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 see if a language, L, over an alphabet, Σ is regular and, if so, what is the smallest number of states that a DFA recognizing L can have. Roughly speaking, we know of four basic techniques:

(1) the Myhill-Nerode Theorem;
(2) consequences of linear algebra, applied to the adjacency matrix of a DFA;
(3) the Pumping Lemma;
(4) derived results: if L, L' are languages and you know that L is regular and $L \cap L'$ is nonregular (or $L \cup L'$ is nonregular), then L' must be nonregular.

This year in CPSC 421, we focus on (1) and (4) above, and omit the Pumping Lemma.

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," but the same is true of the Myhill-Nerode theorem (and the proofs of both theorems are similar).

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.

Research supported in part by an NSERC grant.

In an appedix to this article, we will briefly explain some consequences of linear algebra. Of the above three techniques, they are the easiest to use. Their disadvantages are (1) they don't always give good results on many common examples of non-regular languages, and (2) to prove that these techniques work, we require some theorems in linear algebra.

1. The Myhill-Nerode Theorem: Part 1

Definition 1.1. If L is a language over an alphabet Σ, and $s \in \Sigma^*$, we define the *accepting futures of s in L* to be

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

We will also use the abbreviation AccFut.

Example 1.2. 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

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

Note that we have

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

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 1.3. *If a language, L, over an alphabet, Σ, has at least n distinct accepting futures (i.e., n distinct values of $\text{AccFut}_L(s)$ 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

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

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

Example 1.4. 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

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

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.

2. The Myhill-Nerode Theorem: Part 2

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

Theorem 2.1. *Let L be a language over an alphabet Σ, and assume that there is a finite number, n, of distinct values of*

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

as s varies over Σ^*. 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 2.2. 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 ϵ; we then comptue

$$\mathrm{AccFut}_L(a) = b\Sigma^* \cup L, \quad \mathrm{AccFut}_L(b) = L, \tag{1}$$

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 ϵ.

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 ϵ;

(3) from q_0, associated to ϵ, based on (1) we have the transition rules

$$\delta(q_0,a)=q_1,\quad \delta(q_0,b)=q_0. \tag{2}$$

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 Σ^*, we introduce a new state q_2 associated to ab and to Σ^*, and declare

$$\delta(q_1,a)=q_1,\quad \delta(q_1,b)=q_2. \tag{3}$$

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 Σ^*, which is associated to q_2, we declare

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

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 δ is given by (2)–(4) 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.

3. Derived Results

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

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

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 $k \in \mathbb{N}$, $\text{FutAcc}_{L'}(0^k)$ has a unique shortest length string, which is 1^k; hence the languages $\text{FutAcc}_{L'}(0^k)$ 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.

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 Σ, we set

$$\text{Count}_L(k) \stackrel{\text{def}}{=} \big|L \cap \Sigma^k\big|,$$

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

For example, if

$$\text{Count}_L(k) = \big(C + o(1)\big)10^k/k \tag{5}$$

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

$$\big(C + o(k)\big)10^k/k$$

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, Σ, 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

$$\text{Count}_L(k) \stackrel{\text{def}}{=} \big|L \cap \Sigma^k\big|$$

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

$$\mathrm{Count}_L(k) = c_1\mathrm{Count}_L(k-1) + \cdots + c_n\mathrm{Count}_L(k-n) \tag{6}$$

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

As an application, (6) easily implies that

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

is not regular; it also shows that the language

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

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

$$\mathrm{Count}_L(k) = p(k)\lambda^k(1+o(1)). \tag{7}$$

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 (7) generalize those of (5).

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 (7), λ 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