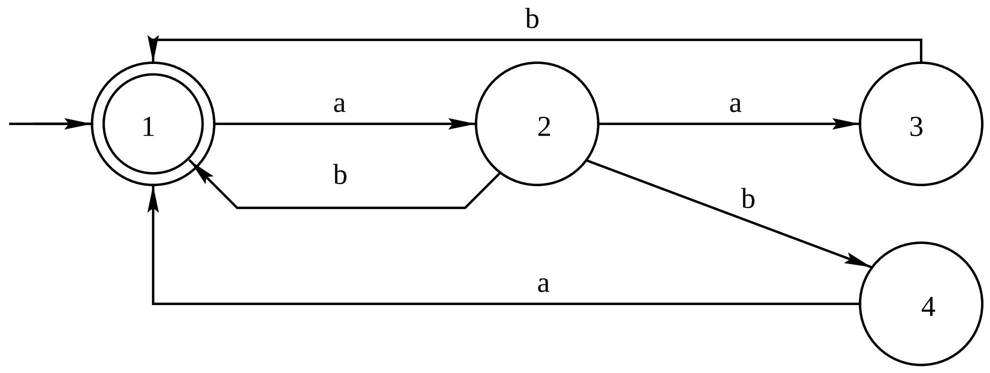
Solutions: Homework 2

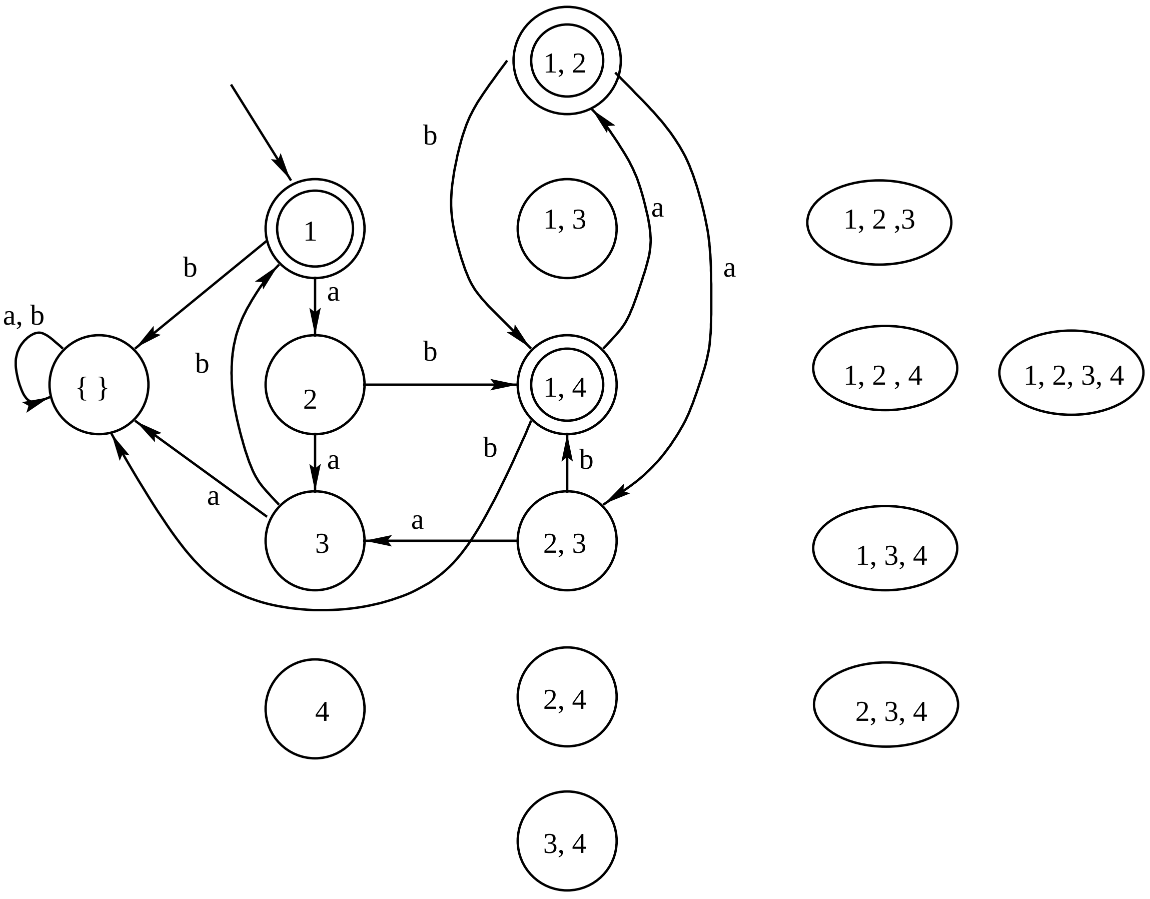
Prepared by Chien-Chung Huang

1

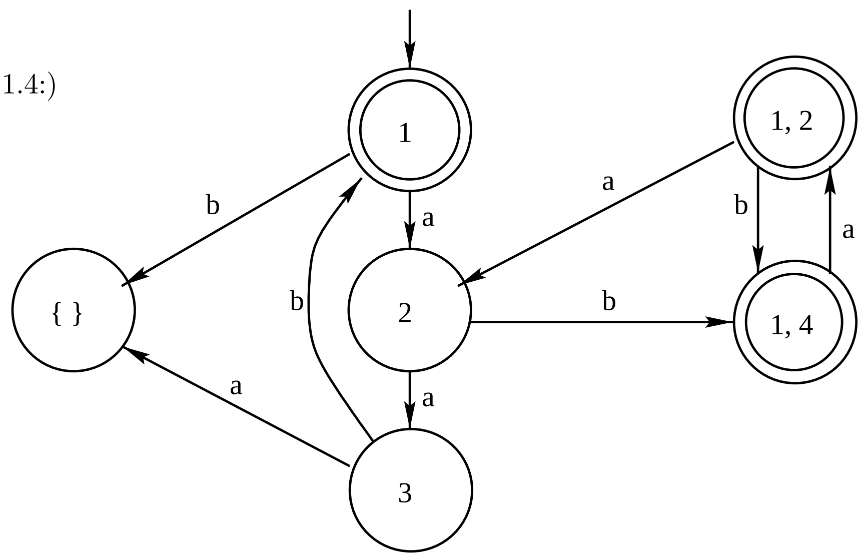
1.1:)



1.2+1.3:)

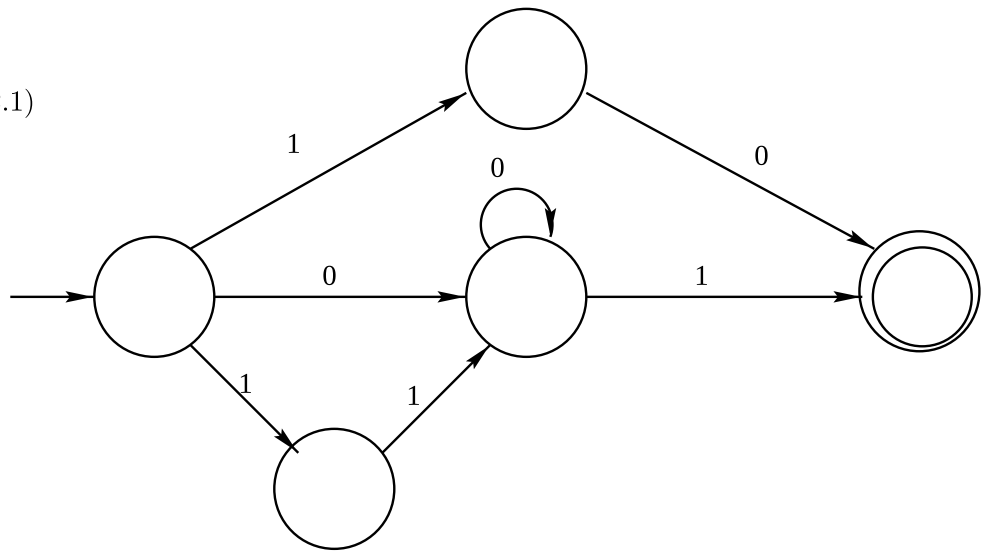


1.4:)

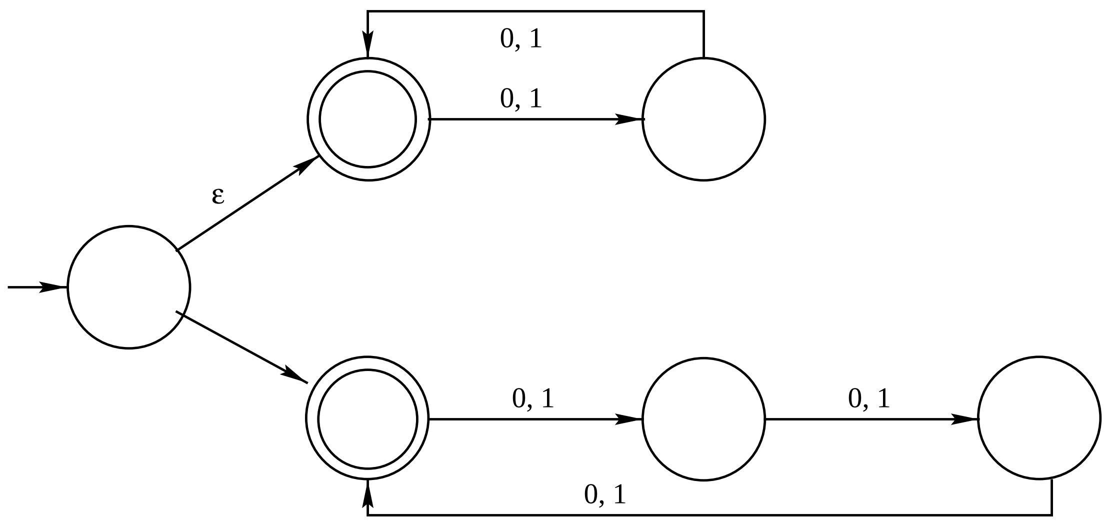


2

2.1)



2.2)



3

a) Answer:
$(0 \cup 1)^*(000 \cup 111)(0 \cup 1)^*$

A string in that language will have the form: some sequence of symbols, followed by three zeros or three ones, followed by some sequene of symbols. Hence the regular expression.

b) Answer:
Since the three 0's can appear before or after the three 1's we have:

$((0 \cup 1)^*000(0 \cup 1)^*111(0 \cup 1)^*) \ \cup \ ((0 \cup 1)^*111(0 \cup 1)^*000(0 \cup 1))^*$

c) Answer:
In any such string, the first two symbols must be 01 or 10, the next two symbols must be again 01 or 10, the next two symbols must be again 01 or 10, and so on. So we have: $(01 \cup 10)^*$

d) Answer:
Let *digit* be the regular expression $0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9$
and *posdigit* be the regular expression $1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9$
and *period* be the regular expression consisting of the single symbol. (period)

The required expression is:
$(0 \cup \mathit{posdigit}\ (\mathit{digit})^*)\ \mathit{period}\ (0 \cup (\mathit{digit})^*\mathit{posdigit})$

4

Informal description

The trick is to add a new final state q_f, to which all existing final states can reach by the empty string ε, and from which there are no transitions out. The old final states are then made non-final.

Formally

If $M = (Q, \Sigma, \delta, q_0, F)$ is the original NFA, design a new NFA $M' = (Q', \Sigma, \delta', q_0', F')$, where

$$\begin{aligned}
Q' &= Q \cup q_f, \quad \text{where } q_f \notin Q \\
F' &= \{q_f\} \\
q_0' &= q_0 \\
\delta'(q,a) &= \begin{cases} \emptyset & \text{if } q = q_f \\ q & \text{if } q \in F, a = \varepsilon \\ \delta(q,a) & \text{otherwise} \end{cases}
\end{aligned}$$

5

This solution written by Amit Chakrabarti

This is a difficult but truly beautiful problem, so let us give two different proofs of the result.

Approach 1, informal: Since L is regular it must be recognized by a DFA $M = (Q, \Sigma, \delta, q_0, F)$. We will construct an NFA M' for $\textsc{Half}(L)$, proving that $\textsc{Half}(L)$ is also regular. The idea is that as M' reads an input x it tries to guess a string y that would ensure $xy \in L$ and it checks whether or not $xy \in L$ by clevery running both strings x and y through M simultaneously. M' begins by guessing a *magic state*: the state M will end up in after reading all of x. Each time M' reads the next character of x, it

1. keeps track of what state M would be in if it were reading x,
2. guesses the next character of y, and
3. keeps track of what state M would be in if it were reading y, starting from the magic state guessed earlier.

Having read x, M' accepts if both of the following happen:

1. M *does* end up in the initially guessed magic state after reading x, and
2. M ends up in an accept state after reading y, starting from the magic state.

Thus, we see that at any point of time, M' needs to keep track of three pieces of information: the magic state, the state reached by reading the current prefix of x, and the state reached by reading the current prefix of y. With this intuition we are ready to describe M' formally.

Approach 1, formal: Let $M' = (\{q_{\text{start}}\} \cup Q \times Q \times Q, \Sigma, \delta', q_{\text{start}}, F')$, where δ' and F are given by

$$\begin{aligned}
\delta'(q_{\text{start}}, \varepsilon) &= \{(q, q_0, q) : q \in Q\}\,, \\
\delta'(q_{\text{start}}, a) &= \emptyset, && \forall\, a \in \Sigma\,, \\
\delta'((q_{\text{magic}}, q_x, q_y), \varepsilon) &= \emptyset, && \forall\, q_{\text{magic}}, q_x, q_y \in Q\,, \\
\delta'((q_{\text{magic}}, q_x, q_y), a) &= \{(q_{\text{magic}}, \delta(q_x, a), \delta(q_y, b)) : b \in \Sigma\}, && \forall\, a \in \Sigma;\;\; q_{\text{magic}}, q_x, q_y \in Q\,, \\
F' &= \{(q, q, q_f) : q \in Q \text{ and } q_f \in F\}\,.
\end{aligned}$$

Then, by the discussion in the informal section, $\mathcal{L}(M') = \text{HALF}(L)$.

Approach 2, informal: Again, we start with a DFA $M = (Q, \Sigma, \delta, q_0, F)$ for the language L. If x is a string in $\text{HALF}(L)$, and y is a string such that $|x| = |y|$ and $xy \in L$, then the string xy must trace a path π through M from q_0 to some state in F, say q_n. Imagine tracing *two paths* through M simultaneously, using your left and right index fingers:

1. Start by placing your left finger at q_0 and your right finger at q_n.
2. Read x from left to right and, simultaneously, y from right to left.
3. After reading each pair of characters, one from x and one from y, move your left finger one step forward along the path π and your right finger one step backward.

Since $|x| = |y|$, it is clear that you will finish reading x and y at exactly the same time and at that point your two fingers will meet at the midpoint of the path π. Now, we turn this observation into a machine that will accept x (and will accept nothing spurious). First of all, observe that as we read x we can't quite know what path the right index finger will trace, because (1) we don't know y and (2) even if we did know y, there could be two or more transitions leading *in* to a state of $q \in Q$ on alphabet symbol $a \in \Sigma$. In other words, reversing of the arrows of a DFA might make it nondeterministic. In order to make the right hand movement deterministic, we will borrow an idea from the subset construction we studied in class and actually keep track of *all possible states* that the right index finger could be in.

Approach 2, formal: Let us define a function $\delta^{-1} : 2^Q \to 2^Q$ as follows: for any set $S \subseteq Q$,

$$\delta^{-1}(S) = \{q \in Q : \exists\, a \in \Sigma\ (\delta(q,a) \in S)\}\,.$$

In words, $\delta^{-1}(S)$ is the set of states of M that we can get to by following one arrow *backwards* from some state in S. Now, define $M'' = (Q \times 2^Q, \Sigma, \delta'', (q_0, F), F'')$, where δ'' and F'' are given by

$$\begin{aligned} \delta''((q,S),a) &= (\delta(q), \delta^{-1}(S)), && \forall\, q \in Q, S \subseteq Q, a \in \Sigma\,, \\ F'' &= \{(q,S) \in Q \times 2^Q : q \in S\}\,. \end{aligned}$$

Then, by the discussion in the informal section, $\mathcal{L}(M'') = \textsc{Half}(L)$.

Approach $2\frac{1}{2}$: Actually it's possible to follow Approach 2 and end up with an NFA instead of a DFA by noting that we don't *have* to make the movement of the right finger deterministic. Of course we will now have to add a special start state and use ε-transitions from it to put the right finger on one of the final states of M. So, based on M, we could build an NFA M_1 instead of the DFA M'' we just built: $M_1 = (\{q_{\text{start}}\} \cup Q \times Q, \Sigma, \delta_1, q_{\text{start}}, F_1)$, where

$$\begin{aligned} \delta_1(q_{\text{start}}, \varepsilon) &= \{(q_0, q_f) : q_f \in F\}\,, \\ \delta_1(q_{\text{start}}, a) &= \emptyset\,, && \forall\, a \in \Sigma\,, \\ \delta_1((q,q'), \varepsilon) &= \emptyset\,, && \forall\, q, q' \in Q\,, \\ \delta_1((q,q'), a) &= \{(\delta(q), q'') : \exists\, b \in \Sigma \text{ such that } \delta(q'', b) = q'\}\,, && \forall\, q, q' \in Q, a \in \Sigma\,, \\ F_1 &= \{(q,q) : q \in Q\}\,. \end{aligned}$$