"Math section" notes

Contents

1 Sections 1, 2

1.1 Opening problems

First section.

Example 1.56 (Sipser p. 68).

There is more than one solution, as students demonstrated in section. One has both states accepting with the start state leading to other if an 'a' is seen. The other state self-loops with 'a's and returns to the start state if a 'b' is seen.

The trick to this accepting the language is exploiting NFA's ability to have any number of outgoing ages for a letter in the alphabet. In this case, the start node has no outgoing edges for 'b,' so a 'b' seen in the start state rejects the string. This rejects b and strings like abb. It forces an a to proceed a b.

What is the smallest DFA distinguishing $ababba$ and $ababaab$?

The trick here is ignoring the alphabet and focusing on string length. One string is of even length, the other odd, so a two state DFA can distinguish these strings with either state marked as accepting.

What is the smallest DFA distinguishing $abaaabb$ and $abaaaba$?

A good language is one in which every b must be proceeded by an a. This requires 3 states.

Prove every NFA can be converted to one with a single accept state.

A new accepting state with epsilon transitions from all existing final states suffices.

Prove that all-NFAs, which accept a string only if EVERY computation path for the string ends in an accept state, recognize the class of regular languages.

Since DFAs fit this definition we know they're a subset of all-NFAs. Since DFAs recognize the class of regular languages so must all-NFAs then.

1.2 What this section is about

This section is a mathematically inclined presentation of extensions to the course with some new theoretical material. It will contain minimal, if any, review of material presented in lecture. Its goal is to provide a broader view of computer science theory and challenge interested students.

The section is open to anyone in the course. No sign-up is required and participation in other sections is not an issue. Attendance is optional.

1.3 Professor Lewis's problem

Find the smallest DFA that distinguishes two strings, that is, the smallest DFA which accepts one string and rejects the other.

The CS121 forum has a topic on this issue.

A DFA recognizes a language, a subset of the strings generated by the concatenation of symbols in an alphabet. Importantly, we are free to pick this language constrained by the above requirement—one string must be in it, the other out. This means the membership of other strings is of no importance.

Taking Professor Lewis's suggestion we considered a singleton alphabet $\Sigma = \{a\}$. Strings represented in this way differ only in length—they are essentially numbers. One suggestion for how to differentiate the strings was to use an idea from modular counting. Since every number has a unique prime factorization there is a least prime factor which one of the values has but the other does not. For example, if the values are 15 and 16 the prime factorizations are $5 * 3$ and 2^4. Two is the least prime factor one has the other does not. Using a two state DFA that performs counting mod 2 a different state will be reached for 15 as opposed to 16. Picking one state to be our accept state (it does not matter which) distinguishes these strings. In fact, this technique distinguishes all even and odd length strings.

One more example would be 7 and 21. Seven is prime and 21's prime factor-

ization is $7 * 3$. Taking the least element from the union of both sets with their intersection removes gives three. It's easy to see a three state DFA counting length will disambiguate the two values. For seven it will end in the first state, for 21 it will end in the start state. Here we have to be more careful about which state is accepting (the second state won't work). Picking the start state and using this technique will always effectively disambiguate.

We did not prove this was an optimal technique for disambiguating strings over a singleton alphabet. In fact, here's a simple counterexample. Taking two primes, e.g. 7 and 23, we can see their result differs mod 3 ($7 \bmod 3 = 1$; $23 \bmod 3 = 2$). By our method we would have used a seven state DFA when a three state DFA suffices.

1.4 Myhill-Nerode

The "Myhill-Nerode theorem" (it is a formal statement, but we'll soon see why quotes are appropriate) is a conjunction of ideas from Myhill and Nerode. The original "proof" of the theorem is far removed from our subject at first glance. In an excellent example of the emerging field changing over time into more digestible textbook entries their work was transformed and gives us the following two relevant statements:

1. A language L is regular iff the distinguishing extension relation (defined below) on strings in the language has a finite number of equivalence classes.
2. The smallest DFA recognizing L has states equal to the number of equivalence classes in the distinguishing extension relation.

For a language L and strings x and y, a "distinguishing extension" is a string z such that only one of xz and yz are in L. The relation, R_L, means there is no distinguishing extension for the strings. This relation is an equivalence relation (transitive, reflexive, symmetric) and gives us the needed equivalence classes above.

Some examples may be helpful.

$L = \{0^n 1^n : n \geq 0\}$. We want to show an infinite number of strings (an infinite subset of Σ^*) have distinguishing relations, implying this language is not regular. Consider the infinite subset of strings $S = \{0^*\}$. Picking arbitrary $m, n \in \mathrm{Z}^+, m \neq n$, we can see that 0^m and 0^n have a distinguishing extension (either 1^m or 1^n). Since S is an infinite subset and any pair of strings has a distinguishing extension there are an infinite number of equivalence classes for strings in S relative to L. This implies L is not regular.

$L = \{ww : w \in \{a, b\}^*\}$. Once again we consider an infinite subset of $\Sigma^*, S = \{a^n b : n \geq 0\}$. Any two elements of this set have distinct numbers of zeros, m, n

and thus distinct distinguishing extensions $a^n b$ and $a^m b$. This also provides an infinite number of equivalence classes, showing the language is not regular.

A considerable amount of discussion is available online for those of you interested further.

The simples proof I have seen is by Kalyanasundaram.

Lemma 1. If L is recognized by a DFA with k states, then $index(L) \leq k$.

Lemma 2. If $index(L) = k < \infty$ then there exists a DFA with k states that recognizes L.

Given these two lemmas we can prove the theorem. Since every regular language is recognized by at least one DFA which has a finite number of states the number of equivalence classes of L by our distinguishing relation must be finite, too (lemma 1). Now assume a finite number of equivalence clases, then there exists a DFA recognizing L (lemma 2). Futher, both lemmas give us that the size of the DFA in states equals the number of equivalence classes.

Proof of Lemma 1.

Proof by contradiction. L is recognized by a k state DFA, M with start state q_0 but $index(L) > k$. This means there exists a set $|S| > k$ pairwise distinguishable by L.

Since $|S| > k, \exists x, y \in S, x \neq y$ such that $\delta(q_0, x) = \delta(q_0, y)$ by the pigeonhole principle (which we discussed in lecture). We want to show x, y are pairwise indistinguishable, so consider $z \in \Sigma^*$:

$$\delta(q_0, xz) = \delta(\delta(q_o, x), z) = \delta(\delta(q_o, y), z) = \delta(q_0, yz)$$

So $xz \in L \leftrightarrow yz \in L$ and x, y are pairwise indistinguishable.

Proof of Lemma 2.

We construct the DFA M with $k = index(L)$ states. Let $S = \{s_0 \dots s_k\}$ be the largest set pairwise distinguishable by L. M has states $Q = \{q_o \dots q_k\}$corresponding to stings in S. For transitions we consider each $a \in \Sigma$ and define $\delta(q_i, a) = q_j$ by considering $z_i a$. We know $s_i a =_L s_j$ for some $s_j \in S$. Let the starting state be the string ϵ is congruent to, and the valid states be all q_i where $x_i \in L$.

Assume for now $\delta(q_i, w) = q_j \leftrightarrow s_i w =_L s_j$.

Now to prove correctness we consider $x \in L$. We know $x =_L s_i$ for some s_i such that $s_i \in L$ since $x = \epsilon x =_L x_i \implies \delta(q_0, x) = q_i$. Since $q_i \in F$ x is recognized by M.

Proof of our assumption. Induction on $|w|$.

Base case. $|w| = 0, w = \epsilon$. $\delta(q_i, w) = \delta(q_i, \epsilon) = q_i$. True since $s_i \epsilon = s_i$.

$|w| = 1$ claim is also true by definition of transition function δ.

For $|w| > 1$ we use induction. Let $w = va$ where $|v| = l$ and $|a| = 1, a \in Sigma$

$$\delta(q_i, w) = \delta(q_i, va) = \delta(\delta(q_i, v), a) = \delta(q_{j1}, a) = q_{j2}$$

where $q_{j1} = \delta(q_i, v)$ and $q_{j2} = \delta(q_{J1}, a)$ so $s_{j1} =_L x_i v$ and $x_{j2} =_L x_{j2} a$ which is true since

$$x_i w = x)iva =_L x_{j1} a =_L x_{j2}$$

1.5 Rethinking Professor Lewis's problem

Professor Lewis's problem is like a promise problem (which we'll see in this section later) for DFAs. Think of it as finding the minimum size language (well-defined per the above) such that one of a pair of strings is in it and the other out. There are, of course, an infinite number of such languages with the latter property but there must be a finite number with minimum size.

Of course, this argument depends on there always being a regular language distinguishing two strings. There must be since we can easily build a DFA distinguishing two strings by checking each element in turn using a unique state.