Riley Rice
CS321
10-23-2024

Homework 2

Problem 1: [5 points]

Suppose $M_1 = (Q_1, \sum, \delta_1, S_1, F_1)$ and $M_2 = (Q_2, \sum, \delta_2, S_2, F_2)$ are two DFAs over the same alphabet. Construct another DFA that decides the language $L(M_1) \setminus L(M_2)$, i.e., the language consisting of all strings in $L(M_1)$ but not in $L(M_2)$. You must describe the DFA using the formal definition. (This problem is about DFAs only, and has nothing to do with NFAs or ϵ-NFAs.)

Hint: Use the theorems in Lec 4, i.e., regular languages are closed under set intersection and complement.

Solution:

This new DFA $L(M') = L(M_1) \setminus L(M_2)$, i.e M' is equivalent to:

$Q' = Q_1 \times Q_2$ (Use all the states from both DFAs)
$\sum = \sum$ (Uses the same alphabet as $L(M_1)$ and $L(M_2)$)
$\delta'((q_1, q_2), c) = (\delta_1(q_1, c), \delta_2(q_2, c))$ (Use both transition functions)
$S' = (S_1, S_2)$ (Use both starting states)
$F' = \{(q_1, q_2) \in Q' | q_1 \in F_1 \text{ and } q_2 \notin F_2\}$ (Only accept when the state is accepted in M_1 and not M_2)

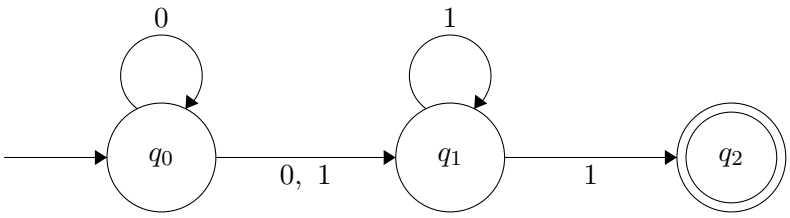
Explanation:

In order to come to this I broke down what $L(M_1) \setminus L(M_2)$ means which is equivalent to $L(M_1) \cap \overline{L(M_2)}$. What this means is that we want to find all the strings in $L(M_1)$ that aren't in $L(M_2)$. The good thing about looking at it this way is that we have some theorems for complements and the intersection of DFAs that can be used to create the new DFA and it's a lot easier to apply them in this form. The first thing we want to do is find the complement for $L(M_2)$ which is defined by making $F' = Q \setminus F = \{x \in Q | x \notin F\}$, or in other words accept on all the elements that are in set Q but not in set F which essentially inverts what is accepted for a DFA. You can see that I created the complement of $L(M_2)$ by accepting on states that are accepted by $L(M_1)$ but not accepted by $L(M_2)$ this makes it so the new DFA will only accept on strings that are in $L(M_1)$ but not $L(M_2)$. Then with the accepting states figured out now we have to get the intersection of $L(M_1)$ and $\overline{L(M_2)}$ which is pretty easy. According to the theorem all we have to do is run the two DFAs in parallel which I do by setting Q' to the cartesian product of the states from both DFAs. I then run both the transition functions and use the two starting states at the same time. Finally we have created a new DFA $L(M')$ which is equivalent to $L(M_1) \cap \overline{L(M_2)}$ which is what we are looking for.

Problem 2: [10 points]

Use the algorithm discussed in class to convert the following NFA to an equivalent DFA, i.e., a DFA that decides the same language as this NFA does. List all steps of the execution of the algorithm. Do not include any states that are unreachable. You can describe the DFA either in the form of a transition graph or using the formal definition; if you do the latter, it must contain all five components.

NFA:

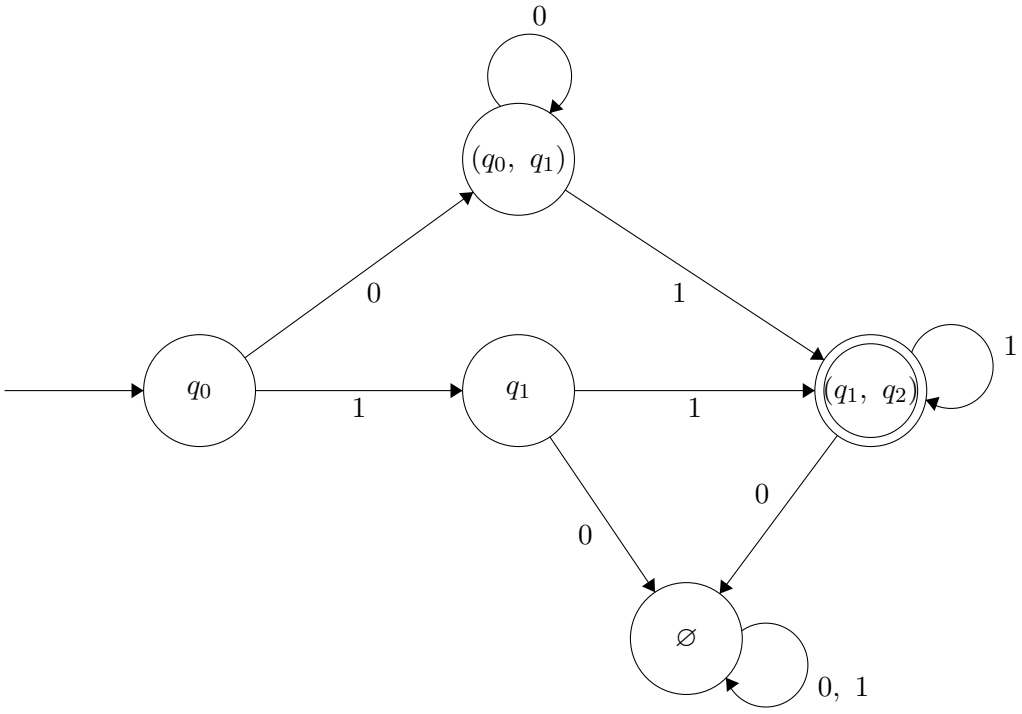


Solution:

NFA $\rightarrow$ DFA Conversion Table:

	0	1
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_1\}$
$\{q_1\}$	$\varnothing$	$\{q_1, q_2\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
$\{q_1, q_2\} \in F$	$\varnothing$	$\{q_1, q_2\}$
$\varnothing$	$\varnothing$	$\varnothing$

Equivalent DFA Transition Graph:



Problem 3: [5 points]

For any string ω (over some alphabet $\sum$), let $\tilde{\omega}$ denote the reverse of ω, i.e., writing ω from right to left. For example, if $\omega = bbbaba$ then $\tilde{\omega} = ababbb$. Furthermore, for any language L, let

$$\tilde{L} = \{\tilde{\omega} | \omega \in L\},$$

i.e., $\tilde{L}$ is the set of all strings that are the reverse of some string in L.

Prove that if L is a regular language, then $\tilde{L}$ is also a regular language.

Hint: There might be multiple ways to prove this. I personally think the easiest way is to construct a regular expression that represents $\tilde{L}$.

Solution:

Note: Regular expressions can only represent regular languages

let L be a regular language over the alphabet $\sum = \{a, b\}$ which can be represented by the regular expression $r = a^*b$ or all strings that consist of zero or more a's followed by a b. Some example strings in L are:

- b
- ab
- aab
- $aaab$

We now want to find $\tilde{L}$, or the set of strings that are the reverse of every string in L. So let's reverse the regular expression r into $\tilde{r} = \tilde{a^*b} = ba^*$, and so $\tilde{r} = ba^*$, which only uses closed operations, and represents the reverse regular language $\tilde{L}$. This represents all strings that start with a b that are followed by zero or more a's. Some example strings in $\tilde{L}$ are:

- b
- ba
- ba
- $baaa$

To verify this, let's take some example strings from L and reverse them and make sure that they are in $\tilde{L}$.

- If $\omega = b \in L$, then $\tilde{\omega} = b \in \tilde{L}$ (True)
- If $\omega = ab \in L$, then $\tilde{\omega} = ba \in \tilde{L}$ (True)
- If $\omega = aab \in L$, then $\tilde{\omega} = baa \in \tilde{L}$ (True)

All the reversed strings are in $\tilde{L}$ and is accurately represented by the reversed regular expression $\tilde{r} = ba^*$ which proves that the languages L and $\tilde{L}$ are indeed regular and closed under reversal.

QED