COMP0003 Theory of Computation
Exercises I: DFAs, NFAs, and regexes

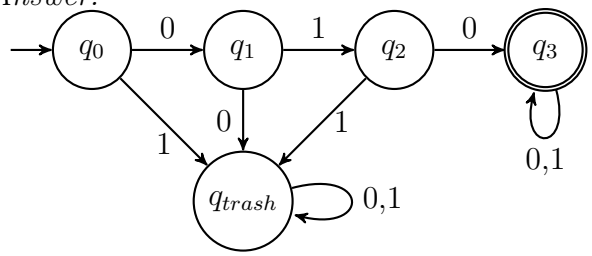
Yuzuko Nakamura

19 & 21 November 2019

1 DFAs

Exercise 1. Construct a DFA (pictorially and in tuple form) that accepts all strings that start with "`010`" ($\Sigma = \{\texttt{0}, \texttt{1}\}$).

Answer:



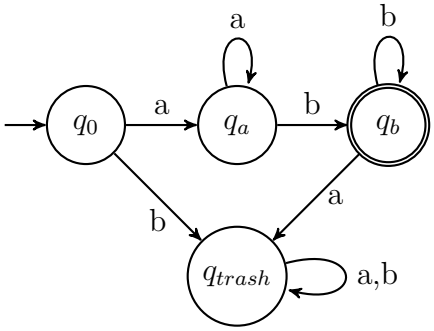
$Q = \{q_0, q_1, q_2, q_3, q_{trash}\}$; $\Sigma = \{\texttt{0}, \texttt{1}\}$; $q_0 = q_0$; $F = \{q_3\}$

q	s	$\delta(q, s)$
q_0	0	q_1
q_0	1	q_{trash}
q_1	0	q_{trash}
q_1	1	q_2
q_2	0	q_3
q_2	1	q_{trash}
q_3	0	q_3
q_3	1	q_3
q_{trash}	0	q_{trash}
q_{trash}	1	q_{trash}

It's okay if they leave out q_{trash} from their diagram, but it should appear in their mathematical specification

Exercise 2. Construct a DFA (pictorially and in tuple form) over the alphabet $\Sigma = \{\texttt{a}, \texttt{b}\}$ that accepts strings with one or more `a`'s followed by one or more `b`'s (for example, should accept "`ab`", "`aaab`", and "`aaaabbbbbb`" but not "`bbb`", "`abba`", or "`ba`".

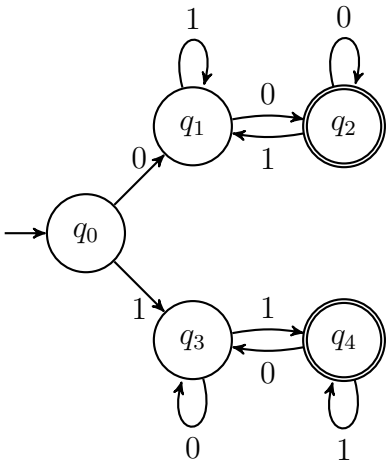
Answer:



$Q = \{q_0, q_a, q_b, q_{trash}\}$; $\Sigma = \{\texttt{a}, \texttt{b}\}$; $q_0 = q_0$; $F = \{q_b\}$

q	s	$\delta(q, s)$
q_0	a	q_a
q_0	b	q_{trash}
q_a	a	q_a
q_a	b	q_b
q_b	a	q_{trash}
q_b	b	q_b
q_{trash}	0	q_{trash}
q_{trash}	1	q_{trash}

Exercise 3. What is the language of the following DFA?

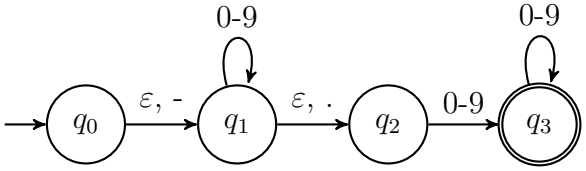


Answer: The set of all strings that begin and end with `0` *or that begin and end with* `1`*. Can also be written in set-builder notation. Can also be written as a regex (see Exercise 7).*

2 NFAs

Exercise 4. Create an NFA that recognizes simple, properly-formatted numbers. The alphabet is $\Sigma = \{$`-, ., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9`$\}$, and a properly formatted number consists of (optional) a sign (`-`), (optional) zero or more digits in front of a decimal point, followed by one or more digits.

Answer:



Exercise 5. In class, we showed that DFAs are closed under complement by switching the set of accepting and non-accepting states. Would the same technique work with NFAs? Why or why not?

Answer: No. The technique shown in class relied on the deterministic nature of DFAs – that reading in the same input would reliably put you in one particular state at the end of input. This property goes away for NFAs. Switching the set of accepting and non-accepting states might not reject all the strings that used to be accepted previously. The only strings that would be rejected would be ones where ALL traces were in an accept state in the NFA, which is too permissive.

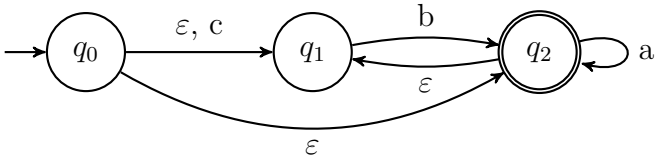
Exercise 6. Use NFAs to show that regular languages are closed under reversal. That is, if L is a regular language, then $L^R = \{w \mid w^R \text{ (w written backwards)} \in L\}$ is also a regular language.

Answer: Given NFA N, create NFA N' to be the same as N except: Reverse the direction of arrows; make a new start state connected by ε transitions to all accepting states (and turn those states non-accepting), and make the original start state an accept state.

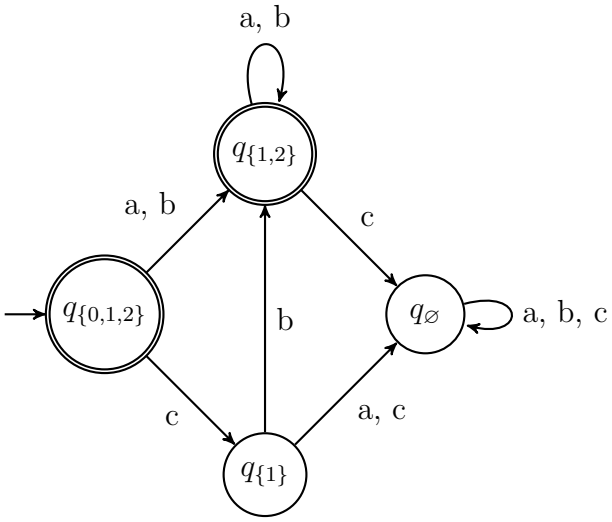
If a string s was accepted by the original NFA, then it can be written as an equivalent string $w = w_0...w_n$ and we know that we start in q_0, end in $q_n \in F$, and transition legitimately: $q_i \in \delta(q_{i-1}, w_{i-1})$. In our reverse NFA, we can take a string equivalent to s^R, $\varepsilon w^R = \varepsilon w_n...w_0$. The first epsilon gets us from the new initial state to $q_n \in F$. From there, if there was an edge from q_{i-1} to q_i, then in our reverse NFA, there is an edge from q_i to q_{i-1} that we can take. This will land us eventually in q_0, which is now our accept state, so we accept.

If a string s was not accepted by the original NFA, then there was no way to get from q_0 to an accept state in the original NFA. In our reverse NFA, this behavior will not change. Each string w equivalent to s (with some potential ε's added in) will get stuck in at least one state, unable to make a valid transition, and such a state will always be encountered in the reverse direction as well. The only change we made was to add ε arrows from the new start state to the old accepting states. However, this doesn't allow us to accept a string that would have been rejected by the original NFA.

Exercise 7. Convert the following NFA to an equivalent DFA.



Answer: Something like this:



They may choose to include more states (not reachable) or omit the trash ($q_\varnothing$) state.

3 Regexes

Exercise 8. Write the languages featured in Exercises 1-4 and 7 as regular expressions.

Answer:
Exercise 1: $010\Sigma^*$
Exercise 2: a^+b^+ *or* aa^*bb^*
Exercise 3: $(0\Sigma^*0)\cup(1\Sigma^*1)$
Exercise 4: $(\varepsilon \cup -)(0\text{-}9)^*(\varepsilon \cup .)(0\text{-}9)^+$
Exercise 7: $\varepsilon\cup(a\cup b)^+\cup cb(a\cup b)^*$