COMS W4115 Assignment 1

Programming Languages and Translators

Xijiao Li (xl2950)

October 19, 2020

Problem 1

a.

1*0(01*0+1)*
ϵ is not valid since it has 0 number of "0"'s.

b.

0*1*1
ϵ is not valid since it is not ending with a "1".

Problem 2

From the graph we know that the NFA recognizes all strings that contain two 0's, between which there can optionally be a substring s and s must has length equal to a multiple of 3. A regular expression for the NFA is (0+1)*0((0+1)(0+1)(0+1))*0(0+1).

Problem 3

a.

$coms4115:\ [a-z][a-z0-9]^*$
Based on maximal munch, we will pick $[a-z][a-z0-9]^*$, since it can fully cover the input.

b.

$while : [a-z][a-z0-9]^*$
Based on maximal munch, we pick $[a-z][a-z0-9]^*,\ for|while|if|else$, since they can fully cover input. Based on the rule of thumb, we pick $[a-z][a-z0-9]^*$ due to its higher priority.

c.

$123abc:\ [0-9]^*[a-z],\ [a-z][a-z0-9]^*$
We need to do partition here, since no rule can cover the full input. Based on maximal munch, we pick $[0-9]^*[a-z]$ first since it can cover the most number of characters "123a" among other rules. Then we have "bc" left, and based on maximal munch, we pick $[a-z][a-z0-9]^*$.

d.

$dictionary : [a-z][a-z0-9]^*$
Based on maximal munch, we pick $[a-z][a-z0-9]^*,\ dictionary$, since they

can fully cover input. Based on the rule of thumb, we pick $[a-z][a-z0-9]^*$ due to its higher priority.

e.

The empty string: none
No rules apply, since all rules require the valid inputs to contain at least one character.

Problem 4

a.

$(p+q)^* + (p+r)^* + (q+r)^*$

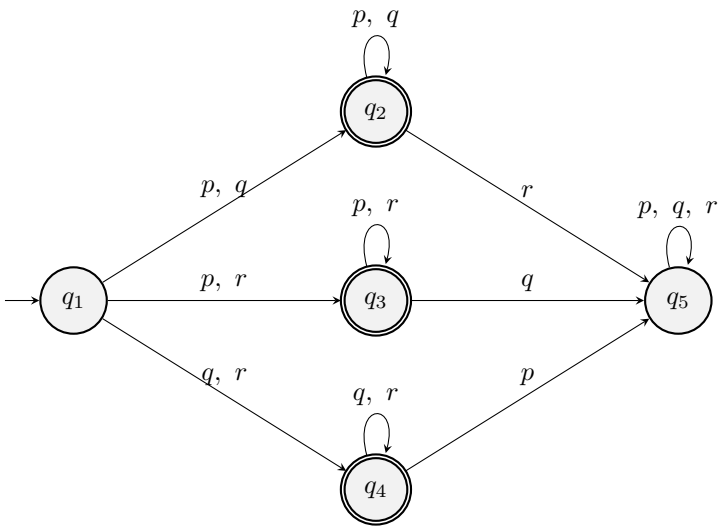
b.

NFA $M = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ is defined as:

$$Q = \{q1, q2, q3, q4, q5\}$$
$$\Sigma = \{p, q, r\}$$
$$S = q_1$$
$$F = \{q_2, q_3, q_4\}$$
$$\delta :$$

state \input	p	q	r
q_1	q_2, q_3	q_2, q_4	q_3, q_4
q_2	q_2	q_2	q_5
q_3	q_3	q_5	q_3
q_4	q_5	q_4	q_4
q_5	q_5	q_5	q_5

The state machine is as follows:

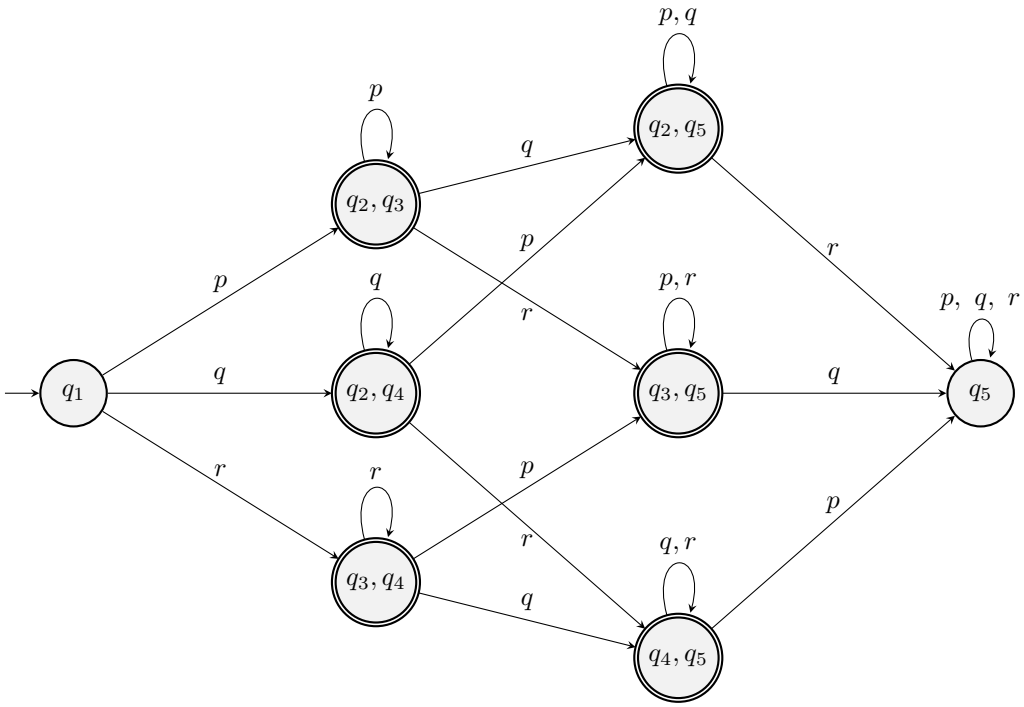


c.

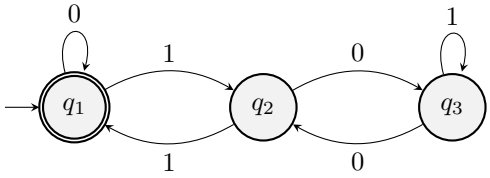
The transition table is as follows:

state \input	p	q	r
q_1	q_2, q_3	q_2, q_4	q_3, q_4
q_2, q_3	q_2, q_3	q_2, q_5	q_3, q_5
q_2, q_4	q_2, q_5	q_2, q_4	q_4, q_5
q_3, q_4	q_3, q_5	q_4, q_5	q_3, q_4
q_2, q_5	q_2, q_5	q_2, q_5	q_5
q_3, q_3	q_3, q_5	q_5	q_4, q_5
q_4, q_5	q_5	q_4, q_5	q_4, q_5
q_5	q_5	q_5	q_5

The state machine for DFA is as follows:



5.



Let x denote the scanned binary input.
q_1 refers to the state that $x \mod 3 \equiv 0$, q_2 refers to the state that $x \mod 3 \equiv 1$, and q_3 refers to the state that $x \mod 3 \equiv 2$. So q_1 will be our initial state as well as the accepting state, and we begin to calculate the reminder from 0.
Suppose that the current x has a reminder of 0.

1. the next input is 0: x is doubled.
   $2x \mod 3 \equiv 0$, and it will stay at q_1.

2. the next input is 1: x is doubled and incremented by 1.
   $2x + 1 \mod 3 \equiv 1$, and it will go to q_2.

Suppose that the current x has a reminder of 1.

1. the next input is 0: x is doubled.
   $2x \mod 3 \equiv= 2$, and it will go to q_3.

2. the next input is 1: x is doubled and incremented by 1.
   $2x + 1 \mod 3 \equiv 0$, and it will go to q_1.

Suppose that the current x has a reminder of 2.

1. the next input is 0: x is doubled.
   $2x \mod 3 \equiv 1$, and it will go to q_2.

2. the next input is 1: x is doubled and incremented by 1.
   $2x + 1 \mod 3 \equiv 2$, and it will stay at q_3.