York University
CSE 2001 – Unit 3.0 DFA Machines and Simple loop Java programs, NFA, Regular Expressions, and Extended Regular Expressions

Instructor: Jeff Edmonds

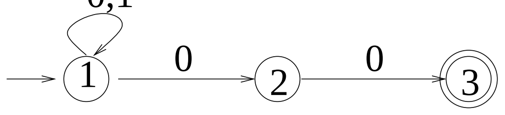
Don't cheat by looking at these answers prematurely.

1. Suppose that the CPU of a DFA is required to remember the current value of x and of y where x is a 100 digit number and $y \in \{1, 2, 3, 4, 5\}$. How many states does this DFA require?

   - Answer: $10^{100} \times 5$

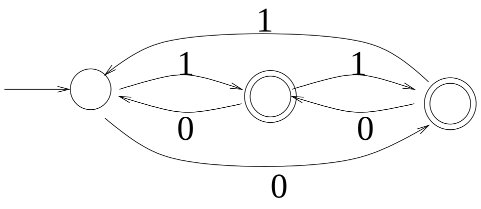
2. Give an NFA for the language $L_1 = \{\omega \in \{0,1\}^* \mid \omega \text{ ends in } 00\}$. Also give a regular expression for it.

   - Answer:



   $\{0,1\}^*00$

3. Consider the following DFA



   (a) If the number of 0's in the input string is 28 and the number of 1's is 10, does this DFA: Definitely-Accept Definitely-Reject Depends-on-the-order-of-the-characters

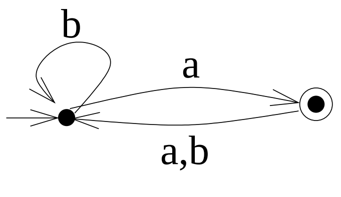
   - Answer: A 0 makes the DFA cycle forward around the circle and a 1 makes it cycle backwards. Hence, we move 28-10=18=0 mod 3. We are definitely back at node the first node. So we reject.

   (b) Denote the language accepted by the above machine with L_2. Describe this language.

   - Answer: $\{w \mid$ # of 0s - # of 1s $\neq 0$ mod 3 $\}$ or $\{w \mid$ # of 1s - # of 0s $\neq 0$ mod 3 $\}$.

4. Let $L = \{\alpha \in \{a,b\}^* \mid \alpha$ ends in a block of a's and this block has odd length $\}$. For example $\alpha = aabaabaaa$ is accepted because the last block of a's has length three which is odd. However $\alpha = aabaab$ does not end in a block of a's and if you wanted to say that it does then this hypothetical block would have length zero which is even. Either way, this string is rejected. Build a DFA for this.

   - Answer:



   (a) A DFA is defined as $M = \langle Q, \Sigma, \delta, q_{start}, F\rangle$. For your DFA, what are each of these: Q, Σ, δ, q_{start}, and F?

- Answer: $Q = \{q_{even}, q_{odd}\}$, $\Sigma = \{a, b\}$, $q_{start} = q_{even}$, $F = \{q_3\}$, and $\delta =$

δ	a	b
q_{even}	q_{odd}	q_{even}
q_{odd}	q_{even}	q_{even}

(b) For each state, what does the the CPU "knows" about the input string read in so far.

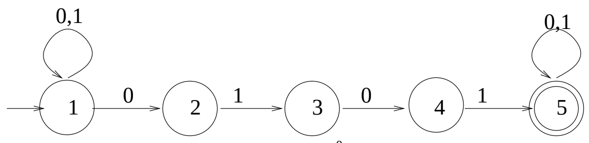
- Answer: Whenever an a is read, the machine toggles between the two states, q_{even} which says that the end block of a's has even length and q_{odd} that its length is odd. Whenever a b is read, the length of the end block of a's goes to zero, and hence the state is reset to q_{even}.

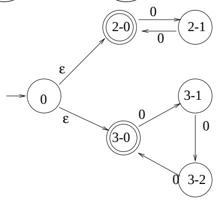
(c) Give a regular expression of L.

- Answer: $a(aa)^* \cup \{a,b\}^* ba(aa)^*$

5. Consider a new kind of finite automaton called an *all-paths-NFA*. An all-paths-NFA M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that recognizes $x \in \Sigma^*$ if *every* possible computation of M on x ends in a state from F. Note, in contrast, that an ordinary NFA accepts a string if *some* computation ends in an accept state.

(a) Consider the following two machines. What language is accepted if these machines are viewed as all-paths-NFA?





- Answer:
  i. $L(M) = \emptyset$. Every string has a computation path that is not accepted, namely the one that stays in the first state.
  ii. $L(m) = \{w \in \{0\}^* \mid |w| = 0 \ mod\ 2 \ and\ |w| = 0 \ mod\ 3\} = \{w \in \{0\}^* \mid |w| = 0 \ mod\ 6\}$. Each string has two computation paths. One ends in a $2-?$ state and the other that ends in a $3-?$ state. For both of these to end in an accept state, the number of zeros must be 0 mod 6.

(b) Prove that all-paths-NFA recognize the same class of languages as regular NFA. I.E. Given an all-paths-NFA M, covert it into a regular NFA M' such that $L(M) = L(M')$. Similarly, given a regular NFA M', covert it into an all-paths-NFA M such that $L_{\text{all-paths-NFA}}(M) = L(M')$.

- Answer:

First Proof: An all-paths-NFA M can be covert into a DFA directly using the clone method given in class, with only one change. Recall the locations of the clones indicate where all the computation paths end. Instead of accepting the string if there is a clone on at least one accept state, accept if all the clones are on accept states. More formally, instead of defining the accepting states of the DFA to be $F_{DFA} = \{S \mid S$ contains an accepting state of $M\}$, define it to be $F_{DFA} = \{S \mid S$ contains only accepting states of $M\}$.

Second Proof:

$\Rightarrow$**:** [all-paths-NFA M accepts string w]
iff [all computation paths through M on input w lead to an accept state]
iff [all computation paths through $\overline{M}$ on input w lead to a reject state, where $\overline{M}$ is the same as M with accept and reject states switched, ie $\overline{F} = Q - F$]
iff [regular NFA $\overline{M}$ rejects string w]

iff [DFA $\overline{M'}$ rejects string w, where $\overline{M'}$ is the DFA constructed from NFA $\overline{M}$ using the clone method given in class]
iff [DFA M' accepts string w, where M' is the same as $\overline{M'}$ with accept and reject states switched, ie $F' = \overline{Q'} - \overline{F'}$]
In conclusion $L_{\text{all-paths-NFA}}(M) = L(M')$.

$\Leftarrow$: Similar to above.

6. One proves $(L_1^* \cup L_2^*) = (L_1 \cup L_2)^*$ by letting L_1 and L_2 be arbitrary yet unspecified languages and letting ω be an arbitrary yet unspecified string. Then you argue that if $\omega \in (L_1^* \cup L_2^*)$, then $\omega \in (L_1 \cup L_2)^*$. Conversely you prove that if $\omega \in (L_1 \cup L_2)^*$, then $\omega \in (L_1^* \cup L_2^*)$.

   One disproves $(L_1^* \cup L_2^*) = (L_1 \cup L_2)^*$ by providing concrete languages L_1 and L_2 and a concrete string ω and either proving that $(\omega \in (L_1^* \cup L_2^*)$ and $\omega \notin (L_1 \cup L_2)^*)$ or proving that $(\omega \in (L_1 \cup L_2)^*$ and $\omega \notin (L_1^* \cup L_2^*))$. Your example should be as simple as possible.

   Prove or disprove the following.

   (a) $(L_1^* \cup L_2^*) = (L_1 \cup L_2)^*$.

   - Answer: This is not true. Let $L_1 = \{a\}$ and $L_1 = \{b\}$. $(L_1^* \cup L_2^*)$ contains strings that either only contain a's or only contain b's. On the other hand, $(L_1 \cup L_2)^*$ contains strings that contains only a's and b's. Let $\omega = ab$. It is in the second and but not the first. Everything in the first is in the second.

   (b) $(L_1 \circ L_2)^* = (L_1 \cup L_2)^*$.

   - Answer: This is not true. Let $L_1 = \{a\}$ and $L_1 = \{b\}$. $(L_1 \circ L_2)^*$ contains strings whose characters are paired, one being an a and the other being a b, like $ab{\cdot}ab{\cdot}ab{\cdot}ba$. On the other hand, $(L_1 \cup L_2)^*$ can order the a's and b's in any order and have an odd length. Let $\omega = aaa$. It is in the second and but not the first.

   (c) $(L_1 \cup L_2) \circ L_3 = (L_1 \circ L_3) \cup (L_2 \circ L_3)$

   - Answer: This is true. Let L_1, L_1, and ω be arbitrary.

     $\omega \in (L_1 \cup L_2) \circ L_3$
     iff $\omega = \alpha\beta$, where $\alpha \in (L_1 \cup L_2)$ and $\beta \in L_3$
     iff $\omega = \alpha\beta$, where ($\alpha \in L_1$ or $\alpha \in L_2$) and $\beta \in L_3$
     iff $\omega = \alpha\beta$, where ($\alpha \in L_1$ and $\beta \in L_3$) or ($\alpha \in L_2$ and $\beta \in L_3$)
     iff $(\omega \in (L_1 \circ L_3))$ or $(\omega \in (L_2 \circ L_3))$
     iff $\omega \in (L_1 \circ L_3) \cup (L_2 \circ L_3)$

7. Every subset of a regular language is regular.

   - Answer: False. 0^n1^n is a subset of $\{0,1\}^*$, but the second is regular and the first is not.