[\alpha(f(w)) \leq z \leq \beta(f(U))] \geq 2/3$.*

A *query* to some word w is asking for the value of $w[i]$, for some i . The *query complexity* is the number of queries made to the word $w \in \Sigma^*$. The *time complexity* is the usual definition, where we assume that the following operations are performed in constant time: arithmetic operations, a uniform random choice of an integer from any finite range, and a query to the input.

In this paper, we introduce the new notion of equivalence testing for finite representations of languages, such as automata or pushdown automata. First we define the notion of ε -equivalence for finite structures.

Definition 2.5. *Let $\varepsilon \geq 0$. Let L_1, L_2 be two languages.*

L_1 is ε -included into L_2 , if every but finitely many words of L_1 are ε -close to L_2 .

L_1 is ε -equal to L_2 , if both L_1 is ε -contained in L_2 and L_2 is ε -contained in L_1 .

Let R_1 and R_2 be two finite representations of the languages L_1 and L_2 respectively. R_1 is ε -equivalent to R_2 , if L_1 is ε -equal to L_2 .

Definition 2.6 (Equivalence tester). *Let $\varepsilon > 0$, and let \mathcal{R} be a family of finite representations of languages. A deterministic (resp. probabilistic) ε -equivalence tester for \mathcal{R} is a deterministic (resp. probabilistic) algorithm A such that, given as input finite representations R_1, R_2 from \mathcal{R} :*

- (1) *If R_1 and R_2 define the same language, then A accepts (resp. with probability at least $2/3$);*
- (2) *If R_1 and R_2 are not ε -equivalent, then A rejects (resp. with probability at least $2/3$).*

3 Embedding of a String by Statistics

We will define several statistics over words and study their robustness [20, 21] and soundness. Robustness means that far words have far statistics, and soundness means that close words have close statistics. Despite the difficulty of computing the edit distance with moves, one can efficiently approximate the statistics of a word. This will directly give us a tolerant tester and then an approximation algorithm for the normalized edit distance with moves.

We will first study the robustness of our first statistics, the block statistics. Then we will extend the robustness to the uniform statistics, which have the advantage of being also sound.

3.1 Statistics, Robustness and Soundness

3.1.1 Block statistics

In this section, w and w' are two words of size n over Σ , such that k divides n . Let $\varepsilon = \frac{1}{k}$. We define the *block statistics* the statistics of the block letters of w , that is the vector $\text{b-stat}(w)$ of dimension $|\Sigma|^k$ whose u -coordinate, for $u \in \Sigma^k$, is $\text{b-stat}(w)[u] \stackrel{\text{def}}{=} \Pr_{j=1, \dots, n/k} [w[j]_b = u]$.

The *block distribution* of w is the uniform distribution on block letters $w[1]_b, \dots, w[\frac{n}{k}]_b$ (with some possible repetitions). Let X be the random vector of size $|\Sigma|^k$ whose coordinates are 0 except the u -coordinate which is 1, for a randomly chosen u according to the block distribution of w . The expectation of X satisfies $\mathbb{E}(X) = \text{b-stat}(w)$.

Last, note that when w and w' have same size, $w \equiv_k w'$ iff $\text{b-stat}(w) = \text{b-stat}(w')$. We relate the distance between two words to the ℓ_1 -distance of their respective block statistics.

Lemma 3.1 (Robustness). $\text{dist}(w, w') \leq (\frac{1}{2}|\text{b-stat}(w) - \text{b-stat}(w')| + \varepsilon) \times n$.

Proof. If $\text{b-stat}(w) = \text{b-stat}(w')$, then the distance $\text{dist}(w, w')$ is at most εn as we only need to move εn block letters. Otherwise, we will construct a word w'' from w such that $\text{b-stat}(w'') = \text{b-stat}(w')$, using at most $\frac{n}{2}|\text{b-stat}(w) - \text{b-stat}(w')|$ substitutions. Applying the triangle inequality and the previous case, we obtain the desired result.

Collect in X_+ the positions i of block letters $w[i]_b$ such that $\text{b-stat}(w)[w[i]_b] > \text{b-stat}(w')[w[i]_b]$, and in X_- the positions j such that $\text{b-stat}(w)[w'[j]_b] < \text{b-stat}(w')[w'[j]_b]$. Note that X_+ and X_- have the same cardinality, which is $\frac{n}{2k}|\text{b-stat}(w) - \text{b-stat}(w')|$. Initially we let $w'' = w$. Until $X_+ \neq \emptyset$ repeat the following: take any $i \in X_+$ and $j \in X_-$; replace in w'' the letters of $w''[i]_b = w[i]_b$ with those of $w'[j]_b$ (using at most k substitutions); remove i from X_+ and j from X_- . The resulting word w'' satisfies the required conditions. \square

3.1.2 Uniform Statistics

In this section, w and w' are again two words of size n over Σ . We want to construct a tolerant tester, which is not only an ε -tester, but also accepts words that are ε' -close, for some constant $0 < \varepsilon' < \varepsilon$. We fix some integer k and let $\varepsilon = \frac{1}{k}$. We consider three different statistics: the original statistics of blocks $\text{b-stat}(w)$, the block uniform statistics $\text{bu-stat}(w)$, and the uniform statistics $\text{u-stat}(w)$.

We define these new statistics like the block statistics, using variants of the block distribution. The *uniform distribution* of w corresponds to a uniform and random choice of a subword of size k of w . This is very much related to the previous work of [8], where the subwords of length k were referred to by the term ‘‘shingles’’.

To define the block uniform distribution of w we first partition w into bigger consecutive blocks of size K , where $K = \lfloor \frac{\varepsilon^3 n}{8 \ln(|\Sigma|) |\Sigma|^{2/\varepsilon}} \rfloor$. To simplify, we assume that k divides $(K - k - 1)$, that n is divisible by K , and that $n = \Omega(\frac{\ln|\Sigma|}{\varepsilon^3} |\Sigma|^{2/\varepsilon})$. We call the new blocks the *big blocks*. Now the *block uniform distribution* is defined by the following two-step procedure: First, in every big block choose uniformly a random $0 \leq t \leq k - 1$, and delete the first t letters and the last $k - 1 - t$ letters; then take uniformly a random block letter in the remaining subword of the original word.

We will prove that u-stat is both robust and sound, which leads to an estimator of the distance for far away instances, whereas b-stat is only robust. For instance, the words $(01)^n$ and $(10)^n$ are $\frac{1}{2n}$ -close, whereas their block statistics are $\Omega(1)$ -far. The proof of the robustness of u-stat will use in an intermediate step the robustness of the block uniform statistics bu-stat . For the soundness of u-stat , the proof is much simpler.

Lemma 3.2 (Soundness). *Let $n = \Omega(\frac{1}{\varepsilon})$. If $\text{dist}(w, w') \leq \varepsilon^2 n$ then $|\text{u-stat}(w) - \text{u-stat}(w')| \leq 6.1\varepsilon$.*

Proof. First, remember that there are at most $n - k + 1$ subwords of size k in w . Assume that $\text{dist}(w, w') = 1$. In case of a simple edit operation (insertion, deletion, substitution) on a letter, $|\text{u-stat}(w) - \text{u-stat}(w')| \leq 2 \times \frac{k}{n-k+1}$. For a move operation, if $w = A \cdot B \cdot C \cdot D$ and $w' = A \cdot C \cdot B \cdot D$ where a subword B has been moved, there are three border areas where we may choose a word of length k in w which does not exist in w' . Conversely, there are similar borders in w' . For each border, there are $k - 1$ possible subwords that intersect it, hence $|\text{u-stat}(w) - \text{u-stat}(w')| \leq 2 \times \frac{3(k-1)}{n-k+1}$.

If $\text{dist}(w, w') \leq \varepsilon^2 n$ and $n = \Omega(\frac{1}{\varepsilon})$, then by the triangle inequality $|\text{u-stat}(w) - \text{u-stat}(w')| \leq \varepsilon^2 n \times \frac{6.1k}{n} = 6.1\varepsilon$, since $k = \frac{1}{\varepsilon}$. \square

We now show that the robustness for $\text{b-stat}(w)$ implies the robustness for $\text{bu-stat}(w)$, which then will imply the robustness for $\text{u-stat}(w)$. For a big block B_i , where $i = 1, \dots, \frac{n}{K}$, we denote by v_{i,t_i} the subword of B_i after deleting the first t_i letters and the last $k - 1 - t_i$ letters of B_i . Let v be the concatenations of the words v_{i,t_i} . Then by the definition of $\text{bu-stat}(w)$ we have $\text{bu-stat}(w) = \frac{K}{n} \sum_{i=1}^{n/K} \mathbb{E}_{t_i=0, \dots, k-1}(\text{b-stat}(v_{i,t_i})) = \mathbb{E}_v(\text{b-stat}(v))$.

Intuitively one would like to use this equation directly for extending the robustness of b-stat to bu-stat . However, this will not work since one would need to use a triangle inequality in the wrong direction. Instead we use a more elaborate proof based on a Chernoff-Hoeffding bound argument.

Lemma 3.3. *There exists a word v obtained from w after deleting $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ letters, so that $|\text{bu-stat}(w) - \text{b-stat}(v)| \leq \frac{\varepsilon}{2}$.*

Proof. Fix a coordinate $u \in \Sigma^k$. For every $i = 1, \dots, \frac{n}{K}$, let X_i be the random variable $X_i \stackrel{\text{def}}{=} \text{b-stat}(v_{i,t_i})[u]$, where t_i is chosen uniformly in $\{0, \dots, k-1\}$. We denote by v the random word obtained from the concatenation of the words v_{i,t_i} . Note that v is obtained from w after deleting $(k-1) \times \frac{n}{K} = O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ letters.

The variables $(X_i)_i$ are independent random variables such that $0 \leq X_i \leq 1$ and $\mathbb{E}_v(\text{b-stat}(v)[u]) = \frac{K}{n} \sum_i \mathbb{E}(X_i) = \text{bu-stat}(w)[u]$. By the Chernoff-Hoeffding bound we then get that, for any $t \geq 0$,

$$\Pr[|\text{bu-stat}(w)[u] - \text{b-stat}(v)[u]| \geq t] \leq 2e^{-2(\frac{n}{K})t^2}.$$

We repeat the same argument for every u -coordinate, and using a union bound, we conclude that:

$$\Pr[|\text{bu-stat}(w) - \text{b-stat}(v)| \geq |\Sigma|^k \times t] \leq |\Sigma|^k \times 2e^{-2(\frac{n}{K})t^2}.$$

If we set $t = \frac{\varepsilon}{2|\Sigma|^k} = \frac{1}{2k|\Sigma|^k}$, and use the definition of K , we conclude that there exists with non zero probability a word v that satisfies the required property about the statistics, completing the proof. \square

Combining the robustness of block statistics, the previous lemma, and the next lemma, which easily relies bu-stat to u-stat , we get our robustness lemma (proofs are in Appendix A.1).

Lemma 3.4. $|\text{bu-stat}(w) - \text{u-stat}(w)| = O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4 n})$.

Lemma 3.5 (Robustness). *Let $n = \Omega(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^5})$. If $\text{dist}(w, w') \geq 5\varepsilon n$ then $|\text{u-stat}_k(w) - \text{u-stat}_k(w')| \geq 6.5\varepsilon$.*

3.2 Approximating the Edit Distance with Moves

In order to construct an efficient tolerant tester of the distance between two words, we need to efficiently approximate $\text{u-stat}(w)$. We will also need to approximate $\text{b-stat}(w)$ for the second part of the paper. For this, we state a more general result that implies the approximability of our statistics. There are several methods which can be used to obtain a Chernoff-Hoeffding type bound on vectors. In our simple case, the use of Chernoff-Hoeffding bound together with a direct union bound is polynomially tight using an argument similar to the one of [4].

Lemma 3.6. *Let f be a function from $\{1, \dots, M\}$ to \mathbb{R}^D , such that $f(x)$ has non-negative coordinates and has unit ℓ_1 -norm, for every x . Let $\{Y_1, \dots, Y_N\}$ be random variables over $\{1, \dots, M\}$ independently distributed according to the same probabilistic distribution d . Then for every $t > 0$,*

$$\Pr[|\mathbb{E}_d(f(Y)) - \frac{1}{N} \sum_{i=1}^N f(Y_i)| \geq D \times t] \leq D \times 2e^{-2Nt^2}.$$

Proof. Let $\mu = \mathbb{E}_d(f(Y))$ and $\hat{\mu}_N = \frac{1}{N} \sum_{i=1}^N f(Y_i)$. For every coordinate $u \in \{1, \dots, D\}$, $\Pr[|\mu[u] - \hat{\mu}_N[u]| \geq t] \leq 2e^{-2Nt^2}$, by the Chernoff-Hoeffding bound for the random variables $X_i = f(Y_i)[u]$ which are between 0 and 1 and whose expectation is $\mu[u]$. We conclude using a union bound. \square

As a corollary we can approximate both block and uniform statistics using a number of samples independent of n . The variables Y_i denote the position of the selected block letters u of w , and X_i denote the corresponding vectors of size $|\Sigma|^k$ whose u -coordinate is one and others are zero. Let stat denote either b-stat or u-stat . Then we define $\widehat{\text{stat}}_N(w) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1, \dots, N} X_i$.

Corollary 3.7. *There exists $N \in O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^3})$ for which $\Pr[|\text{stat}(w) - \widehat{\text{stat}}_N(w)| \geq \varepsilon] \leq \frac{1}{3}$, where stat denotes either b-stat or u-stat .*

Using the Soundness and Robustness Lemmas, we can construct a one-sided error tester for the equality of two words which is also $(\varepsilon^2, 5\varepsilon)$ -tolerant:

Uniform Tester(w, w', ε):

Let $N = \Theta(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^3})$, and $k = \frac{1}{\varepsilon}$

Compute $\widehat{\text{u-stat}}_N(w)$ and $\widehat{\text{u-stat}}_N(w')$ using the same N uniformly random indices in $\{1, \dots, n - k + 1\}$

Accept if $|\widehat{\text{u-stat}}_N(w) - \widehat{\text{u-stat}}_N(w')| \leq 6.25\varepsilon$

Reject otherwise

Theorem 3.8. *For any $\varepsilon > 0$, and two words w, w' of the same size of order $\Omega(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^3})$, the above test:*

- (1) *accepts if $w = w'$ with probability 1;*
- (2) *accepts if w and w' are ε^2 -close with probability at least $2/3$;*
- (3) *rejects if w and w' are 5ε -far with probability at least $2/3$.*

Moreover its query and time complexities are in $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$.

From this $(\varepsilon^2, 5\varepsilon)$ -tolerant tester, one can derive an $(\frac{\varepsilon^2}{25}, 5\sqrt{\varepsilon})$ -approximation algorithm of the distance following the approach of [19].

Corollary 3.9. *There exists a randomized algorithm A such that, given two words w, w' of the same size of order $\Omega(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^3})$, A outputs ε' satisfying $\varepsilon' \in [\frac{\varepsilon^2}{25}, 5\sqrt{\varepsilon}]$ with probability at least $2/3$, where $\varepsilon = \frac{\text{dist}(w, w')}{|w|}$. Moreover the query and time complexities of A are in $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$.*

4 Geometric Embedding of a Language

4.1 General Observations

We want to use the notion of block statistics in order to efficiently characterize a language. We choose this statistics vector because it is the simplest to manipulate.

Using the previous section, we can embed a word w into its block statistics $\text{b-stat}(w) \in \mathbb{R}^{|\Sigma|^{1/\varepsilon}}$. This characterization is approximately one-to-one from Lemma 3.1 if the size of the words is fixed.

This directly implies the following proposition.

Proposition 4.1. *Given unbounded computation resources, every (computable) language L is ε -testable using a number of queries that is independent of the input size n .*

Proof. Given the input size n and setting $k = \lceil 2/\varepsilon \rceil$, we can calculate (using our unbounded resources) every word u of length n in L , and write down $\text{b-stat}_k(u)$. For the input word w we can then $\frac{1}{2}\varepsilon$ -approximate $\text{b-stat}_k(w)$ using Lemma 3.6, and then check whether there is any word u as above for which $|\text{b-stat}_k(u) - \widehat{\text{b-stat}}_k(w)| \leq \frac{1}{2}\varepsilon$. \square

The above proposition is not only time inefficient but it also does not allow the computation to complete in a preprocessing stage. The reason is that the block statistics does not characterize words of different lengths, as $\text{b-stat}(w_0) = \text{b-stat}(w_0^t)$ for every positive integer t , if w_0 is any word whose size is a multiple of k .

This means that the set of block statistics $\text{b-stat}(w)$ of all the elements $w \in L$ is not a good characterization of a general language L . For instance, the word $w_0^{3 \times 2^{s-1}}$ is $(1 - 1/k^{2^{s-1}})$ -far from the language $\{w_0^{2^t} : t \geq 1\}$, for

every positive integer s . Moreover, it is not hard to construct using the appropriate powers a language whose testing algorithm requires arbitrarily intensive computations.

To construct a test that works for all n using only one preprocessing stage, one might consider only block statistics of loops of a language (as provided by an appropriate pumping lemma). This makes sense when any word of a language can be decomposed into loops up to a few remaining letters. Regular languages have this property, and context-free languages also share it when any permutation between block letters is allowed (see Section 4.5.2).

4.2 Regular Languages

We fix a finite alphabet Σ , and an automaton A (possibly non deterministic) on Σ with a set of states Q of size m , that recognizes a regular language L . Let k be a positive integer and $\varepsilon = \frac{1}{k}$. We consider only words whose size is divisible by k , as any word of length n of L , for n large enough, is close to such a word. Define A^k , the k -th power of A , as the automaton on Σ^k with set of states Q such that the transitions of A^k are exactly all sequences of k consecutive transitions of A . Then A and A^k recognize the same language. In the general case, one can modify A^k such that A^k recognizes the language of words of L where the last $(|w| - k \lfloor \frac{|w|}{k} \rfloor)$ letters are deleted.

We will characterize L by the block statistics of its loops on the block alphabet. We remark that the statistics of the A^k -loops basically only depend on L and k , from Proposition A.3 in Appendix A.2.

Definition 4.2. A word v over Σ^k is an A^k -loop if there exist two words u, w over Σ^k and an accepting path of A^k for uvw , such that the state of the automaton after reading u (following the above accepting path) is identical to the state after reading uv .

A finite set of A^k -loops is A^k -compatible if all the loops can occur one after the other (in any order) in one accepting path of A^k .

We define the geometric embedding of L by the union of convex hulls of every compatible set of loops.

Definition 4.3. $\mathcal{H} \stackrel{\text{def}}{=} \bigcup_{\substack{v_1, \dots, v_t: A^k\text{-compatible loops} \\ t \geq 0}} \text{Convex-Hull}(\mathbf{b}\text{-stat}(v_1), \dots, \mathbf{b}\text{-stat}(v_t)).$

This definition is motivated by a standard result on finite automata: one can rearrange any word of a regular language into a sequence of small compatible loops. We formulate this fact in our context.

Proposition 4.4. Let $w \in L$. Then $w \equiv_k v u_1 u_2 \dots u_t$, where $|v|_b, |u_1|_b, \dots, |u_t|_b \leq m$ and $\{u_1, u_2, \dots, u_t\}$ is an A^k -compatible set of A^k -loops (non necessarily distinct).

A consequence together with Caratheodory's theorem is that one can equivalently define \mathcal{H} when the loop sizes and the number of compatible loops are bounded (see Appendix A.2 for the proof). Recall that even if this new characterization explicitly depends on A^k (that is on A and ε), the set \mathcal{H} only depends on L and ε (see Appendix A.2).

Proposition 4.5. $\mathcal{H} = \bigcup_{\substack{v_1, \dots, v_t: A^k\text{-compatible loops} \\ t = \lceil |\Sigma|^{1/\varepsilon} + 1, |v_i|_b \leq m}} \text{Convex-Hull}(\mathbf{b}\text{-stat}(v_1), \dots, \mathbf{b}\text{-stat}(v_t)).$

Another consequence of this proposition is that if a word w belongs to L , then it has to satisfy approximately $\mathbf{b}\text{-stat}(w) \in \mathcal{H}$ (Lemma 4.6). This can be understood as an approximate Parikh classification of regular languages, whereas the original Parikh characterization was for context-free languages [18]. The converse is also approximately true (Theorem 4.8). Missing proofs are in Appendix A.3.

Lemma 4.6. For every $w \in L$ there exists w' , so that

$$0 \leq |w| - |w'| \leq \frac{m}{\varepsilon}, \text{dist}(w, w') \leq \frac{m}{\varepsilon}, \\ |\mathbf{b}\text{-stat}(w) - \mathbf{b}\text{-stat}(w')| \leq \frac{2m}{\varepsilon|w|}, \text{ and } \mathbf{b}\text{-stat}(w') \in \mathcal{H}.$$

Lemma 4.7. For every $X \in \mathcal{H}$ and every n there exists $w \in L$, such that

$$0 \leq |w| - n \leq (|\Sigma|^{1/\varepsilon} + 3) \frac{2m}{\varepsilon}, \text{ and } |X - \mathbf{b}\text{-stat}(w)| \leq (|\Sigma|^{1/\varepsilon} + 2) \frac{3m}{\varepsilon n}.$$

Proof. Let $X \in \mathcal{H}$, that is $X = \sum_{i=1}^l \lambda_i \cdot \mathbf{b}\text{-stat}(u_i)$, where $l = |\Sigma|^k + 1$, $|u_i|_b \leq m$, $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$. Fix any integer n . We choose non negative integers $(r_i)_{i=1,2,\dots,l}$ that respectively approximate $\lambda_i \frac{\varepsilon n}{|u_i|_b}$, that is satisfy $0 \leq |r_i - \lambda_i \frac{\varepsilon n}{|u_i|_b}| \leq 1$, and such that $0 \leq \sum_i r_i |u_i|_b - \varepsilon n \leq m$. It is always possible to satisfy this last condition due to the degree of freedom on the choices of r_i and the upper bound $|u_i|_b \leq m$: We let $j \geq 0$ be the minimum integer so that $\sum_{i=1}^j \lceil \lambda_i \frac{\varepsilon n}{|u_i|_b} \rceil |u_i|_b + \sum_{i=j+1}^l \lfloor \lambda_i \frac{\varepsilon n}{|u_i|_b} \rfloor |u_i|_b \geq 0$, and set $r_i = \lceil \lambda_i \frac{\varepsilon n}{|u_i|_b} \rceil$ for $i \leq j$ and $r_i = \lfloor \lambda_i \frac{\varepsilon n}{|u_i|_b} \rfloor$ for $i > j$.

Define the word $w' = u_1^{r_1} u_2^{r_2} \dots u_l^{r_l}$. Then its block length is close to εn : $0 \leq |w'|_b - \varepsilon n \leq m$. Moreover its block statistics satisfies

$$\begin{aligned} |\mathbf{b}\text{-stat}(w') - X| &= \left| \sum_i \left(r_i \frac{|u_i|_b}{|w'|_b} - \lambda_i \right) \mathbf{b}\text{-stat}(u_i) \right| \leq \sum_i |r_i \frac{|u_i|_b}{|w'|_b} - \lambda_i| \\ &\leq \sum_i |r_i \frac{|u_i|_b}{|w'|_b} - r_i \frac{|u_i|_b}{\varepsilon n}| + \sum_i |r_i \frac{|u_i|_b}{\varepsilon n} - \lambda_i| \leq \sum_i r_i |u_i|_b \times \left| \frac{1}{|w'|_b} - \frac{1}{\varepsilon n} \right| + \sum_i \frac{m}{\varepsilon n} \\ &\leq (m + \varepsilon n) \times \left(\frac{1}{\varepsilon n} - \frac{1}{m + \varepsilon n} \right) + l \frac{m}{\varepsilon n} = \frac{m}{\varepsilon n} + l \frac{m}{\varepsilon n}. \end{aligned}$$

Using A^k -compatibility, we can get a word of L from w' by inserting few block letters. Let $v_0 u_{i_1} v_1 u_{i_2} v_2 \dots u_{i_l} v_l \in L$ be the witness of the A^k -compatibility of the loops u_1, \dots, u_l , such that $|v_j|_b \leq m$ for every j , and where (i_1, \dots, i_l) is a permutation of $(1, \dots, l)$. Then $w = v_0 u_{i_1}^{r_{i_1}} v_1 u_{i_2}^{r_{i_2}} v_2 \dots u_{i_l}^{r_{i_l}} v_l \in L$ by construction. Moreover $0 \leq |w|_b - |w'|_b \leq (l+1)m$, and $|\mathbf{b}\text{-stat}(w') - \mathbf{b}\text{-stat}(w)| \leq \frac{2(l+1)m}{\varepsilon n}$, so we conclude. \square

Theorem 4.8. *Let $w \in \Sigma^n$ and $X \in \mathcal{H}$ be such that $|\mathbf{b}\text{-stat}(w) - X| \leq \delta$. Then*

$$\text{dist}(w, L) \leq \left(\frac{\delta}{2} + \left(1 + O\left(\frac{m|\Sigma|^{1/\varepsilon}}{\varepsilon^2 n} \right) \right) \varepsilon \right) n.$$

4.3 Construction of \mathcal{H}

One of the remaining tasks is to efficiently construct \mathcal{H} for a given automaton A with m states. One could try to enumerate all A^k -loops of size at most m over Σ^k . This is not efficient enough due to the possible large number of loops, $O(|\Sigma|^{km})$. Nevertheless, since we only care about block statistics of compatible loops one can enumerate them using a standard reduction to matrix multiplication over the appropriate algebra. The complexity is then just polynomial in the number of possible corresponding block statistics, $\binom{m+|\Sigma|^k}{|\Sigma|^k} = O(m^{|\Sigma|^k})$, since a block statistics of a word v of size at most m over Σ^k basically corresponds to a partition of m into $|\Sigma|^k$ parts.

We explain the procedure and prove the following lemma in Appendix A.4.

Lemma 4.9. *Given A and ε , a set H of $(|\Sigma|^{1/\varepsilon} + 1)$ -tuples of vectors can be computed in time $m^{|\Sigma|^{O(1/\varepsilon)}}$ such that $|H| \leq m^{|\Sigma|^{O(1/\varepsilon)}}$ and $\mathcal{H} = \bigcup_{S \in H} \text{Convex-Hull}(S)$.*

For a regular language, the set \mathcal{H} is a subset of the unit ball of $\mathbb{R}^{|\Sigma|^k}$ for the ℓ_1 -norm. Let us consider the grid $\mathcal{G}_\varepsilon = \{0, \frac{\varepsilon}{|\Sigma|^k}, \frac{2\varepsilon}{|\Sigma|^k}, \dots, 1\}^{|\Sigma|^k}$ of the cube $[0, 1]^{|\Sigma|^k}$ with step $\frac{\varepsilon}{|\Sigma|^k}$. Let \mathcal{H}_ε be the set of points of \mathcal{G}_ε that are at distance at most $\frac{\varepsilon}{2}$ from \mathcal{H} (for the ℓ_1 -distance). Since $|\mathcal{G}_\varepsilon| = (k|\Sigma|^k + 1)^{|\Sigma|^k} = 2^{|\Sigma|^{O(1/\varepsilon)}}$, then $|\mathcal{H}_\varepsilon| = 2^{|\Sigma|^{O(1/\varepsilon)}}$. Moreover, one can easily construct it from H .

Proposition 4.10. *Given A and ε , the set \mathcal{H}_ε can be computed in time $m^{|\Sigma|^{O(1/\varepsilon)}}$.*

4.4 Applications

Theorem 4.11. *For every real $\varepsilon > 0$ and regular language L over a finite alphabet Σ , there exists an ε -tester for L whose query complexity is in $O\left(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4}\right)$ and whose time complexity is in $2^{|\Sigma|^{O(1/\varepsilon)}}$.*

Moreover, given an automaton with m states that recognizes L , the tester can be constructed in time $m^{|\Sigma|^{O(1/\varepsilon)}}$.

Proof. We fix $\varepsilon > 0$, and automaton A with m states that recognizes L . We construct a 3ε -tester for L whose correctness directly follows from the previous section. Let w be a word given as input. We assume that $|w|/(\frac{m|\Sigma|^{1/\varepsilon}}{\varepsilon^2})$ is large enough, otherwise we just run the automaton on w .

The tester is in two steps: a preprocessing step and the testing step itself. Given A and ε , one can compute \mathcal{H}_ε in time $m^{|\Sigma|^{O(1/\varepsilon)}}$ from Proposition 4.10. Now the testing part consists of computing an estimation $\widehat{\text{b-stat}}_N(w)$ of $\text{b-stat}(w)$ as in Corollary 3.7, where $N = \Theta(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^3})$, using $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ queries to w . If $\widehat{\text{b-stat}}_N(w)$ is at distance at most 2ε from \mathcal{H}_ε , the tester accepts, and otherwise it rejects. \square

Theorem 4.12. *There exists a deterministic algorithm T such that, given two automata A and B over a finite alphabet Σ with at most m states and a real $\varepsilon > 0$, $T(A, B, \varepsilon)$:*

- (1) *accepts if A and B recognize the same language;*
- (2) *rejects if A and B are not 2ε -equivalent.*

Moreover the running time complexity of T is in $m^{|\Sigma|^{O(1/\varepsilon)}}$.

Proof. Fix $\varepsilon > 0$. The algorithm simply computes the respective discrete approximations $\mathcal{H}_{A,\varepsilon}$ and $\mathcal{H}_{B,\varepsilon}$ of \mathcal{H}_A and \mathcal{H}_B corresponding to the automata A and B . If they are equal, the tester accepts, and otherwise it rejects.

The correctness is again omitted and follows from the previous section. \square

One can get a tolerant equivalent tester for automata using uniform statistics following the more complex approach of Appendix C.

4.5 Extensions

4.5.1 Infinite Regular Languages

We now consider an application to infinite words over a finite alphabet Σ . In this section, all words are infinite unless we explicitly state otherwise. We will show how to extend our results to nondeterministic Büchi automata. A *Büchi automaton* is simply a finite automaton A on which the notion of acceptance has been modified as follows. For a word $w \in \Sigma^\omega$ over Σ and a corresponding (infinite) path in A , we denote by $\text{Inf}_A(w)$ the set of states of A which are reached infinitely many times by the path. We say that w is *accepted* by A if there exists a path for w such that $\text{Inf}_A(w)$ contains an accepting state of A . We say that A recognizes the language of accepted infinite words. Such languages are called ω -regular languages.

For every integer n , we denote by w_n the prefix of w of size n . We say that two words w, w' are ε -close if the superior limit $\overline{\lim}_{n \rightarrow \infty} \text{dist}(w_{|_n}, w'_{|_n})/n$ is at most ε . Similarly, we say that a word w is ε -close to a language L if there exists a word $w' \in L$ which is ε -close to w . Last, the *block statistics* $\text{b-stat}(w)$ of w is the set of accumulation points of the sequence $(\text{b-stat}(w_{|_n}))_n$.

If we adapt the geometric embedding \mathcal{H} of the previous section, for this distance, an equivalence tester for two Büchi automata follows from the one previously defined for regular languages (over finite words). In this tester (whose correctness is in Appendix B.1), we modify the Definition 4.3 of \mathcal{H} , by simply restricting ourselves to A^k -compatible loops of (strongly) connected components of accepting states of A^k (we could also extend Theorem 4.11 to lasso words as in [9]).

Definition 4.13. *For every connected component C of A^k , let \mathcal{H}_C be the convex hull of the vector set $\{\text{b-stat}(w) : w \text{ is a loop in } C \text{ s.t. } |w|_b \leq m\}$. We denote by \mathcal{H}' the union $\bigcup_C \mathcal{H}_C$ where C ranges over all connected components of A^k that contain an accepting state and are reachable from an initial state.*

Theorem 4.14. *Theorem 4.12 is also valid for nondeterministic Büchi automata, with \mathcal{H}' taking the place of \mathcal{H} .*

4.5.2 Context-Free Languages

We construct here an exponential time test and an equivalence tester for context-free languages, given by their grammar, or by their push-down automaton (the two representations are polynomially equivalent so we will switch back

and forth between them as convenient). In comparison, the problem of whether two context-free grammars define the same language is not decidable.

We prove this in two parts. First, we move from the original grammar to a “block grammar” that approximates the block-statistics of the original language, just like for automata we use the k -power construction. The easiest way to do so is through a pushdown automaton with no ϵ -transitions, which requires the grammar to be in Greibach Normal Form. Fortunately, it is now known how to produce such a grammar with only a polynomial increase.

Lemma 4.15 ([7]). *Every context-free grammar is equivalent to a grammar in Greibach Normal Form (for which there exists a polynomial size equivalent pushdown automaton without ϵ -transitions), whose representation size is polynomial in that of the original grammar.*

Using this, we can now move from the original grammar to the “grammar of blocks” (proof in Appendix B.2).

Lemma 4.16. *For every fixed k and terminal alphabet Σ , there exists a polynomial time algorithm that converts a context-free grammar G over Σ to a context-free grammar G' over the terminal alphabet Σ^k , such that a word is recognizable by G if and only if its block representation (after appropriate “rounding” of the word if the letter count is not divisible by k) is recognizable by G' .*

We then use the original Parikh theorem about spectra of context-free languages, that provides a formula defining a semi-linear set on the letter counts of all possible words, which we call the *Parikh formula*.

Lemma 4.17 ([18]). *The computation of the Parikh formula can be reduced to the computation of the letter counts of all possible productions from a nonterminal to a word whose height is at most quadratic in the grammar size (and so the word size is at most exponential in the grammar size) and which contains at most one nonterminal character.*

From the spectrum one can clearly calculate the set \mathcal{H} that approximates the block-statistics of all large enough words, and then construct an appropriate \mathcal{H}_ϵ . We are thus done using the following (proof in Appendix B.2).

Lemma 4.18. *Given a nonterminal A , all letter counts of all productions of A into words of size up to l with no nonterminals can be calculated in time polynomial in the grammar size and l (where the alphabet is fixed). Similarly we can calculate all letter counts of all productions of A into words of size up to l containing a single nonterminal B .*

With the above lemmas we can construct string testers for a context-free language (which are not possible for the usual edit distance without moves, as the counter example in [1] works for the edit distance as well as the Hamming distance), as in Theorem 4.11. We can also construct equivalence testers for context-free grammars, as in Theorem 4.12. We sketch here how to construct an equivalence test.

Theorem 4.19. *There exists a deterministic algorithm T such that, given two context-free grammars A and B over a finite alphabet Σ whose representation size is at most m , and a real $\epsilon > 0$, $T(A, B, \epsilon)$:*

- (1) *accepts if A and B recognize the same language;*
- (2) *rejects if A and B recognize languages that are not 2ϵ -equivalent.*

Moreover the running time complexity of T is exponential in $m^{|\Sigma|^{O(1/\epsilon)}}$.

Proof. For $k = \frac{1}{\epsilon}$, using Lemma 4.16 we construct the corresponding grammars A' and B' over Σ^k . Then using Lemmas 4.18 and 4.17, we construct $\mathcal{H}_{A',\epsilon}$ and $\mathcal{H}_{B',\epsilon}$, and accept if and only if these two sets are identical. The exponential time is because Lemma 4.17 requires us to use an exponential l in the statement of Lemma 4.18. \square

Before we close we note a corollary for regular expressions with squaring. Although these recognize only regular languages, their (exact) equivalence problem is EXPSPACE-complete by [16], so the exponential time algorithm given here can be considered as a slight improvement (proof in Appendix B.2). Regular expressions with squaring can be converted into equivalent pushdown automata with a polynomially related size. Applying the previous theorem, we obtain the following result.

Theorem 4.20. *There exists a deterministic algorithm T such that, given two regular expressions with squaring A and B over a finite alphabet Σ whose length is at most m , and a real $\epsilon > 0$, $T(A, B, \epsilon)$:*

- (1) *accepts if A and B recognize the same language;*
- (2) *rejects if A and B recognize languages that are not 2ϵ -equivalent.*

Moreover the running time complexity of T is exponential in $m^{|\Sigma|^{O(1/\epsilon)}}$.

References

- [1] N. Alon, M. Krivelevich, I. Newman, and M. Szegedy. Regular languages are testable with a constant number of queries. *SIAM Journal on Computing*, 30(6):1842–1862, 2000.
- [2] Z. Bar-Yossef, T.S. Jayram, R. Krauthgamer, and R. Kumar. Approximating edit distance efficiently. In *Proceedings of the ACM Symposium on Theory of Computing*, 2004.
- [3] T. Batu, F. Ergun, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, and R. Sami. A sublinear algorithm for weakly approximating edit distance. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 316–324, 2003.
- [4] T. Batu, L. Fortnow, R. Rubinfeld, W. Smith, and P. White. Testing that distributions are close. In *41st IEEE Symposium on Foundations of Computer Science*, pages 259–269, 2000.
- [5] M. Blum and S. Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, 1995.
- [6] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.
- [7] N. Blum and R. Koch. Greibach normal form transformation revisited. *Information and Computation*, 150(1):112–118, 1999.
- [8] A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–30, 1997.
- [9] H. Chockler and O. Kupferman. ω -regular languages are testable with a constant number of queries. *Theoretical Computer Science*, 329:71–92, 2002.
- [10] G. Cormode. *Sequence Distance Embeddings*. PhD thesis, University of Warwick, 2003.
- [11] G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 667–676, 2002.
- [12] F. Ergün, S. Muthukrishnan, and S. Sahinalp. Comparing sequences with segment rearrangements. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science*, pages 183–194, 2003.
- [13] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.
- [14] F. Magniez and M. de Rougemont. Property testing of regular tree languages. In *Proceedings of 31st International Colloquium on Automata, Languages and Programming*, volume 3142 of *Lecture Notes in Computer Science*, pages 932–944. Verlag, 2004.
- [15] W. Masek and M. Paterson. A faster algorithm for computing string edit distance. *Journal of Computer and System Sciences*, 20(1):18–31, 1980.
- [16] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 125–129, 1972.
- [17] I. Newman. Testing membership in languages that have small width branching programs. *SIAM Journal on Computing*, 31(5):1557–1570, 2002.
- [18] R. Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, 1966.
- [19] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. Technical Report TR04-010, ECCS, 2004. <http://eccs.uni-trier.de/eccc-reports/2004/TR04-010/index.html>.
- [20] R. Rubinfeld. On the robustness of functional equations. *SIAM Journal on Computing*, 28(6):1972–1997, 1999.
- [21] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):23–32, 1996.
- [22] D. Shapira and J. Storer. Edit distance with move operations. In *Proceedings of Symposium on Combinatorial Pattern Matching*, volume 2373 of *Lecture Notes in Computer Science*, pages 85–98. Verlag, 2002.
- [23] D. Shapira and J. Storer. Large edit distance with multiple block operations. In *Proceedings of Symposium on String Processing and Information Retrieval*, pages 369–377, 2003.
- [24] L. Stockmeyer and A. Meyer. Word problems requiring exponential time. In *Proceedings of 5th ACM Symposium on Theory of Computing*, pages 1–9, 1973.

A Missing Proofs

A.1 Proof of Lemmas 3.4 and 3.5

The following simple result is well known and easy to check.

Proposition A.1. *Let $A \subseteq B$ be two finite subsets and let μ_A, μ_B be their respective uniform distributions. Then $|\mu_A - \mu_B| = 2 \frac{|B| - |A|}{|B|}$.*

Proof of Lemma 3.4. The proof consists of proving that both underlying block distributions are at ℓ_1 -distance at most $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4 n})$. Then the definitions of the vectors u-stat and bu-stat directly imply the result.

The uniform distribution consists of choosing uniformly at random a subword u of w of length k , that is an integer $z \in \{1, 2, \dots, n - k - 1\}$. The block uniform distribution consists of choosing uniformly at random a big block, an integer $0 \leq t \leq k - 1$, and then a subword u of length k at position i in the big block that satisfies $(i - 1) = t \pmod k$. In an equivalent way, the block uniform distribution is the uniform distribution over all subwords of size k that are inside some big block.

The number of subwords of size k that cross boundaries of big blocks is $(k - 1) \times (\frac{n}{K} - 1)$. Therefore using Proposition A.1, the ℓ_1 -distance between the distributions is upper bounded by $2 \times (k - 1) (\frac{n}{K} - 1) \times \frac{1}{n - k} = O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4 n})$. \square

Proof of Lemma 3.5. We assume that $n / (\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^6})$ is large enough so that the $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ of Lemma 3.3 is upper bounded by $\frac{\varepsilon n}{16}$, and the $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4 n})$ of Lemma 3.4 is upper bounded by $\frac{\varepsilon}{8}$.

Using Lemmas 3.3 and 3.4, we get subwords v and v' that respectively come from w and w' after deleting at most $\frac{\varepsilon n}{16}$ letters from each, so that $|\text{u-stat}(w) - \text{b-stat}(v)| \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{8}$ and $|\text{u-stat}(w') - \text{b-stat}(v')| \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{8}$.

From the hypothesis on w and w' , and using the triangle inequality on dist, we obtain that $\text{dist}(v, v') \geq 5\varepsilon n - \frac{\varepsilon n}{8}$. Therefore, using Lemma 3.1, we get that $|\text{b-stat}(v) - \text{b-stat}(v')| \geq 8\varepsilon - \frac{\varepsilon}{4}$, which implies from the construction of v, v' that $|\text{u-stat}(w) - \text{u-stat}(w')| \geq 8\varepsilon - \frac{\varepsilon}{4} - 2(\frac{\varepsilon}{2} + \frac{\varepsilon}{8}) = 6.5\varepsilon$. \square

A.2 Automata Independence Proofs

We show that loops in different automata for the same language are related, by correlating them with a definition that only depends on the language itself.

Definition A.2. *A word v over Σ^k is an (L, k) -loop if there exist two words u, w over Σ^k such that $uv^t w \in L$ for every integer t .*

One can remark that A^k -loops and (L, k) -loops are nearly the same, and in particular share the same statistics, according to the following result.

Proposition A.3. *Every A^k loop is also an (L, k) -loop, and on the other hand for every (L, k) -loop v there exists $t \geq 1$ such that v^t is an A^k loop. In particular, the set of statistics of loops of an automaton deciding the language L depends only on L and k .*

Proof. The first direction is clear.

The second direction requires some indications. Let us now consider the family of L words $(uv^t w)_{t \geq 1}$, and in particular let us consider one of the accepting paths for $uv^m w$. By a counting argument, there exist $0 \leq t < t' \leq m$, such that this accepting path reaches the same state after both $|uv^t|_b$ steps and $|uv^{t'}|_b$ steps. Hence the automaton contains some loop that corresponds to $v^{t'-t}$. Moreover, there is an accepting path that contains this loop, namely the one for $uv^m w$, and so it is indeed an A^k -loop. \square

However, for our purpose we need to consider not only loops, but sets of compatible loops. Here is the corresponding definition that depends on the language.

Definition A.4. A sequence of words v_1, \dots, v_l over Σ^k is a compatible (L, k) -loop sequence if there exists a permutation $\sigma : \{1, \dots, l\} \rightarrow \{1, \dots, l\}$, and words u_0, u_1, \dots, u_l over Σ^k , such that for every t_1, \dots, t_l we have $u_0 v_{\sigma(1)}^{t_1} u_1 v_{\sigma(2)}^{t_2} u_2 \dots u_{l-1} v_{\sigma(l)}^{t_l} u_l \in L$ (note that this in particular implies that v_1, \dots, v_l are (L, k) -loops).

Proposition A.5. Every compatible sequence of A^k -loops is also a compatible sequence of (L, k) -loops. On the other hand, for every compatible sequence v_1, \dots, v_l of (L, k) -loops there exist $t_1, \dots, t_l \geq 1$ such that $v_1^{t_1}, \dots, v_l^{t_l}$ is a compatible sequence of A^k -loops. In particular, the geometric set \mathcal{H} , constructed from any automaton A^k deciding the language L , depends only on L and k .

Proof. The proof follows the very same methods of the proof of Proposition A.3. \square

Proof of Proposition 4.5. The inclusion \supseteq is straightforward.

For the \subseteq inclusion, let us first state Caratheodory's theorem: In dimension d , any convex hull of N points p_1, \dots, p_N can be decomposed into the union of convex hulls of $(d + 1)$ points $p_{i_1}, \dots, p_{i_{d+1}}$ (with some possible repetitions), where the union is over every possible choices of these points. Hence this inclusion would have been also straightforward without the length assumption on the loops as the dimension here is $d = |\Sigma|^k$. To overcome the length constraint we use the fact that any loop $v = v_i$ of size $|v|_b > m$ can be decomposed (after a possible reordering of the block letters) into $v = u_1 u_2$, where u_1 and u_2 are loops which are also compatible with the other loops v_j . Repeating this argument inductively, we first prove that \mathcal{H} is not smaller if we upper bound the loop sizes by m . Then applying Caratheodory's theorem, we conclude the proof. \square

A.3 Proofs of Lemma 4.6 and Theorem 4.8

Proof of Lemma 4.6. First, recall that the block statistics $\mathbf{b}\text{-stat}(w)$ is invariant under block letter permutations. Moreover, if w' is w on which one block letter has been either deleted or inserted then $|\mathbf{b}\text{-stat}(w) - \mathbf{b}\text{-stat}(w')| \leq \frac{2}{\varepsilon|w|}$.

Let $w \in L$. Applying Proposition 4.4, we can delete less than m block letters from w so that the resulting word is a concatenation $w' \equiv_k u_1 u_2 \dots u_l$, where the u_i are A^k -compatible A^k -loops. Since w' is obtained from w using at most m deletions of block letters, we have $|\mathbf{b}\text{-stat}(w) - \mathbf{b}\text{-stat}(w')| \leq \frac{2m}{\varepsilon|w|}$. This concludes the proof since $\mathbf{b}\text{-stat}(w') \in \mathcal{H}$. \square

Proof of Theorem 4.8. Let $n = |w|$. For simplicity, we assume that k divides n , otherwise we just delete at most $k - 1$ letters from w so that the new length is dividable by k . From Lemma 4.7, there exists a word $w' \in L$, such that $0 \leq |w'| - n \leq (l + 2)\frac{2m}{\varepsilon}$ and $|\mathbf{b}\text{-stat}(w') - X| \leq (l + 1)\frac{3m}{\varepsilon n}$, where $l = 1 + |\Sigma|^k$. We again assume that k divides $|w'|$.

Assume that $|w| = |w'|$. Then, using Lemma 3.1, we get that $\text{dist}(w, w') \leq (\frac{1}{2}(\delta + (l + 1)\frac{3m}{\varepsilon n}) + \varepsilon)n$.

If w and w' have different sizes, we artificially increment the size of w by adding at most $(l + 2)2m$ block letters at the end of w (recall that adding a block letter adds k to the word size). The deviation of its block statistics is then at most $(l + 2)2m \times \frac{2}{\varepsilon n}$, so we asymptotically get the same bound. \square

A.4 Details of the Construction of \mathcal{H}

We proceed recursively on the length t of paths between two possible states of A^k , for $t = 1, \dots, m$. Let P_t be an $m \times m$ matrix where the entry (i, j) is the set of block statistics corresponding to a path of length t between the states i and j . Let us consider the algebra of sets of distributions over Σ^k with the operations \cup, \odot_t , where \odot_t is distributive over \cup and defined for singletons by $\{\vec{x}\} \odot_t \{\vec{y}\} = \{\frac{1}{t+1}\vec{x} + \frac{t}{t+1}\vec{y}\}$. If we denote by \circ_t the matrix multiplication over this algebra, then the matrices P_t satisfy the following simple inductive equation, where P_1 is directly given by A^k (by setting each non-empty entry of P_1 to be the set of unit vectors corresponding to the block letters labeling the corresponding arcs in A^k):

$$P_{t+1} = P_1 \circ_t P_t.$$

Proof of Lemma 4.9. We first compute as we explained above the matrices $(P_t)_{t=1,\dots,m}$. At the end of the process, the diagonals of those matrices contain the block statistics of all A^k -loops of length at most m . Then, a tuple of $(|\Sigma|^{1/\varepsilon} + 1)$ loops is compatible if and only if there exists an accepting path of the automaton which passes through all states of the respective origins of the loops, a condition that can also be checked in polynomial time by using matrix multiplication over an appropriate algebra. Using Proposition 4.5, we know that including in H the statistics of the corresponding compatible sets is sufficient. The upper bounds on the size and the time complexity of the decomposition come from the previous observation that at most $m^{|\Sigma|^k}$ block statistics are considered. \square

B Details of Extensions

B.1 Infinite Regular Languages

We now enumerate intermediate results from which Theorem 4.14 directly follows. We first state the following lemmas whose proofs are based on the ones of Lemmas 4.6 and 4.7. We fix a Büchi automaton A that recognizes L .

Lemma B.1. *Let $w \in L$. Then $\text{b-stat}(w) \subseteq \mathcal{H}'$.*

Proof. We consider an accepting path in A^k for w (as a block word). This path can clearly move from a (strong) connectivity class to another only finitely many times, and hence apart from a finite number of positions this path is fully contained in one of the connectivity classes of A^k . From this the remainder of the proof is straightforward. \square

Lemma B.2. *Let $X \in \mathcal{H}'$. Then there exists $w \in L$ such that $\text{b-stat}(w) = \{X\}$.*

If we relaxed a little bit the conclusion of the above lemma, one can remark that, for any fixed $\delta > 0$, there exists a lasso word $w = uv^\omega$, where $|v|$ is divisible by k , such that $\text{b-stat}(w) = \{\text{b-stat}(v)\}$ and $|\text{b-stat}(v) - X| \leq \delta$.

Based on these lemmas and Lemma 3.1, we can state that \mathcal{H}' is a robust characterization of L , whenever the block statistics have a unique accumulation point.

Lemma B.3. *Let $w \in \Sigma^\omega$ and $X \in \mathcal{H}'$ be such that $\text{b-stat}(w)$ is a singleton $\{Y\}$ satisfying $|Y - X| \leq \delta$. Then w is $(\frac{\delta}{2} + \varepsilon)$ -close to L .*

Now the crucial point for concluding the proof of Theorem 4.14 is the following lemma, which allows us to consider only words whose block statistics have a unique accumulation point.

Lemma B.4. *Let $w \in L$, and let $w' \in \Sigma^\omega$ be ε' -far from w . Then there exists another word $v \in L$ such that $\text{b-stat}(v)$ has only one element, $\text{b-stat}(v) \subseteq \text{b-stat}(w)$, and v is $(\varepsilon' - \varepsilon)$ -far from w' .*

Proof. Let (n_i) be an increasing sequence of integers such that $\text{dist}(w_{|n_i}, w'_{|n_i})$ has a limit at least ε' , when $i \rightarrow \infty$. We can extract again from (n_i) a subsequence (m_j) for which $\text{b-stat}(w_{|m_j})$ has limit X . Note that this subsequence still satisfies that $\text{dist}(w_{|m_j}, w'_{|m_j})$ has a limit at least ε' , when $j \rightarrow \infty$.

Observe that $X \in \mathcal{H}'$ by Lemma B.1, and therefore, using Lemma B.2, there exists a word $v \in L$ such that $\text{b-stat}(v) = \{X\}$. From Lemma 3.1, we get that $\text{dist}(w_{|m_j}, v_{|m_j})$ has superior limit at most ε . Therefore, by the triangle inequality, $\text{dist}(w'_{|m_j}, v_{|m_j})$ has superior limit at least $(\varepsilon' - \varepsilon)$, which concludes the proof. \square

B.2 Context-Free Languages

Proof of Lemma 4.16. We first use Lemma 4.15 to move from G to an equivalent grammar in Greibach Normal Form, and then to a pushdown automaton A with no ε -transitions. Then we take the k power of A , replacing each path of k transitions over Σ with one transition over Σ^k . If some of the resulting transitions consume more than one letter off the stack, we add new states and ε -transitions as necessary. Finally, we move from this automaton to the equivalent block grammar G' (which by now is not necessarily in normal form). Each of the above transitions causes no more than a polynomial blowup in the representation size, and hence we are done. \square

Proof of Lemma 4.18. We provide here only the proof for finding the productions into words with no nonterminals, as the proof for words with a single given nonterminal is an easy extension. We first move the grammar to Chomsky Normal Form (which causes a polynomial increase in the grammar size), so that there will be no production in the grammar with more than two nonterminals. Note that this transformation preserves the original nonterminals (while it may also add some new nonterminals), so we can do this transition and still preserve the list of required words that are derivable from a nonterminal A .

We maintain an array, that for every letter (nonterminal or terminal) provides a list of possible letter counts of the words of terminals of size up to l that can be produced. The maximum size of every such list is $\binom{l+h}{h} = l^{O(h)}$ for $h = |\Sigma|^k$ the number of possible terminals, and so it is polynomial in l . In the beginning, the list for any terminal α contains a single vector which is ‘1’ on the coordinate corresponding to α and ‘0’ on all other coordinates, while for any nonterminal the initial list is empty. The lists for the terminal letters will not change (and thus remain of size 1) all throughout the algorithm.

We now repeat the following procedure, as long as we can increase the size of any of our lists: Given a production of the grammar, say “ $B \rightarrow u$ ”, we look at the current lists of the nonterminals that appear in u . For every possibility of picking a vector from the list of every such nonterminal (if the same nonterminal appears twice in u then we can pick two vectors from its list or the same vector twice), we sum up these vectors together with the (unique) vectors of the terminals in u , and add this vector sum to the list of B if its weight is no more than l . As we assumed that there are no more than two nonterminals in every production, this step takes at most quadratic time in the maximum size of a list.

The above procedure must stop after at most a number of steps that is equal to the number of nonterminals times the maximum size of a list, and so after a polynomial number of steps we arrive at lists which can no longer be enlarged in the above manner. It is not hard (and left to the reader) to see that at this point, the list for every nonterminal A will be exactly the list of letter counts of all its possible productions into words of size up to l . \square

Proof of Theorem 4.20. A regular expression with squaring A can be converted to an equivalent pushdown automaton whose size is polynomial in m . The conversion follows the vein of converting a regular expression without squaring to a (normal) nondeterministic automaton, only here in addition we convert each square sign to a corresponding state s that pushes down the stack a symbol unique to it: After pushing down the symbol s makes an ϵ -transition to the state corresponding to the beginning of the subexpression that is squared, and when s is reached again and finds its symbol already in the stack it pops the symbol and makes an ϵ -transition to the state corresponding to the expression element following the squared part.

We convert A and B to their respective pushdown automata (and then to their corresponding context-free grammars), and then we use Theorem 4.19. \square

C Tolerant equivalence testers

Considering the soundness and robustness of the uniform statistic (Lemma 3.2 and Lemma 3.5), we could make the property testers as well as the equivalence testers for nondeterministic automata (and their extensions) tolerant, if instead of reducing the original automaton to its “block version”, we reduce it to its “shingle version”. In other words, we would like to construct an automaton that recognizes a word w over Σ^k if and only if it consists of all subwords of size k of some word u recognizable by the original automaton. This can be done for most computational models discussed above, and to illustrate this we sketch the construction for nondeterministic finite automata.

Lemma C.1. *Given an automaton A with m states over Σ , it is possible to construct an automaton B with $l = m \cdot |\Sigma|^{k-1}$ states over Σ^k in time polynomial in l , where B accepts a word w if and only if there exists a word u accepted by A so that w consists of all length k subwords of u .*

Proof. We assume that A contains no ϵ transitions. Let the “memory automaton” M denote the following automaton consisting of $|\Sigma|^{k-1}$ states and with no accepting state: Each state of M is labeled by a word from Σ^{k-1} . From a word a_1, \dots, a_{k-1} the automaton has a transition to a_2, \dots, a_{k-1}, b for every $b \in \Sigma$, which is labeled by a_1, \dots, a_{k-1}, b .

The automaton B consists of running in parallel the original automaton A and the automaton M , so in effect the state set of B is the product of the state spaces of M and A . The initial states of B are the coupling of any state

a_1, \dots, a_{k-1} of M with any state of A reachable from an initial state by a path reading the letter sequence a_1, \dots, a_{k-1} from the input. The accepting states of B are all couplings of a state of M with an accepting state of A .

Defining the transition function, there is a transition from a state s_1 of A coupled with a state a_1, \dots, a_{k-1} of M to a state s_2 of A coupled with a state a_2, \dots, a_{k-1}, b of M , labeled by a_1, \dots, a_{k-1}, b , for every a_1, \dots, a_{k-1} and b where the original automaton A contained a transition from s_1 to s_2 labeled by b . The proof that B is the required automaton is straightforward and is left to the reader. \square

We conclude by sketching what this implies e.g. for equivalence testers.

Theorem C.2. *There exists a deterministic algorithm T such that, given two automata A and B over a finite alphabet Σ with at most m states and a real $\varepsilon > 0$, $T(A, B, \varepsilon)$:*

- (1) *accepts if A and B recognize languages that are ε^2 -equivalent;*
- (2) *rejects if A and B are not 10ε -equivalent.*

Moreover the running time complexity of T is in $m^{|\Sigma|^{O(1/\varepsilon)}}$.

Proof. We use the same proof of Theorem 4.12, only instead of calculating the geometric embedding of the powers A^k and B^k , we calculate the geometric embedding of the automata derived from A and B by Lemma C.1 for an appropriate $k = O(1/\varepsilon^2)$, and accept if these are close enough. This closeness can be checked e.g. from an appropriate discretization of the embedding. \square