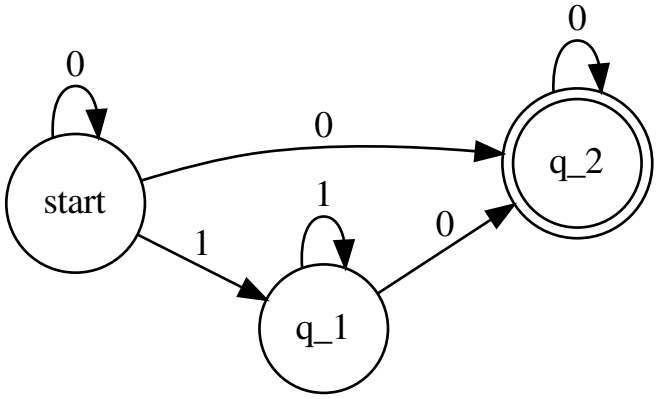
Homework #3
Automata and Computation Theory
Fall 2018

Written by Eric Rothman

September 27, 2018

1 Problem 1

Give an NFA (both a state diagram and a formal description) recognizing the language 010^+ with three states. The alphabet is $\{0, 1\}$. You dont need to have a formal proof. Automoata M:



Formal Definition: $M = \{\{start, q\_1, q\_2\}, \{0, 1\}, \delta, start, \{q\_2\}\}$

$\delta =$

	0	1	ϵ
start	$\{start, q\_2\}$	q_1	$\emptyset$
q_1	$\{q\_2\}$	$\{q\_1\}$	$\emptyset$
q_2	$\{q\_2\}$	$\emptyset$	$\emptyset$

2 Prob 2

This question studies the number of states in a DFA equivalent to an NFA. Recall that in class we showed an NFA with 4 states that recognizes the language which consists of all binary strings that have a 1 in the third position from the end. For any integer k, it is easy to generalize this construction to an NFA with k + 1 states that recognizes the language which consists of all binary strings that have a 1 in the kth position from the end. The general transformation from an NFA to a DFA will give us a DFA with at most 2^{k+1} states recognizing the same language.

Show that, any DFA that recognizes the same language must have at least 2^k states.

Smallest DFA

This is not necessary for the proof, but I thought I might as well show that it is possible to make a DFA that recognizes the language in only 2^k states.

Automata $M_1 = \{Q, 0, 1, \delta, q_0, F\}$

Where: 1) $Q = \{0, 1\}^k$. So Q consists of a different state for every possible binary string of length k. The idea for this automata is that it keeps track of the last k states that have been inputted. Each state is written as $w_1 w_2 ... w_k$ where each $w \in \{0, 1\}$

2) $q_0 = 0^k$. The automata pretends that the string being inputted is concatenated with a string of 0 of length k. This makes it so that no string less than length k can be accepted.

3) $F = \{w_1 w_2 ... w_k \in Q | w_1 = 1\}$ This is all the states where the spot k away from the end is 1

4) $\delta(w_1 w_2 ... w_k, b) = w_2 w_3 .... w_k b$ Each state transitions to the state with w_1 removed from the string and b appended to it.

Proof of algorithm:
Let $a \in L(M_1)$
So for the last k entries of $a = w_1 w_2 ... w_k$, $w_1 = 1$
So the value in a k positions away from the end is 1
So $a \in$ language.
Now let $a \in$ language
So the value in a k positions away from the end is 1
So for the last k entries of $a = w_1 w_2 ... w_k$, $w_1 = 1$
So $a \in L(M_1)$.
So $L(M_1) =$ specified language.

Proof

Prove that any DFA that recognizes the same language must have at least 2^k states.

FSOC let $\exists$ DFA M_2 such that M_2 has less than 2^k states and M_2 recognizes the language which consists of all binary strings that have a 1 in the kth position from the end.

Since M_2 has less than 2^k states, it can "remember" less than 2^k pieces of unique information.

Now let $a, b \in \{0,1\}^k$ such that a and b are different strings. So $\exists i \in \mathcal{Z}$ such that $1 \leq i \leq k$ and $a_i \neq b_i$. Since a anb b are different strings, there must be a position in each of them that do not match, so an i must exist.

Since $a_i \neq b_i$, one of them is a 1 and the other is a 0.

WLOG let $a_i = 1$ and $b_i = 0$.

Now let $z = 0^{i-1}$.

Since $a_i = 1$ and $b_i = 0$, az should be accepted by M_2 and bz should not be accepted.

So the i'th bit must be remembered by M_2 so that it can differentiate between the az and bz.

Since there are k different values that i could be, M_2 must remember all k possibilities that i might be, which is the last k values inputted from the input string.

Since each k position can have two different possibilities, M_2 must have the capacity to remember all 2^k possibilities for the input string.

This is a contradiction since M_2 can only remember less than 2^k unique things, and each of the 2^k possibilities are unique.

So to remember them all M_2 would need atleast 2^k states.

Since there is a contradiction, the claim must be true.

So any DFA that recognizes the language which consists of all binary strings that have a 1 in the kth position from the end must have atleast 2^k states.

3 Problem 3

Say that string x is a prefix of string y if a string z exists where $xz = y$ and that x is a proper prefix of y if in addition $x \neq y$. Let A be a regular language. Show that the class of regular languages is closed under the following operation.

NOEXTEND(A) = $\{w \in A | w$ is not the proper prefix of any string in $A\}$

Let A be recognized by DFA $M = \{Q, \mathcal{E}, \delta, q_0, F\}$ The construction written below will not work for NFA since whether or not something is a proper prefix may be dependent on all current states at time of string completion. That can't be properly expressed in NFA where whether or not something is recognized only depends on one instance, not what every current state is at time of completion.

Construction

This will be proven later in claim 1, but an element is a proper prefix if there is a way to get to an accept state from another accept state in a DFA.

Since every regular language has a DFA, we can remove all proper prefixes by making all accept states from which there is a way to get to another accept state not accept states.

So in other words, remove all accept states that have a path to another accept state, including itself, from the set of accept states.

Luckily there is an easy way to find if there is a path from any node q to any other node s in a directed graph, which is the DFS.

Let $F_2 = \{w \in F |$ when DFS is run on M starting at w, it encounters at least one $b \in F\}$

The only thing that should be changed to exclude the proper prefixes is which states count as accept states since the rest of the elements in the language need to run through the automata as normal.

So the automata for NOEXTEND(A) is:

$M_2 = \{Q, \mathcal{E}, \delta, q_0, F/F2\}$

Claim 1

String $a \in A$ is a proper prefix iff there is a path from the accept state a ends on when run through M_1 to any accept state in F.

First Direction

PROVE: If string $a \in A$ is a proper prefix then there is a path from the accept state a ends on when run through M_1 to any accept state in F.

Let $a \in A$ be a proper prefix So $\exists z \in \mathcal{E}^*$ such that $y = az$, $y \in A$ and $a \neq y$.

FSOC let there not be a path from the accept state a ends on, arbitrarily called q_0 when run through M_1 to any accept state in F

Since there is no path from a to any accept state there is no circular transitions from q_0.

Since M_1 is a DFA, there must be two transitions from q_0.

Since there are no circular transitions, both transitions must be to a different state.

Since the automata is a DFA, there is only on current state at a time.

So when y is passed through M_1, since the first part of y is a, it will arrive at q_0.

Then since $a \neq y$ and y is created from concating a with another string, the length of y is greater than the lenght of a.

So when y is passed through M_1, it will arrive at q_0 with input left.

So it will move past q_0.

Since there is only one current state at a time, when y is past q_0, it must travel along a path and arrive at another accept state to be accepted.

Since there is no path from q_0 to any accept state, y is not able to arrive at any accept state.

So $y \notin L(M_1)$

So $y \notin A$

This is a contradiction since y was defined as being in A.

Since there is a contradiction, the statement must be true.

So if string $a \in A$ is a proper prefix then there is a path from the accept state a ends on when run through M_1 to any accept state in F.

Other Way

Let $a \in A$. PROVE: If there is a path from the accept state a ends on when run through M_1 to any accept state in F then string a is a proper prefix

FSOC let a not be a proper prefix.

So there is not a $z \in \mathcal{E}^*$ such that $y = az$, $y \in A$, and $y \neq a$.

Now lets look at the state a ends on when run through M_1, and lets call it q_1 arbitrarily.

There exists a path from q_1 to an accept state in F.

So there exists a string $b \in \mathcal{E}^*$ such that when the current state is q_1 and b is inputted into the automata, the automata will end at an accept state and $b \neq \epsilon$.

That is because every path in a DFA can be traversed by inputting the correct string, since every transition in an automata is connected to a certain symbol in the language.

If there was not a string inputted that could get from q_1 to an accept state, there would not be a path from q_1 to that accept state.

Now let $y = ab$.

Since $a \in A$, $a \in \mathcal{E}^*$ and since $b \in \mathcal{E}^*$, y is a valid input string for the DFA.

When y is inputted into M_1 it will first go to q_1 since it begins with a, and a goes to q_1 when inputted into M_1.

Then it will traverse the automata starting at q_1 with the input of b.

This was defined such that when b was inputted and the state is q_1, the automata will traverse to an accept state.

So the current state becomes another accept state.

Then the automata stops since the string is all inputted.

So y ends on an accept state, so $y \in A$.

So $y = ab$, $y \in A$, and $y \neq a$ since $b \neq \epsilon$.

This is a contradiction with the fact that there is not a $z \in \mathcal{E}^*$ such that $y = az$, $y \in A$, and $y \neq a$.

Since there is a contradiction when assumed false, the statement must be true.

So if there is a path from the accept state a ends on when run through M_1 to any accept state in F then string a is a proper prefix

Conclusion

Since both directions are proven, it must be true that string $a \in A$ is a proper prefix iff there is a path from the accept state a ends on when run through M_1 to any accept state in F.

Proof

First Direction

$NOEXTEND(A) \subset L(M_2)$

let $a \in NOEXTEND(A)$ be an arbitrary element.

Since $a \in NOEXTEND(A)$, a is not a proper prefix and $a \in A$.

Since $a \in A$, when put through M_1 it ends on an accept state. Let that state arbitrary be called q_1.

By the fact a is not a proper prefix and by Claim I, there is no path from q_1 to any accept state in F.

Since there is not path from q_1 to an accept state in F, no path from q_1 to any accept state was found when DFS was run starting at q_1 since DFS has been proven to return all nodes that the starting node has a path to, and only those nodes.

So $q_1 \notin F2$.

So $q_1 \in F/F2$.

So q_1 is an accept state of M_2

Since the only thing that changed between M_1 and M_2 is which states are accept states, and q_1 is still and accept state in M_2, when a is put through M_2, it still ends on q_1.

So $a \in L(M_2)$.

So $NOEXTEND(A) \subset L(M_2)$

Other Direction

$L(M_2) \subset NOEXTEND(A)$.

let $b \in L(M_2)$ be an arbitrary element.

So when run through M_2, the automata ends on an accept state, arbitrarily named q_2.

Since q_2 is an accept state of M_2, $q_2 \in F$ and $q_2 \notin F2$

Since $q_2 \in F$, q_2 is an accept state of M_1.

Since the only thing that changed between M_1 and M_2 is which states are accept states, and q_1 is still an accept state in M_1, when b is put through M_1, it still ends on q_2.

So $b \in A$.

Since $q_2 \notin F2$ and $q_2 \in F$, when DFS was run on M_1 starting at q_2, it did not find a path to any accept state of M_1.

Since DFS has been proven to return every node that the starting node has a path to and since DFS did not return any accept state when started at q_2, there is no path from q_2 to any state in F.

Since there are no paths from q_2 to any accept state of M_1, by Claim I any string that ends on q_2 is not a proper prefix.

So b is not a proper prefix.

Since b is not a proper prefix and $b \in A$, $b \in NOEXTEND(A)$.

So $L(M_2) \subset NOEXTEND(A)$

Conclusion

Since $L(M_2) \subset NOEXTEND(A)$ and $NOEXTEND(A) \subset L(M_2)$, $L(M_2) = NOEXTEND(A)$.

So the automata recognizes NOEXTEND(A).

So NOEXTEND(A) is a regular language if A is a regular language.

4 Problem 4

Let $\sum = \{0, 1\}$.

a

Write a regular expression for the language L consisting of all strings in $\mathcal{E}^*$ with exactly one occurrence of the substring 111.

$$(0 \cup 01 \cup 011)^*111(0 \cup 01 \cup 011)^*$$

b

Write a regular expression for the language L consisting of all strings in $\mathcal{E}$ that do not end with 10.

$$\epsilon \cup 0 \cup (1 \cup 0)^*(00 \cup 1)$$