Rhea Mae Edwards

CS 321: Homework #3

1. a) {$w \in \{a, b\}^*$ | w has an even number of b's}

   (a*ba*ba*)*

   b) {$w \in \{a, b\}^*$ | w does not contain the substring ab}

   b*a*

   c) {$w \in \{a, b\}^*$ | w contains substring ab an even number of times}

   (a*(ab)b*a*(ab)b*)*

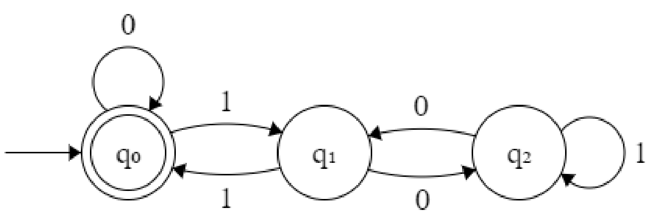
   d) {$w \in \{a, b\}^*$ | w has an even number of a's and even number of b's}

   ((bb)*(abab)*(aa)*(baba)*)*

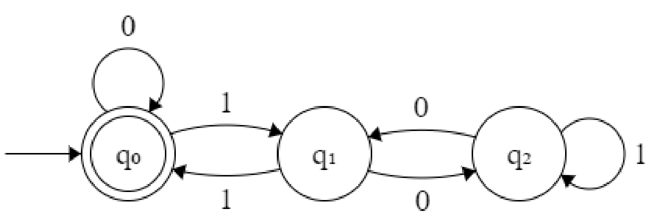
   A number of all of the minimally described possible combinations of having an even number of a's and an even number of b's in an expression.

2. L(M) = {$w \in \{0, 1\}^*$ | w and rev(w) are both a multiple of 3}   //Prove this.

   w DFA:



   rev(w) DFA:



   They're DFAs are equal!

   Overall, you would first reverse all of the transitions' directions, and then swap the accept state and the start state; leading us to have equal DFAs up above.

   How so? Well…

   The idea is with $w \in M$, there is a path from state q_0 to state q_1 when following transitions by reading through the characters of w. There is also a path from state q_1 to state q_0 following those transitions backwards when reading through the character in the reverse, also known as rev(w).

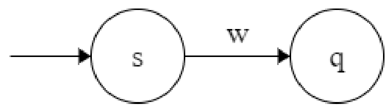
3.

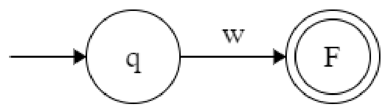
Warmup/Partial Credit:

$\{qw \mid q \in Q$ and $w \in \Sigma^*$ and $\delta^*(s, w) = q$ and $\delta^*(q, w) \in F\}$ //Show that language is regular.

Note: String in this language consist of one character from Q and then any number of character from Σ.

Two finger construction with comparing the two statements:





"$q \in Q$ and $w \in \Sigma^*$ and $\delta^*(s, w) = q$" roughly meaning starting at state *s*, you read in a string of characters *w*, you would then transfer to state *q*.

"q and $\delta^*(q, w) \in F$" roughly meaning being in state *q*, you read in a string of characters *w*, you would transition to the accepting state *F*.

With this two finger construction comparison, the overall transition function can be represented as…

$\delta^*((s, q), w) = (q, F)$

Full Question:

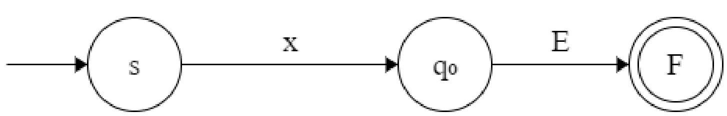
If A is a language over alphabet Σ, define:

$$\text{undouble}(A) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid ww \in A\}.$$

Show that if A is regular, then so is undouble(A).

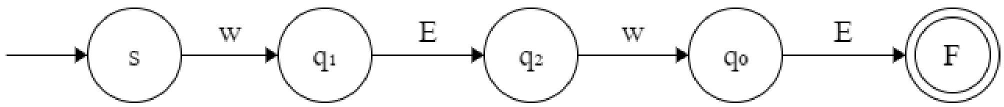
Example: if $A = \{\varepsilon, 0, 11, 0010, 0101\}$ then undouble(A) $= \{\varepsilon, 1, 01\}$.

Proving that A is regular, means that there is a NFA along with a regular expression that would both accepts parts of A. Having *x* representing a regular expression, there's a following NFA accepting A…



Including an epsilon transition (E) as a precaution at this level of thinking, and a state q_0 being a state after reading *x*.

Let's now describe *x* as *ww*, where $w \in A$. According to the problem statement, *w* will be accepted by undouble(A), $w \in$ undouble(A). Now applying this redefined variable to the NFA described above…



For the epsilon transition between states q_1 and q_2, would be where the NFA "guesses" that it's at the midpoint of a string being passed through this NFA. Then following, if *w* read again through the NFA, it'll be accepted at the final state.

The transition function representing this NFA can be represented as…

$\delta^*((s, q_2), w\varepsilon) = (q_2, f)$

We can conclude that strings (*w*) that are accepted by undouble(A) are accepted by this NFA.

With the NFA above to represent undouble(A), we can say that undouble(A) is regular.