CS 360
Naomi Nishimura

Decision problems for regular languages

1 Problems considered

The following are examples of decision problems for regular languages discussed in class.

1. Given an NFA M and a string x, does M accept x?
2. Given a DFA M, is $L(M)$ empty?
3. Given a DFA M, is $L(M)$ finite?
4. Given two DFA's M_1 and M_2, do they accept any of the same strings?
5. Given two DFA's M_1 and M_2, is $L(M_1) \subseteq L(M_2)$?
6. Given two DFA's M_1 and M_2, do they accept the same language?
7. Given two regular expressions α_1 and α_2, is $L(\alpha_1) = L(\alpha_2)$?

2 Alternate proofs

The textbook gives one way of testing the emptiness of regular languages. Here we give another way of testing emptiness and also a way of testing finiteness.

Testing emptiness

We first show that for D a DFA with n states, $L(D)$ is nonempty if and only if D accepts a string of length less than n. One direction is trivial: If D accepts a string of length less than n, then clearly $L(D)$ is nonempty. For the other direction, we use a proof by contradiction. Suppose instead that $L(D)$ is nonempty and the length of the shortest string in $L(D)$ is at least n. We pick a shortest string w and observe that since $|w| \geq n$ and $w \in L(D)$ (which is by definition a regular language), we can conclude that we can express w as a decomposition $w = xyz$ such that $|xy| \leq n$, $|y| > 0$, and for all $k \geq 0$, the string xy^kz is in $L(D)$. In particular, for $k = 0$, $xz \in L(D)$, and since $|y| > 0$ we can conclude that $|xz| < |w|$. However this contradicts the assumption that w is a shortest string in $L(D)$.

The algorithm for testing emptiness then consists of trying each string of length less than n to see if it is accepted by D. If no such string is accepted, then we can conclude that $L(D)$ is empty. This algorithm can be executed in a finite amount of time as the number of strings of length less than n on the alphabet of D is finite and each can be processed in finite time.

Testing finiteness

We prove a result about DFA's similar to that above; we show that for D a DFA with n states, $L(D)$ is infinite if and only if D accepts a string w such that $n \leq |w| < 2n$. Again one direction is trivial: If D accepts such a string, then by the pumping lemma w can be used to form an infinite family of strings all in $L(D)$.

To see that if $L(D)$ is infinite, D accepts a string w such that $n \leq |w| < 2n$, we again make use of the pumping lemma. Suppose instead that $L(D)$ is infinite but D does not accept any string such that $n \leq |w| < 2n$. Clearly D must accept at least one string of length $2n$ or greater, since otherwise all strings $L(D)$ are of length less than n, and they are finite in number, a contradiction. Let w be a shortest string of length at least $2n$. Then by the pumping lemma, we can express w as a decomposition $w = xyz$ such that $|xy| \leq n$, $|y| > 0$, and for all $k \geq 0$, the string xy^kz is in $L(D)$. In particular, for $k = 0$, $xz \in L(D)$, and since $|y| > 0$ we can conclude that $|xz| < |w|$. If $|xz| \geq 2n$, then this contradicts the assumption that w is a shortest string of length at least $2n$. If $|xz| < 2n$, then since $|xy| \leq n$ and $|xyz| \geq 2n$, we can conclude that $|xz| \geq n$, contradicting the assumption that $L(D)$ contains no strings of length $n \leq |xz| < 2n$. In either case, the proof is complete.

The result implies an algorithm: we try all strings of length $n \leq |w| < 2n$. Again, the number of strings is finite. Either at least one is accepted and hence $L(D)$ is infinite, or by the result above, $L(D)$ is finite.