Solution to Problem Set 1

CS373 - Summer 2012

Due: Wednesday June 27th at 9:00 AM

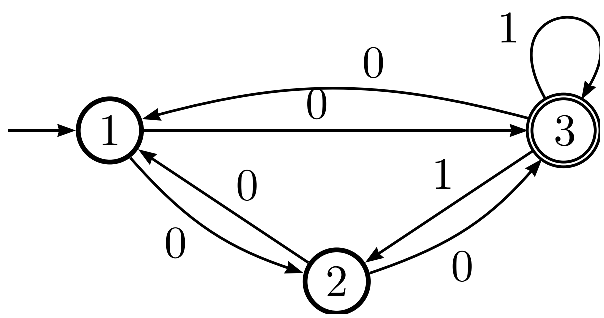
This assignment is worth 100 points.

1. NFA to DFA conversion

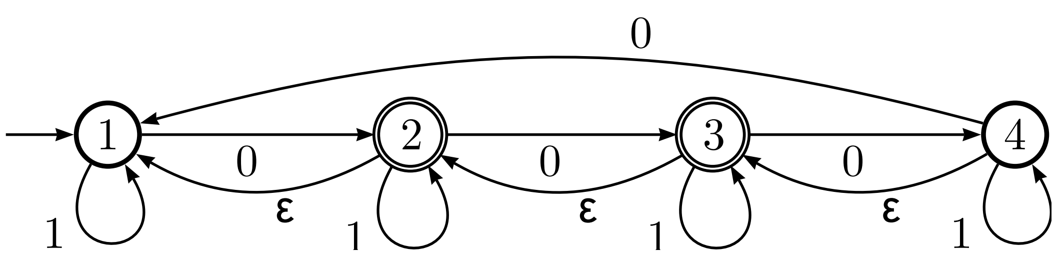
   [**Category**: Comprehension, **Points**: 10]

   Convert the following NFAs to DFAs:

   (a)

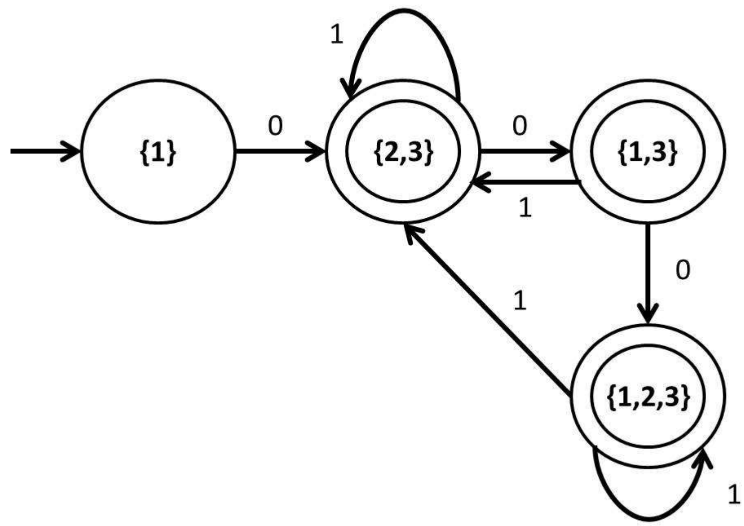


   (b)

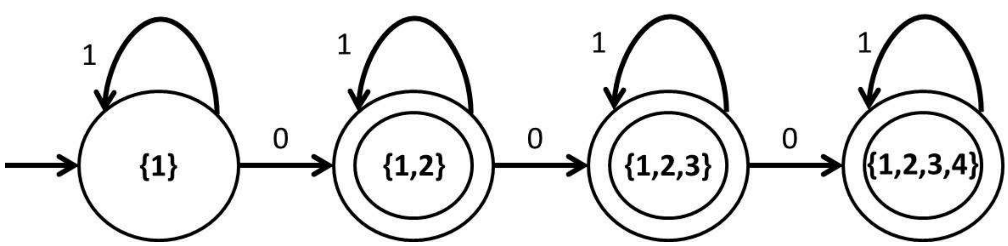


Solution:

(a)



(b)



2. Epsilon Transitions
   [**Category**: Proof, **Points**: 20]

   Provide a method for removing ε-transitions from an NFA without changing the number of states. That is, given $N = (Q, \Sigma, \delta, q_0, F)$, show that you can always construct another NFA $N' = (Q', \Sigma, \delta', q_0', F')$ where $L(N) = L(N')$, $|Q| = |Q'|$, and δ' has no ε-transitions. Prove that your method is correct.

   Solution: Let E_q be the set of states reachable from q by ε-transitions. We can define this recursively by $q \in E_q$ and $\delta(q', \varepsilon) \in E_q$ if $q' \in E_q$. In other words, this is the "epsilon closure" of state q.

   We create the new transition function δ' and final states F' as follows:

   $\delta'(q, a) = \{\delta(q', a) \mid q' \in E_q\} \cup \delta(q, a)$

   $F' = \{E_q \cap F \neq \emptyset\}$

3. Closure properties
   [**Category**: Proof, **Points**: 45]

Prove that regular languages are closed under the following operations (assume L is regular):

(a) $\lhd(L) = \{x_n x_{n-1} \cdots x_2 x_1 \mid x_i \in \Sigma, x_1 x_2 \cdots x_{n-1} x_n \in L\}$

(b) $\dagger(L) = \{x \mid xy \in L, y \in \Sigma^*\}$

(c) $\bowtie(L) = \{yx \mid x \in \Sigma^*, y \in \Sigma^*, xy \in L\}$

Solution:

(a) Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA that recognizes L, we can construct $N' = (Q', \Sigma, \delta', q_0', F')$ be the NFA that recognizes $\lhd(L)$, where:

$$Q' = Q \cup q_0'$$

For any $q \in Q'$ and any $a \in \Sigma$

$$\delta'(q, a) = \begin{cases} F & \text{if } q = q_0' \text{ and } a = \epsilon \\ \{q' \mid \delta(q', a) = q\} & \text{if } q \in Q \end{cases}$$

$$F' = q_0$$

(b) Let $M = (Q, \Sigma, \delta, q_0, F)$ be the DFA that recognizes L, we can construct $M' = (Q, \Sigma, \delta, q_0, F')$ be the DFA that recognizes $\dagger(L)$, where:

$$F' = \{q \in Q \mid q \text{ exists on a path between } q_0 \text{ and } q_f \in F\}$$

(c) Let $M = (Q, \Sigma, \delta, q_0, F)$ be the DFA that recognizes L, we can construct $M' = (Q', \Sigma, \delta', q_0', \{q_f\})$ be the an NFA that recognizes $\bowtie(L)$, where:
$Q' = Q \times Q \cup \{q_0', q_f\}$
And we'll define the new transition function as follows:
Jump from the new start state to any state in the original machine. Remember this state:
$\delta'(q_0', \varepsilon) = (q, q)$ for $q \in Q$
Take any of the previous transitions, taking care to remember where we started:
$\delta'((q_a, q_b), a) = (q_a, \delta(q_b, a))$ for $q_b \in Q$
If we reach an ending state in the previous machine, wrap around:
$\delta'((q_a, q_b), \varepsilon) = (q_a, q_0)$ for $q_b \in F$
If a transition would take us back to where we started, we should transition to the end state:
$\delta'((q_a, q_b), a) = q_f$ if $\delta(q_b, a) = q_a$

4. Regular Expressions
[**Category**: Comprehension, **Points**: 25]

Write a regular expression for the language:

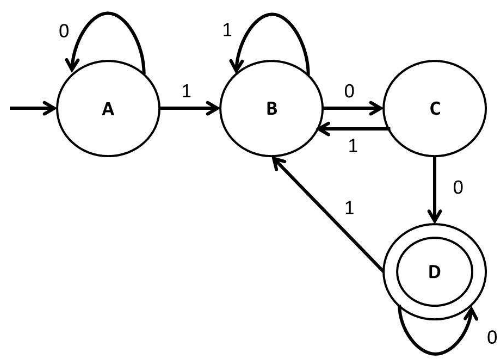
$$L = \{w \mid w \in \{0,1\}^*, w = \langle n \rangle, n \in \mathbb{N}, n \equiv 4 \mod 5\}$$

In other words, L is the language of binary strings that encode natural numbers that have remainder 4 when divided by 5. For example, $1001 = \langle 9 \rangle \in L$, but $110 = \langle 6 \rangle \notin L$ Ignore leadning zeros (so the strings 00010 and 0010 are both encodings of the same natural number, $\langle 2 \rangle$.)

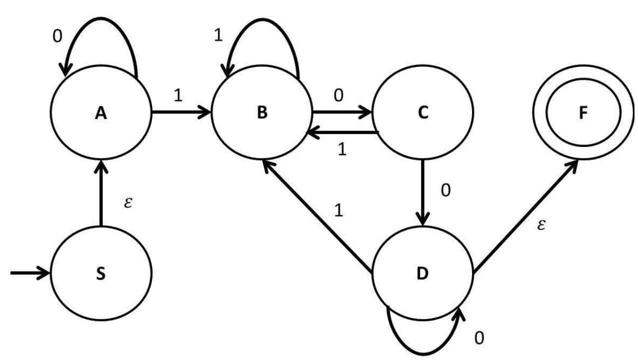
HINT: Generating this regular expression *ex nihilo* (from nothing) is probably very difficult. Don't forget alternative ways of generating regular expressions.

Solution:

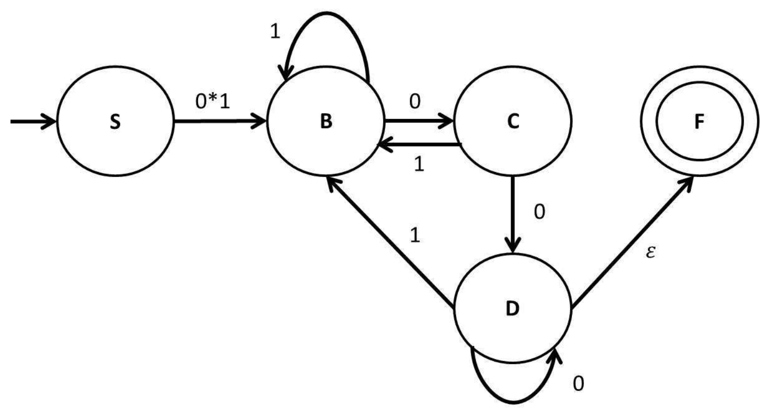
By the description of the problem, we know that the w needs to contain at least one 1 and has at least two 0s at the end of w. Therefore, we can construct a DFA for the language L:



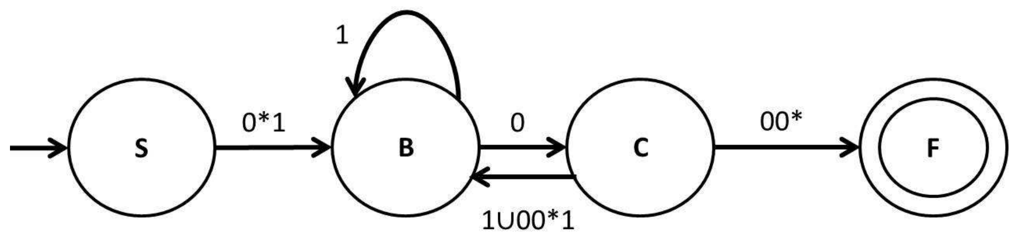
Then we can construct a GNFA of this DFA by the description in the textbook(p.70). Notice that we don't show the ϕ transitions between the states because they do not change the result of conversion.



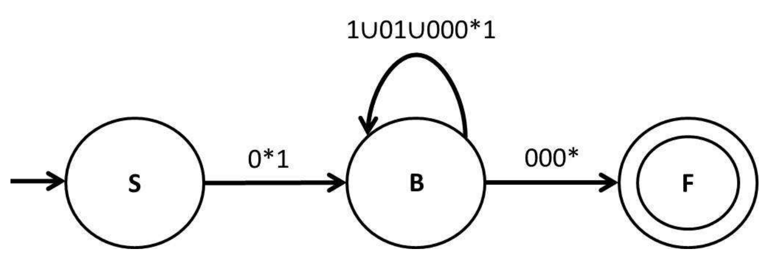
Then we rip state A and get:



Ripping state D:



Ripping state C:



Ripping state B and the transition between the start state and the final state is the regular expression we need for this problem.