APPROXIMATE SATISFIABILITY AND EQUIVALENCE*

ELDAR FISCHER†, FRÉDÉRIC MAGNIEZ‡, AND MICHEL DE ROUGEMONT§

Abstract. Inspired by Property Testing, for every $\varepsilon > 0$ we relax the classical satisfiability $U \models F$ between a finite structure U of a class $\mathbf{K}$ and a formula F, to a notion of ε-satisfiability $U \models_\varepsilon F$, and relax the classical equivalence $F_1 \equiv F_2$ between two formulas F_1 and F_2 to ε-equivalence $F_1 \equiv_\varepsilon F_2$. We consider strings and trees with the norm of the edit distance with moves, and show that, unlike their exact counterparts, these approximate notions can be efficiently decided.

We use a statistical embedding of words (resp. trees) into ℓ_1, which generalizes the original Parikh mapping, obtained by sampling $O(f(\varepsilon))$ finite samples of the words (resp. trees). We give a tester for equality and membership in any regular language, in time independent of the size of the structure. Using our geometrical embedding, we can also test the equivalence between two regular properties over words, defined by regular expressions or Monadic Second Order formulas. Our equivalence tester has polynomial time complexity in the size of the automaton (or regular expression), for any fixed ε, whereas the exact version of the equivalence problem is PSPACE-complete. We also prove versions of some of these results for trees, but with a worse time complexity.

Last, we extend the geometric embedding, and hence the testing algorithms, to infinite regular languages and to context-free languages. For context-free languages, the equivalence tester has an exponential time complexity for any fixed ε, whereas the exact version is not even decidable.

1. Introduction. Let $\mathbf{K}$ be a class of finite structures with a distance dist between structures. In the classical setting, satisfiability is the decision problem of whether $U \models F$ for a structure $U \in \mathbf{K}$ and a formula F, and equivalence is the decision problem of whether $F_1 \equiv F_2$, *i.e.* whether $U \models F_1$ iff $U \models F_2$ for all $U \in \mathbf{K}$, for two formulas F_1 and F_2. Equivalence is typically very hard to decide as a function of the size of the formulas, and in some cases undecidable. For any $\varepsilon > 0$, two structures are *ε-close* if their normalized distance is at most ε, and otherwise they are *ε-far*. We introduce the notions $U \models_\varepsilon F$ and $F_1 \equiv_\varepsilon F_2$ based on Property Testing. Let $U \models_\varepsilon F$ if there exists a U' ε-close to U such that $U' \models F$, and otherwise $U \not\models_\varepsilon F$. Let $F_1 \equiv_\varepsilon F_2$ if all but finitely many structures that satisfy $U \models F_1$ satisfy also $U \models_\varepsilon F_2$, and all but finitely many structures that satisfy $U \models F_2$ satisfy also $U \models_\varepsilon F_1$.

An *ε-tester* for a property P defined by a formula F on $\mathbf{K}$, is a randomized algorithm which takes a finite structure $U \in \mathbf{K}$ of size n as input, and distinguishes with high probability between $U \models F$ and $U \not\models_\varepsilon F$. A property P is *testable* if there exists a randomized algorithm A such that, for every $\varepsilon > 0$ as input, $A(\varepsilon)$ is an ε-tester of P whose time complexity only depends on ε, *i.e.* it is independent of the size n. An *ε-equivalence tester* for a logic $\mathcal{L}$ is an algorithm that can distinguish between $F_1 \equiv F_2$ and $F_1 \not\equiv_\varepsilon F_2$, for any two formulas $F_1, F_2 \in \mathcal{L}$.

We consider the class of strings and trees with a specific distance, and show that these approximate notions can be efficiently distinguished for important properties. These decision methods are robust, in the sense that they are adaptable to noisy inputs.

Property testing of regular languages was first considered in [2] for the Hamming

* A preliminary version of this paper appeared in *Proceedings of 21st IEEE Symposium on Logic in Computer Science*, pages 421–430, 2006. Supported in part by the French ANR Defis program under contract ANR-08-EMER-012 (QRAC project), and the French ANR Sesur program under contract ANR-07-SESU- 013 (VERAP project), and by Israel Science Foundation grants number 55/03 and 1101/06

†Faculty of Computer Science, Technion – Israel institute of technology, Haifa 32000, Israel. eldar@cs.technion.ac.il

‡LRI, Univ Paris-Sud, CNRS; F-91405 Orsay, France. magniez@lri.fr

§LRI, Univ Paris 2; F-91405 Orsay, France. mdr@lri.fr

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. The *edit distance* between two words (resp. trees) is the minimal number of insertions, deletions and substitutions of a letter (resp. node) required to transform one word (resp. tree) into the other. The *edit distance with moves* considers one additional operation: Moving one arbitrary substring (resp. subtree) to another position in a single step. Our results depend on this last specific distance and in particular do not apply to the edit distance without moves.

We develop a statistical embedding of words into ℓ_1 that has similarities with the Parikh mapping [18]. Based on this embedding, we develop an ε-tester (**Theorem 3.9**) for the equality between two words whose complexity is $|\Sigma|^{O(1/\varepsilon)}$, where $|\Sigma|$ is the alphabet size. Our equality tester is also *tolerant*, that is it is not only an ε-tester (which by itself would have been trivial to construct), but it also accepts with high probability words that are ε^2-close to each other. The 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 weak approximation was pointed out in [19]. Based on this observation and our tolerant tester, we directly get an approximation algorithm for the normalized edit distance with moves between two words (**Corollary 3.10**), whose complexity is $|\Sigma|^{O(1/\varepsilon)}$. To our knowledge this is the first such approximation algorithm whose complexity is independent of the size n.

Computing the edit distance with moves is NP-hard [22], but it was approximated within an $\tilde{O}(\ln n)$ factor in only near linear time [10]. It has been used in [15] for testing regular languages, where the tester is more efficient and simpler than the one of [2], and can be generalized to tree regular languages. We note that the edit distance without moves, whose value always 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 [3].

We then 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 4.9**). Discretizing this representation gives us a new ε-tester (**Theorem 4.17**) for regular languages whose query complexity is $|\Sigma|^{O(1/\varepsilon)}$ and whose time complexity is $2^{|\Sigma|^{O(1/\varepsilon)}}$. Whereas the complexity of previous testers for regular languages depended exponentially on the number of states m of the corresponding automaton (whether it is deterministic or non-deterministic), here the tester construction requires time $m^{|\Sigma|^{O(1/\varepsilon)}}$, which is polynomial in m for a fixed ε. In addition, the automaton here is only used in a preprocessing step to construct the tester, which is independent of the size of the input.

Using again discretization, we construct an ε-equivalence tester (**Theorem 4.18**) for nondeterministic finite automata in deterministic polynomial time, that is $m^{|\Sigma|^{O(1/\varepsilon)}}$, where the exact decision version of this problem is PSPACE-complete by [23]. We then extend this result to the ε-equivalence testing of Büchi automata (**Theorem 5.11**) after generalizing our definitions to deal also with languages of infinite words, and a deterministic exponential time algorithm for the ε-equivalence testing of context-free grammars (**Theorem 5.12**), for which the exact decision version is not even recursively computable. In particular our equivalence testers distinguish whether MSO (Monadic second-order) formulas on strings are equivalent or not even ε-equivalent. As exact model checking is not feasible in many settings, it would be

interesting to generalize this approach to other logics in the future.

Last we consider 2-ranked ordered trees, but our results generalize to any ranked trees. When trees are interpreted as graphs, their edit distance with moves is very related to the minimal number of edges one has to add or remove in order to get one tree from the other. This distance was well investigated in the context of property testing in bounded-degree graphs [12]. We define a compression of trees by a relabeling of the tree. Basically, all small subtrees are removed and encoded into the labels of their ancestor nodes. Such a compression removes a large fraction of the 2-degree nodes, and can therefore be used to approximately encode any ranked tree T with a word $w(T)$. Since our ℓ_1-embedding of $w(T)$ can be approximately sampled from samples on T, some of our previous results on words can be extended to trees. Then the tree isomorphism problem is testable (**Theorem 6.8**). This result is surprising by itself since there is a negative result in the context of dense graphs [1]. Lastly regular tree languages have an $(\varepsilon^4, O(\varepsilon))$-tolerant tester (**Theorem 6.9**) whose query complexity is $(|\Sigma|+1)^{O(1/\varepsilon^5)}$ and time complexity is $2^{(|\Sigma|+1)^{O(1/\varepsilon^5)}}$. Again, as opposed to previous testers for tree regular languages [15], here the automaton is only used in a preprocessing step to build the tester, in time exponential in the tree automaton size for a fixed ε.

2. Preliminaries. Let $\mathbf{K}$ be a class of finite structures U, such as words or trees. A property P is a subset of $\mathbf{K}$. A formula F over $\mathbf{K}$ is defined in some logic such as First-Order Logic or Monadic Second-Order Logic. We use here the logical characterization of regular languages of words (resp. trees) as Monadic Second Order Logic properties. We say that $U \in \mathbf{K}$ *satisfies* P, or $U \models P$, if $U \in P$. When P is defined by a formula F, we extend this notation to F. Instead of properties, we may speak of classes or languages, and in particular regular languages of words and trees.

2.1. Distances on Words and Trees. An *elementary operation* on a word w is an insertion, a deletion or a substitution of a single letter, or the *move* of a whole subword of w to another position. The *edit distance with moves* $\mathsf{dist}(w, w')$ between w and w' is the minimal number of elementary operations performed on w to obtain w'.

The above distance is extended to trees by generalizing the elementary operations. An *elementary operation* (see Figure 2.1) on an unranked ordered tree T is either an insertion or a deletion of a node [24], the substitution of a label, or the move of an entire subtree [15]. More precisely, a *move* (u, v, i) moves in one step u (and the corresponding subtree rooted at u) to be the i-th successor of v, shifting every j-th successor of v for $j \geq i$ by one. As a consequence, the new parent of u is now v. When trees are specified to be r-ranked, we will restrict ourselves only to insertions, deletions and moves that maintain an r-ranked tree.

2.2. Approximate Satisfiability and Equivalence. We define the notion of approximate satisfiability as in property testing [13]. Let $\mathbf{K}$ be a class of finite structures U with a distance measure dist between structures. Since property testing is an approximate notion of verification for dense instances, or equivalently for normalized distances, we first define a suitable notion of closeness for any distance dist. We say that $U, U' \in \mathbf{K}$ are *ε-close* if their distance is at most $\varepsilon \times M$, where M is a normalization factor, that is the maximum of $\mathsf{dist}(V, V')$ when V and V' range over $\mathbf{K}$ and have respectively same sizes as U and U'. We say that the two structures are *ε-far* if they are not ε-close. For words and trees, M is set to be the maximal size of the respective structures, since this is always on the order of the maximal distance.

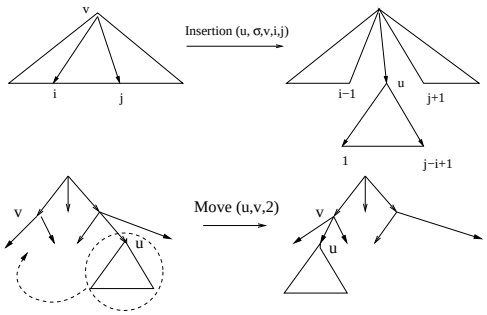

FIG. 2.1. *Examples of elementary operations on trees.*

DEFINITION 2.1. *Let P be a property on $\mathbf{K}$. A structure $U \in \mathbf{K}$* ε-satisfies *P, or $U \models_\varepsilon P$ for short, if U is ε-close to some $U' \in \mathbf{K}$ such that $U' \models P$.* When P is defined by a formula F we extend this notation to F. Note that $U \not\models_\varepsilon P$ means that U is ε-far from every U' such that $U' \models P$.

DEFINITION 2.2 (Property Tester [13]). *Let $\varepsilon > 0$. An* ε-tester *for a property $P \subseteq \mathbf{K}$ is a randomized algorithm A such that, for any structure $U \in \mathbf{K}$ as input:*
(1) If $U \models P$, then A accepts with probability at least $2/3$;
(2) If $U \not\models_\varepsilon P$, then A rejects with probability at least $2/3$. If in addition the algorithm is guaranteed to always accept if $U \in P$, then we call it a one-sided error *ε-tester.*
When (1) is amended as follows for some $0 < \varepsilon_0 < \varepsilon$:
(1') If $U \models_{\varepsilon_0} P$, then A accepts with probability at least $2/3$;
then we say that the tester is a tolerant $(\varepsilon_0, \varepsilon)$-tester *[19].* Approximation algorithms are related to tolerant testers [19]. Let $\alpha, \beta : \mathbb{R} \to \mathbb{R}$ and $f : \mathbf{K} \to \mathbb{R}$. An (α, β)-*approximation* of f is a randomized algorithm that, for any input $U \in \mathbf{K}$, outputs a value z such that $\Pr[\alpha(f(U)) \leq z \leq \beta(f(U))] \geq 2/3$.

A *query* to a structure U depends on the model for accessing the structure. For a word w, a query is asking for the value of $w[i]$ for some i. For a tree T, a query is asking for the value of the label of v for some node v, and potentially for the index of its parent and its j-th successor, for some j. We also assume that the algorithm may query the input size. The *query complexity* is the number of queries made to the structure. The *time complexity* is the usual definition, where we assume that the following operations are performed in constant time: arithmetic operations, a uniformly random choice of an integer from any finite range not larger than the input size, and a query to the input.

DEFINITION 2.3. *A property $P \subseteq \mathbf{K}$ is* testable*, if there exists a randomized algorithm A such that, for every real $\varepsilon > 0$ as input, $A(\varepsilon)$ is an ε-tester of P, and the query and time complexities of the algorithm A depend only on ε.*

We extend these definitions to any formula F that defines P. We then introduce the new notion of equivalence testing for two properties P_1 and P_2, and in particular when the properties are definable by two formulas F_1 and F_2 over a logic $\mathcal{L}$.

DEFINITION 2.4. *Let $\varepsilon > 0$. Let F_1 and F_2 be two formulas on $\boldsymbol{K}$. Then F_1 is* ε-equivalent *to F_2, or $F_1 \equiv_\varepsilon F_2$ for short, if all but finitely many structures $U \in \mathbf{K}$ that satisfy $U \models F_1$ satisfy also $U \models_\varepsilon F_2$, and all but finitely many structures $U \in \mathbf{K}$ that satisfy $U \models F_2$ satisfy also $U \models_\varepsilon F_1$.*

DEFINITION 2.5 (Equivalence tester). *Let $\varepsilon > 0$. A (deterministic)* ε-equivalence tester *for $\mathcal{L}$ is a (deterministic) algorithm A such that, given as input $F_1, F_2 \in \mathcal{L}$:*

(1) If $F_1 \equiv F_2$, then A accepts;
(2) If $F_1 \not\equiv_\varepsilon F_2$, then A rejects. The probabilistic version would require modifying the above conditions to hold for A only with probability 2/3.

3. Words. We will define several statistics over words and study their robustness [20, 21] and tolerance. Robustness means that far words have far statistics, and tolerance 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 tolerant. Last we will see how to use these statistics to efficiently decide approximate satisfiability and equivalence.

3.1. Statistical Embeddings. Let k be an integer and $\varepsilon = 1/k$. For a word w over a finite alphabet Σ, we will define and study statistics of subwords of k consecutive letters of w for several probability distributions over the subwords. We will call Σ^k the *block alphabet*, and its elements *block letters*. We will also denote by $|w|$ the size of w, by $w[i]$ the i-th letter of w and by $w[i \dots j]$ the subword $w[i]w[i+1]\cdots w[j]$, for $i \leq j$.

In this section, w and w' are two words of size n over Σ, such that k divides n. We implicitly decompose any word w into consecutive subwords of size k, $w = w[1]_b w[2]_b \cdots w[\varepsilon n]_b$, where $w[i]_b \in \Sigma^k$ is the i-th block letter of w. We then denote by $|w|_b$ the number of block letters of w, namely $|w|_b = |w|/k$. The *block statistics* $\mathsf{b\text{-}stat}(w) \in \mathbb{R}^{\Sigma^k}$ is the statistics of the block letters of w, that is, for every $u \in \Sigma^k$, the value $\mathsf{b\text{-}stat}(w)[u]$ is equal to the probability that we get $w[j]_b = u$ for a uniformly random choice of $j \in \{1, \dots, n/k\}$.

The *block distribution* of w is the uniform distribution on the block letters $w[1]_b, \dots, w[\varepsilon n]_b$ (with 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 then satisfies $\mathrm{E}(X) = \mathsf{b\text{-}stat}(w)$.

We want to construct statistics that are both robust and tolerant. Since the block statistics will turn out to be non-robust, we therefore define other statistics using variants of the block distribution. The *uniform distribution* $\mathsf{u\text{-}stat}(w)$ corresponds to a uniform and random choice of a (consecutive) 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".

As an example, for binary words, $k = 2$ and respectively $\varepsilon = 0.5$, there are 4 possible subwords of length 2, which we take in lexicographic order. For the binary word $w = 000111$, $\mathsf{b\text{-}stat}(w) = (1/3, 1/3, 0, 1/3)$, whereas $\mathsf{u\text{-}stat}(w) = (2/5, 1/5, 0, 2/5)$ as there are 2 blocks 00, 1 block 01, no block 10 and 2 blocks 11 among the possible 5 blocks.

The block uniform distribution will serve as a link between the block and the uniform distributions. To define the *block uniform distribution* $\mathsf{bu\text{-}stat}(w)$ 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 $n = \Omega(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^5})$. We call the new blocks the *big blocks*. Now $\mathsf{bu\text{-}stat}(w)$ 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 (non-consecutive) subword of the original word.

In order to construct efficient algorithms based on these statistics, we need to efficiently approximate them. 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 a Chernoff-Hoeffding bound together with a direct union bound is polynomially tight using an argument similar to the one of [4].

LEMMA 3.1. *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 $t>0$, $\Pr\left[|\mathrm{E}_d(f(Y)) - \frac{1}{N}\sum_{i=1}^N f(Y_i)| \geq D\times t\right] \leq D \times 2e^{-2Nt^2}$.*

Proof. Let $\mu = \mathrm{E}_d(f(Y))$ and $\hat{\mu}_N = \frac{1}{N}\sum_{i=1}^N f(Y_i)$. For each $u \in \{1,\dots,D\}$, the i-th coordinate of μ and $\hat{\mu}_N$ satisfy $\Pr[|\mu[u] - \hat{\mu}_N[u]| \geq t] \leq 2e^{-2Nt^2}$, 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. □

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 other coordinates are zero. Let stat denote either $\mathsf{b\text{-}stat}$ or $\mathsf{u\text{-}stat}$. Then we define $\widehat{\mathsf{stat}}_N(w) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1,\dots,N} X_i$.

COROLLARY 3.2. *There exists $N \in O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^3})$ for which $\Pr[|\mathsf{stat}(w) - \widehat{\mathsf{stat}}_N(w)| \geq \varepsilon] \leq \frac{1}{3}$, where stat denotes either $\mathsf{b\text{-}stat}$ or $\mathsf{u\text{-}stat}$.*

Proof. By definition, $\mathsf{stat}(w) = \mathrm{E}_d(f(Y))$ where f gives the stat vector of a block, d is the uniform distribution over the relevant blocks in the w (*e.g.* the ones starting at a multiple of k for the $\mathsf{b\text{-}stat}$ statistic), and $\widehat{\mathsf{stat}}_N(w) = \frac{1}{N}\sum_{i=1}^N f(Y_i)$. From Lemma 3.1 where $D = |\Sigma|^{1/\varepsilon}$ and $D\times t = \varepsilon$, *i.e.* $t = \varepsilon/|\Sigma|^{1/\varepsilon}$, we conclude that $\Pr[|\mathsf{stat}(w) - \widehat{\mathsf{stat}}_N(w)| \geq \varepsilon] \leq D \times 2e^{-2Nt^2} \leq 1/3$ if $N \geq c \times \ln D/t^2 = c\times\ln(|\Sigma|)|\Sigma|^{2/\varepsilon}/\varepsilon$ for an appropriate global constant c. □

3.2. Robustness and Tolerance. Note that $\mathsf{b\text{-}stat}(w) = \mathsf{b\text{-}stat}(w')$ iff w' can be obtained by a permutation of the block letters of w (since w and w' have the same size). We then say that $w \equiv_b w'$. This can be extended when the equality is only approximate, by relating the distance between two words to the ℓ_1-distance of their respective block statistics.

LEMMA 3.3 (Robustness). $\mathsf{dist}(w,w') \leq (\frac{1}{2}|\mathsf{b\text{-}stat}(w) - \mathsf{b\text{-}stat}(w')| + \varepsilon)\times n$.

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

Choose sets X_+ and X_- satisfying the following. For every block letter $\alpha \in \Sigma^k$ for which $\mathsf{b\text{-}stat}(w)[\alpha] > \mathsf{b\text{-}stat}(w')[\alpha]$, the set X_+ contains exactly $\mathsf{b\text{-}stat}(w)[\alpha] - \mathsf{b\text{-}stat}(w')[\alpha]$ indices i for which $w[i]_b = \alpha$. And for every block letter $\beta \in \Sigma^k$ for which $\mathsf{b\text{-}stat}(w')[\beta] > \mathsf{b\text{-}stat}(w)[\beta]$, the set X_- contains exactly $\mathsf{b\text{-}stat}(w')[\beta] - \mathsf{b\text{-}stat}(w)[\beta]$ indices j for which $w'[j]_b = \beta$. Note that X_+ and X_- exist and have the same

cardinality, which is $\frac{n}{2k}|\mathsf{b\text{-}stat}(w)-\mathsf{b\text{-}stat}(w')|$. Initially we let $w''=w$. While $X_+\neq\emptyset$ 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. □

We now prove that u-stat is both robust and tolerant (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 for an even k their block statistics are $\Omega(1)$-far. The proof of the robustness of u-stat will use as an intermediate step the robustness of the block uniform statistics bu-stat. For the tolerance of u-stat, the proof is much simpler.

LEMMA 3.4 (Tolerance). *Let $n=\Omega(\frac{1}{\varepsilon})$. If $\mathsf{dist}(w,w')\leq\varepsilon^2 n$ then $|\mathsf{u\text{-}stat}(w)-\mathsf{u\text{-}stat}(w')|\leq 6.1\varepsilon$.*

Proof. First, remember that there are $n-k+1$ contiguous subwords of size k in w. Assume that $\mathsf{dist}(w,w')=1$. In case of a simple edit operation (insertion, deletion, substitution) on a letter, $|\mathsf{u\text{-}stat}(w)-\mathsf{u\text{-}stat}(w')|\leq 2\times\frac{k}{n-k+1}$. For a move operation, if $w=ABCD$ and $w'=ACBD$ 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 $|\mathsf{u\text{-}stat}(w)-\mathsf{u\text{-}stat}(w')|\leq 2\times\frac{3(k-1)}{n-k+1}$.

This means that if $\mathsf{dist}(w,w')\leq\varepsilon^2 n$ and $n=\Omega(\frac{1}{\varepsilon})$, then by repeated use of the triangle inequality $|\mathsf{u\text{-}stat}(w')-\mathsf{u\text{-}stat}(w')|\leq\varepsilon^2 n\times\frac{6.1k}{n}=6.1\varepsilon$, since $k=\frac{1}{\varepsilon}$. □

We now show that the robustness for $\mathsf{b\text{-}stat}(w)$ implies the robustness for $\mathsf{bu\text{-}stat}(w)$, which will then imply the robustness for $\mathsf{u\text{-}stat}(w)$. For a big block B_i, where $i=1,\ldots,\frac{n}{K}$, 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 $\mathsf{bu\text{-}stat}(w)$ we have $\mathsf{bu\text{-}stat}(w)=\frac{K}{n}\sum_{i=1}^{n/K}\mathrm{E}_{t_i=0,\ldots,k-1}(\mathsf{b\text{-}stat}(v_{i,t_i}))=\mathrm{E}_v(\mathsf{b\text{-}stat}(v))$.

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

Proof. Fix a coordinate $u\in\Sigma^k$. For every $i=1,\ldots,\frac{n}{K}$, let X_i be the random variable $X_i\stackrel{\text{def}}{=}\mathsf{b\text{-}stat}(v_{i,t_i})[u]$, where t_i is chosen uniformly in $\{0,\ldots,k-1\}$. 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 $\mathrm{E}_v(\mathsf{b\text{-}stat}(v)[u])=\frac{K}{n}\sum_i\mathrm{E}(X_i)=\mathsf{bu\text{-}stat}(w)[u]$. By the Chernoff-Hoeffding bound we then get that, for any $t\geq 0$, $\Pr\left[|\mathsf{bu\text{-}stat}(w)[u]-\mathsf{b\text{-}stat}(v)[u]|\geq t\right]\leq 2e^{-2(\frac{n}{K})t^2}$.

We repeat the same argument for every u-coordinate, and using a union bound, we conclude that $\Pr\left[|\mathsf{bu\text{-}stat}(w)-\mathsf{b\text{-}stat}(v)|\geq|\Sigma|^k\times t\right]\leq|\Sigma|^k\times 2e^{-2(\frac{n}{K})t^2}$. If we set $t=\frac{\varepsilon}{2|\Sigma|^k}=\frac{1}{2k|\Sigma|^k}$, and use the definition of K, we conclude that with non-zero probability v is a word that satisfies the required property about the statistics, completing the proof. □

Combining the robustness of block statistics, the previous lemma, and the next lemma, which easily relates bu-stat to u-stat, we get our robustness lemma.

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

PROPOSITION 3.6. *Let $A \subseteq B$ be two finite multisets and let μ_A, μ_B be their respective uniform distributions. Then $|\mu_A - \mu_B| = 2\frac{|B|-|A|}{|B|}$.*

LEMMA 3.7. $|\mathsf{bu\text{-}stat}(w) - \mathsf{u\text{-}stat}(w)| \in O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4 n})$.

Proof. The proof consists of proving that the underlying distributions are at ℓ_1-distance at most $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4 n})$. Then the definitions of the vectors $\mathsf{u\text{-}stat}$ and $\mathsf{bu\text{-}stat}$ 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 $z \in \{1, 2, \ldots, n-k-1\}$. 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 $(i-1) = t \mod k$. 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 3.6, the ℓ_1-distance between the distributions is upper bounded by $2 \times (k-1)(\frac{n}{K} - 1) \times \frac{1}{n-k} \in O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4 n})$. □

LEMMA 3.8 (Robustness). *For any large enough $n \in \Omega(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^5})$, if $\mathsf{dist}(w, w') \geq 5\varepsilon n$ then $|\mathsf{u\text{-}stat}_k(w) - \mathsf{u\text{-}stat}_k(w')| \geq 6.5\varepsilon$.*

Proof. We assume that $n/(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^5})$ is large enough so that the $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ of Lemma 3.5 is upper bounded by $\frac{\varepsilon n}{16}$, and the $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4 n})$ of Lemma 3.7 is upper bounded by $\frac{\varepsilon}{8}$.

Using Lemmas 3.5 and 3.7, 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 $|\mathsf{u\text{-}stat}(w) - \mathsf{b\text{-}stat}(v)| \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{8}$ and $|\mathsf{u\text{-}stat}(w') - \mathsf{b\text{-}stat}(v')| \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{8}$.

From the hypothesis on w and w', and using the triangle inequality on dist, we obtain that $\mathsf{dist}(v, v') \geq 5\varepsilon n - \frac{\varepsilon n}{8}$. Therefore, using Lemma 3.3, we get that $|\mathsf{b\text{-}stat}(v) - \mathsf{b\text{-}stat}(v')| \geq 8\varepsilon - \frac{\varepsilon}{4}$, which implies from the construction of v, v' that $|\mathsf{u\text{-}stat}(w) - \mathsf{u\text{-}stat}(w')| \geq 8\varepsilon - \frac{\varepsilon}{4} - 2(\frac{\varepsilon}{2} + \frac{\varepsilon}{8}) = 6.5\varepsilon$. □

Using the Tolerance 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 $\widehat{\mathsf{u\text{-}stat}}_N(w)$ and $\widehat{\mathsf{u\text{-}stat}}_N(w')$ using the same N uniformly random indices in $\{1, \ldots, n-k+1\}$
Accept if $|\widehat{\mathsf{u\text{-}stat}}_N(w) - \widehat{\mathsf{u\text{-}stat}}_N(w')| \leq 6.25\varepsilon$
Reject otherwise

THEOREM 3.9. *For any $\varepsilon > 0$, and two words w, w' of the same size of order $\Omega(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^5})$, the algorithm **Uniform Tester(w, w', ε)**:*
(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 $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$.

Proof. If $w = w'$, then $\widehat{\mathsf{u\text{-}stat}}_N(w) = \widehat{\mathsf{u\text{-}stat}}_N(w')$ because we used the same indices to calculate both, hence (1). If w and w' are ε^2-close, then $|\mathsf{u\text{-}stat}(w) - \mathsf{u\text{-}stat}(w')| \leq 6.1\varepsilon$ by Lemma 3.4, and **Uniform Tester(w, w', ε)** accepts with probability at

least 2/3 using Corollary 3.2, hence (2). Finally, if w and w' are 5ε-far, then $|\mathsf{u\text{-}stat}(w) - \mathsf{u\text{-}stat}(w')| \geq 6.5\varepsilon$ by Lemma 3.8, and **Uniform Tester**(w, w', ε) rejects with probability at least 2/3 using Corollary 3.2, hence (3). □

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

COROLLARY 3.10. *There exists an $(\varepsilon^2, 5\varepsilon)$-approximation algorithm for computing the normalized distance $\varepsilon = \mathsf{dist}(w, w')/|w|$ between two words w, w' of the same size in $\Omega(\frac{\ln(|\Sigma|/\varepsilon)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$, with query and time complexities in $O(\frac{(\ln(|\Sigma|/\varepsilon))|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$.*

4. Languages. We want to use the notion of block statistics in order to efficiently characterize a language. We choose this statistics vector for the sake of clarity of the explanation since it is the simplest to manipulate. This work can be extended to the uniform statistic, leading to tolerant testers following the more complex approach in Subsection 4.5, which are more important.

Using the previous section, we can embed a word w into its block statistics $\mathsf{b\text{-}stat}(w) \in \mathbb{R}^{|\Sigma|^{1/\varepsilon}}$. This characterization is approximately one-to-one (*i.e.* words far from each other are embedded into far vectors) by Lemma 3.3 if the size of the words is fixed.

This directly implies the following proposition which states the existence of a tester for any computable language. However, this proposition does not upper bound the required time complexity for constructing such a tester in a preprocessing stage. The rest of the paper is devoted to studying time bounds for specific classes of languages.

PROPOSITION 4.1. *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 every word u of length n in L, and write down $\mathsf{b\text{-}stat}_k(u)$. For the input word w we can then $\frac{1}{2}\varepsilon$-approximate $\mathsf{b\text{-}stat}(w)$ by a vector $\widehat{\mathsf{b\text{-}stat}}(w)$ using Lemma 3.1, and then check whether there is any word u as above for which $|\mathsf{b\text{-}stat}(u) - \widehat{\mathsf{b\text{-}stat}}(w)| \leq \frac{1}{2}\varepsilon$. □

The above proposition is not only time inefficient, but it also does not allow to move the computation to a preprocessing stage. The reason is that the block statistics does not characterize words of different lengths, as $\mathsf{b\text{-}stat}(w_0) = \mathsf{b\text{-}stat}(w_0^t)$ 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 $\mathsf{b\text{-}stat}(w)$ of all words $w \in L$ 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 the "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 5.2).

4.1. Geometrical Embedding of Regular Languages. We fix an automaton A (possibly a non-deterministic one) over the alphabet Σ with a set of states Q of size m, that recognizes the regular language L. We assume that A contains no ϵ-transitions (transitions that consume no input letters).

A *path of A for a word w* of size n is a sequence of states $\pi = (s_1, s_2, \ldots, s_n)$ such that s_1 is an initial state and (s_i, s_{i+1}) is a transition of A reading the symbol $w[i]$. We say that π is *accepting* when s_n is an accepting state. When we do not precise the word w, we will simply refer to an (accepting) path of A. The *state of A on π after reading u* is $s_{|u|}$, when u is a prefix of w.

Let q be a state in Q, and let v be a word over Σ. Then v is an *A-loop on q* if there exist two words u, w over Σ and an accepting path π of A for uvw, such that the state q of the automaton on π after reading u is identical to the state after reading u and v. If there exists a state q, such that v is an *A-loop on q*, we will simply talk about an A-loop.

Let k be a positive integer and set $\varepsilon = \frac{1}{k}$. Since our embedding will be defined using the block statistics, we will only consider words whose size is divisible by k. This restriction is acceptable as any word of length n of L, for n large enough, is close to such a word. In the general case, one can modify A such that A recognizes the language of the words of L whose last $(|w| - k\lfloor \frac{|w|}{k} \rfloor)$ letters have been deleted, preserving the size of A up to an additive constant in $O(k)$.

A natural modification of A consists of defining a new automaton whose alphabet is exactly the same as the support of the block statistics, namely the block alphabet Σ^k. Define A^k, the *k-th power of A*, as the automaton over Σ^k with a set of states Q such that the transitions of A^k are exactly all sequences of k consecutive transitions of A. There is a natural embedding between words accepted by A^k and words of length a multiple of k accepted by A.

Definition 4.2. *A finite set $\{v_1, v_2, \ldots, v_l\}$ of A^k-loops is* A^k-compatible *if all the loops can occur on the same accepting path π of A^k, where each loop v_i is a loop on some state of π.*

We will characterize L by the block statistics of its loops on the block alphabet. The geometric embedding of L is the union of convex hulls of every compatible set of loops.

Definition 4.3. *Let $\mathcal{H}$ be the union of* $\mathsf{Convex\text{-}Hull}(\mathsf{b\text{-}stat}(v_1), \mathsf{b\text{-}stat}(v_2), \ldots, \mathsf{b\text{-}stat}(v_t))$ *when $v_1, \ldots, v_t$ range over all sets of A^k-compatible loops, for every $t \geq 0$.*

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. Recall that $|w|_b$ denote the size of w in term of block letters, and that $w \equiv_k w'$ if w' can be obtained by a permutation of the block letters of w.

Proposition 4.4. *Let $w \in L$. Then $w \equiv_k uv_1v_2 \ldots v_l$, where $|u|_b < m$ and $|v_1|_b, \ldots, |v_l|_b \leq m$ and $\{v_1, v_2, \ldots, v_l\}$ is an A^k-compatible set of A^k-loops.*

Proof. Let π be an accepting path for w. The construction of the loops is perform inductively starting with an empty collection of loops. Using the pigeonhole principle, one can find in w a loop v on a state q of π, where $|v|_b \leq m$. We remove v from w and collect it in our collection of loops. Remove also from π the subpath corresponding to v. The argument is repeated until the remaining part of w has no more loops.

Let u be the resulting subword, let π_0 be the resulting subpath, and let $(v_i)_{1 \leq i \leq l}$ be the collected loops, for some l. From the pigeonhole principle, u has block-size less than m. By construction $u \in L$ since π_0 is an accepting path for u. Moreover, π_0 is the witness that proves the A^k-compatibility of the collected loops $(v_i)_{1 \leq i \leq l}$. Last we conclude by observing that $w \equiv_k uv_1v_2 \ldots v_l$. □

Let us now state Caratheodory's theorem.

THEOREM 4.5 (Caratheodory). *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 choice of these points from $p_1, \dots, p_N$.* A consequence together with Caratheodory's theorem is that one can equivalently define $\mathcal{H}$ with the loop sizes and the number of compatible loops being bounded. 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 as we explain later in Proposition 4.11.

PROPOSITION 4.6. *$\mathcal{H}$ equals the union of $\mathsf{Convex\text{-}Hull}(\mathsf{b\text{-}stat}(v_1), \mathsf{b\text{-}stat}(v_2), \dots, \mathsf{b\text{-}stat}(v_t))$ when $v_1, \dots, v_t$ range over A^k-compatible loops such that $|v_i|_b \leq m$ and $t = |\Sigma|^{1/\varepsilon} + 1$.*

Proof. The inclusion $\supseteq$ is straightforward.

For the $\subseteq$ inclusion, we use Caratheodory's theorem. Using the same argument as in the proof of Proposition 4.4, we can overcome the length constraint. Indeed, 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 that are also compatible with the other loops v_j. Repeating this argument inductively, we first prove that the resulting set is the same also when we upper bound the loop sizes by m. The number of possible loops is then finite. Then applying Caratheodory's theorem, we conclude the proof. □

Another consequence of this proposition is that if a word w belongs to L, then it has to satisfy approximately $\mathsf{b\text{-}stat}(w) \in \mathcal{H}$ (Lemma 4.7 below). This can be understood as an approximate Parikh classification of regular languages, whereas the original, more involved, Parikh characterization was for context-free languages [18]. The converse is proved in Theorem 4.9.

As an example, let $L = (0110)^*(11)^*$ (and A be its smallest automaton), and $k = 2$. The A^k-loops of L are $(0110)^l$ and $(11)^l$, for any l. These loops are A^k-compatible. Let $s_1 = \mathsf{b\text{-}stat}((0110)^l) = (0, 1/2, 1/2, 0)$, $s_2 = \mathsf{b\text{-}stat}((11)^l) = (0, 0, 0, 1)$. Then by definition, $\mathcal{H}_L = \mathsf{Convex\text{-}Hull}(s_1, s_2)$. Notice that $s_2' = \mathsf{u\text{-}stat}((11)^l) = (0, 0, 0, 1)$ is identical to s_2, but that $s_1' = \mathsf{u\text{-}stat}((0110)^l) = (\frac{l-1}{4l-1}, \frac{l}{4l-1}, \frac{l}{4l-1}, \frac{l}{4l-1})$ does not converge to s_1. Taking the limit of s_1' and s_2' with respect to $l \to \infty$ we get that $\mathsf{Convex\text{-}Hull}((1/4, 1/4, 1/4, 1/4), (0, 0, 0, 1))$ is the corresponding polytope for $\mathsf{u\text{-}stat}$. We will get back to the polytops for $\mathsf{u\text{-}stat}$ when we deal with tolerant testing.

LEMMA 4.7. *For every $w \in L$ there exists w', so that $0 \leq |w| - |w'| \leq \frac{m}{\varepsilon}$, $\mathsf{dist}(w, w') \leq \frac{m}{\varepsilon}$, $|\mathsf{b\text{-}stat}(w) - \mathsf{b\text{-}stat}(w')| \leq \frac{2m}{\varepsilon|w|}$, and $\mathsf{b\text{-}stat}(w') \in \mathcal{H}$.*

Proof. First, recall that the block statistics $\mathsf{b\text{-}stat}(w)$ is invariant under block letter permutations. Moreover, if w' is obtained from w by inserting or deleting one block letter then $|\mathsf{b\text{-}stat}(w) - \mathsf{b\text{-}stat}(w')| \leq \frac{2}{\varepsilon|w|}$.

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 $|\mathsf{b\text{-}stat}(w) - \mathsf{b\text{-}stat}(w')| \leq \frac{2m}{\varepsilon|w|}$. This concludes the proof since $\mathsf{b\text{-}stat}(w') \in \mathcal{H}$. □

LEMMA 4.8. *For every $X \in \mathcal{H}$ and every n there exists $w \in L$, such that $0 \leq |w| - n \leq (|\Sigma|^{1/\varepsilon} + 3)\frac{2m}{\varepsilon}$ and $|X - \mathsf{b\text{-}stat}(w)| \leq (|\Sigma|^{1/\varepsilon} + 2)\frac{3m}{\varepsilon n}$.*

Proof. Let $X \in \mathcal{H}$, that is $X = \sum_{i=1}^{l} \lambda_i \, \mathsf{b\text{-}stat}(v_i)$, where $l = |\Sigma|^k + 1$, $|v_i|_b \leq m$, $0 \leq \lambda_i \leq 1$, $\sum_i \lambda_i = 1$, and $(v_i)_i$ are A^k-compatible loops. Fix any integer n. We choose non-negative integers $(r_i)_{i=1,2,\dots,l}$ that respectively approximate $\lambda_i \frac{\varepsilon n}{|v_i|_b}$, that is satisfy $0 \leq |r_i - \lambda_i \frac{\varepsilon n}{|v_i|_b}| \leq 1$, and such that $0 \leq \sum_i r_i |v_i|_b - \varepsilon n \leq m$. 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 $|v_i|_b \leq m$: We let $j \geq 0$ be the minimum integer so that $\sum_{i=1}^{j} \lceil \lambda_i \frac{\varepsilon n}{|v_i|_b} \rceil |v_i|_b + \sum_{i=j+1}^{l} \lfloor \lambda_i \frac{\varepsilon n}{|v_i|_b} \rfloor |v_i|_b \geq 0$, and set $r_i = \lceil \lambda_i \frac{\varepsilon n}{|v_i|_b} \rceil$ for $i \leq j$ and $r_i = \lfloor \lambda_i \frac{\varepsilon n}{|v_i|_b} \rfloor$ for $i > j$.

Define the word $w' = v_1^{r_1} v_2^{r_2} \dots v_l^{r_l}$. Then its block length is close to εn: $0 \leq |w'|_b - \varepsilon n \leq m$. Moreover its block statistics satisfies

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

Using the A^k-compatibility of loops $(v_i)_i$, we can get a word of L from w' by inserting few block letters. More precisely, we can find words $u_1, u_2, \dots, u_{l+1}$ over A^k, an accepting path π of A^k for the concatenated word $u = u_1 u_2 \dots u_{l+1}$, and a permutation σ over $\{1, 2, \dots, l\}$, such that $v_{\sigma(i)}$ is an A^k-loop on q_i, where q_i is the state of A^k on π after reading $u_1 u_2 \dots u_i$, for $i = 1, 2, \dots, l$, We also assume without lost of generality that each subword u_i has size at most the number of states of A, that is $|u_i|_b \leq m$. Then $w = u_1 v_{\sigma(1)}^{r_{\sigma(1)}} u_2 v_{\sigma(2)}^{r_{\sigma(2)}} \dots u_l v_{\sigma(l)}^{r_{\sigma(l)}} u_{l+1} \in L$ by construction. Moreover $0 \leq |w|_b - |w'|_b \leq (l+1)m$, and $|\mathsf{b\text{-}stat}(w') - \mathsf{b\text{-}stat}(w)| \leq \frac{2(l+1)m}{\varepsilon n}$, so we conclude. □

THEOREM 4.9. *Let $w \in \Sigma^n$ and $X \in \mathcal{H}$ be such that $|\mathsf{b\text{-}stat}(w) - X| \leq \delta$. Then*

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

Proof. 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 divisible by k. From Lemma 4.8, there exists a word $w' \in L$, such that $0 \leq |w'| - n \leq (l+2)\frac{2m}{\varepsilon}$ and $|\mathsf{b\text{-}stat}(w') - X| \leq (l+1)\frac{3m}{\varepsilon n}$, where $l = 1 + |\Sigma|^k$. We again assume that k divides $|w'|$.

Assume that $|w| = |w'|$. Then, using Lemma 3.3, we get that $\mathsf{dist}(w, w') \leq (\frac{1}{2}(\delta + (l+1)\frac{3m}{\varepsilon n}) + \varepsilon)n$.

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. □

4.2. Automata Independency. We now show that the statistics of the A^k-loops basically only depend on L and k. The loops in different automata for the same language are related, using a definition that only depends on the language itself.

DEFINITION 4.10. *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.*

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

PROPOSITION 4.11. *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. For the second direction, consider the family of words $(uv^t w)_{t\geq 1}$ in L, and fix an accepting path π of A^k for $uv^m w$. By a counting argument, there exist $0 \leq t < t' \leq m$, such that the accepting path π reaches the same state after both $|uv^t|_b$ steps and $|uv^{t'}|_b$ steps. Hence $v^{t'-t}$ is indeed an A^k-loop. □

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 4.12. *A finite set of (L,k)-loops $\{v_1,\dots,v_l\}$ is* (L,k)-compatible *if there exists a permutation $\sigma : \{1,\dots,l\} \to \{1,\dots,l\}$, 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 4.13. *Every A^k-compatible set of A^k-loops is also an (L,k)-compatible set of (L,k)-loops. On the other hand, for an (L,k)-compatible set $\{v_1,\dots,v_l\}$ 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 an A^k-compatible set 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 is similar to the proof of Proposition 4.11. Again the first direction is clear. For the second direction, a counting argument is applied to every loop $v_{\sigma(i)}$. Consider the family of words $(u_0 v_{\sigma(1)}^{t_1} u_1 v_{\sigma(2)}^{t_2} u_2 \dots u_{l-1} v_{\sigma(l)}^{t_l} u_l)_{t_1,t_2,\dots,t_l\geq 1}$ in L, and fix an accepting path π for $u_0 v_{\sigma(1)}^m u_1 v_{\sigma(2)}^m u_2 \dots u_{l-1} v_{\sigma(l)}^m u_l$. Using the same counting argument as in the proof of Proposition 4.11, we find integers $0 \leq t_i < t_i' \leq m$ such that π reaches the same state after both reading $u_0 v_{\sigma(1)}^m u_1 v_{\sigma(2)}^m u_2 \dots u_{l-1} v_{\sigma(i)}^{t_i}$ and reading $u_0 v_{\sigma(1)}^m u_1 v_{\sigma(2)}^m u_2 \dots u_{l-1} v_{\sigma(i)}^{t_i'}$. Hence the loops $(v_{\sigma(i)}^{t_i'-t_i})_i$ are A^k-compatible. □

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. There are several obstacles. One of them is to find an appropriate representation of $\mathcal{H}$ for deciding membership and inclusion efficiently. Since we are only interested in an approximate version of these tasks, we define, for a regular language, a simpler approximate representation $\mathcal{H}_\varepsilon$ of $\mathcal{H}$. We will achieve this in constant space and polynomial time by discretization.

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 $\frac{\varepsilon}{|\Sigma|^k}$.

DEFINITION 4.14. *Let $\mathcal{H}_\varepsilon$ be the set of points of $\mathcal{G}_\varepsilon$ 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 $|\mathcal{H}_\varepsilon| = 2^{|\Sigma|^{O(1/\varepsilon)}}$. We then simply represent $\mathcal{H}_\varepsilon$ by the brute force enumeration of its elements.

Our goal is to prove that $\mathcal{H}_\varepsilon$ can be constructed in polynomial time in the automaton size m. For this we will construct in polynomial time a polynomial-size set H of tuples of vectors such that $\mathcal{H} = \bigcup_{S\in H} \mathsf{Convex\text{-}Hull}(S)$. One could try to enumerate all A^k-loops of size at most m over Σ^k, but this is not efficient enough due to the possible large number of loops, $O(|\Sigma|^{km})$.

Instead of that, we enumerate block statistics of compatible loops using a standard reduction to matrix multiplication over an appropriate algebra. The complexity of the construction is then just polynomial in the number of possible corresponding block statistics, which is $\binom{m+|\Sigma|^k}{|\Sigma|^k} = O(m^{|\Sigma|^k})$, since block statistics of a word v of size at most m over Σ^k basically correspond to a partition of a number bounded

by m into $|\Sigma|^k$ parts. We proceed recursively on the length t of paths between two possible states of A^k, for $t = 1, \ldots, m$. 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 $\mathcal{D}_t$ 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 $\{\overrightarrow{x}\} \odot_t \{\overrightarrow{y}\} = \{\frac{1}{t+1}\overrightarrow{x} + \frac{t}{t+1}\overrightarrow{y}\}$. If we denote by $\circ_t$ the matrix multiplication over the algebra $\mathcal{D}_t$, 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): $P_{t+1} = P_1 \circ_t P_t$.

LEMMA 4.15. *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 $\mathcal{H} = \bigcup_{S \in H} \mathsf{Convex\text{-}Hull}(S)$.*

Proof. We first compute as we explained above the matrices $(P_t)_{t=1,\ldots,m}$. 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 the algebras $\mathcal{D}_t$. Using Proposition 4.6, 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 $O(m^{|\Sigma|^k})$ block statistics are considered. □

We now state that the construction of $\mathcal{H}_\varepsilon$ in polynomial time in the automaton size m.

PROPOSITION 4.16. *Given A and ε, the set $\mathcal{H}_\varepsilon$ can be computed in time $m^{|\Sigma|^{O(1/\varepsilon)}}$.*

Proof. We can construct $\mathcal{H}_\varepsilon$ by brute force enumeration of the elements of $\mathcal{G}_\varepsilon$. First, construct the set of tuples of vectors H defined in Lemma 4.15. Consider one by one the points X of $\mathcal{G}_\varepsilon$. Then, decide if $X \in \mathcal{H}_\varepsilon$ by checking if X is at ℓ_1-distance at most ε from $\mathsf{Convex\text{-}Hull}(S)$, where S is any tuple of vectors from H. Since the size of $\mathcal{G}_\varepsilon$ is $2^{|\Sigma|^{O(1/\varepsilon)}}$ and the size of H is at most $m^{|\Sigma|^{O(1/\varepsilon)}}$, the running time of the above procedure is in $m^{|\Sigma|^{O(1/\varepsilon)}}$. □

4.4. Property and Equivalence Testers. THEOREM 4.17. *For every real $\varepsilon > 0$ and regular language L over a finite alphabet Σ, there exists an ε-tester T for L whose query complexity is in $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ and whose time complexity is in $2^{|\Sigma|^{O(1/\varepsilon)}}$. Moreover, given an automaton with m states which recognizes L, the tester T can be constructed in time $m^{|\Sigma|^{O(1/\varepsilon)}}$.*

Proof. We fix $\varepsilon > 0$, and an automaton A with m states that recognizes L. For simplicity, we construct a 3ε-tester T for L. An ε-tester can be deduced from this by replacing ε with $\varepsilon/3$.

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 T has a preprocessing step followed by the testing step itself. Given A and ε, the preprocessing step computes $\mathcal{H}_\varepsilon$ in time $m^{|\Sigma|^{O(1/\varepsilon)}}$ from Proposition 4.16. Now the testing part consists of computing an estimation $\widehat{\mathsf{b\text{-}stat}}_N(w)$ of $\mathsf{b\text{-}stat}(w)$ as in Corollary 3.2, 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 $\widehat{\mathsf{b\text{-}stat}}_N(w)$ is 2ε-close to $\mathcal{H}_\varepsilon$, then the tester accepts, and otherwise it rejects.

The time complexity of T is clear. Let us now study its correctness. By Corollary 3.2, $|\mathsf{u\text{-}stat}(u) - \widehat{\mathsf{u\text{-}stat}}_N(u)| \leq \varepsilon$ with probability at least $2/3$, and we assume that this is the case in the analysis of T.

First, if $w \in L$, then by Lemma 4.7 $\mathsf{b\text{-}stat}\, w$ is $\varepsilon/10$-close to $\mathcal{H}$, and hence $6\varepsilon/10$-close to $\mathcal{H}_\varepsilon$. Then $\widehat{\mathsf{b\text{-}stat}}_N(w)$ is 1.6ε-close to $\mathcal{H}_\varepsilon$. Therefore T accepts w with probability at least $2/3$.

Conversely, if w is 3ε-far from L, then by Theorem 4.9, $\mathsf{b\text{-}stat}(w)$ is 3.9ε-far from $\mathcal{H}$, and hence 3.4ε-far from $\mathcal{H}_\varepsilon$. Then $\widehat{\mathsf{b\text{-}stat}}_N(w)$ is then 2.4ε-far from $\mathcal{H}_\varepsilon$. Therefore T rejects w with probability at least $2/3$. □

THEOREM 4.18. *There exists a deterministic algorithm T such that, for every $\varepsilon > 0$ as input, $T(\varepsilon)$ is an ε-equivalence tester for automata over a finite alphabet Σ. Moreover the running time complexity of T is in $m^{|\Sigma|^{O(1/\varepsilon)}}$, where m is the input automata size.*

Proof. Fix $\varepsilon > 0$. We construct for simplicity a 2ε-equivalence tester, that could be transformed into an ε-equivalence tester by replacing ε with $\varepsilon/2$. Given two automata A and B, 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 respectively to the automata A and B. If they are equal, the tester accepts, and otherwise it rejects.

Let us study the correctness of T. Let L_A (resp. L_B) be the language recognized by A (resp. B). If $L_A = L_B$, then their respective sets $\mathcal{H}_A$ and $\mathcal{H}_B$ are equal by Proposition 4.13, and so their discrete approximations $\mathcal{H}_{A,\varepsilon}$ and $\mathcal{H}_{B,\varepsilon}$ are also identical. Therefore the algorithm accepts.

Assume now that A and B are not 2ε-equivalent. For instance assume that L_A is not 2ε-contained in L_B. Let (w_n) be an infinite sequence of words that are in L_A but are 2ε-far from L_B. We only consider just one word w_n such that $|w_n| = \Omega(\frac{m|\Sigma|^{1/\varepsilon}}{\varepsilon^2})$. Then, from Lemma 4.7, $\mathsf{b\text{-}stat}(w_n)$ is at most 0.25ε-far from $\mathcal{H}_A$ and at most 0.75ε-far from $\mathcal{H}_{A,\varepsilon}$. Now, by the contraposition of Theorem 4.9, since w_n is 2ε-far from L_B we get that $\mathsf{b\text{-}stat}(w_n)$ is at ℓ_1-distance at least $(2 - 0.25)\varepsilon$ from $\mathcal{H}_B$ and at least 1.25ε away from $\mathcal{H}_{B,\varepsilon}$. Therefore $\mathcal{H}_{A,\varepsilon}$ and $\mathcal{H}_{B,\varepsilon}$ are not identical and the algorithm rejects. □

4.5. Tolerant Testing. The tolerance and robustness of the uniform statistic (Lemma 3.4 and Lemma 3.8) allow us to 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". Recall that $\mathsf{b\text{-}stat}$ is not tolerant, but $\mathsf{u\text{-}stat}$ is.

We construct an automaton that recognizes a word w over Σ^k if and only if it consists of all contiguous 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. Let $w = \mathsf{Shingle}(u)$, where $\mathsf{Shingle}(u)$ is the word where all contiguous subwords of u of size k are concatenated in order. For example if $u = 00110$ and $k = 2$, then $w = \mathsf{Shingle}(u) = 00011110$. Notice that $\mathsf{b\text{-}stat}(w) = \mathsf{u\text{-}stat}(u)$.

LEMMA 4.19. *Given an automaton A with m states over Σ, it is possible to construct an automaton A' with $l = m \times |\Sigma|^{k-1}$ states over Σ^k in time polynomial in l, where A' accepts a word w if and only if there exists a word u accepted by A so that $w = \mathsf{Shingle}(u)$.*

Proof. 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 $a_1, \ldots, a_{k-1}$ the automaton has a transition to $a_2, \ldots, a_{k-1}, b$ for every $b \in \Sigma$, which is labeled by $a_1, \ldots, a_{k-1}, b$.

The automaton A' consists of running in parallel the original automaton A and the automaton M, so in effect the state set of A' is the product of the state spaces of M and A'. The initial states of A' are the coupling of any state $a_1, \ldots, a_{k-1}$ of M with any state of A reachable from one of its original initial states by a path reading the letter sequence $a_1, \ldots, a_{k-1}$ from the input. The accepting states of A' are all couplings of a state of M with an accepting state of A.

For the transition function, there is a transition from a state s_1 of A coupled with a state $a_1, \ldots, a_{k-1}$ of M, to a state s_2 of A coupled with a state $a_2, \ldots, a_{k-1}, b$ of M, labeled by $a_1, \ldots, a_{k-1}, b$, for every $a_1, \ldots, a_{k-1}$ and b where the original automaton A contained a transition from s_1 to s_2 labeled by b. Clearly A' is the required automaton. □

Let $\mathcal{H}_{A'}$ be the polytope associated with the automaton A', as in the construction of section 4.3. We can view this polytope as the polytope $\mathcal{H}$ for the $\mathsf{u\text{-}stat}$ statistic.

We can now present the tolerant membership and equivalence testers.

THEOREM 4.20. *For every real $\varepsilon > 0$ and regular language L over a finite alphabet Σ, there exists an $(\varepsilon^2, 7\varepsilon)$-tolerant tester T for L whose query complexity is in $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ and whose time complexity is in $2^{|\Sigma|^{O(1/\varepsilon)}}$. Moreover, given an automaton with m states which recognizes L, the tester T can be constructed in time $m^{|\Sigma|^{O(1/\varepsilon)}}$.*

Proof. For $\varepsilon > 0$ and an automaton A that recognizes L, the tolerant tester T for L is similar to the ε-tester of Theorem 4.17.

We assume that $|w|/(\frac{m|\Sigma|^{1/\varepsilon}}{\varepsilon^2})$ and $|w|/(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ are both large enough, as otherwise we just run the automaton on all words that are ε^2-close to the input word w.

The tester T estimates $\mathsf{u\text{-}stat}(u)$ with $\widehat{\mathsf{u\text{-}stat}}_N(u)$. If $\widehat{\mathsf{u\text{-}stat}}_N(u)$ is 7.75ε-close to $\mathcal{H}_{A',\varepsilon}$, the tester accepts, and otherwise it rejects. By Corollary 3.2, $|\mathsf{u\text{-}stat}(u) - \widehat{\mathsf{u\text{-}stat}}_N(u)| \leq \varepsilon$ with probability at least $2/3$, and we assume from now on that this is the case in the analysis of T.

If $\mathsf{dist}(u, L) \leq \varepsilon^2 n$, then there exists $u_0 \in L$ such that u_0 is ε^2-close to u. Therefore, by Lemma 3.4 $|\mathsf{u\text{-}stat}(u) - \mathsf{u\text{-}stat}(u_0)| \leq 6.1\varepsilon$, *i.e.* $|\mathsf{b\text{-}stat}(w) - \mathsf{b\text{-}stat}(w_0)| \leq 6.1\varepsilon$ for $w = \mathsf{Shingle}(u)$ and $w_0 = \mathsf{Shingle}(u_0)$. Moreover, by Lemma 4.7 $\mathsf{b\text{-}stat}(w)$ is $\varepsilon/10$-close to $\mathcal{H}_{A'}$, and hence 0.6ϵ-close to $\mathcal{H}_{A',\varepsilon}$. Thus $\widehat{\mathsf{u\text{-}stat}}_N(u)$ is 7.7ε-close to $\mathcal{H}_{A',\varepsilon}$.

For the second direction, we argue by contraposition: If $\widehat{\mathsf{u\text{-}stat}}_N(u)$ is 7.8ε-close to $\mathcal{H}_{A',\varepsilon}$, then $\mathsf{u\text{-}stat}(u)$ is 8.3ε-close to $\mathcal{H}_{A'}$, *i.e.* $\mathsf{b\text{-}stat}(w)$ is 7.8ε-close to $\mathcal{H}_{A'}$ for $w = \mathsf{Shingle}(u)$. Moreover, by Lemma 4.8, there exists $u_0 \in L$ such that $w_0 = \mathsf{Shingle}(u_0)$ satisfies $|\mathsf{b\text{-}stat}(w_0) - \mathsf{b\text{-}stat}(w)| \leq \varepsilon/10$ and $|w| \leq |w_0| \leq |w|(1 + \varepsilon^2/100)$. Therefore $|\mathsf{u\text{-}stat}(u_0) - \mathsf{u\text{-}stat}(u)| \leq \varepsilon/10$ and $|u| \leq |u_0| \leq |u|(1 + \varepsilon/100)$. By Lemma 3.8, if u and u_0 have same size then $\mathsf{dist}(u, u_0) \leq (8.4 \times 5/6.5)\varepsilon n \leq 6.9\varepsilon n$. In the general case, we first delete at most an $\varepsilon/100$ fraction of the letters from u_0, which induces an additional deviation in the uniform statistics that is upper bounded using Lemma 3.4 by $\varepsilon/50$. Hence u is 7ε-close to L. □

THEOREM 4.21. *There exists a deterministic algorithm T such that, for every $\varepsilon > 0$ as input, $T(\varepsilon)$ is an $(\varepsilon^2, O(\varepsilon))$-equivalence tester for automata over a finite alphabet Σ. Moreover the running time complexity of T is in $m^{|\Sigma|^{O(1/\varepsilon)}}$, where m is the input automata size.*

Proof. We use the same proof of Theorem 4.18, only instead of calculating the

geometric embedding of the given automata A and B, we calculate the geometric embedding of the shingle automata A' and B' derived from them for $k = O(1/\varepsilon)$, and accept if and only if these are $O(\varepsilon)$-close. The running time comes from the discretization as in Theorem 4.18. ∎

5. Generalizations.

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. 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 $\mathrm{Inf}_A(w)$ 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 $\mathrm{Inf}_A(w)$ contains an accepting state of A. We say that A *recognizes* the language of accepted infinite words. Such languages are called *ω-regular languages*.

We could easily extend Theorem 4.17 to lasso words as in [9], but we prefer to study in this section the general case of infinite words. Defining the distance over infinite words requires to normalize the edit distance with moves. For every integer n, we denote by $w_{|_n}$ the prefix of w of size n. The distance between two words w, w' is defined as the superior limit $\mathsf{dist}(w, w') = \overline{\lim}_{n\to\infty} \mathsf{dist}(w_{|_n}, w'_{|_n})/n$. Then the distance of a word w to the language L is defined as $\mathsf{dist}(w, L) = \inf_{v\in L} \mathsf{dist}(w, v)$. Two words w, w' are *ε-close* if their distance is at most ε.

The *block statistics* $\mathsf{b\text{-}stat}(w)$ of w is the convex closure of the set of accumulation points of the sequence $(\mathsf{b\text{-}stat}(w_{|_n}))_n$. It is either a one element set or an infinite compact set. If $w = uv^\omega$ for two finite words u, v where the size of u is a multiple of k, then $\mathsf{b\text{-}stat}(w) = \mathsf{b\text{-}stat}(v^\omega)$ is the one element set $\{\mathsf{b\text{-}stat}(v')\}$ where $v' = v^i$ is such that its length is a multiple of k.

We now state some simple facts on infinite words that derive from concatenations of finite words of bounded size.

PROPOSITION 5.1. *Let $(w_i)_i$ be a sequence of words on Σ of globally bounded size, and $(\delta_i)_i$ be a sequence of reals converging to zero, such that $\mathsf{b\text{-}stat}(w_1 w_2 \cdots w_i)$ is in a δ_i-neighborhood of a compact convex set C for every i. Let w be the infinite word $w_1 w_2 \cdots$. Then w satisfies $\mathsf{b\text{-}stat}(w) \subseteq C$.*

Proof. Let c be the global bound on the size of the words, and let $\varepsilon > 0$ be any real number. Also, for every l denote by i_l the largest integer such that $w_1 w_2 \cdots w_{i_l}$ does not contain more than l letters. Thus $w_{|_l}$ can be constructed from $w_1 w_2 \cdots w_{i_l}$ by appending less than c letters.

Now assume that $l_1, l_2, \ldots$ are such that $\mathsf{b\text{-}stat}(w_{|_{l_i}})$ converges. Let j_1 be such that all δ_j for $j \geq j_1$ are smaller than $\varepsilon/2$. Let j_2 be such that for all $j \geq j_2$, both $l_j \geq 4ck/\varepsilon$ and $i_{l_j} \geq j_1$. Now for every word $w_{|_{l_j}}$ for $j \geq j_2$, its statistic is not more than $\varepsilon/2$-away from that of $w_1 w_2 \cdots w_{i_{l_j}}$, whose statistic is in turn not more than $\varepsilon/2$ away from a point of C. Since C is compact and the existence of an appropriate j_2 was shown for every $\varepsilon > 0$, this means that the convergence point of the statistic $\mathsf{b\text{-}stat}(w_{|_{l_i}})$ indeed lies in C. ∎

PROPOSITION 5.2. *Let $(v_i)_i$ and $(w_i)_i$ be two sequences of words on Σ of bounded size, such that $v_1 v_2 \cdots v_i$ is ε-close to $w_1 w_2 \cdots w_i$ for every i. Let v and w be respectively the infinite words $v_1 v_2 \cdots$ and $w_1 w_2 \cdots$. Then v is 2ε-close to w.*

Proof. Let c be the global bound on the size of the words, and let $\varepsilon' > 2\varepsilon$ be a real number. Also, denote by i_l the largest integer such that both $w_1 w_2 \cdots w_{i_l}$ and

$v_1 v_2 \cdots v_{i_l}$ have no more than l characters. Note that either $w_{|_l}$ can be constructed from $w_1 w_2 \cdots w_{i_l}$ by appending less than c characters, or the same holds for $v_{|_l}$ and $v_1 v_2 \cdots v_{i_l}$ (or both hold).

Now let l_1 be such that for every $l \geq l_1$ we have $l \geq 2c/(\varepsilon' - 2\varepsilon)$. For any $l \geq l_1$, assume without loss of generality that $w_{|_l}$ can be constructed from $w_1 w_2 \cdots w_{i_l}$ by appending less than c characters. Also $v_1 v_2 \cdots v_{i_l}$ can be constructed from $w_1 w_2 \cdots w_{i_l}$ by at most εl operations, and in particular its length is more than $(1-\varepsilon)l - c$ (since it cannot be smaller than that of $w_1 w_2 \cdots w_{i_l}$ by more than εl). Thus $v_{|_l}$ can be constructed from $v_1 v_2 \cdots v_{i_l}$ by appending less than $\varepsilon l + c$ letters. Summing up we have less than a total of $2(\varepsilon l + c)$ operations to get from $w_{|_l}$ to $v_{|_l}$, which total not more than $\varepsilon' l$. Having proved the existence of l_1 for every $\varepsilon' > 2\varepsilon$, we are done. □

We fix a Büchi automaton A with m states that recognizes a language L. By adapting our geometric embedding for this distance, an equivalence tester for Büchi automata follows similarly to the one constructed for regular languages over finite words. In this tester, we modify the definition of $\mathcal{H}$ (Definition 4.3), by restricting ourselves to the loops of (strongly) connected components of the accepting states of A^k, and by distinguishing the convex hulls in $\mathcal{H}$. Formally, $\mathcal{H}$ is a finite *family* of convex hulls.

Definition 5.3. *For every reachable strongly connected component C which contains an accepting state q of A^k, let H_C be the convex hull of the vector set $\{\mathsf{b\text{-}stat}(w) : w$ is a finite A^k-loop of $C\}$. We denote by $\bar{\mathcal{H}}$ the family $\{H_C : C \in \mathcal{C}\}$, where $\mathcal{C}$ is the set of all connected components having an accepting state of A^k that are reachable from an initial state, and after deleting sets which are contained in other sets (i.e. if $H_C \subseteq H_D$ then we only keep H_D).*

Note that the size of the family $\bar{\mathcal{H}}$ is at most the size of the automaton A, and that we can restrict ourselves to the A^k-loops whose block sizes are at most the size of A. Moreover, elements in $\bar{\mathcal{H}}$ are not counted with multiplicity since these will not be useful for our purpose. As in Section 4.2, it can be observed that the construction is independent from the automaton representation, but this requires proving some more lemmas first. We define $\bar{\mathcal{H}}_\varepsilon$ similarly to Definition 4.14.

Definition 5.4. *For every family $\bar{\mathcal{H}} = \{H_C : C \in \mathcal{C}\}$, we define $\bar{\mathcal{H}}_\varepsilon = \{H_{C,\varepsilon} : C \in \mathcal{C}\}$, where $H_{C,\varepsilon}$ is the set of points of $\mathcal{G}_\varepsilon$ that are at distance at most $\frac{\varepsilon}{2}$ from H_C (for the ℓ_1-distance).*

Then $\bar{\mathcal{H}}_\varepsilon$ can again be computed efficiently as in Proposition 4.16.

Proposition 5.5. *Let $\varepsilon > 0$. Given $\varepsilon > 0$ and a Büchi automaton A of size m, the set $\bar{\mathcal{H}}_\varepsilon$ can be computed in time $m^{|\Sigma|^{O(1/\varepsilon)}}$.*

Proof. As in the proof of Proposition 4.16, we construct $\bar{\mathcal{H}}_\varepsilon$ using Lemma 4.15. The only difference is that the construction is done separately for every connected component C of $\mathcal{C}$. Since the number of connected components of A^k is at most m, the construction time has the claimed order. □

We now enumerate intermediate results from which the Equivalence Tester (Theorem 5.11) will follow. We first state some lemmas about the classification by $\bar{\mathcal{H}}$. Due to the nature of infinite languages, the one about the robustness of $\bar{\mathcal{H}}$ is more involved and approximate.

Lemma 5.6. *Let $w \in L$, and n be a number divisible by k. Then there exists $H_C \in \bar{\mathcal{H}}$, a finite word u_0 and two infinite sequences of finite words $v_1, v_2, \ldots$ and $u_1, u_2, \ldots$ such that*

1. *$w = u_0 u_1 v_1 u_2 v_2 \cdots$;*
2. *$v_1, v_2, \ldots$ are A^k-loops within the same connected component C of A^k;*

3. for every $i > 0$ the word sizes satisfy $n \leq k|u_i| \leq |v_i| \leq n(k+1)^{m+1}$;

4. $\mathsf{dist}(w, v_1 v_2 \cdots) \leq 2\varepsilon$ and $\mathsf{b\text{-}stat}(v_1 v_2 \cdots) \subseteq H_C$.

Proof. Consider an (infinite) accepting path π in A^k for w (as a block word). Since this is an infinite walk on the finite directed graph of the automaton A^k, there exists a connected component C such that all but at most a finite number of the states in π are inside C. Let u_0 be the subword of w corresponding to the positions in which π has not yet converged inside C. Let w_1 be such that $w = u_0 w_1$.

We now construct u_i, v_i by induction. Assume that $u_1, \ldots, u_{i-1}$ and $v_1, \ldots, v_{i-1}$ were already constructed, and set w_i such that $w = u_0 u_1 v_1 u_2 v_2 \cdots u_{i-1} v_{i-1} w_i$. Let also π_i denote the subsequence of π corresponding to the positions in w_i. We now look at the automaton states corresponding to $\pi_i[\frac{n}{k}(k+1)], \pi_i[\frac{n}{k}(k+1)^2], \ldots, \pi_i[\frac{n}{k}(k+1)^{m+1}]$. By the pigeon hole principle there exist $1 \leq j_1 < j_2 \leq m+1$ such that $\pi_i[\frac{n}{k}(k+1)^{j_1}] = \pi_i[\frac{n}{k}(k+1)^{j_2}]$. We now set the subwords on Σ^k $u_i = w_i[1 \ldots \frac{n}{k}(k+1)^{j_1} - 1]_b$ and $v_i = w_i[\frac{n}{k}(k+1)^{j_1} \ldots \frac{n}{k}(k+1)^{j_2} - 1]_b$. From this construction the first three items in the assertion of the lemma hold.

Using Proposition 5.1, $\mathsf{b\text{-}stat}(v_1 v_2 \cdots) \subseteq H_C$ follows from the fact that $\mathsf{b\text{-}stat}(v_i) \in H_C$ for every i (because v_i is a loop in C) together with the existence of a constant upper bound on the length of all v_i. Using Proposition 5.2, $\mathsf{dist}(w, v_1 v_2 \cdots) \leq 2\varepsilon$ follows from $k|u_i| \leq |v_i|$ (and so $u_1 v_1 u_2 v_2 \cdots u_i v_i$ is ε-close to $v_1 v_2 \cdots v_i$ for every i) together with the constant upper bound on all $|v_i|$ (note also that clearly $\mathsf{dist}(w_1, u_0 w_1) = 0$). This concludes the proof of the fourth and remaining assertion of the lemma. □

We also need to know that in fact there are words "filling" the whole of $\bar{\mathcal{H}}$.

LEMMA 5.7. *If A is a Büchi automaton that recognizes L, and $\bar{\mathcal{H}}$ is its corresponding family (as defined in definition 5.3), then for every $H_C \in \bar{\mathcal{H}}$ there is a word $w \in L$ satisfying $\mathsf{b\text{-}stat}(w) = H_C$.*

Proof. Let $v_1, \ldots, v_s$ be the words corresponding to all simple loops in the component C of A^k, and let u be a word for which there is a path from an initial state of A^k to the first state in the loops corresponding to v_1. Furthermore, let u_i be a word for which there is a path from the first state of the loop of v_i to the first state of the loop of v_{i+1}, where u_s leads on a path from the first state of the loop of v_s to that of v_1.

Finally, set $w_i = (v_1)^{(si+1)!} u_1 (v_2)^{(si+2)!} u_2 \cdots (v_s)^{(si+s)!} u_s$. Then the word $u w_1 w_2 \cdots$ satisfies the requirement. □

Now we can prove that $\bar{\mathcal{H}}$ depends only on the language L and k (and not on the specific automaton A).

LEMMA 5.8. *If A, B are two Büchi automata that recognize the same language, then $\bar{\mathcal{H}}_A = \bar{\mathcal{H}}_B$.*

Proof. To show $\bar{\mathcal{H}}_A \subseteq \bar{\mathcal{H}}_B$, let $H \in \bar{\mathcal{H}}_A$. Using first Lemma 5.7, and then using the resulting word with Lemma 5.6 for $\bar{\mathcal{H}}_B$, we get that there is $H' \in \bar{\mathcal{H}}_B$ that contains H. Going the other way around, there is $H'' \in \bar{\mathcal{H}}_A$ that contains H' and hence also H. Since we removed containments from $\bar{\mathcal{H}}_A$ it follows that $H'' = H$ and hence $H = H'$. Showing that $\bar{\mathcal{H}}_B \subseteq \bar{\mathcal{H}}_A$ is done similarly. □

The soundness of $\bar{\mathcal{H}}_{A,\varepsilon}$ will follow from the following straightforward variant on the finite case.

LEMMA 5.9. *Let $H_{C,\varepsilon} \in \bar{\mathcal{H}}_{A,\varepsilon}$ and $l \geq n = 10\lceil(|\Sigma|^{1/\varepsilon} + 2)\frac{3m^2}{\varepsilon^2}\rceil$. For every X in $H_{C,\varepsilon}$, there is a finite A^k-loop v in C such that $0 \leq |v| - l \leq \varepsilon l$ and $|X - \mathsf{b\text{-}stat}(v)| \leq \varepsilon$.*

Proof. Consider the language L_C of the finite A^k-loops in C. This language is clearly recognizable by an automaton of size at most $m^2 + 1$. Applying Lemma 4.8

for L_C for a vector $X' \in H_C$ which is $\varepsilon/2$-close to $X \in H_{C,\varepsilon}$, we get a finite A^k-loop v in C satisfying the assertions of the lemma. □

Based on the above lemmas, we can state that $\bar{\mathcal{H}}$ is a robust characterization of L.

LEMMA 5.10. *If A, B are two Büchi automata such that $\bar{\mathcal{H}}_{A,\varepsilon} = \bar{\mathcal{H}}_{B,\varepsilon}$, then $A \equiv_{8\varepsilon} B$.*

Proof. Set $n = \max\{\lceil (|\Sigma|^{1/\varepsilon} + 2)\frac{3m^2}{\varepsilon^2} \rceil, 4km\}$, where m is the maximum of the sizes of A and B. In fact, we will prove that *every* word $w \in L_A$ (not just most words) is 8ε-close to L_B. The other direction is identical, and therefore we omit it.

Let $w \in L_A$. Applying Lemma 5.6 (with the above n) we get a word w' that is 2ε-close to w, and satisfying $\mathsf{b\text{-}stat}(w') = \mathsf{b\text{-}stat}(v_1 v_2 \cdots) \subseteq H_C$. Also, all the loops satisfy $|v_i| \geq n$, while still being of bounded size.

By hypothesis, there exists $H_D \in \bar{\mathcal{H}}_B$ such that $H_{C,\varepsilon} = H_{D,\varepsilon}$. Therefore, by Lemma 5.9, for every A^k-loop v_i in C there exists a B^k-loop z_i in D, such that $|\mathsf{b\text{-}stat}(v) - \mathsf{b\text{-}stat}(z)| \leq \varepsilon$ and $0 \leq |z| - n \leq \varepsilon n$.

Now we construct $w'' \in L_B$ together with an accepting path π of w'' in B. First, we let y_0 be any finite word with a finite path π_0 from an initial state of B to some state in its connected component D. Now, for every i, assuming that we already constructed $y_1, \ldots, y_{i-1}$ and a path π_{i-1} for $y_0 y_1 z_2 y_2 z_2 \cdots y_{i-1} z_{i-1}$, we construct y_i and π_i as follows. We look at the last state q reached by π_{i-1}, and let y_i be any word of size at most $2m$ for which there is a path in B from q to the first state in a loop corresponding to z_i that also goes through some accepting state in D. We then let π_i be the continuation of π_{i-1} with the path for y_i and the loop for z_i.

Finally, we set $w'' = y_0 y_1 z_1 y_2 z_2 \cdots$ and π to be the limit of π_i. Clearly π is an accepting path in B for w''. We finally show that w'' is 8ε-close to w by showing that $\mathsf{dist}(w', w'') \leq 6\varepsilon$ and then using the triangle inequality. Now because of the nature of block words and Lemma 5.9, we have $\mathsf{dist}(v_i, z_i) \leq 2\varepsilon$ for every $i > 0$. Hence, for every i we have $\mathsf{dist}(v_1 v_2 \cdots v_i, y_1 z_1 y_2 z_2 \cdots y_i z_i) \leq 3\varepsilon$ (notice that in particular $k|y_i| \leq |z_i|$). Since the sizes of v_i, y_i and z_i all have a global bound, this implies that the distance of the infinite words w' and w'' is not more than 6ε using Proposition 5.2. This concludes the proof. □

Theorem 4.18 is then valid for nondeterministic Büchi automata, with $\bar{\mathcal{H}}$ taking the place of $\mathcal{H}$. Its proof is a direct consequence of the above lemmas.

THEOREM 5.11. *There exists a deterministic algorithm T such that, for every $\varepsilon > 0$ as input, $T(\varepsilon)$ is an ε-equivalence tester for Büchi automata over a finite alphabet Σ. Moreover the running time complexity of T is in $m^{|\Sigma|^{O(1/\varepsilon)}}$, where m is the input automata size.*

Proof. Fix $\varepsilon > 0$. For simplicity an 8ε-tester is constructed. The tester is the same as the one of Theorem 4.18, only here we check the two families of sets $\bar{\mathcal{H}}_{A,\varepsilon}$ and $\bar{\mathcal{H}}_{B,\varepsilon}$ for equality.

Whenever the given automata A and B recognize the same language, the tester accepts since we have $\bar{\mathcal{H}}_A = \bar{\mathcal{H}}_B$ by Proposition 5.8, and therefore $\bar{\mathcal{H}}^A_\varepsilon = \bar{\mathcal{H}}^B_\varepsilon$. On the other hand, if A and B are not even 8ε-equivalent, then the contraposition of Lemma 5.10 shows that $\bar{\mathcal{H}}^A_\varepsilon \neq \bar{\mathcal{H}}^B_\varepsilon$, and therefore the tester rejects. □

This result has a direct application for the Logic LTL, Linear Time Logic. A classical construction associates a Büchi automaton to an LTL formula, whose size can be exponential in the size of the formula. When exact model checking is infeasible, we can still use our approximate equivalence tester with an arbitrarily small parameter ε.

5.2. Context-Free Languages. We construct an exponential time equivalence tester for context-free languages, given by their grammar or by their push-down automaton (the two representations are polynomially equivalent so we can switch back and forth between them as convenient).

THEOREM 5.12. *There exists a deterministic algorithm T such that, for every $\varepsilon > 0$ as input, $T(\varepsilon)$ is an ε-equivalence tester for context-free grammars over a finite alphabet Σ. Moreover, the running time complexity of T is exponential in $m^{|\Sigma|^{O(1/\varepsilon)}}$, where m is the input grammars' representation size.*

By comparison, the exact decision problem of whether two context-free grammars define the same language is not decidable. By the same methods, we can also construct an exponential (preprocessing) time property tester that makes a polynomial number of queries (note that over the Hamming distance norm, context-free languages are not necessarily testable).

The proof uses the original Parikh theorem about the spectra of context-free languages, that provides a formula defining a semi-linear set on the letter counts of all possible words. The exponential blow-up in the grammar size comes from this step. From the spectrum one can calculate the set $\mathcal{H}$ that approximates the block-statistics of all large enough words, and then construct an appropriate $\mathcal{H}_\varepsilon$. Equivalence testability for context-free grammars follows from comparing the constructed sets as in Theorem 4.18 .

Before proceeding we note the following corollary for regular expressions with squaring.

COROLLARY 5.13. *There exists a deterministic algorithm T such that, for every $\varepsilon > 0$ as input, $T(\varepsilon)$ is an ε-equivalence tester for regular expressions with squaring signs over a finite alphabet Σ whose running time is exponential in $m^{|\Sigma|^{O(1/\varepsilon)}}$, where m is the input expression size.*

Proof. We claim that regular expressions with squaring can be converted to equivalent pushdown automata with a polynomially related size, over which we can use the algorithm provided by Theorem 5.12. This conversion in fact follows the standard conversion of a regular expression without squaring to a (non-pushdown) nondeterministic automaton. The first difference is that before we read the input we push down a unique symbol s down the stack (using an ϵ-transition), and in the end we only accept if this symbol is indeed at the top of the stack. Apart from that, we will only have stack operations while dealing with square signs, while all other regular expression operators are treated exactly as in the standard conversion of a regular expression.

The i-th square sign is converted to a corresponding state s_i that pushes down the stack a specific symbol, with respect to i, unless this symbol was already at the top of the stack. After pushing down the symbol, s_i makes an ϵ-transition to the state corresponding to the beginning of the subexpression that is squared. When s_i is reached with its unique symbol already at the top of the stack (which happens only when it is reached again after repeating the part of the automaton corresponding to the squared expression), it pops this symbol from the stack and makes an ϵ-transition to the state corresponding to the expression element following the square sign.

Finally, it is not hard to see that in an accepting run of the automaton, whenever a state related to a beginning of a squared subexpression is reached, the special state s_i causes the automaton to go through the portion related to that subexpression exactly twice before moving to the state corresponding to its end. Hence, the special states have the same effect as the square signs in the translated expression. □

Although regular expressions with squaring still recognize only regular languages, their (exact) equivalence problem is EXPSPACE-complete by [16], so the exponential time approximation algorithm given here can be considered as a slight improvement over the exact decision setting. We now turn to the proof of Theorem 5.12.

We also note that a property tester can be constructed using the same groundwork. Recall that under the usual Hamming distance, not all context free languages are testable. The following tester will also serve us later when considering languages of trees.

THEOREM 5.14. *For every real $\varepsilon > 0$ and context free language L over a finite alphabet Σ, there exists an ε-tester T for L whose query complexity is in $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ and whose time complexity is in $2^{|\Sigma|^{O(1/\varepsilon)}}$. Moreover, given a grammer of size m which recognizes L, the tester T can be constructed in time exponential in $m^{|\Sigma|^{O(1/\varepsilon)}}$.*

To construct an equivalence tester and a property tester we first lay some groundwork, starting with normal forms.

DEFINITION 5.15. *A context free grammar with a start symbol S is said to be* ϵ-free *if apart from possibly containing the rule $S \to \epsilon$ (here ϵ denotes the empty word), no production rule has the word ϵ or any word containing S in its right hand side.*

A context free grammar is said to be in Chomsky Normal Form *if apart from possibly containing $S \to \epsilon$, all its production rules are each either into a single terminal or into a word of exactly two nonterminals.*

A context free grammar is said to be in Greibach Normal Form *if apart from possibly containing $S \to \epsilon$, all its production rules are into words consisting of one terminal followed by any number of nonterminals.*

It is well known that any context free gammar is equivalent to one in Chomsky Normal Form of size polynomial in the size of the original grammar, and that this transformation can be done in polynomial time (the standard transformation generally works, only one needs to take care to eliminate productions to a single variable or to the empty word only after eliminating productions to words of size larger than 2). A similar result with respect to Greibach Normal Form was proven more recently in [7].

LEMMA 5.16 ([7]). *Every context-free grammar is equivalent to a grammar in Greibach Normal Form, whose representation size is polynomial in that of the original grammar. Furthermore, there exists an algorithm for this transformation taking polynomial time.*

The main theorem of [7] is formulated only for ϵ-free grammars, but a grammar can be first made ϵ-free by transforming it to Chomsky Normal Form. The computation time bound is not explicitly mentioned in [7], but it follows immediately from the proof there.

It is also well known how to convert context free grammars to nondeterministic pushdown automata and the other way around with a polynomial blowup in the size. For the Greibach Normal Form, the following simple observation is important.

PROPOSITION 5.17. *The standard conversion procedure of a grammar in Greibach Normal Form into a nondeterministic pushdown automaton yields an automaton without ϵ-transitions, i.e. a nondeterministic automaton for which every transition consumes exactly one letter from the input string.*

Using the above we can now efficiently move from the original grammar to a "block grammar" that approximates the block-statistics of the original language, similarly to the k-power construction that we used for regular languages.

LEMMA 5.18. *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 (where words whose length is not a multiple of k are truncated to the nearest multiple) is recognizable by G'.

Proof. We first use Lemma 5.16 to move from G to an equivalent grammar in Greibach Normal Form, and then use Proposition 5.17 to move to a pushdown automaton A with no ϵ-transitions. Then we take the k'th power of A: We use the same state space as A, but our transitions are those corresponding to sequences of k transitions over Σ. If some of the resulting transitions consume more than one letter off the stack, we add additional states and ϵ-transitions as necessary (we may also need to make some states accepting to account for possible truncations of words whose size is not a multiple of k). 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 steps causes no more than a polynomial blowup in the representation size if k is fixed, and so we are done. □

Lemma 5.18 implies that if we have a way of comparing the spectra of the block grammars corresponding to our two languages, that is comparing the possible block letter counts of the words in the corresponding languages, then we can construct an equivalence tester.

For this end we use the Parikh theorem [18], that provides a formula defining a semi-linear set on the letter counts of all possible words, which we call the *Parikh formula*. This formula is in fact a sequence of matrices $A_1, \ldots, A_r$ and vectors $b_1, \ldots, b_r$, where a vector w of nonnegative integers corresponds to the letter counts of a word in the language if and only if there exists an index i and a vector u of nonnegative integers such that $w = A_i u + b_i$.

It is not hard to see that from the matrices $A_1, \ldots, A_r$ appearing in the Parikh formula one can efficiently calculate the set $\mathcal{H}$ that approximates the block-statistics of all large enough words, and then construct an appropriate $\mathcal{H}_\varepsilon$. A look at the original proof in [18] already brings us half-way towards calculating these matrices.

LEMMA 5.19 ([18]). *The computation of the Parikh formula can be reduced (in time polynomial in the required data size) to the computation of the letter counts of all possible productions from a nonterminal character to a word which contains at most one nonterminal character, whose heights are at most quadratic in the grammar size (and so the word size is at most exponential in the grammar size).*

The exponential running time of our equivalence tester is a result of the above lemma requiring information concerning all letter counts of words of size exponential in the grammar size. Providing this data can be done with an overhead that is only polynomial in the maximum word size (where as before k and $|\Sigma|$ are fixed), using a successive relaxation method.

LEMMA 5.20. *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.*

Proof. 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). Note that this transformation preserves the original nonterminals (while it may 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 $\binom{l+h}{h} = l^{O(h)}$ where $h = |\Sigma|^k$ is the number of possible terminals, and so (for fixed Σ and k) it is polynomial in l. In the beginning, the list L_α 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 A the initial list L_A 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 "$A \to u$", u may either be a single terminal α or a sequence of two nonterminals BC. In the first case, we just add the vector assigned to L_α also to the list L_A maintained for A. In the second case, we look at the current lists for B and C, and for every pair of vectors $v \in L_B$ and $w \in L_C$ we add $w + v$ to L_A, but only if the weight of $w + v$ is no more than l. Note that this step takes time that is at most quadratic in the maximum representation size of a single 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 that can no longer be enlarged in the above manner. It is not hard (and left to the reader) to see that all through the algorithm, no list L_A will contain vectors that are not in fact vectors of letter counts of a word possibly produced from A. Also, every count of letters of a word of size up to l produced from A will eventually make it into L_A. Thus when the algorithm stops the lists indeed contain the correct output. □

With the above lemmas we can now finalize the proof of Theorem 5.12 as well as Theorem 5.14.

Proof. [Proof of Theorem 5.12 and Theorem 5.14] To construct an equivalence tester, for $k = \frac{1}{\varepsilon}$, using Lemma 5.18 we construct the corresponding grammars A' and B' over Σ^k. Using Lemma 5.20 in conjunction with Lemma 5.19, we calculate the Parikh formulae for the two grammars, from which we calculate $\mathcal{H}_{A,\varepsilon}$ and $\mathcal{H}_{B,\varepsilon}$. As before, we now accept if and only if these two sets are identical. The exponential time comes from the fact that Lemma 5.19 requires us to use an exponential l in the statement of Lemma 5.20.

To construct a property test for a grammer A, we first calculate $\mathcal{H}_{A,\varepsilon}$ as above, and then given a word w we approximate $\mathsf{b\text{-}stat}(w)$ using sampling and check it against $\mathcal{H}_{A,\varepsilon}$ as was done in the property tester for regular languages. □

6. Trees. The purpose of this section is to extend our results to Σ-trees, *i.e.* trees whose nodes have labels in some finite label alphabet Σ. This will be done by defining an efficient word encoding of trees. Word encodings of trees are well studied. A simple one is given by a Depth First Search from the root of the tree. Such a DFS encoding maps tree regular languages to context free languages. We cannot directly use such an encoding, since edit distance with moves on the resulting words does not translate well to the edit distance with moves on trees. We proceed in two steps: We first compress a tree T into another tree T_k where the number of 2-degree nodes has been reduced, and then linearize T_k to obtain a word T' with labels. Samples on T' can be directly obtained from samples in T.

We consider 2-ranked labeled ordered trees, but our results can be extended to any ranked trees. Recall that a 2-ranked tree is a tree with at most 2 successors, *i.e.*

every node has 0, 1 or 2 successors. The *size* of a tree is the number of its nodes, which we will denote by n. The *degree* of a node is the number of its successors. Let k be an integer and $\varepsilon = 1/k$.

We first define the *k-compression* of a tree T, denoted by T_k, which basically consists of removing every node whose subtree has size $< k$, and encoding the removed subtrees into the labels of their direct ancestor nodes. In a second step, we linearize T_k with a few move operations and obtain a word $w(T)$ which approximately encodes T and such that $\mathsf{u\text{-}stat}(w(T))$ can be approximately sampled from samples on T. The two processes are illustrated in Figure 6.1. Then some of our previous results on words can be extended to trees.

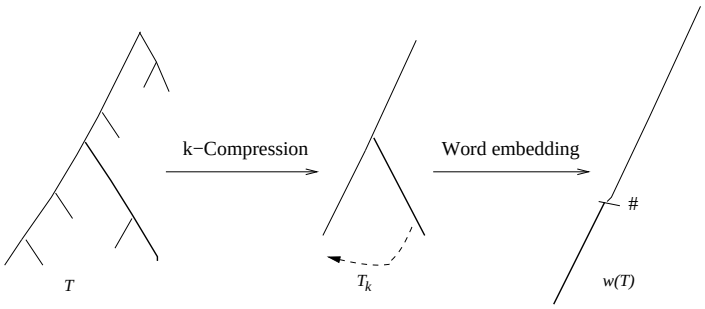

FIG. 6.1. *Tree T in (a), its k-compression T_k in (b) and its linearization in (c).*

6.1. Compression. Initial labels are called *simple labels*. We introduce new labels to encode subtrees. A *tree label* is simply the encoding of a subtree from a node. The labels of the tree encoded by a tree label have all to be simple. By construction, the leaves of our compressed trees will all have tree labels. A *mixed label* for a node v is an ordered pair (a, t) or (t, a), where a is a simple label (namely the one of v before the compression) and t is a tree label (namely that encodes the subtree from a successor of v). Notice that an internal node of a compressed tree might have either a simple label or a mixed label. Moreover, an internal node with a simple label has degree 1 or 2, whereas an internal node with a mixed label always has degree 1. The *size* of a label is the size of the encoded subtrees: 1 for simple labels, l for labels encoding trees with l nodes, and $l + 1$ for a mixed label (a, t) or (t, a) where a is simple and t encodes a tree with l nodes.

DEFINITION 6.1. *A* compressed tree *is a tree T' whose leaves have tree labels, internal nodes of degree* 2 *have simple labels, and all other nodes have either simple or mixed labels.*

We will say that a compressed tree T' *encodes* a tree T, if replacing the tree labels of T' by the corresponding subtrees leads to the tree T.

More precisely, the *k-tree alphabet*, denoted $\Sigma^{(k)}$, is the set of any possible labels that come from a k-compression, that is when tree labels encode trees of sizes between k and $2k - 1$. The smallest compressed subtree is of size k and the largest of size $2k - 1$, a tree t with two subtrees of size $k - 1$ on each successor. Therefore $|\Sigma^{(k)}| \leq (|\Sigma| + 1)^{O(k)}$. The *size* of a $\Sigma^{(k)}$-tree is the sum of the sizes of its labels, where the *size* of a simple label is 1; the *size* of a tree label is the size of the encoded tree; and the *size* of a mixed label is the sum of the sizes of its labels, that is 1 plus the size of its tree label part.

We now define our compression explicitly.

DEFINITION 6.2. *Let T be a tree and $k \geq 1$ be an integer. The* k-compression T_k *of T is the compressed tree that encodes T, where the subtrees of size less than k have been removed.*

The new label of a node v of T_k can be computed using $O(k)$ queries to T by the following procedure **Encode**(T, v, k) that either computes the label of v on T_k, or rejects v if it has been removed during the compression.

Encode(T, v, k)
If v is not the root of T, and the subtree from v in T has size $< k$ (this can be checked with k queries using DFS) then Reject
Let u_1 and u_2 be the successors of v (or $u_1 = u_2$ if v has only one successor)
If the subtrees from u_1 and u_2 in T have size $< k$ then return the tree label of the subtree of v in T
If no subtree from u_1 and u_2 in T has size $< k$ then return the simple label of v
If only the subtree from u_2 in T has size $< k$ then return the pair of the simple label of v followed by the tree label of the subtree from u_2 in T
If only the subtree from u_1 in T has size $< k$ then return the pair of the tree label of the subtree from u_1 in T followed by the simple label of v

Our procedure ensures that the size of every label of T_k is less than $2k$.

PROPOSITION 6.3. *When* ***Encode***(T, v, k) *does not reject v, it outputs a label of size at most $2k - 1$.*

PROPOSITION 6.4. *Let T be a tree and $k \geq 1$ be an integer. Then* ***Encode*** *satisfies the following for every node $v \in T$:*

1. *$v \in T_k$ if and only if* ***Encode***(T, v, k) *does not reject.*
2. *If $v \in T_k$, then its new label in T_k is given by the outcome of* ***Encode***(T, v, k).

The k-compression of a tree T is almost a word, as the number of remaining 2-degree nodes in T_k is small.

LEMMA 6.5. *T_k has at most εn 2-degree nodes.*

Proof. Only nodes with simple labels can have degree 2 in T_k. To every 2-degree node of T_k we will associate a distinct part of T_k of size at least k. Then the lemma follows since T_k has at most size n.

The construction is bottom-up. We start with the tree $T'_k = T_k$, remove some nodes and continue until there are no more 2-degree nodes with simple labels in T'_k. We will maintain the following invariant of T'_k, which T_k initially satisfies by assumption: *(*): Every 2-degree node of T'_k with a simple label has two successors whose subtrees in T_k have size at least k.*

The iteration procedure is now described. Let v be a lowest node of T'_k with degree 2 and a simple label. By Property (*), the node v has two successors u_1 and u_2 whose subtrees in T_k have size $\geq k$. We remove from T'_k the remaining subtree of u_1. Therefore v has now degree 1 in T'_k. Moreover, since the subtree from u_2 is still in T'_k, we guarantee that the new T'_k still satisfies Property (*). We can do this transformation at most n/k times, and so there are at most εn nodes of degree 2 in T_k. □

6.2. Word Embedding. Lemma 6.5 implies that any tree T is 3ε-close to another tree T', such that T'_k has no 2-degree nodes and is 2ε-close to T_k. The tree T' is inductively defined by the following procedure **Linearize**(T, k). For simplicity, the construction of T'_k from T_k is described (see Figure 6.2). Let # be a new symbol.

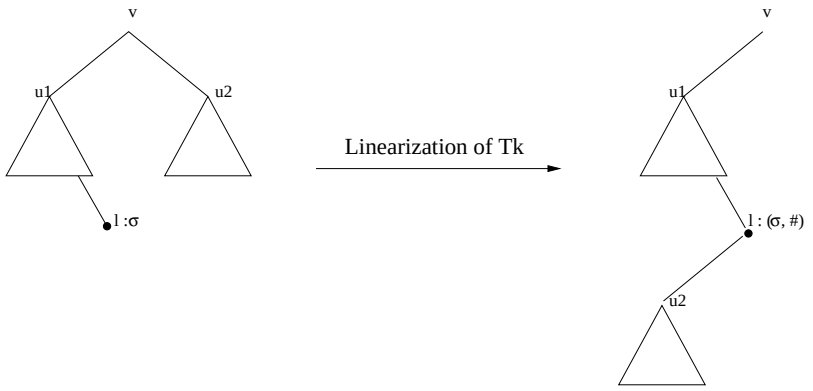

FIG. 6.2. *Linearization of T_k.*

Linearize(T, k)
Let $T' = T$
While there exists a 2-degree node v in T'_k do
  Let (u_1, u_2) be the two successors of v
  Let l be the rightmost leaf of the subtree of u_1 in T'_k
  Move u_2 to become the (unique) successor of the rightmost leaf l of the subtree of u_1 in T'_k
    (perform the analogous move operation also on T)
  Change the tree label σ of l to a mixed label $(\sigma, \#)$
Return T'

Note that we only mark the leaves where we move subtrees but not their ancestors. One reason is that this would require too many types of markers. But the main reason is that since testers can tolerate a small fraction of moves, this information is not useful.

Using T'_k one can now define the word embedding of T.

DEFINITION 6.6. *Let T be a tree and $k \geq 1$ be an integer. Let $w(T)$ be the word over the k-tree alphabet which enumerates the labels of T'_k from its root.*

We will then use the uniform statistics of $w(T)$ in order to apply the results of previous sections on words. Since each letter of $w(T)$ might encode a tree of size up to $O(1/\varepsilon)$ we need to apply an ε^2-tester on words, so we can get an $O(\varepsilon)$-tester on trees. The following procedure can approximate $\mathsf{u\text{-}stat}(w(T))$ (with block-size k^2) with additive error ε^2 by taking $O(1/\varepsilon^6)$ samples from T_k.

Statistics(T, k)
Take a random v in T until **Encode**(T, v, k) does not reject
Let $i = 1$ and $u_1 = v$
Iterate $k^2 - 1$ times
  If u_i has at least one successor in T_k then let v be the leftmost one
  If u_i has no successor then
    Using a backtracking of depth k^4 in T, search the first 2-degree node w in T_k such that u_i is on the left subtree of w
    If the search fails then go back the beginning of the algorithm
  Let $i = i + 1$ and $u_i = w$
Output the labels of $u_1 u_2 \dots u_{k^2}$ using **Encode**$(T, \cdot, k)$

LEMMA 6.7. ***Statistics***$(T, 1/\varepsilon)$ *outputs a label drawn from a probabilistic distribution which is at* ℓ_1*-distance at most* ε^2 *from* u-stat$(w(T))$ *(with block-size* $1/\varepsilon^2$*). Moreover, the expected query and time complexities of* ***Statistics***$(T, 1/\varepsilon)$ *are in* $O(1/\varepsilon^6)$.

Proof. Let B be the set of nodes such that the backtracking search fails in **Statistics**$(T, 1/\varepsilon)$, where the rightmost leaf of T_k has been removed. Let C be the set of 2-degree nodes in T_{k^4}, whose size is at most $\varepsilon^4 n$ by Lemma 6.5. We will prove that B is not larger than C by constructing an injection from C to B. For any node $v \in C$ with left successor v_1, let $u_v \in B$ be the rightmost leaf of v_1. Conversely, for any node $u \in B$, let $v_u \in C$ be the first node reached along a backtracking in T_k from u to its root. Assuming that v_u is well defined for any $u \in B$, it is clear that $u_{v_u} = u$. Now, v_u is well defined unless u is the right most leaf of T_k, which is not the case by assumption. Thus by the existence of a left inverse, we deduce that the mapping from C to B is indeed an injection.

Thus, because of the backtracking of depth k^4, **Statistics**$(T, 1/\varepsilon)$ samples all but $\varepsilon^4 n$ leaves of T_k. Therefore the fraction of missing subwords of $w(T)$ is $\varepsilon^4 \times k^2 = \varepsilon^2$, and we can conclude (using e.g. Proposition 3.6) the correctness of the algorithm. To calculate its expected query complexity, first note that the number of queries in the main loop is dominated by $(k^2 - 1)(k^4 + 2) = O(k^6)$. Note further that the expected number of queries needed to complete the first step is constant (because T has at most an ε fraction of nodes for which **Encode**$(T, \cdot, k)$ will fail). Finally, there are at most an ε^4 fraction of nodes for which a depth k^4 backtracking fails, and for each such node v there are at most k^2 nodes for which the main loop leads to failing on v, and so also the expected number of times that the loop aborts and due to backtracking failure is also a constant. Thus $O(k^6)$ is indeed the expected number of queries in the entire algorithm. □

First our equality tester for words can be applied to trees as an equality tester since any elementary operation on $w(T)$ corresponds to $O(1)$ elementary operations on T. Note that this tester is not tolerant in general, since the sizes of $w(T)$ and $w(T')$ can be significantly different whereas T and T' are close.

THEOREM 6.8. *Tree equality is* ε*-testable with query and time complexities in* $(|\Sigma| + 1)^{O(1/\varepsilon^5)}$.

Then our regular language tester can also be extended since we also get that an automaton on trees T of size m corresponds [11] to a push-down automaton on words $w(T)$ whose number of states and stack alphabet have both size m.

THEOREM 6.9. *For every real* $\varepsilon > 0$ *and 2-ranked regular tree language* L *over a finite alphabet* Σ*, there exists an* ε*-tester for* L *whose query complexity is in* $(|\Sigma| +$

$1)^{O(1/\varepsilon^5)}$ *and whose time complexity is in* $2^{(|\Sigma|+1)^{O(1/\varepsilon^5)}}$. *Moreover, given a 2-ranked tree automaton with m states that recognizes L, the tester can be constructed in time exponential in* $m^{(|\Sigma|+1)^{O(1/\varepsilon^5)}}$.

Proof. Let A be a tree automaton that recognizes L and let T be any tree. We now define a push-down automaton B on any word $w(T)$. Note that both A and B are non deterministic. The push-down automaton B reads $w(T)$ from right to left, and evaluates $w(T)$ as A on T bottom-up with a few modifications:

- B has the same set of states as A. Moreover, the initial and the accepting states of B are the ones of A, but with the added condition that the stack of B has to be empty when accepting. The stack alphabet is is also set to be the set of states of A.
- When a simple label is read, one of the following steps is chosen non deterministically.
  - The automaton B evaluates the current label just as A. This corresponds to a move to the predecessor.
  - The automaton B evaluates the current label as A with a pair of successors whose states are (in any order) the current state of B and the symbol on its stack. Then the symbol of the stack is pulled. This corresponds to a branching between a previously evaluated branch of the tree and the current one.
- When a tree label t is read, the automaton B modifies its state to a state that A can reach after reading t. This corresponds to the evaluation of a bounded-size subtree.
- When a mixed label $(t, \#)$ is read, the current state is pushed on the stack, the state of B is set to any initial state, and t is read just as in the previous step. This corresponds to the end of the evaluation of a branch, and the exploration of a new branch.
- When a mixed label (t, a) (respectively (a, t)) is read, where a is not $\#$, the automaton B first computes a state q that A can reach after reading t from initial states. Then B evaluates the current label just as A with a pair of successors whose states are in q and the current state of B (respectively the current state of B and q). This corresponds to a branching between a bounded-size branch of the tree and the current one.

By construction, for any tree $T \in L$, there exists an accepting path in B for $w(T)$. Conversely, if $w(T)$ is accepted by B then T is $O(\varepsilon)$-close to a tree in L.

The proof is then concluded by using the above construction together with Lemma 6.7 and then using a tester for the context-free language. We use a tester construction similar to that of Theorem 5.14, with the difference that we need a tolerant one which uses samples from (an approximation of) $\mathsf{u\text{-}stat}(w)$ instead of samples from $\mathsf{b\text{-}stat}(w)$. Its construction is almost identical to that of Theorem 5.14, only instead of taking a k power of the automaton corresponding to the Greibach normal form, we take the corresponding automaton for k-shingles. □

For the equivalence testing problem, there already exists a deterministic exponential time algorithm for the exact version of the problem, so it seems that our current approach does not reduce the complexity. However, in the previous construction, if L is a tree language such that the number of occurrences of # symbols in $w(T)$ for $T \in L$ is constant, then the language of possible $w(T)$ is regular. This is the case for example for the binary encoding for regular unranked tree of constant depth. We can then apply the equivalence tester for regular languages of words, and test the

equivalence between such classes of tree languages in polynomial time. The general case remains an open problem.

7. Conclusion and Open Problems. Under the edit distance with moves, we can not only efficiently test for regular languages, but can also approximately compare whole languages, given by regular expressions or nondeterministic automata.

The approximate comparison notion, that of ε-equivalence testing, holds particular promise in view of the fact that exact equivalence of languages is usually a computationally very hard problem to decide. For regular expressions for example we need polynomial time for the approximate algorithm, as compared to polynomial space for the exact problem.

Still, the edit distance with moves is a relatively weak norm. It would be interesting to extend ε-equivalence testing of nondeterministic automata and regular expressions to the stronger norm of the edit distance without moves. Also, it would be interesting to get rid of the dependency of the *degree* of the polynomial time expression on ε, *i.e.* find an ε-equivalence tester whose running time has a globally constant degree, and where only the coefficients depend on ε.

Moving to context free grammars, the exponential dependency of the running time of the equivalence testing algorithm (as well as the property tester) is much better than the undecidability of the exact problem (with regards to property testing, one should note that under the Hamming distance as opposed to our norm, context free grammars are generally not testable at all). Still, it may be the case that the edit distance with moves allows even for an equivalence tester whose running time is polynomial in the size of the grammars. It would be interesting to investigate this further.

Finally, with regards to tree languages, where the edit distance with moves is a very appealing and natural norm, we have just started to scratch the surface. It would be interesting to see how far one can go there. In particular, it would be interesting to see whether polynomial time equivalence testing of tree regular languages is feasible.

REFERENCES

[1] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. *Combinatorica*, 20(4):451–476, 2000.

[2] N. Alon, M. Krivelevich, I. Newman, and M. Szegedy. Regular languages are testable with a constant number of queries. *SIAM J. Comp.*, 30(6):1842–1862, 2000.

[3] T. Batu, F. Ergün, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, and R. Sami. A sublinear algorithm for weakly approximating edit distance. In *Proc. STOC*, pp. 316–324, 2003.

[4] T. Batu, L. Fortnow, R. Rubinfeld, W. Smith, and P. White. Testing that distributions are close. In *Proc. FOCS*, pp. 259–269, 2000.

[5] M. Blum and S. Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, 1995.

[6] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comp. Syst. Sci.*, 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 *Proc. 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. *Theor. Comp. Sci.*, 329:71–92, 2002.

[10] G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. In *Proc. SODA*, pp. 667–676, 2002.

[11] J. Doner, Tree acceptors and some of their applications. *Journal of Computer and System Science*, 4:406–451, 1970.
[12] O. Goldreich, and D. Ron. Property Testing in Bounded Degree Graphs. *Algorithmica*, 32(2):302–343, 2002.
[13] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.
[14] W. Masek and M. Paterson. A faster algorithm for computing string edit distance. *J. Comp. Syst. Sci.*, 20(1):18–31, 1980.
[15] F. Magniez and M. de Rougemont. Property testing of regular tree languages. *Algorithmica*, to appear. Downloadable at `http://www.lri.fr/ magniez/PAPIERS/mr-algorithmica06.pdf`
[16] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. FOCS*, pp. 125–129, 1972.
[17] I. Newman. Testing membership in languages that have small width branching programs. *SIAM J. Comp.*, 3142(5):1557–1570, 2002.
[18] R. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966.
[19] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. Journal of Computer and System Sciences,72(6):1012-1042 2006
[20] R. Rubinfeld. On the robustness of functional equations. *SIAM J. Comp.*, 28(6):1972–1997, 1999.
[21] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comp.*, 25(2):23–32, 1996.
[22] D. Shapira and J. Storer. Edit distance with move operations. In *Proc. Symp. Combinatorial Pattern Matching*, pp. 85–98, 2002.
[23] L. Stockmeyer and A. Meyer. Word problems requiring exponential time. In *Proc. STOC*, pp. 1–9, 1973.
[24] K. C. Tai. The tree-to-tree correction problem. *J. ACM*, 26:422–433, 1979.