- **Please give details of your answer. A direct answer without explanation is not counted.**
- **Your answers must be in English.**
- **Please carefully read problem statements.**
- **During the exam you are not allowed to borrow others' class notes.**
- **Try to work on easier questions first.**

Problem 1 (5 pts)

Consider the following language

$$\{ww^R \mid w \in \{0,1\}^*\}$$

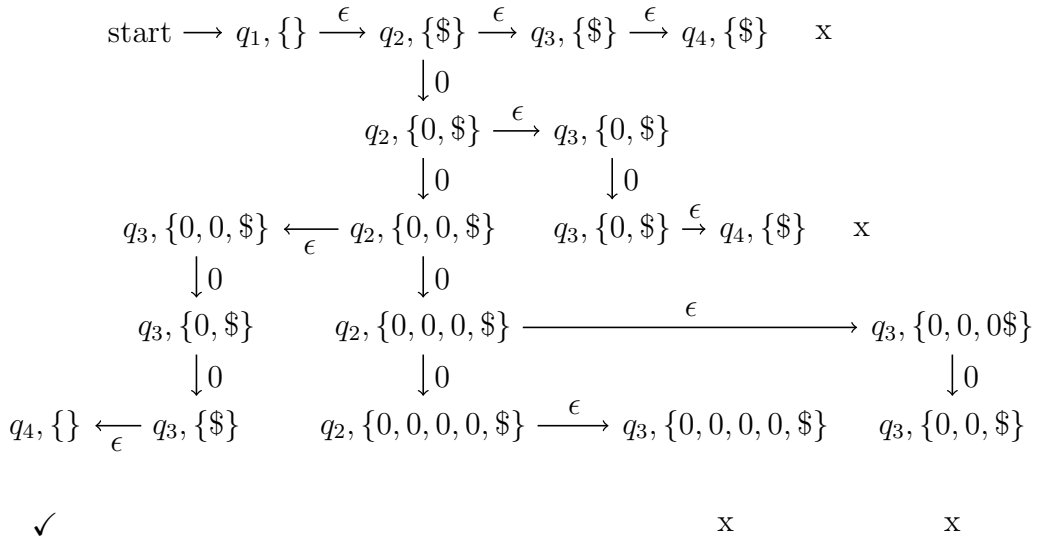
and its 4-state PDA mentioned in the lecture. Draw a tree to simulate the following input string.

0000

Note that the tree must be a complete one. You cannot just use one path to accept/reject the string.

Answer

The tree is as following.

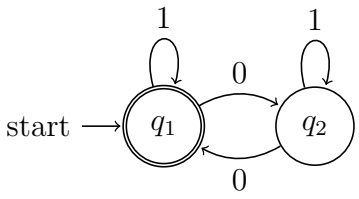


Problem 2 (25 pts)

Consider the following language

$$\{w \mid w \in \{0,1\}^* \text{ has an even number of zeros}\}$$

and its DFA



(a) (10 pts) In our lecture, we describe a way to transform each link to a rule of CFG. Use that way to generate a CFG for this language.

(b) (15 pts) Generate a CFG for this language with the smallest number of rules. **Explain why yours has the smallest number.**

Answer

(a) We can get the following CFG.

$$V = \{R_0, R_1, R_2\}\ \Sigma = \{0,1\}\ S = R_1$$

The rules are

$$R_1 \longrightarrow 0R_2 \mid 1R_1 \mid \epsilon$$

$$R_2 \longrightarrow 0R_1 \mid 1R_2$$

(b) The language with smallest rules are

$$V = \{R_0\}\ \Sigma = \{0,1\}\ S = R_0$$

The rules are

$$R_0 \longrightarrow R_0 0 R_0 0 R_0 \mid 1R_0 \mid \epsilon$$

We explain that

$$\text{language of CFG} = \text{language of DFA}$$

First, by the first rule, any string generated by the CFG must have an even number of zeros. Thus,

$$\text{language of CFG} \subset \text{language of DFA}$$

Second, for any string that has an even number of zeros, we can represent it as

$$1^*01^*0...01^*01^*$$

By our rules, 1^* can be generated by the second and third rules. Thus,

$$\text{language of CFG} \supset \text{language of DFA}$$

Therefore, the proof is complete.

One rule is impossible: It is because the CFG with only one rule can generate at most one string. However, there are a lot of strings in the language.

Two rules are impossible: It is because we need to generate 1 and 11. By the above arguement, we need two rules without 0, but we also need to generate 0. There must be another rule for 0.

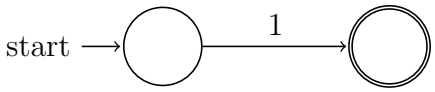
Thus, it is impossible to have a CFG for this language with less than 3 rules.

Another solution

$$R_0 \longrightarrow 0R_0 0R_0 \mid 1R_0 \mid \epsilon$$

Problem 3 (45 pts)

Consider the following NFA,



We assume $\Sigma = \{1\}$

(a) (5 pts) Modify it to be a PDA and give the formal definition. Note that you can use only **two states**.

(b) (15 pts) We would like to use the procedure in Lemma 2.27 to transform the PDA obtained in (a) to a CFG. To begin, we modify the PDA to another PDA with no more than 3 states satisfying

(i) It has a single accept state.

(ii) It empties its stack before accepting.

(iii) Each transition either pushes a symbol onto the stack or pops one off the stack, but it does not do both at the same time.

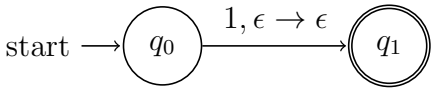
Then please generate the CFG.

(c) (20 pts) Convert the CFG obtained in (b) to CNF. **You must show the details.**

(d) (5 pts) Simplify the CNF in (c) to the smallest number of rules. **You must explain why some rules in your CNF are not needed and therefore the rule set can be simplified to your answer.**

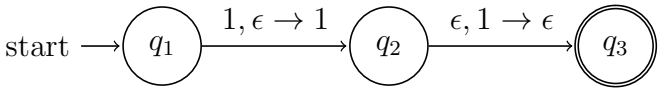
Answer

(a) The PDA diagram



$Q = \{q_0, q_1\}$
$\Sigma = \{1\}$
$\Gamma = \emptyset$
start state $= q_0$
$F = \{q_1\}$
$\delta(q, \sigma, \gamma) = \begin{cases} \{(q_1, \epsilon)\}, & q = q_0, \sigma = 1, \gamma = \epsilon \\ \emptyset, & \text{otherwise} \end{cases}$ $, q \in Q, \sigma \in \Sigma_\epsilon, \gamma \in \Gamma_\epsilon$

(b) To satisfy those condition, consider



By Lemma 2.27, we can generate a CFG from above PDA. For simplicity, we denote A_{q_i,q_j}

as A_{ij}

$$
\begin{aligned}
S &\to A_{13} \\
A_{11} &\to A_{11}A_{11}|A_{12}A_{21}|A_{13}A_{31}|\epsilon \\
A_{12} &\to A_{11}A_{12}|A_{12}A_{22}|A_{13}A_{32} \\
A_{13} &\to A_{11}A_{13}|A_{12}A_{23}|A_{13}A_{33}|1A_{22} \\
A_{21} &\to A_{21}A_{11}|A_{22}A_{21}|A_{23}A_{31} \\
A_{22} &\to A_{21}A_{12}|A_{22}A_{22}|A_{23}A_{32}|\epsilon \\
A_{23} &\to A_{21}A_{13}|A_{22}A_{23}|A_{23}A_{33} \\
A_{31} &\to A_{31}A_{11}|A_{32}A_{21}|A_{33}A_{31} \\
A_{32} &\to A_{31}A_{12}|A_{32}A_{22}|A_{33}A_{32} \\
A_{33} &\to A_{31}A_{13}|A_{32}A_{23}|A_{33}A_{33}|\epsilon
\end{aligned}
$$

(c) First we remove ϵ,

$$
\begin{aligned}
S &\to A_{13} \\
A_{11} &\to A_{11}A_{11}|A_{12}A_{21}|A_{13}A_{31}|A_{11} \\
A_{12} &\to A_{11}A_{12}|A_{12}A_{22}|A_{13}A_{32}|A_{12} \\
A_{13} &\to A_{11}A_{13}|A_{12}A_{23}|A_{13}A_{33}|A_{13}|1A_{22}|1 \\
A_{21} &\to A_{21}A_{11}|A_{22}A_{21}|A_{23}A_{31}|A_{21} \\
A_{22} &\to A_{21}A_{12}|A_{22}A_{22}|A_{23}A_{32}|A_{22} \\
A_{23} &\to A_{21}A_{13}|A_{22}A_{23}|A_{23}A_{33}|A_{23} \\
A_{31} &\to A_{31}A_{11}|A_{32}A_{21}|A_{33}A_{31}|A_{31} \\
A_{32} &\to A_{31}A_{12}|A_{32}A_{22}|A_{33}A_{32}|A_{32} \\
A_{33} &\to A_{31}A_{13}|A_{32}A_{23}|A_{33}A_{33}|A_{33}
\end{aligned}
$$

Then we remove unit rules,

$$
\begin{aligned}
S &\to A_{11}A_{13}|A_{12}A_{23}|A_{13}A_{33}|1A_{22}|1\\
A_{11} &\to A_{11}A_{11}|A_{12}A_{21}|A_{13}A_{31}\\
A_{12} &\to A_{11}A_{12}|A_{12}A_{22}|A_{13}A_{32}\\
A_{13} &\to A_{11}A_{13}|A_{12}A_{23}|A_{13}A_{33}|1A_{22}|1\\
A_{21} &\to A_{21}A_{11}|A_{22}A_{21}|A_{23}A_{31}\\
A_{22} &\to A_{21}A_{12}|A_{22}A_{22}|A_{23}A_{32}\\
A_{23} &\to A_{21}A_{13}|A_{22}A_{23}|A_{23}A_{33}\\
A_{31} &\to A_{31}A_{11}|A_{32}A_{21}|A_{33}A_{31}\\
A_{32} &\to A_{31}A_{12}|A_{32}A_{22}|A_{33}A_{32}\\
A_{33} &\to A_{31}A_{13}|A_{32}A_{23}|A_{33}A_{33}
\end{aligned}
$$

Finally we add a new variable and rules to be a CNF,

$$
\begin{aligned}
S &\to A_{11}A_{13}|A_{12}A_{23}|A_{13}A_{33}|1A_{22}|1\\
A_{11} &\to A_{11}A_{11}|A_{12}A_{21}|A_{13}A_{31}\\
A_{12} &\to A_{11}A_{12}|A_{12}A_{22}|A_{13}A_{32}\\
A_{13} &\to A_{11}A_{13}|A_{12}A_{23}|A_{13}A_{33}|BA_{22}|1\\
A_{21} &\to A_{21}A_{11}|A_{22}A_{21}|A_{23}A_{31}\\
A_{22} &\to A_{21}A_{12}|A_{22}A_{22}|A_{23}A_{32}\\
A_{23} &\to A_{21}A_{13}|A_{22}A_{23}|A_{23}A_{33}\\
A_{31} &\to A_{31}A_{11}|A_{32}A_{21}|A_{33}A_{31}\\
A_{32} &\to A_{31}A_{12}|A_{32}A_{22}|A_{33}A_{32}\\
A_{33} &\to A_{31}A_{13}|A_{32}A_{23}|A_{33}A_{33}\\
B &\to 1
\end{aligned}
$$

(d) Because S is the start variable, the first rule applied is

$$S \to A_{11}A_{13}|A_{12}A_{23}|A_{13}A_{33}|1A_{22}|1.$$

If we take

$$S \to A_{11}A_{13},$$

we argue that no string can be generated. The reason is that we must have

$$S \to A_{1?} \cdots A_{?1} A_{13},$$

where $A_{?1}$ becomes a terminal. However there is no such a rule. By a similar argument, we have that none of

$$S \to A_{12} A_{23}$$
$$S \to A_{13} A_{33}$$
$$S \to B A_{22}$$

can generate a string. Therefore, because from S we can only generate 1, all other rules are useless. Thus, $S \to 1$ becomes the CFG with the smallest number of rules.

Common Mistakes

(a)

- F is a set.

(c)

- You cannot remove $A_{11} \to A_{11}A_{11}$ while removing $A_{11} \to \epsilon$

Problem 4 (25 pts)

(a) (15 pts) Design a Turing machine with no more than 10 states (including q_{accept} and q_{reject}) that accepts the language

$$\{\#w\#a^n | w \in \{0,1\}^* \text{ and is a binary representation for } n\},$$

where $\Sigma = \{\#, 0, 1, a\}, \Gamma = \{\#, 0, 1, a, \times, \sqcup\}$. For example, the following strings are accepted

- ##
- #0#
- #01#a
- #110#aaaaaa

(b) (10 pts) Use the Turing machine obtained in (a) to simulate the following strings

(i) #010#aa

(ii) #10#a

Answer

(a) The concept is in each round, minus w by one as well as cross off one a. Here we describe a Turing machine M that recognize this language.

$M_1 =$ "On input string s

1. Find the right most element of w

2. (Minus w by one) Find the right most 1 and replace it by 0, and replace every 0 right of it by 1.

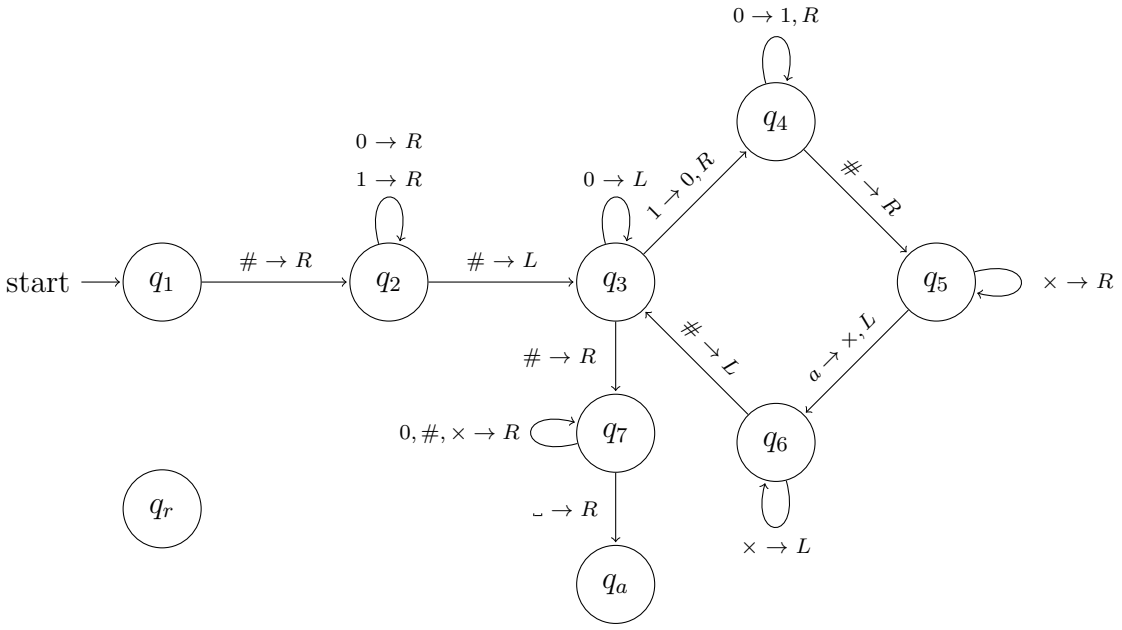
3. If in stage **2** the tape contains no 1 but still contains a, *reject*.

4. If in stage **2** the tape contains neither 1 nor a, *accept*.

5. (Cross off one a) Find the left most a and cross it off (i.e., replace it with $\times$).

6. If in stage **5** the tape contains no a, *reject*.

7. Go to stage **1**."



The state diagram of M_1. Note that links not shown go to q_r

(b) (i) For #010#aa

$q_1\#010\#aa\sqcup$ $\#0q_310\#aa\sqcup$ $\#000q_4\#\times a\sqcup$ $\#0q_300\#\times\times\sqcup$ $\#000q_7\#\times\times\sqcup$

$\#q_2010\#aa\sqcup$ $\#00q_40\#aa\sqcup$ $\#000\#q_5\times a\sqcup$ $\#q_3000\#\times\times\sqcup$ $\#000\#q_8\times\times\sqcup$

$\#0q_210\#aa\sqcup$ $\#001q_4\#aa\sqcup$ $\#000\#\times q_5a\sqcup$ $q_3\#000\#\times\times\sqcup$ $\#000\#\times q_8\times\sqcup$

$\#01q_20\#aa\sqcup$ $\#001\#q_5aa\sqcup$ $\#000\#q_6\times\times\sqcup$ $\#q_7000\#\times\times\sqcup$ $\#000\#\times\times q_8\sqcup$

$\#010q_2\#aa\sqcup$ $\#001q_6\#\times a\sqcup$ $\#000q_6\#\times\times\sqcup$ $\#0q_700\#\times\times\sqcup$ $\#000\#\times\times\sqcup q_a$

$\#01q_30\#aa\sqcup$ $\#00q_31\#\times a\sqcup$ $\#00q_30\#\times\times\sqcup$ $\#00q_70\#\times\times\sqcup$

(ii) For #10#a

$q_1\#10\#a\sqcup$ $\#10q_2\#a\sqcup$ $\#0q_40\#a\sqcup$ $\#01q_6\#\times\sqcup$ $\#00\#q_5\times\sqcup$

$\#q_210\#a\sqcup$ $\#1q_30\#a\sqcup$ $\#01q_4\#a\sqcup$ $\#0q_31\#\times\sqcup$ $\#00\#\times q_5\sqcup$

$\#1q_20\#a\sqcup$ $\#q_310\#a\sqcup$ $\#01\#q_5a\sqcup$ $\#00q_4\#\times\sqcup$ $\#00\#\times\sqcup q_r$

For this problem, (a) and (b) are graded together, so you only see one number as your total points of this problem.

Alternative solution of 4(a)

1. This answer also crosses off one a while it minuses w by 1. But when it minuses w, it doesn't need to find the right most 1 of w first.

   M_2 = "On input string s

   1. Find the right most element of w

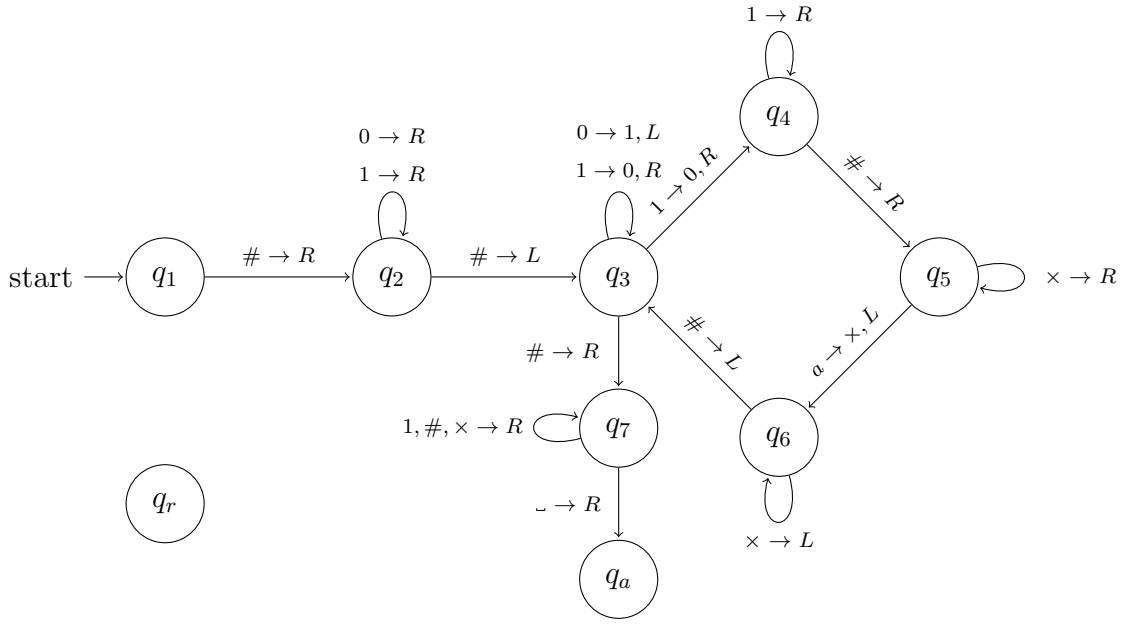
   2. (Minus w by one) Go left and replace every scanned 0 with 1, until touching a #, then go to stage **3**. (This indicate before this stage, w is 0). Or until a 1, then replace the 1 with 0 and go to stage **4**.

   3. If the tape contains no a *accept*, else, *reject*.

   4. (Cross off one a) Find the left most a and cross it off (i.e., replace it with $\times$).

   5. If in stage **4** the tape contains no a, *reject*.

   6. Go to stage **1**."



The state diagram of M_2. Note that links not shown go to q_r

2. This answer crosses off nearly half of the a string instead of crossing it one by one.

   $M_3 =$ "On input string s

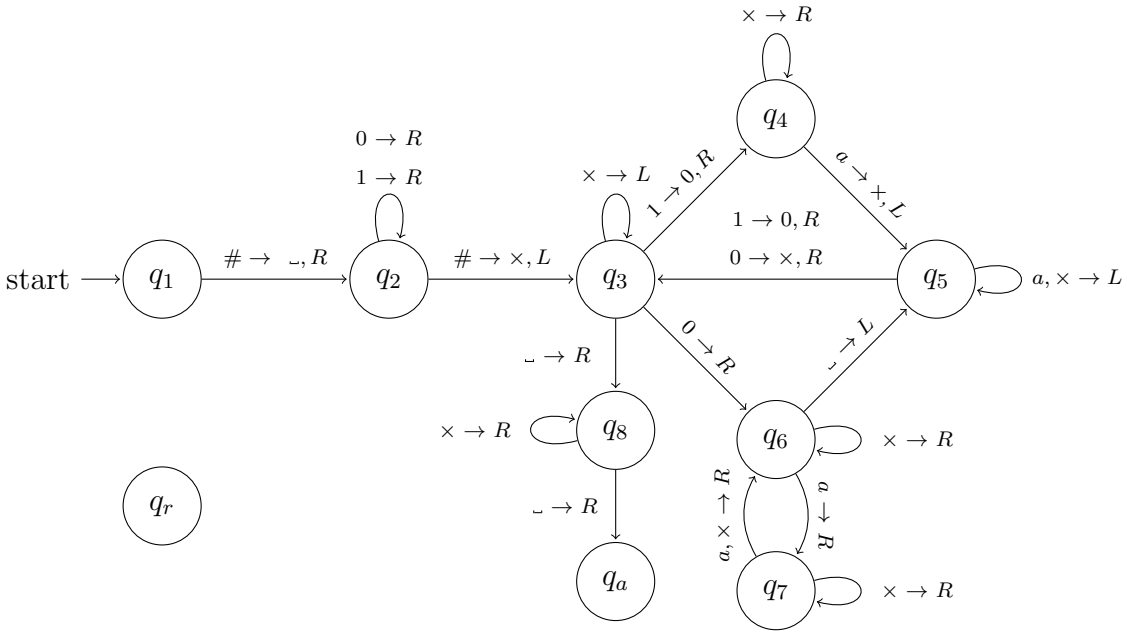
   1. If the right most digit of w(not including $\times$) is 1, go to stage **2**; it's 0, go to stage **3**; it has no character left, go to stage **4**.

   2. (Minus w by one) Replace this 1 with 0 and cross off one a. If no a left, *reject*.

   3. (Divide w by two) Cross off the 0. If the length of a is odd, *reject*, else, cross off half of the a string.

   4. If the tape contains no a *accept*, else, *reject*.

   5. Go to stage **1**."



The state diagram of M_3. Note that links not shown go to q_r