# NON-REGULAR LANGUAGES, THE MYHILL-NERODE THEOREM, AND LINEAR ALGEBRA TESTS

JOEL FRIEDMAN

## Contents

**Disclaimer:** The material may sketchy and/or contain errors, which I will elaborate upon and/or correct in class. For those not in CPSC 421/501: use this material at your own risk...

There are a number of ways to test if a language, $L$, over an alphabet, $\Sigma$ is regular or not, and—if so—what is the smallest number of states that a DFA recognizing $L$ can have. Most techniques we know can be organized as follows:

(1) results on languages over an alphabet of size one;
(2) "derived results," where proven results on languages give similar results on related languages;
(3) the Myhill-Nerode Theorem;
(4) consequences of linear algebra, applied to the adjacency matrix of a DFA;
(5) the Pumping Lemma;

This year in CPSC 421, we focus on (1,2,3), and and omit (4,5).

New to this article as of Fall 2021[1] is the focus on the simple classification of languages over alphabets of size one, i.e., consisting of a single letter. In addition, one can this classifaction to derive results for certain languages over larger alphabets. This is probably the easiest way to produce examples of non-regular languages "from scratch," without assuming certain facts from linear algebra.

*Date*: Sunday 17th October, 2021, at 13:17(get rid of time in final version).
Research supported in part by an NSERC grant.

[1] Based on discussion of September 28, 2021, and, in particular a question of Markus de Medeiros.

The Pumping Lemma is by far the most awkward technique to use; its advantage is that there is an analogous Pumping Lemma for context-free languages. [In CPSC 421 this year we are skipping over the chapter on context-free languages, so we have little motivation to cover the Pumping Lemma.] Another advantage of the Pumping Lemma is that it is easy to prove "from scratch," rather than requiring theorems in linear algebra, but the same is true of (1,2,3) above, and, in particular, the proof of the Myhill-Nerode theorem is similar to that of the Pumping Lemma.

This article focuses on the Myhill-Nerode theorem; this theorem is stronger than the Pumping Lemma, in that any result of the Pumping Lemma can be proven (usually more simply and directly) using the Myhill-Nerode theorem. Furthermore, the Myhill-Nerode theorem allows you to build (at least in principle) the DFA with the smallest number of states that recognizes a given regular language.

The Cayley-Hamilton theorem and Jordan canonical form are results in linear algebra that allow us to quickly prove numerous results about DFA's and non-regular laguages. The connection to linear algebra is that we may view DFA as a directed graph, whose adjacency matrix can be studied with linear algebra. We briefly discuss these ideas in an appedix to this article, Of the above techniques, they are the easiest to use, but (1) they don't always give good results on many common examples of non-regular languages, and (2) some CPSC 421/501 have not taken a one-term course in linear algebra. However, there is a strong argument that a first-term linear algebra course is more relevant to computer than, say, a second-term calculus course, at least as of 2021[2]; hence, in the future one may be able to assume that all CPSC 421/501 students are familiar enough with matrices to easily apply the methods of Appendix A.

## 1. Languages Over an Alphabet Consisting of a Single Letter

In class we will explain that if $M = (Q, \Sigma, \delta, q_0, F)$ is a DFA over a language where $\Sigma$ consists of a single letter, then by elimianting redundant states and renaming states, one can assume that $Q$ consists of the $n$ states, $Q = \{v_1, \ldots, v_n\}$, with $q_0 = v_1$, where the (sole) transition from $v_i$ with $i < n$ is to $v_{i+1}$ (and the transition from $v_n$ is to an arbitrary element of $Q$, which may be $v_1$ or $v_n$ itself).

[SEE DRAWINGS FROM CLASS NOTES, OCT 5]

We therefore deduce the following theorem.

**Theorem 1.1.** *A language, $L \subset \Sigma^*$, over $\Sigma = \{a\}$ is regular iff it is* eventually $p$-periodic *for some $p \in \mathbb{N}$, in the sense that there exists an $n_0 \in \mathbb{Z}_{\geq 0}$ such that*

$$\forall n \geq n_0, \quad a^n \in L \iff a^{n+p} \in L. \tag{1}$$

*In other words, for sufficiently large $n$, whether or not $a^n$ belongs to $L$ depends only on the class of $n$ modulo $p$ (i.e., the remainder of $n$ divided by $p$, commonly denoted $n \bmod p$ in computer science). so that $n \bmod p$ returns the remainder of $n$ divided by $p$, i.e., a number between $0$ and $p-1$).*

In (1) we understand $n$ to be an integer; we could more precisely write $\forall n \in \mathbb{Z}_{\geq n_0}$ or $\forall n \in \mathbb{Z}_{\geq 0},\ n \geq n_0$, but it seems less cumbersome to leave (1). In the homework in 2021, we identify $\Sigma^*$ above with $\mathbb{Z}_{\geq 0}$, and define what it means for a subset of $\mathbb{Z}_{\geq 0}$ to be eventually periodic.

[2] We thank Vee Kay for discussions on this point.

**Example 1.2.** . Let

$$L = \{a^{(n^3)} \mid n \in \mathbb{N}\} = \{a, a^8, a^{27}, a^{64}, \ldots\}$$

be the language of strings of $a$'s whose length is a perfect cube. Then since the interval between any two perfect cubes is

$$(n+1)^3 - n^3 = 3n^2 + 3n + 1,$$

which is arbitrarily large, $L$ cannot be $p$ periodic (since for any fixed $p$ and large $n$, $a^{n^3+1}, a^{n^3+2}, \ldots, a^{n^3+p}$ are not in $L$, which would imply that $L$ is finite).

One can similarly show that if $n_1 < n_2 < \cdots$ is an infinite (increasing) sequence of non-negative integers, then if

$$L = \{a^{n_i} \mid i \in \mathbb{N}\} = \{a^{n_1}, a^{n_2}, \ldots\}$$

is regular, we must have $n_{i+1} - n_i$ must be bounded above over all $i$ (see Exercise 6.1.6).

## 2. Derived Results: Part 1

Here is a standard way of producing more lower bounds (or examples of non-regular languages) from others.

**Proposition 2.1.** *Let $L_1, L_2 \subset \Sigma^*$ be lanuages over $\Sigma$. If $L_1$ is regular, and $L_1 \cap L_2$ is non-regular, then $L_2$ is non-regular.*

This follows immediately from the fact that the intersection of any two regular languages is, again, regular.

**Example 2.2.** Let $L \subset \Sigma^*$ be the language over $\Sigma = \{a, b\}$ of strings/words whose length is a perfect cube. Then $L$ is non-regular, since $L \cap \{a\}^*$ is, according to Example 1.2, non-regular.

## 3. The Myhill-Nerode Theorem: Part 1

**Definition 3.1.** If $L$ is a language over an alphabet $\Sigma$, and $s \in \Sigma^*$, we define the *accepting futures of $s$ in $L$* to be

$$\mathrm{AcceptingFuture}_L(s) \overset{\mathrm{def}}{=} \{s' \in \Sigma^* \mid ss' \in L\}.$$

We will also use the abbreviation AccFut.

**Example 3.2.** Let $\Sigma = \{0, 1, \ldots, 9\}$, and

$$L = \text{DIV-BY-2} = \{0, 2, 4, 6, 8, 10, 12, \ldots\}.$$

In class we gave a 5 state DFA that recognizes this language. We have

$$\begin{aligned}
\mathrm{AccFut}_L(\epsilon) &= L = \{0, 2, 4, 6, 8, 10, \ldots\} \\
\mathrm{AccFut}_L(0) &= \{\epsilon\} \\
\mathrm{AccFut}_L(00) &= \emptyset \\
\mathrm{AccFut}_L(1) &= \Sigma^*(0, 2, 4, 6, 8) \\
\mathrm{AccFut}_L(2) &= \{\epsilon\} \cup \Sigma^*(0, 2, 4, 6, 8)
\end{aligned}$$

Note that we have

$$\mathrm{AccFut}_L(2) = \mathrm{AccFut}_L(4) = \mathrm{AccFut}_L(2238) = \cdots,$$

so many strings have the same accepting future with respect to $L$.

In class we similarly gave 6 different strings with different accepting futures for the language DIV-BY-3. We also explained why if DIV-BY-3 has 6 different accepting futures, then any DFA recognizing DIV-BY-3 must have at least 6 different states. More generally we have the following observation.

**Proposition 3.3.** *If a language, $L$, over an alphabet, $\Sigma$, has at least $n$ distinct accepting futures (i.e., $n$ distinct values of $\mathrm{AccFut}_L(s)$ with $s \in \Sigma^*$), then any DFA recognizing $L$ has at least $n$ states.*

*Proof.* If $s, s' \in \Sigma^*$ are taken to the same state in a DFA, then for any $t \in \Sigma^*$, $st$ lands in an accepting state of the DFA iff $s't$ does. Hence if

$$\mathrm{AccFut}_L(s_1), \ldots, \mathrm{AccFut}_L(s_n)$$

are distinct, then a DFA recognizing $L$ must take $s_1, \ldots, s_n$ to distinct states. □

**Example 3.4.** Let $\Sigma = \{0,1\}$ and

$$L = \{0^n 1^n \mid n \in \mathbb{N}\} = \{01, 0011, 000111, 0^4 1^4, \ldots\}$$

we have

$$\mathrm{AccFut}_L(0) = \{1, 011, 00111, \ldots\}$$
$$\mathrm{AccFut}_L(00) = \{11, 0111, 001111, \ldots\}$$
$$\mathrm{AccFut}_L(000) = \{111, 01111, \ldots\}$$

and, more generally, $\mathrm{AccFut}_L(0^k)$ has a unique shortest string, namely $1^k$. Hence $\mathrm{AccFut}_L(0^k)$ for $k \in \mathbb{N}$ are all distinct, and so $L$ is not regular.

## 4. The Myhill-Nerode Theorem: Part 2

The second part of the Myhill-Nerode is a converse to the proposition in the last section.

**Theorem 4.1.** *Let $L$ be a language over an alphabet $\Sigma$, and assume that there is a finite number, $n$, of distinct values of*

$$\mathrm{AccFut}_L(s)$$

*as $s$ varies over $\Sigma^*$. Then there exists a DFA with $n$ states that recognizes $L$.*

Actually, much more is true in the above theorem: one can actually build the DFA, and to do so one does not need to describe all of $\mathrm{AccFut}_L(s)$ for strings, $s$—rather, one needs only to be able to tell for $s, s' \in \Sigma^*$ whether or not $\mathrm{AccFut}_L(s)$ equals $\mathrm{AccFut}_L(s')$.

To prove the theorem we consider the languages:

(1) $\mathrm{AccFut}_L(\epsilon)$;
(2) $\mathrm{AccFut}_L(a)$, $\mathrm{AccFut}_L(b)$;
(3) $\mathrm{AccFut}_L(aa)$, $\mathrm{AccFut}_L(ab)$, $\mathrm{AccFut}_L(ba)$, $\mathrm{AccFut}_L(bb)$;
(4) etc.

Each time we see a new language, i.e., a new value of $\mathrm{AccFut}_L(s)$, we introduce a new state; the transition rule $\delta\colon Q \times \Sigma \to Q$ is given by

$$\delta(\mathrm{AccFut}_L(s), \sigma) = \mathrm{AccFut}_L(s\sigma)$$

(where we identify a value of $\mathrm{AccFut}_L(s)$ with its corresponding state). It is much easier to understand the theorem and its proof from an example.

**Example 4.2.** Let $\Sigma = \{a, b\}$ and

$$L = \{s \in \Sigma^* \mid s \text{ contains } ab \text{ as a substring}\}.$$

We begin by computing

$$\mathrm{AccFut}_L(\epsilon) = L,$$

which we associate with a state $q_0$ and to the string $\epsilon$; we then comptue

$$\mathrm{AccFut}_L(a) = b\Sigma^* \cup L, \quad \mathrm{AccFut}_L(b) = L, \tag{2}$$

which gives us a new state, $q_1$ associated to $b\Sigma^* \cup L$ and associate to the input string $a$; we do not introduce a new state for $b$, since we have already seen the language $\mathrm{AccFut}_L(b) = L$ associated to $q_0$ and $\epsilon$.

At this point:

(1) We have determined two states, $q_0, q_1$, of our DFA;
(2) $q_0$ is the initial state of the DFA, since the initial state is the state you reach on input $\epsilon$;
(3) from $q_0$, associated to $\epsilon$, based on (2) we have the transition rules

$$\delta(q_0, a) = q_1, \quad \delta(q_0, b) = q_0. \tag{3}$$

As a next step we want to determine $\delta(q_1, \sigma)$ for $\sigma = a, b$. We compute that

$$\mathrm{AccFut}_L(aa) = b\Sigma^* \cup L, \quad \mathrm{AccFut}_L(ab) = \Sigma^*;$$

since we have already encountered $b\Sigma^* \cup L$ but not $\Sigma^*$, we introduce a new state $q_2$ associated to $ab$ and to $\Sigma^*$, and declare

$$\delta(q_1, a) = q_1, \quad \delta(q_1, b) = q_2. \tag{4}$$

Next we want to determine $\delta(q_2, \sigma)$ for $\sigma = a, b$. We compute that

$$\mathrm{AccFut}_L(aba) = \Sigma^*, \quad \mathrm{AccFut}_L(abb) = \Sigma^*;$$

since we have already seen $\Sigma^*$, which is associated to $q_2$, we declare

$$\delta(q_2, a) = q_2, \quad \delta(q_2, b) = q_2. \tag{5}$$

At this point we have determined $\delta(q, \sigma)$ for all $\sigma = a, b$ and all states, i.e., $q_0, q_1, q_2$, without introducing new states. Hence $Q = \{q_0, q_1, q_2\}$, and $\delta$ is given by (3)–(5) above.

Finally, since

$$\epsilon \notin L, \quad \epsilon \notin b\Sigma^* \cup L, \quad \epsilon \in \Sigma^*,$$

the state $q_2$ is an accepting (or final) state, and $q_0, q_1$ are not. (The general principle here is that $\epsilon \in \mathrm{AccFut}_L(s)$ iff $s \in L$.) This determines the DFA.

Abstractly, the reason why the above construction works is that if

$$\mathrm{AccFut}_L(s) = \mathrm{AccFut}_L(s')$$

for some $s, s'$, then for any $\sigma \in \Sigma$ we have

$$\mathrm{AccFut}_L(s\sigma) = \mathrm{AccFut}_L(s'\sigma).$$

In the above example we have

$$\mathrm{AccFut}_L(a) = \mathrm{AccFut}_L(aa),$$

and this implies that for $\sigma = a, b$ we have

$$\mathrm{AccFut}_L(a\sigma) = \mathrm{AccFut}_L(aa\sigma);$$

hence the transition from the state of $aa$ to that of $aa\sigma$ is the same as of that from $a$ to $a\sigma$.

Notice that in the above construction we don't need to determine *all of* $\mathrm{AccFut}_L(s)$ for strings, $s \in \Sigma^*$; it suffices to know for certain $s, s' \in \Sigma^*$ whether or not $\mathrm{AccFut}_L(s)$ and $\mathrm{AccFut}_L(s')$ are equal.

## 5. Derived Results: Part 2

Once we prove that certain langauges are not regular, we can infer that other languages are not regular. Here is one such examples of a "derived result."

Above we have proven that $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ is not regular; this is also done in the textbook by Sipser (and similar textbooks) using the Pumping Lemma. Consider the language

$$L' = \bigl\{ s \in \{0,1\}^* \mid s \text{ has the same number of 0's and 1's} \bigr\}.$$

Then $L'$ is not regular, for if $L'$ were regular then

$$L' \cap 0^* 1^*$$

would also be regular, which is impossible since $L = L' \cap 0^* 1^*$.

Note that the Myhill-Nerode theorem gives a more direct proof that $L'$ is not regular: for any $k \in \mathbb{N}$, $\mathrm{FutAcc}_{L'}(0^k)$ has a unique shortest length string, which is $1^k$; hence the languages $\mathrm{FutAcc}_{L'}(0^k)$ are distinct for $k \in \mathbb{N}$, and so the Myhill-Nerode theorem implies that $L'$ is not regular.

In the the appendix we will also explain how to use linear algebra tests to show that $L'$ is not regular.

## 6. EXERCISES

### 6.1. Exercises on Languages over $\Sigma = \{a\}$.

**For all these exercises, recall that $\mathbb{Z}_{\geq 0}$ refers to the set $\{0, 1, 2, \ldots\}$, i.e., the set of non-negative integers.**

**Unless otherwise instructed, DO NOT USE THE MYHILL-NERODE THEOREM for the exercises in this subsection.** You can assume the results of Section 1, but **do not assume anything further, such as consequences of this section that we discussed in class; do not merely quote results stated in class.**

**Exercise 6.1.1.** Let $L$ be a language over the alphabet $\Sigma = \{a\}$.

6.1.1(a) Let $m \in \mathbb{Z}_{\geq 0}$. **Briefly explain** why if $L$ is finite and $a^m \in L$, then any DFA accepting $L$ must have at least $m + 1$ states.

6.1.1(b) Let $m \in \mathbb{N}$. **Briefly explain** why if $L$ contains $a^m, a^{m+1}, a^{m+2}, \ldots$, but does not contain $a^{m-1}$, then $L$ must have at least $m + 1$ states.

**Exercise 6.1.2.** Let $L = \{a^{3n} \mid n \in \mathbb{Z}_{\geq 0}\}$. What is the minimum number of states in a DFA needed to recognize $L$? **Explain this as briefly as possible.** Give such a DFA.

**Exercise 6.1.3.** Let $L = \{a^{3n} \mid n \in \mathbb{N}\}$. What is the minimum number of states in a DFA needed to recognize $L$? **Explain this as briefly as possible.** Give such a DFA.

**Exercise 6.1.4.** Let $L \subset \Sigma^*$ be a regular language with $\Sigma = \{a\}$. Hence $L$ is eventually $p$-periodic for some $p \in \mathbb{N}$. Let $p \in \mathbb{N}$ be the smallest $p$ for which $L$ is eventually $p$-periodic; **we call this value of** $p$ **the** ***period of*** $L$.

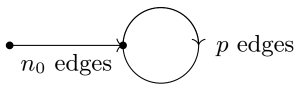
6.1.4(a) Show that if $p' \in \mathbb{N}$ is any multiple of $p$, then $L$ is also eventually $p'$-periodic.
6.1.4(b) Show that for any positive naturals $m < m'$, if $L$ is eventually $m$-periodic and eventually $m'$-periodic, then $L$ is eventually $(m' - m)$-periodic.
6.1.4(c) Show that if $L$ is $p'$-periodic for some $p' \in \mathbb{N}$, then $p'$ must be a multiple $p$. [Hint: If not, then $p'$ mod $p$ is strictly between 1 and $p$.]

[Hence for any $p' \in \mathbb{N}$, $L$ is eventually $p'$-periodic iff $p'$ is a multiple of $p$.]

**Exercise 6.1.5.** Let $L$ be a regular language over the alphabet $\Sigma = \{a\}$ that contains all strings of sufficiently large even length. What can you infer about the period of $L$, in view of Exercise 6.1.4?

**Exercise 6.1.6.** Let $L \subset \{a\}^*$ be an infinite, regular language over the alphabet $\Sigma = \{a\}$, and hence is $p$-periodic for some $p \geq 1$. Let $p$ be the smallest such value (i.e., the *period of* $L$ is $p$, in the terminology of Exercise 6.1.4).

6.1.6(a) Let $n, t \in \mathbb{N}$, and let $L \subset \{a\}^*$ be a language such that $a^n \in L$ but $a^{n+1}, \ldots, a^{n+t} \notin L$. **Briefly explain** why $p \geq t + 1$.
6.1.6(b) Consider the following depiction of a DFA over a size one alphabet as a digraph (*digraphs*, i.e., *directed graphs* are explained in [Sip], Chapter 0):



**Briefly explain** why the number of states in such a DFA is $n_0 + p$. [If you like, you may draw a diagram that looks closer to the one drawn in class.]
6.1.6(c) Show that under the assumptions of part (1), if, moreover, $a^{n-s} \in L$ for some $s \in \mathbb{N}$ with $s \leq t$ and $n - s \geq 0$, then either (1) $n - s \leq n_0 - 1$, or (2) $p \geq t + s + 1$. [Hint: consider whether or not the DFA reaches $a^{n-s}$ before the circular part of the DFA. You might draw an example to illustrate the general case.]
6.1.6(d) Conclude that the number of states in any DFA that recognizes is at least

$$\min(n + 2 + t - s, t + s + 1).$$

**Exercise 6.1.7.** Let $L \subset \{a\}^*$ be an infinite, regular language over the alphabet $\Sigma = \{a\}$, such that $a^{100}, a^{110} \in L$, but $a^{111}, a^{112}, \ldots, a^{120} \notin L$.

6.1.7(a) Use Exercise 6.1.6 to show that any DFA recognizing $L$ must have at least 21 states.
6.1.7(b) Give a language $L$ that satisfies the above conditions and is recognized by a DFA with exactly 21 states.

**Exercise 6.1.8.** Let

$$L = \{a^p \mid p \text{ is a prime number}\} = \{a^2, a^3, a^5, a^7, \ldots\}.$$

What does it mean to conjecture that $(a^2L) \cap L$ is infinite? Identify this as a well-known conjecture in number theory.

**Exercise 6.1.9.** Add another exercise or two.

## Appendix A. Consequences of Linear Algebra

**The following matertial is not required in CPSC 421 this year.**

If $L$ is a language over an alphabet $\Sigma$, we set

$$\mathrm{Count}_L(k) \overset{\mathrm{def}}{=} \bigl|L \cap \Sigma^k\bigr|,$$

which counts how many words of length $k$ over $\Sigma$ lie in $L$. Theorems in linear algebra show that $\mathrm{Count}_L(k)$ must satisfy certain conditions if $L$ is regular and accepted by a DFA with $n$ states.

For example, if

$$\mathrm{Count}_L(k) = \bigl(C + o(1)\bigr)10^k/k \tag{6}$$

where $C$ is a positive real number, then facts from linear algebra show that $L$ is not regular; similarly if 10 above is replaced with any positive real.

As an application, since the number of primes less than $N$ is $N/\log N + o(N)$ (this is called the Prime Number Theorem), the language PRIMES of primes written in base 10 is asymptotically

$$\bigl(C + o(k)\bigr)10^k/k$$

for some $C > 0$ (the constant $C$ depends on whether or not leading 0's are allowed). It follows that PRIMES is not regular.

Let us briefly describe more general consequences of linear algebra. We call these consequences "linear algebra tests."

A DFA with $n$ states has an *adjacency matrix*, which is an $n \times n$ matrix whose $i, j$ entry counts the number of symbols (in the alphabet, $\Sigma$, of the DFA) that take you from state $i$ to state $j$ in the DFA. It follows a DFA has adjacency matrix, $M$, and recognizes the langauge, $L$, then

$$\mathrm{Count}_L(k) \overset{\mathrm{def}}{=} \bigl|L \cap \Sigma^k\bigr|$$

is a sum of the $1, j$ components of $M^k$ over all final states $j$, were state 1 is the initial state. The Cayley-Hamilton theorem implies that $\mathrm{Count}_L(k)$ satisfies an $n$-term recurrence equation

$$\mathrm{Count}_L(k) = c_1 \mathrm{Count}_L(k-1) + \cdots + c_n \mathrm{Count}_L(k-n) \tag{7}$$

for all $k \ge n$ for fixed integers $c_1, \ldots, c_n$.

As an application, (7) easily implies that

$$L = \bigl\{a^m \mid m \text{ is a perfect square}\bigr\}$$

is not regular; it also shows that the language

$$L = \bigl\{a^m \mid m \in \mathbb{N},\ m \ge 20\bigr\}$$

cannot be recognized by a DFA with 20 states or fewer (which is optimal, since there is a 21 state DFA for this language).

The "Jordan canonical form" therem implies that for any regular language, $L$, there are complex numbers $\lambda_1, \ldots, \lambda_m$ and polynomials $p_1, \ldots, p_m$ such that for $k$ sufficiently large we have

$$\mathrm{Count}_L(k) = \sum_{i=1}^{m} p_i(k)\lambda_i^k.$$

As a consequence, one can show that if

$$\lambda = \lim_{k\to\infty} \frac{\mathrm{Count}_L(k+1)}{\mathrm{Count}_L(k)}$$

exists, then there is a polynomial $p$ such that

$$\mathrm{Count}_L(k) = p(k)\lambda^k(1+o(1)). \tag{8}$$

This implies that if

$$\mathrm{Count}_L(k) = \bigl(C + o(k)\bigr)10^k/k$$

for some $C > 0$, then $L$ is not regular. So the results regarding (8) generalize those of (6).

If $L = \{0^n1^n \mid n \in \mathbb{N}\}$ (which is a favourite example in textbooks of a non-regular language), then $\mathrm{Count}_L(k)$ alternates between 0 and 1. You get the same counting function for the regular language

$$\bigl\{0^{2n} \mid n \in \mathbb{N}\bigr\}.$$

Hence linear algebra tests can fail to provide optimal bounds on DFA's and regularity.

One actually gets more information from linear algebra: for example, in (8), $\lambda$ must be an *algebraic integer*, and $C$ must be an *algebraic number*.

Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1Z4, CANADA.

*E-mail address*: `jf@cs.ubc.ca`

*URL*: `http://www.cs.ubc.ca/~jf`