A new algorithm for testing if a regular language is locally threshold testable

Mikołaj Bojańczyk

Institute of Informatics, Warsaw University

Abstract

A new algorithm is presented for testing if a regular language is locally threshold testable. The new algorithm is slower than existing algorithms, but its correctness proof is shorter. The proof idea is to restate the problem in Presburger arithmetic.

Key words: Formal Languages

A language $L \subseteq A^*$ is called *locally threshold testable* (LTT) if it is a Boolean combination of languages of the form: a) words that have $w \in A^*$ as a prefix; or b) words that have $w \in A^*$ as a suffix; or c) words that have $w \in A^*$ as an infix at least n times. For instance, the language $a^+b^+a^+b^+$ is locally threshold testable, as witnessed by: "words that have a as a prefix, have ab as an infix exactly two times, and have ba as an infix exactly one time". By Gaifman's theorem, locally threshold testable is equivalent to being definable in first-order logic with the successor relation on positions.

This paper presents a new proof of the following theorem:

Theorem 1 It is decidable if a regular language L is locally threshold testable.

As observed by Beauquier and Pin in [1,2], the above problem is decidable by a result of Thérien and Weiss [8]. A polynomial time algorithm – with respect to the minimal deterministic automaton for L – was presented by Pin in [4,5]. The approach of [8,2,1,4,5] uses semigroup theory; the crux is that a language is locally threshold testable if and only if its syntactic semigroup satisfies certain swapping conditions. However, proving that the swapping conditions are sufficient requires involved combinatorics. The proof in this paper does not use semigroups, only Parikh images and Presburger arithmetic. The point is to provide an alternative proof of decidability, even if the algorithm is quite slow (several exponentials).

The rest of the paper is devoted to a proof of Theorem 1. There are two important parameters in the definition of LTT: the maximal size of the words w and the maximal counting threshold n. If the maximal size of words w is at most k, then the language is called k-locally threshold testable. Our approach is to first calculate k (in Section 1) and then n (in Section 2).

1 Calculating k

The results and techniques in this section are not new. Essentially, the Delay Theorem of [7] is reproved, but using automata instead of semigroups. This section is included to make the paper self-contained.

A *k-scanner* is a transducer $f : A^* \to B^*$ that only sees the input through windows of size at most k. The scanner is defined in terms of functions

$$\mathit{short} : A^{\leq k} \to B^* \qquad \mathit{prefix}, \mathit{infix}, \mathit{suffix} : A^k \to B^* \ .$$

The output of the scanner upon reading an input word $w \in A^*$ is defined as follows. If $|w| \leq k$, then the scanner outputs $\mathit{short}(w)$. Otherwise, the scanner begins by outputting the result of *prefix* on the k-letter prefix of w. Then it consecutively processes the k-letter infixes of w (not including the prefix and suffix) from left to right, and appends to the output the value of *infix* for each such infix. Finally, the scanner applies *suffix* to the k-letter suffix, and appends the result to the output.

A 1-scanner is essentially a homomorphism. A 2-scanner can, for instance, output an input letter only if it is different from the previous one.

Consider an automaton $\mathcal{B}$ with input alphabet B, and let $f : A^* \to B^*$ be a scanner. Here and afterward, the term automaton refers to a deterministic finite state automaton. A language $L \subseteq A^*$ is *recognized* by $\mathcal{B} \circ f$ if L is the set of words w such that $f(w)$ is accepted by $\mathcal{B}$. The following theorem shows that the window size of f can be bounded by the size of an automaton for L:

Theorem 2 Let L be a regular language. One can effectively compute a $k \in \mathbb{N}$ such that if L is recognized by $\mathcal{B} \circ f$ for some automaton $\mathcal{B}$ and scanner f, then L is recognized by $\mathcal{B} \circ g$ for some k-scanner g.

Note that in the above theorem, only the scanner f is replaced by g, while the automaton $\mathcal{B}$ stays the same.

Proof. Let m be the window size of f. Let $\mathcal{A}$ be an automaton recognizing L, with states Q. Let A, B be the input alphabets of $\mathcal{A}$, $\mathcal{B}$ respectively; in particular f transforms A^* into B^*. Finally, let $l = |Q|^{|Q|}$, and let $k = 2l$.

Each word $w \in A^*$ gives a transformation on states of $\mathcal{A}$. A word $u \in A^+$ is an *extender* of $w \in A^*$ if the words w and wu give the same state transformation. Since extenders of w are closed under concatenation, we may assume that each extender has length at least $m-1$. To make the extender unique, we chose the first such u in the lexicograhic ordering.

Below, we define an l-scanner $h : A^* \to A^*$, which, intuitively speaking, adds extenders to the input word. The scanner works as follows. If the input word has at most l positions, then it is copied onto the output without changes. Otherwise, the scanner looks successively at each position $i = 1, \ldots, n$ in the input word $a_1 \cdots a_n$, and inspects the last $l-1$ positions (these give the word $a_{i-l-1} \cdots a_i$ when $i > l$, or $a_1 \cdots a_i$ when $i \leq l$). If this word has an extender u, then the scanner outputs $a_i u$, otherwise it only outputs a_i. The above procedure can be easily implemented in terms of maps *prefix*, *infix*, *suffix*. Slightly ahead of time, we remark the key property of this scanner: it will output an extender at least every l positions.

An induction on word length shows that transformations on states of $\mathcal{A}$ are the same for w and $h(w)$. In particular, $\mathcal{B} \circ f$ recognizes L if and only if $\mathcal{B} \circ f \circ h$ recognizes L. The proof of the lemma will be completed by showing that the composition $f \circ h$ is a $2l$-scanner.

We need to define the maps *short*, *prefix*, *infix*, and *suffix* for the composition $f \circ h$. The *short* map is defined enumerating the results of $f \circ h$ on the short words. Among the latter three maps, we only deal with *infix*. Let $a_1 \cdots a_{2l}$ be an infix with $2l$ letters. Inspecting the last l letters $a_{l+1} \cdots a_{2l}$, we can determine what new word $v \in B^*$ will be appended by the l-scanner h to its output. Unfortunately, this word v is not enough to apply the m-scanner f; for this we also need $m-1$ letters from the previous output. We will show that by looking at the letters $a_1 \cdots a_{2l-1}$, we can also determine the last $m-1$ letters that have been produced by h before reading the new position a_{2l}. This information is enough to apply the m-scanner l to the newly produced output v.

Since we have access to positions $a_1 \cdots a_{2l-1}$, we know what output will be produced by the l-scanner h when processing the letters $a_l, \ldots, a_{2l-1}$. We will show that h will output an extender at least once on these positions, which concludes by assumption on extenders having at least $m-1$ letters. Since there are at most $l = |Q|^{|Q|}$ possible state transformations, at least two of the $l+1$ words ϵ, a_l, $\ldots$, $a_l \cdots a_{2l-1}$ induce the same transformation on states of $\mathcal{A}$. In particular, at least two of these words admits an extender, and therefore so does one of the words $a_1 \cdots a_l, \ldots, a_l \cdots a_{2l-1}$. This gives the desired result. $\square$

Corollary 1 For each regular language $L \subseteq A^*$ one can effectively calculate

a $k \in \mathbb{N}$ such that if L is locally threshold testable, then it is already k-locally threshold testable.

Proof. A *counting automaton* is an automaton $\mathcal{B}$ that counts the number of times each letter of the alphabet occurs in the input word, up to a given threshold n. More formally, if the input alphabet is $A = \{a_1, \ldots, a_m\}$, then the state space consists of vectors $(i_1, \ldots, i_m) \in \{0, \ldots, n\}^m$, while the transition function is defined:

$$\delta((i_1, \ldots, i_m), a_j) = (i_1, \ldots, i_{j-1}, \max(n, i_j + 1), i_{j+1}, \ldots, i_m) \ .$$

Clearly a language is k-locally threshold testable if and only if it is recognized by $\mathcal{B} \circ f$, where $\mathcal{B}$ is a counting automaton and f is a k-scanner. The result then follows from Theorem 2. □

2 Calculating n

Using Corollary 1, we have determined a candidate for the k such that L may be k-locally threshold testable. This section is devoted to calculating the counting threshold n. The new ideas of the paper are here.

The *k-footprint* $\pi_k(v)$ of a word $v \in A^*$ is a vector of naturals, which says:

- What words of length at most k are prefixes of v.
- What words of length at most k are suffixes of v.
- For each word w of length at most k, how many times does w occur as an infix of v.

The vector $\pi_k(v)$ therefore has $3(|A| + |A|^2 + \cdots + |A|^k)$ coordinates. For the prefixes and suffixes, only the values $0, 1$ need be used in the vector, however the coordinates for infixes may contain arbitrarily large values.

Numbers $i, j \in \mathbb{N}$ are said to *agree up to threshold* $n \in \mathbb{N}$, if they are either equal or both greater than n. This definition is extended to vectors of naturals coordinatewise. The definition of LTT can be reformulated as follows: a language $L \subseteq A^*$ is k-locally threshold testable if and only if:

(*) There is $n \in \mathbb{N}$ such that any two words whose k-footprints agree up to threshold n must either both belong to L or both be outside L.

Theorem 1 will therefore follow from the following lemma:

Lemma 1 Condition (*) is effective (given L and k).

Before proving the above lemma, we recall the definitions of semilinear sets and Presburger arithmetic. A vector set $X \subseteq \mathbb{N}^m$ is called *semilinear* if it is a finite union of linear sets, which are vector sets of the form:

$$\{\overline{x} + i_1\overline{x}_1 + \cdots + i_n\overline{x}_n : i_1, \ldots, i_n \in \mathbb{N}\} \qquad \text{for some } \overline{x}, \overline{x}_1, \ldots, \overline{x}_n \in \mathbb{N}^m \ .$$

Presburger arithmetic is the first-order theory of the natural numbers with addition $(\mathbb{N}, +)$. Presburger's Theorem [6] states that not only is Presburger arithmetic decidable, but the vector sets defined by formulas with free variables are exactly the semilinear sets. Moreover, this correspondence is effective. For instance, the formula $x = y$ describes the (linear) set $\{(0,0) + i(1,1) : i \in \mathbb{N}\}$.

Proof. Consider the sets of footprints for L and its complement, i.e.

$$X = \{\pi_k(v) : v \in L\} \qquad \text{and} \qquad Y = \{\pi_k(v) : v \notin L\} \ .$$

In Lemma 2, we show that both these sets of vectors are semilinear, and can be effectively calculated. Using the sets X and Y, condition (*) becomes:

> There is some $n \in \mathbb{N}$ such that no two vectors $\overline{x} \in X$ and $\overline{y} \in Y$ can agree up to threshold n.

The above question is a statement in Presburger arithmetic, and can therefore be effectively answered. □

Lemma 2 For a regular language $L \subseteq A^*$, the vector set $\{\pi_k(v) : v \in L\}$ is semilinear and can be effectively calculated.

Proof. For $k = 1$ this follows from Parikh's theorem [3]. The case of $k > 1$ can be reduced to $k = 1$ by considering words over the expanded alphabet $A^{\leq k+1}$, where each position is labeled with the next $k + 1$ letters (or less, if the position is near the end of the word). □

Almost all the results in this paper easily extend to ranked trees. (Note that for trees, we need to use the full power of Parikh's theorem for context-free grammars, and not just regular languages.) The only exception is Theorem 2, for which no tree analogues are known.

I would like to thank the anonymous referee for many helpful comments.

References

[1] D. Beauquier and J.-É. Pin. Factors of words. In *International Colloquium on Automata, Languages and Programming*, volume 372 of *Lecture Notes in Computer Science*, pages 63 – 79, 1989.

[2] D. Beauquier and J.-É. Pin. Languages and scanners. *Theoretical Computer Science*, 84(1):3 – 21, 1991.

[3] R. Parikh. On context-free languages. *Journal of the ACM*, pages 570–581, 1966.

[4] J.-É. Pin. The expressive power of existential first-order sentences of Büchis sequential calculus. In *International Colloquium on Automata, Languages and Programming*, volume 1099 of *Lecture Notes in Computer Science*, pages 300 – 311, 1996.

[5] J.-É. Pin. The expressive power of existential first-order sentences of Büchis sequential calculus. *Discrete Mathematics*, 291(1-3):155 – 174, 2005.

[6] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du I congrès de Mathématiciens des Pays Slaves. Warszawa*, pages 92 – 1001, 1929.

[7] H. Straubing. Finite semigroup varieties of the form $V * D$. *Journal of Pure and Applied Algebra*, 36:53–94, 1985.

[8] D. Thérien and A. Weiss. Graph congruences and wreath products. *Journal of Pure and Applied Algebra*, 36(2):205 – 215, 1985.