Approximate Satisfiability and Equivalence[*]

Eldar Fischer[†] Frédéric Magniez[‡] Michel de Rougemont[§]

Abstract

Inspired by Property Testing, 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 the classical equivalence $F_1 \equiv F_2$ between two formulas F_1 and F_2, to ε-equivalence $F_1 \equiv_\varepsilon F_2$ for $\varepsilon > 0$. We consider the class of strings and trees with the edit distance with moves, and show that 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 on words, defined by Monadic Second Order formulas. Our equivalence tester has polynomial time complexity in the size of the automaton (or regular expression), for a fixed ε, whereas the exact version of the equivalence problem is PSPACE-complete.

Last, we extend the geometric embedding, and hence the tester algorithms, to infinite regular languages and to context-free languages. For context-free languages, the equivalence tester has an exponential time complexity, whereas the exact version is undecidable.

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 whether $U \models F$ for a structure $U \in \mathbf{K}$ and a formula F, and equivalence is the decision problem $F_1 \equiv F_2$, *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 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. $U \models_\varepsilon F$ if there exists a U' ε-close to U such that $U' \models F$, otherwise $U \not\models_\varepsilon F$. $F_1 \equiv_\varepsilon F_2$ if all but finitely many structures that satisfy $U \models F_1$ satisfy also $U \models_\varepsilon F_2$, and conversely.

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.* is independent of the size n. An 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 decided for important properties. These decision methods are robust, in the sense that they are adaptable to noisy inputs. They are well adapted to environments such as XML data on the Web represented by unranked labelled trees or Genomics data represented by strings.

Property testing of regular languages was first considered in [2] for the Hamming distance, and then extended to languages recognizable by bounded width read-once branching programs [16], 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 one 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) which has similarities with the Parikh mapping [17]. Based on this embedding, we develop an ε-tester (**Theorem 3.1**)

[*]Work supported in part by grants *ACI-SI: VERA* and *AlgoQP* of the French Ministry of research, and by an Israel Science Foundation grant number 55/03

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

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

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

for the equality between two words whose complexity is $|\Sigma|^{O(1/\varepsilon)}$, where $|\Sigma|$ is the alphabet size. Our tester is also *tolerant*, that is it is not only an ε-tester, but it also accepts with high probability words that are ε^2-close. 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 [18]. 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 3.2**), 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 [21] but can be approximated within an $\tilde{O}(\ln n)$ factor only in near linear time [10]. It has been used in [13] 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].

Then we extend our embedding to languages. This leads us to an approximate geometrical description of regular languages by finite unions of polytopes, which is robust (**Theorem 3.2**). Discretizing this representation gives us a new ε-tester (**Theorem 3.3**) for regular languages whose query complexity is $|\Sigma|^{O(1/\varepsilon)}$ and time complexity is $2^{|\Sigma|^{O(1/\varepsilon)}}$. Whereas the complexity of previous testers for regular languages depended (exponentially) on the number of states 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 build the tester, which is independent of the size of the input.

Using again discretization, we construct an ε-equivalence tester (**Theorem 3.4**) 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 [22]). We then extend this result to the ε-equivalence testing of Büchi automata (**Theorem 3.5**) (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 3.6**) (for which the exact decision version is not even recursively computable). Equivalence testers decide if MSO (Monadic second-order) formulas on strings are ε-equivalent. Approximate Model-Checking could generalize this approach to other Logics in the future, when exact Model-Checking remains infeasible.

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 closely 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 highly used 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 2-degree nodes, and can therefore be used to encode any ranked tree T into 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 4.1**). This is unlike the context of dense graphs where there is a negative result [1]. In addition, regular tree languages have an $(\varepsilon^4, O(\varepsilon))$-tolerant tester (**Theorem 4.2**) whose query complexity is $|\Sigma|^{O(1/\varepsilon^5)}$ and time complexity is $2^{|\Sigma|^{O(1/\varepsilon^5)}}$. Again, as opposed to previous testers for tree regular languages [13], here the automaton is only used in a preprocessing step to build the tester, in time exponential in the tree automaton size for 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 in the language of $\mathbf{K}$ is defined in some Logic such as First-Order Logic or Monadic Second-Order Logic, and we use the logical characterization of regular properties of words (resp. trees) by Monadic Second Order Logic. We say that $U \in \mathbf{K}$ *satisfies* P, or $U \models P$, when $U \in P$. 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 either an insertion, a deletion or a substitution of a letter, or the *move* of a subword of w into another position. The *edit distance with moves* $\mathsf{dist}(w, w')$ 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 1) on an unranked ordered tree T is either an insertion or a deletion of a node [23], the substitution of a label, or the move of an entire subtree [13]. 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 all the j successors 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 deletions and moves that gives a r-ranked tree.

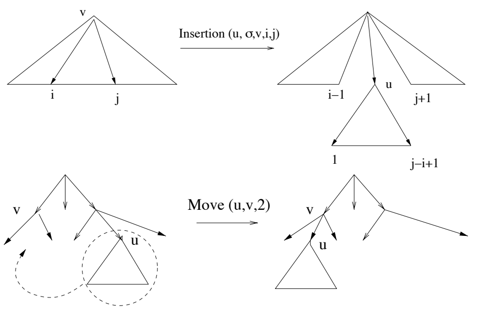

Figure 1. Elementary operations on trees.

2.2 Approximate Satisfiability and Equivalence

We define the notion of approximate satisfiability as in property testing [11]. Let $\mathbf{K}$ be a class of finite structures U with a distance 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'. They 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 the order of the maximum distance. (For dense graphs, M is the square of the maximal size of the respective structures.)

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 (Tester [11]). *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 prob. at least $2/3$;
then we say that the tester is a *(tolerant)* $(\varepsilon_0, \varepsilon)$-*tester* [18]. Approximation algorithms are related to tolerant testers [18]. 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 x, outputs a value z such that $\Pr[\alpha(f(x)) \leq z \leq \beta(f(x))] \geq 2/3$.

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 i, for some i, 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 uniform 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 $\mathbf{K}$. 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 conversely.*

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 with probability $2/3$.

3 Words

We will define several statistics over words and study their robustness [19, 20] and soundness. Robustness means that far words have far statistics, and soundness means that close words have close statistics. Despite the difficulty of computing the edit distance with moves, one can efficiently approximate the statistics of a word. This will directly give us a tolerant tester and then an approximation algorithm for the normalized edit distance with moves.

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

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, $w = w[1]_b w[2]_b \dots w[\varepsilon n]_b$, where $w[i]_b \in \Sigma^k$ is the *i-th block letter* of w. The *block statistics* $\mathsf{b\text{-}stat}(w)$ 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 for a uniformly random choice of $j \in \{1, \dots, n/k\}$ we get $w[j]_b = u$.

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

We want to construct statistics that are both robust and sound. Since the block statistics will appear to be non robust, we 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 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".

For example, for binary words, if $k = 2$ (and so $\varepsilon = 0.5$), 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, defined below, will be a link between block and 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 subword of the original word.

In order to construct an efficient algorithms based on those 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 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 coordinate $u \in \{1, \dots, D\}$, $\Pr[|\mu[u] - \hat{\mu}_N[u]| \geq t] \leq 2e^{-2Nt^2}$, by the Chernoff-Hoeffding bound for the random variables $X_i = f(Y_i)[u]$ which are between 0 and 1 and whose expectation is $\mu[u]$. We conclude using a union bound. $\square$

As a corollary we can approximate both block and uniform statistics using a number of samples independent of n. The variables Y_i denote the position of the selected block letters u of w, and X_i denote the corresponding vectors of size $|\Sigma|^k$ whose u-coordinate is one and others are zero. Let stat denote either $\mathsf{b\text{-}stat}$ 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 3.1. *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 $\mathsf{u\text{-}stat}$.*

3.2 Robustness and Soundness

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 same size). 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 3.2 (Robustness)**.** *$\mathsf{dist}(w, w') \leq (\frac{1}{2}|\mathsf{b\text{-}stat}(w) - \mathsf{b\text{-}stat}(w')| + \varepsilon) \times n$.*

Proof. If $\mathsf{b\text{-}stat}(w) = \mathsf{b\text{-}stat}(w')$, 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.

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

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

Lemma 3.3 (Soundness)**.** *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 at most $n-k+1$ 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 $|\mathsf{u\text{-}stat}(w) - \mathsf{u\text{-}stat}(w')| \leq 2 \times \frac{3(k-1)}{n-k+1}$.

If $\mathsf{dist}(w, w') \leq \varepsilon^2 n$ and $n = \Omega(\frac{1}{\varepsilon})$, then by 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 then will 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 using a Chernoff-Hoeffding bound argument.

Lemma 3.4. *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 there exists with non-zero probability a word v 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.

Lemma 3.5. $|\mathsf{bu\text{-}stat}(w) - \mathsf{u\text{-}stat}(w)| = O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4 n})$.

Lemma 3.6 (Robustness)**.** *Let* $n = \Omega(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^5})$*. If* $\mathsf{dist}(w, w') \geq 5\varepsilon n$ *then* $|\mathsf{u\text{-}stat}_k(w) - \mathsf{u\text{-}stat}_k(w')| \geq 6.5\varepsilon$.

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

```
Uniform Tester($w, w', \varepsilon$):
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 3.1. *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 above test:*
(1) accepts if $w = w'$ *with probability* 1*;*
(2) accepts if w *and* w' *are* ε^2*-close with prob. at least* $2/3$*;*
(3) rejects if w *and* w' *are* 5ε*-far with prob. at least* $2/3$*.*
Moreover its query and time complexities are in $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$.

From this $(\varepsilon^2, 5\varepsilon)$-tolerant tester, one can derive an $(\varepsilon^2, 5\varepsilon)$-approximation algorithm of the distance following the approach of [18].

Corollary 3.2. *There exists an* $(\varepsilon^2, 5\varepsilon)$*-approximation algorithm for computing the normalized distance* $\varepsilon = \mathsf{dist}(w, w')/|w|$ *between every words* w, w' *of the same size in* $\Omega(\frac{\ln(|\Sigma|/\varepsilon)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$*, and with query and time complexities in* $O(\frac{(\ln(|\Sigma|/\varepsilon))|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$.

3.3 Geometric Embedding of a Language

3.3.1 General Observations

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. Nonetheless, this work can be extended to the uniform statistic, leading to tolerant testers, by following a more complex approach that will appear in a future journal version.

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 from Lemma 3.2 if the size of the words is fixed. This means that given unlimited computational power we could test any language with a constant number of queries, by first precomputing the statistics of all possible words of length n in that language.

However, the block statistics do 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 the elements $w \in L$ is not a good characterization of a general language L. For instance, the word $w_0^{3\times 2^{s-1}}$ is $(1-1/k^{2^{s-1}})$-far from the language $\{w_0^{2^t} : t \geq 1\}$, for every positive integer s. Moreover, it is not hard to construct using the appropriate powers a language whose testing algorithm requires arbitrarily intensive computations.

To construct a test that works for all n using only one preprocessing stage, one might consider only block statistics of loops of a language (as provided by an appropriate pumping lemma). This makes sense when any word of a language can be decomposed into loops up to a few remaining letters. Regular languages have this property, and context-free languages also share it when any permutation between block letters is allowed (see Section 3.6).

3.3.2 Regular Languages on strings

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

We will characterize L by the block statistics of its loops on the block alphabet. We remark that the statistics of the A^k-loops basically only depend on L and k (the proof is omitted due to the lack of space).

Definition 3.1. *A word v over Σ^k is an A^k-*loop *if there exist two words u, w over Σ^k and an accepting path of A^k for uvw, such that the state of the automaton after reading u (following the above accepting path) is identical to the state after reading uv.*
*A finite set of A^k-loops is A^k-*compatible *if all the loops can occur one after the other (in any order) in one accepting path of A^k.*

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

Definition 3.2. *Let $\mathcal{H}$ be the union of* $\mathsf{Convex\text{-}Hull}(\mathsf{b\text{-}stat}(v_1), \ldots, \mathsf{b\text{-}stat}(v_t))$ *when $v_1, \ldots, v_t$ range over 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.

Proposition 3.1. *Let $w \in L$. Then there exists a permutation of the block letters of w into $w' = vu_1u_2 \ldots u_l$, such that $|v|_b, |u_1|_b, \ldots, |u_l|_b \leq m$ and $\{u_1, u_2, \ldots, u_l\}$ is an A^k-compatible set of A^k-loops (not necessarily distinct).*

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

Proposition 3.2. *$\mathcal{H}$ equals the union of* $\mathsf{Convex\text{-}Hull}(\mathsf{b\text{-}stat}(v_1), \ldots, \mathsf{b\text{-}stat}(v_t))$ *when $v_1, \ldots, v_t$ range over A^k-compatible loops such that $|v_i|_b \leq m$ and $t = |\Sigma|^{1/\varepsilon} + 1$.*

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 3.7). This can be understood as an approximate Parikh classification of regular languages, whereas the original Parikh characterization was for context-free languages [17]. The converse is also approximately true (Theorem 3.2).

As an example, let $L = (010)^*0^*$ and $k = 2$. Let $s_1 = \mathsf{b\text{-}stat}((010)^2) = (1/3, 1/3, 1/3, 0)$, $s_2 = \mathsf{b\text{-}stat}(00) = (1, 0, 0, 0)$, then $\mathcal{H}_L = \mathsf{Convex\text{-}Hull}(s_1, s_2)$.

Lemma 3.7. *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 $\mathsf{b\text{-}stat}(w') \in \mathcal{H}$.*

Lemma 3.8. *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 $X = \sum_{i=1}^{l} \lambda_i \, \mathsf{b\text{-}stat}(u_i)$, where $l = |\Sigma|^k + 1$, $|u_i|_b \leq m$, $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$. Fix any integer n. We choose non-negative integers $(r_i)_{i=1,2,\ldots,l}$ that respectively approximate $\lambda_i \frac{\varepsilon n}{|u_i|_b}$, that is satisfy $0 \leq |r_i - \lambda_i \frac{\varepsilon n}{|u_i|_b}| \leq 1$, and such that

$0 \leq \sum_i r_i|u_i|_b - \varepsilon n \leq m$. It is always possible to satisfy this last condition due to the degree of freedom on the choices of r_i and the upper bound $|u_i|_b \leq m$: We let $j \geq 0$ be the minimum integer so that $\sum_{i=1}^{j} \lceil \lambda_i \frac{\varepsilon n}{|u_i|_b} \rceil |u_i|_b + \sum_{i=j+1}^{l} \lfloor \lambda_i \frac{\varepsilon n}{|u_i|_b} \rfloor |u_i|_b \geq 0$, and set $r_i = \lceil \lambda_i \frac{\varepsilon n}{|u_i|_b} \rceil$ for $i \leq j$ and $r_i = \lfloor \lambda_i \frac{\varepsilon n}{|u_i|_b} \rfloor$ for $i > j$.

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

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

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

Theorem 3.2. *Let $w \in \Sigma^n$ and $X \in \mathcal{H}$ be such that $|\mathsf{b\text{-}stat}(w) - X| \leq \delta$. Then*
$$\mathsf{dist}(w, L) \leq \left(\tfrac{\delta}{2} + \left(1 + O(\tfrac{m|\Sigma|^{1/\varepsilon}}{\varepsilon^2 n})\right)\varepsilon\right) n.$$

3.3.3 Construction of $\mathcal{H}$

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

Lemma 3.9. *Given A and ε, a set H of $(|\Sigma|^{1/\varepsilon} + 1)$-tuples of vectors can be computed in time $m^{|\Sigma|^{O(1/\varepsilon)}}$ such that $|H| \leq m^{|\Sigma|^{O(1/\varepsilon)}}$ and $\mathcal{H} = \bigcup_{S \in H} \mathsf{Convex\text{-}Hull}(S)$.*

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

Proposition 3.3. *Given A and ε, the set $\mathcal{H}_\varepsilon$ can be computed in time $m^{|\Sigma|^{O(1/\varepsilon)}}$.*

3.4 Property and Equivalence Testers

Theorem 3.3. *For every real $\varepsilon > 0$ and regular language L over a finite alphabet Σ, there exists an ε-tester for L whose query complexity is in $O(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^4})$ and whose time complexity is in $2^{|\Sigma|^{O(1/\varepsilon)}}$.*
Moreover, given an automaton with m states that recognizes L, the tester can be constructed in time $m^{|\Sigma|^{O(1/\varepsilon)}}$.

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

The tester is in two steps: a preprocessing step and the testing step itself. Given A and ε, one can compute $\mathcal{H}_\varepsilon$ in time $m^{|\Sigma|^{O(1/\varepsilon)}}$ from Proposition 3.3. 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 3.1, 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 at distance at most 2ε from $\mathcal{H}_\varepsilon$, the tester accepts, and otherwise it rejects. □

Theorem 3.4. *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 $\varepsilon > 0$. The algorithm simply computes the respective discrete approximations $\mathcal{H}_{A,\varepsilon/2}$ and $\mathcal{H}_{B,\varepsilon/2}$ of $\mathcal{H}_A$ and $\mathcal{H}_B$ corresponding to the automata A and B. If they are equal, the tester accepts, and otherwise it rejects. The correctness proof, omitted here, essentially follows from the previous section.

□

3.5 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 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*.

For every integer n, we denote by w_n the prefix of w of size n. Two words w, w' are *ε-close* if the superior limit $\overline{\lim}_{n\to\infty} \mathsf{dist}(w_{|_n}, w'_{|_n})/n$ is at most ε. Last, the *block statistics* $\mathsf{b\text{-}stat}(w)$ of w is the set of accumulation points of the sequence $(\mathsf{b\text{-}stat}(w_{|_n}))_n$.

By adapting our geometric embedding for this distance, an equivalence tester for two Büchi automata follows from the one previously defined for regular languages (over finite words). In this tester, we modify the Definition 3.2 of $\mathcal{H}$, by simply restricting ourselves to the loops of (strongly) connected components of the accepting states of A^k (we could also extend Theorem 3.3 to lasso words as in [9]).

Definition 3.3. *For every connected component C of A^k, let $\mathcal{H}_C$ be the convex hull of the vector set $\{\mathsf{b\text{-}stat}(w) :$ w is a loop in C s.t. $|w|_b \leq m\}$. We denote by $\mathcal{H}'$ the union $\bigcup_C \mathcal{H}_C$ where C ranges over all connected components of A^k that contain an accepting state and are reachable from an initial state.*

Theorem 3.4 is then valid for nondeterministic Büchi automata, with $\mathcal{H}'$ taking the place of $\mathcal{H}$.

Theorem 3.5. *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.*

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 use our approximate Equivalence tester with a fixed small parameter ε.

3.6 Context-Free Languages

We can construct an exponential time test and an equivalence tester for context-free languages, given by their grammar, or by their push-down automaton (the two representations are polynomially equivalent so we can switch back and forth between them as convenient). In comparison, the exact decision problem of whether two context-free grammars define the same language is not decidable.

The proof uses the original Parikh theorem about spectra of context-free languages, that provides a formula defining a semi-linear set on the letter counts of all possible words. 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 $\mathcal{H}_\varepsilon$.

Therefore we can design string testers for a context-free language (which are not possible for the usual edit distance without moves, as the counter example in [2] works for the edit distance as well as the Hamming distance), in analogy to Theorem 3.3. We explicitly state the equivalence testability for context-free grammars we get as in Theorem 3.4 .

Theorem 3.6. *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 size.*

We note a corollary for regular expressions with squaring. Although they recognize only regular languages, their (exact) equivalence problem is EXPSPACE-complete by [15], so the exponential time algorithm given here can be considered as a slight improvement. Applying the previous theorem, we can obtain an equivalence tester.

4 Trees

To simplify the discussion we will consider only 2-ranked labeled ordered, trees but our results can be extended to any ranked trees. Let Σ be the finite label alphabet. 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 define the k-compression of a tree T, which basically consists of removing every node whose subtree has size $\leq k$, and encoding the removed subtrees into the labels of their ancestor nodes. This compression leads naturally to a word $w(T)$ that encodes T such that $\mathsf{u\text{-}stat}(w(T))$ can be approximately sampled from samples on T. Then some of our previous results on words can be extended to trees.

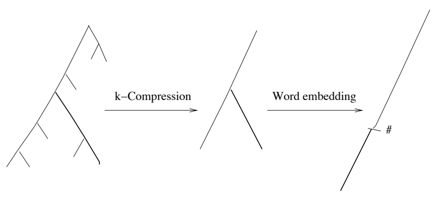

Figure 2. k-compression and word embedding.

4.1 Compression

Initial labels are named *simple labels*. We introduce new *tree labels* for leaves l whose purpose is to encode a subtree from l. We also interpret a simple label on a leaf as a tree label. A *mixed label* is an ordered pair of a simple label and a tree label, whose tree label encodes the subtree of the corresponding successor. Notice that an internal node might have either a simple label or a mixed label. Such a labeled tree *encodes* T when expanding the tree labels (from leaves and from the tree components of mixed labels) leads to the initial tree T.

The *size of a simple label* is 1. The *size of a tree label* is the size of the tree that it encodes. 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.

Definition 4.1. *Let T be a tree and $k \geq 1$ be an integer. The k-*compression *T_k of T is the tree encoding of T such that each tree label has the minimal possible value in $[k, 2k-1]$.*

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 size in $[k, 2k-1]$. Therefore $|\Sigma^{(k)}| = |\Sigma|^{O(k)}$. The new label of a node v can be computed using $O(k)$ queries to T by the following procedure that either computes its label on T_k, or rejects if v is not anymore in T_k.

```
Encode($T, v, k$)
If the subtree from $v$ in $T$ has size $< k$ 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 both size $< k$ then return $v$ and the encoding of the subtree from $v$ in $T$
If no subtree from $u_1$ and $u_2$ in $T$ has size $< k$ then return $v$ and the simple label of $v$
If only the subtree from $u_1$ (resp. $u_2$) in $T$ has size $< k$ then return $v$, and the pair of the simple label of $v$ and the tree label of the subtree from $u_1$ (resp. $u_2$, but with opposite order) in $T$
```

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

Lemma 4.1. *T_k has at most εn 2-degree nodes.*

Proof. In this proof the labels are the ones of T_k. Only nodes with simple labels can have degree 2 in T_k. Moreover, a node has degree 2 in T_k iff it has degree 2 in T and has a simple label. To every 2-degree node of T_k we will associate a distinct part of T of size $\geq k$. Then the lemma follows since T_k has $\leq n$ nodes.

The construction is bottom-up. We start with the tree $T' = T$ and continue until there are no more 2-degree nodes in T' with simple labels. We will maintain the following invariant of T', which T initially satisfies by assumption:

(*): Every 2-degree node of T' with a simple label (in T_k) has two successors whose subtrees in T' have size $\geq k$.

The iteration procedure is now described. Let v be a lowest node of T' with degree 2 and a simple label. By Property (*), the node v has two successors u_1 and u_2 whose subtrees in T' have size $\geq k$. We remove from T' the remaining subtree of u_1. Therefore v has now degree 1 in T'. Moreover since the subtree from u_2 is still in T', we guaranty that the new T' still satisfies Property (*). □

4.2 Word Embedding

From Lemma 4.1 we show 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. First let us fix a new symbol #. To construct T'_k, we recursively eliminate each 2-degree node v with a simple label and successors (u_1, u_2), by moving u_2 to the rightmost leaf l of the subtree of u_1 in T_k and by changing the tree label σ of l to a mixed label $(\sigma, \#)$; equivalently on T we insert a new node u with label # between l and its parent, the left successor of u is l, its right sucessor is u_2.

Then we define $w(T)$ as the word over the k-tree alphabet which enumerates the labels of T'_k from its root. $w(T)$ is also the enumeration of the labels of T_k obtaining by a DFS from its root, where at most εn tree labels t have been modified to mixed labels $(t, \#)$.

We will perform the uniform statistics on $w(T)$ 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, in order to get an $O(\varepsilon)$-tester on trees. An important fact is that $\mathsf{u\text{-}stat}(w(T))$ (with block-size k^2) can be approximately sampled with additive error ε^2 by $O(1/\varepsilon^6)$ samples on T_k.

```
Statistics($T, k$)
(*) Take a random $v$ in $T$ while Encode($T, v, k$) rejects
Let $i = 1$ and $u_1 = v$; and iterate $k^2 - 1$ the following
  If $u_i$ has at least one successor in $T_k$ let be $v$ the left one
  If $u_i$ has no successor then
    Using a backtracking of depth $k^4$ in $T_k$, Search the first 2-degree node $v$ in $T_k$ such that $u_i$ is on the left subtree of $v$
    If the search fails then go back to Step (*)
  Let $i = i + 1$ and $u_i = v$
Outputs the labels of $u_1 u_2 \dots u_i$ using Encode($T, \cdot, k$)
```

Lemma 4.2. ***Statistics**$(T, 1/\varepsilon)$ outputs a probabilistic distribution which is at ℓ_1-distance at most ε^2 from* $\mathsf{u\text{-}stat}(w(T))$ *(with block-size $1/\varepsilon^2$). Moreover, its expected query and time complexities are in $O(1/\varepsilon^6)$.*

Proof. **Statistics**$(T, 1/\varepsilon)$ connects all but $\varepsilon^4 n$ leaves of T_k as in T'_k because of the backtracking. This means that only an ε^2 fraction of subwords is missing from $w(T)$. Then we can conclude. □

First our equality tester for words can be applied to trees as an isomorphism tester since any elementary operation on $w(T)$ corresponds to $O(1)$ elementary operations on T.

Theorem 4.1. *The tree isomorphism problem is tolerantly $(\varepsilon^4, O(\varepsilon))$-testable with query and time complexities in $|\Sigma|^{O(1/\varepsilon^5)}$.*

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

Theorem 4.2. *For every real $\varepsilon > 0$ and 2-tree regular language L over a finite alphabet Σ, there exists a tolerant $(\varepsilon^4, O(\varepsilon))$-tester for L whose query complexity is in $|\Sigma|^{O(1/\varepsilon^5)}$ and whose time complexity is in $2^{|\Sigma|^{O(1/\varepsilon^5)}}$. Moreover, given a 2-tree automaton with m states that recognizes L, the tester can be constructed in time exponential in $m^{|\Sigma|^{O(1/\varepsilon^5)}}$.*

Proof. The push-down automaton basically reads the tree as a word from the bottom up using the tree alphabet with a few modifications. When a label $(t, \#)$ is read, the current state is pushed on the stack and the state goes to an accessible state from reading t with an initial state, as a leaf. Then the symbol of the stack can be pulled while reading a simple label which corresponds to a branching between a previously evaluated branch of the tree and the current one. $\square$

For the equivalence problem, there already exists a deterministic exponential time algorithm for the exact version of the problem. Therefore 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 $\#$ symbols in $w(T)$ for $T \in L$ is constant, then the set of $w(T)$ is regular. It is the case 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.

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. Ergun, 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, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comp. Syst. Sci.*, 47(3):549–595, 1993.

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

[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] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

[12] O. Goldreich, and D. Ron. Property Testing in Bounded Degree Graphs. *Algorithmica*, 32(2): 302–343, 2002.

[13] F. Magniez and M. de Rougemont. Property testing of regular tree languages. In *Proc. ICALP*, pp. 932–944, 2004.

[14] W. Masek and M. Paterson. A faster algorithm for computing string edit distance. *J. Comp. Syst. Sci.*, 20(1):18–31, 1980.

[15] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. FOCS*, pp. 125–129, 1972.

[16] I. Newman. Testing membership in languages that have small width branching programs. *SIAM J. Comp.*, 3142(5):1557–1570, 2002.

[17] R. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966.

[18] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. TR04-010, ECCC, 2004.

[19] R. Rubinfeld. On the robustness of functional equations. *SIAM J. Comp.*, 28(6):1972–1997, 1999.

[20] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comp.*, 25(2):23–32, 1996.

[21] D. Shapira and J. Storer. Edit distance with move operations. In *Proc. Symp. Combinatorial Pattern Matching*, pp. 85–98, 2002.

[22] L. Stockmeyer and A. Meyer. Word problems requiring exponential time. In *Proc. STOC*, pp. 1–9, 1973.

[23] K. C. Tai. The tree-to-tree correction problem. *J. ACM*, 26:422–433, 1979.