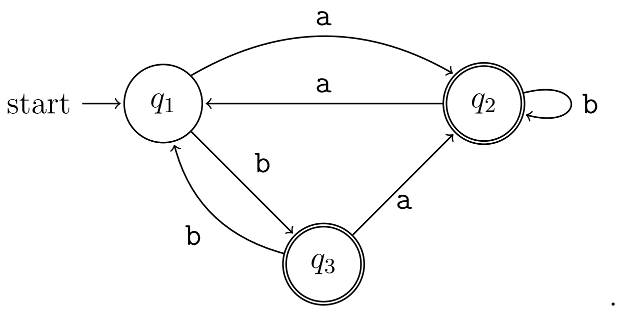
- **Please give details of your answer. A direct answer without explanation is not counted.**
- **Your answers must be in English.**
- **Please carefully read problem statements.**
- **During the exam you are not allowed to borrow others' class notes.**
- **Try to work on easier questions first.**

Problem 1 (5 pts)

Consider the following DFA,



Give a CFG with ≤ 8 rules to describe the above language.

Answer

(1) For each state q_i, use a variable S_i to represent strings ending in q_i.

(2) For each transition function start from state q_i, add rules:

$$\left.\begin{aligned}\delta(q_1, a) &= q_2 \\ \delta(q_1, b) &= q_3\end{aligned}\right\} \Rightarrow S_1 \to aS_2 \mid bS_3$$

$$\left.\begin{aligned}\delta(q_2, a) &= q_1 \\ \delta(q_2, b) &= q_2\end{aligned}\right\} \Rightarrow S_2 \to aS_1 \mid bS_2$$

$$\left.\begin{aligned}\delta(q_3, a) &= q_2 \\ \delta(q_3, b) &= q_1\end{aligned}\right\} \Rightarrow S_3 \to aS_2 \mid bS_1$$

(3) Add $S_i \to \epsilon$ for accept state q_i, and the DFA can be finally described as the following CFG:

$$
\begin{aligned}
S_1 &\to aS_2 \mid bS_3 \\
S_2 &\to bS_2 \mid aS_1 \mid \epsilon \\
S_3 &\to bS_1 \mid aS_2 \mid \epsilon
\end{aligned}
$$

The start variable is S_1.

We can remove S_3 to have

$$
\begin{aligned}
S_1 &\to aS_2 \mid bbS_3 \mid baS_2 \mid b \\
S_2 &\to bS_2 \mid aS_1 \mid \epsilon
\end{aligned}
$$

Common Mistakes

1. Your CFG has more rules than what you are allowed.

Problem 2 (10 pts)

Consider the following language,

$$\{\ a^i b^j c^k \mid i = j \text{ or } i = k,\ i \geq 0, j \geq 0, k \geq 0\ \}$$

Give a CFG with ≤ 10 rules for this language.

Answer

(1) Use two separated rules $S \to A \mid B$ to identify either $i = j$ or $i = k$.

(2) For $S \to A$, i.e. $a^i b^j c^k$ where $i = j$,
To generate $a^i b^i$, there must be a recursive rule such as $S \to aSb$. Thus c shouldn't occur on the right-hand side of S, otherwise, c will appear on the left-hand side of b. So there must be two different variables S_1, S_2 to separately represent $a^i b^i$ and c^k. Therefore,

$$
\begin{aligned}
A &\to S_1 S_2 \\
S_1 &\to aS_1 b \mid \epsilon \\
S_2 &\to cS_2 \mid \epsilon
\end{aligned}
$$

(3) For $S \rightarrow B$, i.e. $a^ib^jc^k$ where $i = k$,

To generate a^ic^i, there must be a recursive rule such as $S \rightarrow aSc$, and b^j should be in the middle between a^i and c^i. So the variable B can be represented as:

$$\begin{aligned} &B \rightarrow S_3 \\ &S_3 \rightarrow aS_3c \mid S_4 \\ &S_4 \rightarrow bS_4 \mid \epsilon \end{aligned}$$

(4) Remove $S \rightarrow A$ and $S \rightarrow B$. We can get the simplified 10-rule CFG as follows:

$$\begin{aligned} &S \ \ \rightarrow S_1S_2 \mid S_3 \\ &S_1 \rightarrow aS_1b \mid \epsilon \\ &S_2 \rightarrow cS_2 \mid \epsilon \\ &S_3 \rightarrow aS_3c \mid S_4 \\ &S_4 \rightarrow bS_4 \mid \epsilon \end{aligned}$$

Common Mistakes

1. Your CFG has more rules than what you are allowed.

2. Some wrongly give CFG for the case of $i = k$ or $j = k$.

Problem 3 (15 pts)

Consider the following CFG,

$$\begin{aligned} &S \ \ \rightarrow S_1 \mid S_2 \\ &S_1 \rightarrow 0S_11 \mid \epsilon \\ &S_2 \rightarrow 1S_20 \mid \epsilon \end{aligned}$$

Convert this CFG to CNF by following the procedure in the textbook. Your number of rules should be ≤ 13. (Hint: you can remove redundant rules after finishing the procedure.)

Answer

Add a new start variable S_0:

$$\begin{aligned} S_0 &\to S \\ S &\to S_1 \mid S_2 \\ S_1 &\to 0S_11 \mid \epsilon \\ S_2 &\to 1S_20 \mid \epsilon \end{aligned}$$

Remove ϵ-rules $S_1 \to \epsilon$ and $S_2 \to \epsilon$:

$$\begin{aligned} S_0 &\to S \mid \epsilon \\ S &\to S_1 \mid S_2 \\ S_1 &\to 0S_11 \mid 01 \\ S_2 &\to 1S_20 \mid 10 \end{aligned}$$

Remove unit rules $S \to S_1$ and $S \to S_2$:

$$\begin{aligned} S_0 &\to S \mid \epsilon \\ S &\to 0S_11 \mid 01 \mid 1S_20 \mid 10 \\ S_1 &\to 0S_11 \mid 01 \\ S_2 &\to 1S_20 \mid 10 \end{aligned}$$

Remove unit rule $S_0 \to S$:

$$\begin{aligned} S_0 &\to 0S_11 \mid 01 \mid 1S_20 \mid 10 \mid \epsilon \\ S &\to 0S_11 \mid 01 \mid 1S_20 \mid 10 \\ S_1 &\to 0S_11 \mid 01 \\ S_2 &\to 1S_20 \mid 10 \end{aligned}$$

Add additional rules $A \to 1$ and $B \to 0$:

$$\begin{aligned} S_0 &\to BS_1A \mid BA \mid AS_2B \mid AB \mid \epsilon \\ S &\to BS_1A \mid BA \mid AS_2B \mid AB \\ S_1 &\to BS_1A \mid BA \\ S_2 &\to AS_2B \mid AB \\ A &\to 1 \\ B &\to 0 \end{aligned}$$

Convert remaining rules into proper form:

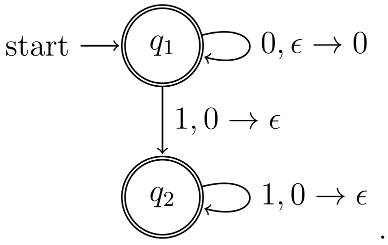
$$\begin{aligned}
S_0 &\to BY_1 \mid BA \mid AY_2 \mid AB \mid \epsilon \\
S &\to BY_1 \mid BA \mid AY_2 \mid AB \\
S_1 &\to BY_1 \mid BA \\
S_2 &\to AY_2 \mid AB \\
Y_1 &\to S_1A \\
Y_2 &\to S_2B \\
A &\to 1 \\
B &\to 0
\end{aligned}$$

Finally, we can simplify the CNF by merging the rules starting with S_0 and S:

$$\begin{aligned}
S &\to BY_1 \mid BA \mid AY_2 \mid AB \mid \epsilon \\
S_1 &\to BY_1 \mid BA \\
S_2 &\to AY_2 \mid AB \\
Y_1 &\to S_1A \\
Y_2 &\to S_2B \\
A &\to 1 \\
B &\to 0
\end{aligned}$$

Problem 4 (20 pts)

Consider the following state diagram of a PDA,



Assume $\Sigma = \{0, 1\}$

(a) What is the corresponding language? Give the formal definition of this PDA including a table for the δ function.

(b) Give a CFG with ≤ 3 rules for this language.

(c) We would like to have a DPDA for the same language. Give the formal definition including a table for the δ function.

Answer

(a) $\{0^m 1^n \mid m \geq n \geq 0\}$.

Becuase the PDA keeps pushing 0s when it sees input 0, and starts popping 0 when it sees input 1, it will accept the string $\{0^m 1^n\}$ for some m, n. In addition, when it starts popping 0s, it is at accept state, and the start state is also an accept state, so $m \geq n \geq 0$.

Its formal definition is $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

- $Q = \{q_1, q_2\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0\}$
- q_1 is the start state.
- $\{q_1, q_2\}$ is the set of accept states.
- The transition function δ is:

	0		1		ϵ		Σ_ϵ
	0	ϵ	0	ϵ	0	ϵ	Γ_ϵ
q_1	$\emptyset$	$\{(q_1, 0)\}$	$\{(q_2, \epsilon)\}$	$\emptyset$	$\emptyset$	$\emptyset$	
q_2	$\emptyset$	$\emptyset$	$\{(q_2, \epsilon)\}$	$\emptyset$	$\emptyset$	$\emptyset$	

(b) We can split the string accepted by the PDA in (a) into $0^n 0^{m-n} 1^n$, so we at least need two rules to generate it. That is, one is for 0^{m-n}, and the other is for 0^n and 1^n. Thus, we can write it as

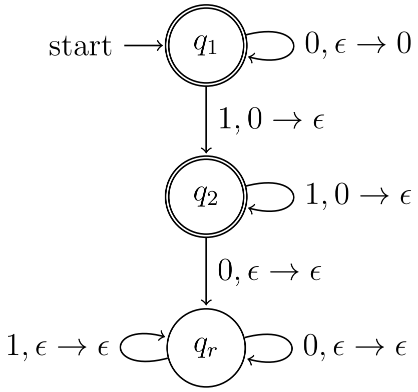
$$\begin{aligned} A &\to 0A1 \mid B \mid \epsilon \\ B &\to 0B \mid \epsilon \end{aligned} \tag{1}$$

In addition, we can further reduce (1) to

$$A \to 0A \mid 0A1 \mid \epsilon$$

because the generating order of 0s is exchangeable.

(c) The DPDA is:


start
q_1
$0, \epsilon \to 0$
$1, 0 \to \epsilon$
q_2
$1, 0 \to \epsilon$
$0, \epsilon \to \epsilon$
$1, \epsilon \to \epsilon$
q_r
$0, \epsilon \to \epsilon$

.

Its formal definition is $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

- $Q = \{q_1, q_2, q_r\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0\}$
- q_1 is the start state.
- $\{q_1, q_2\}$ is the set of accept states.
- The transition function δ is:

	0		1		ϵ		Σ_ϵ
	0	ϵ	0	ϵ	0	ϵ	Γ_ϵ
q_1	$\emptyset$	$(q_1, 0)$	(q_2, ϵ)	$\emptyset$	$\emptyset$	$\emptyset$	
q_2	$\emptyset$	(q_r, ϵ)	(q_2, ϵ)	$\emptyset$	$\emptyset$	$\emptyset$	
q_r	$\emptyset$	(q_r, ϵ)	$\emptyset$	(q_r, ϵ)	$\emptyset$	$\emptyset$	

From the table we can see that for every $q \in Q$, $a \in \Sigma$ and $x \in \Gamma$, exactly one of the values

$$\delta(q, a, x),\ \delta(q, a, \epsilon),\ \delta(q, \epsilon, x) \text{ and } \delta(q, \epsilon, \epsilon)$$

is not $\emptyset$.

Alternative Solution for (c).

1. The last column of the table can also be

q_r	(q_r, ϵ)	$\emptyset$	$\emptyset$	(q_r, ϵ)	$\emptyset$	$\emptyset$

and

q_r	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	(q_r, ϵ)

Common Mistake.

1. δ is:

$$Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to P(Q \times \Gamma_\epsilon),$$

so your δ table should have a column for $\epsilon \in \Sigma_\epsilon$.

Problem 5 (30 pts)

Use the language and diagram in Problem 4.

(a) Modify the PDA to satisfy:

1. Each link is either push or pop;
2. Single accept state;
3. Stack is empty before accepting a string.

Your diagram should have ≤ 4 states. (Hint: you may introduce extra symbols in the stack.)

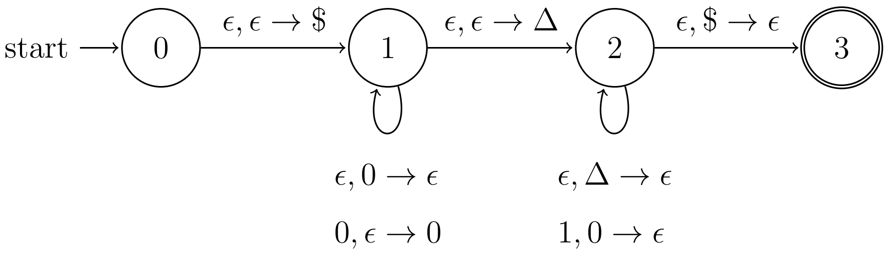
(b) Use (a) and the procedure in Lemma 2.27 to generate a CFG for the language.

(c) Is it possible to simplify results in (b) to those in Problem 4 (b)?

(d) Give two strings s_1 and s_2 with $|s_1| = |s_2| = 3$, $s_1, s_2 \in \{0, 1\}^*$, and s_1 is accepted while s_2 is rejected. Use the PDA obtained in (a) to simulate the two strings. Note that you must draw trees like what we did in the lecture (it is similar to how we simulate NFA).

Answer

(a) Consider the PDA of Problem 4. It accepts the strings with non-empty stack for $m > n$. To make the stack empty when accepting the string, it is necessary to pop some 0s from stack earlier. The idea is to pop 0s nondeterministically before reading 1s from input and accept only some unique number of pops. Therefore, we need at least two states, one is for pushing 0s, and the other is for popping 0s. However, we cannot link these two states with $\epsilon, \epsilon \to \epsilon$ because each operation should be push or pop. To do so, we introduce a new stack variable Δ, and push it once nondeterministically when reading 0s from input, then pop it before reading 1s from input. In addition, we add a start state 0 and an accept state 3, as well as the push/pop operation of \$.


start
0
$\epsilon, \epsilon \to \$$
1
$\epsilon, \epsilon \to \Delta$
2
$\epsilon, \$ \to \epsilon$
3
$\epsilon, 0 \to \epsilon$
$0, \epsilon \to 0$
$\epsilon, \Delta \to \epsilon$
$1, 0 \to \epsilon$

(b) To convert the PDA to CFG, we start from the start variable A_{03} to get

$$\begin{aligned} &A_{03} \to A_{12} \\ &A_{12} \to 0A_{12}1 \mid 0A_{11} \mid A_{22} \qquad (2) \\ &A_{11} \to 0A_{11} \end{aligned}$$

Then, we add rules

$$A_{ik} \to A_{ij}A_{jk} \ (0 \le i, j, k \le 3, \text{ total 64 rules}) \qquad (3)$$

Finally, we add rules

$$A_{ii} \to \epsilon \ (0 \le i \le 3, \text{ total 4 rules}) \qquad (4)$$

(c) Yes.

To simplify the CFG, we first show that rules in (3) are redundant.

Because only A_{03}, A_{12}, A_{00}, A_{11}, A_{22}, A_{33} appear on the left-hand side of (2) and (4), all we have to do is to show that for these A_{ij},

$$A_{ij} \to A_{i?}A_{?j}$$

is redundant. First, we consider the rules

$$A_{ii} \to A_{ij}A_{ji}, 0 \le i \le 3, 0 \le j \le 3$$

For $j = 0$, we have

$$A_{ii} \to A_{i0}A_{0i}.$$

Because A_{00} is the only one among A_{i0} that appear in (2) and (4),

$$\begin{aligned} A_{ii} &\to A_{ix} \cdots A_{y0}A_{0i} \\ &\to \text{a string} \end{aligned}$$

implies that $y = 0$. Then eventually $x = 0$, but $A_{i0}, i \neq 0$ does not generate any string by (2) or (4). Therefore, the only possibility is that $i = 0$ and

$$A_{00} \to A_{00}A_{00} \cdots A_{00}. \qquad (5)$$

For $j = 2$ and 3, the situation is similar because $A_{22} \to \epsilon$ and $A_{33} \to \epsilon$ are what we only have in (2) and (4).

For $j = 1$, note that in (2) we have

$$A_{11} \to 0A_{11}.$$

By a similar argument, all we have is

$$\begin{aligned} A_{11} \to & \text{ a string of } A_{11} \text{ and } 0 \\ \to & \text{ a string of } 0^* \end{aligned} \tag{6}$$

Clearly, (5) and (6) can be handled by rules in (2) and (4), so they are redundant.

For A_{03}, we have

$$\begin{aligned} A_{03} &\to A_{0x}A_{x3} \\ &\to A_{00}A_{03} \\ &\to A_{00}A_{0x} \cdots A_{03} \end{aligned} \tag{7}$$

or

$$\begin{aligned} A_{03} &\to A_{03}A_{33} \\ &\to A_{03}A_{3x} \cdots A_{33}. \end{aligned} \tag{8}$$

The reason is that A_{00} and A_{33} are those of $A_{0?}$ that appear in (2) and (4). For (7), x must be 0 or 3, while for (8), x must be 3 because $A_{3?}$ has only A_{33} in (2) and (4). In the end, the right-hand side of (7) and (8) is a string of A_{00} and A_{03}, or A_{03} and A_{33}. From (2) and (4), they are redundant.

The situation for $A_{12} \to A_{1?}A_{?2}$ is similar. Therefore, all the 64 rules in (4) can be removed.

Then, by changing the variables A_{03} to S, A_{12} to A, A_{11} to B in (b) and removing the redundant rules, we can get

$$\begin{aligned} &S \to A \\ &A \to 0A1 \mid 0B \mid \epsilon \\ &B \to 0B \mid \epsilon \end{aligned} \tag{9}$$

We can replace

$$A \to 0B \mid \epsilon$$

in (9) with

$$A \to B$$

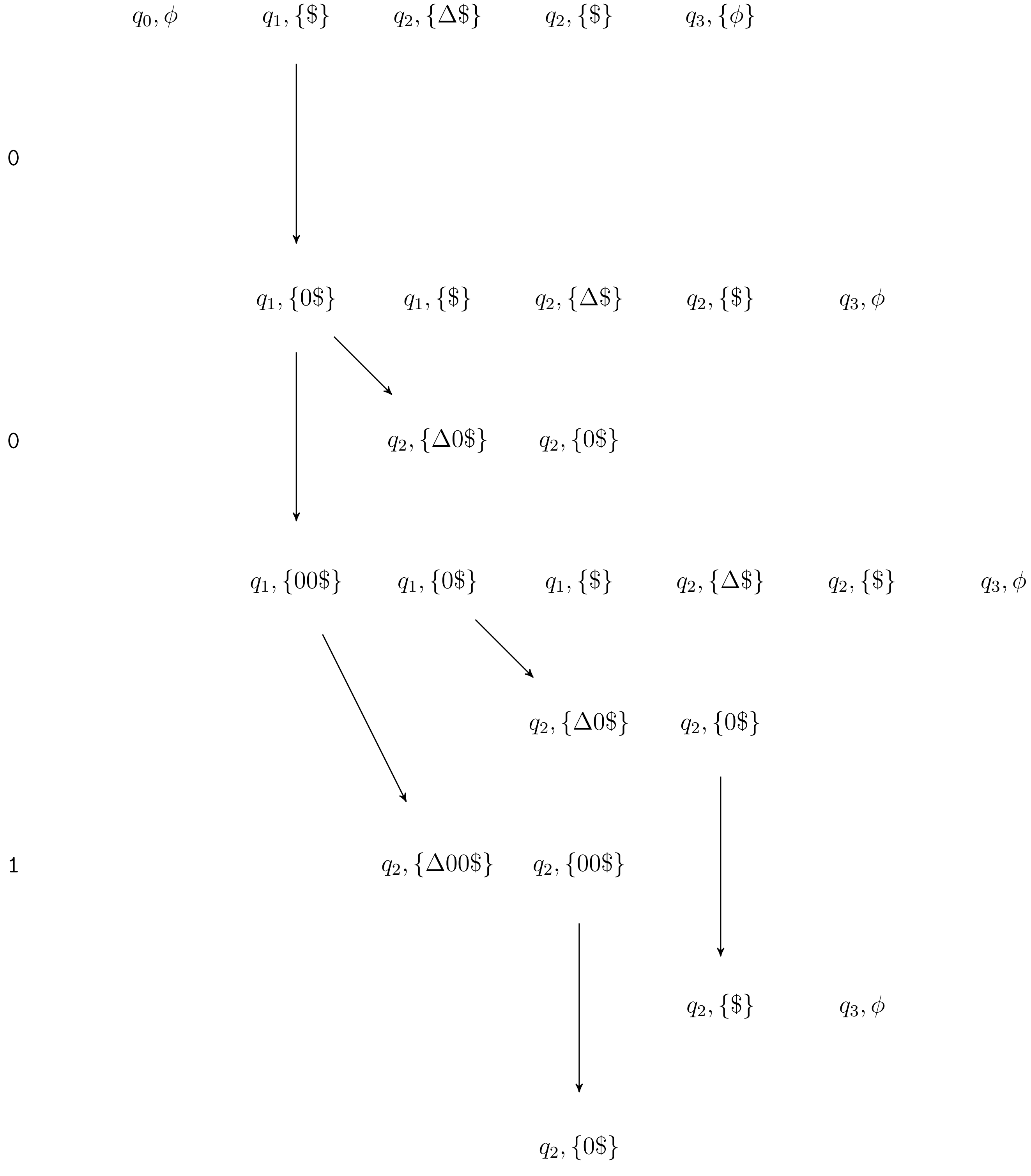
because of

$$B \to 0B \mid \epsilon$$

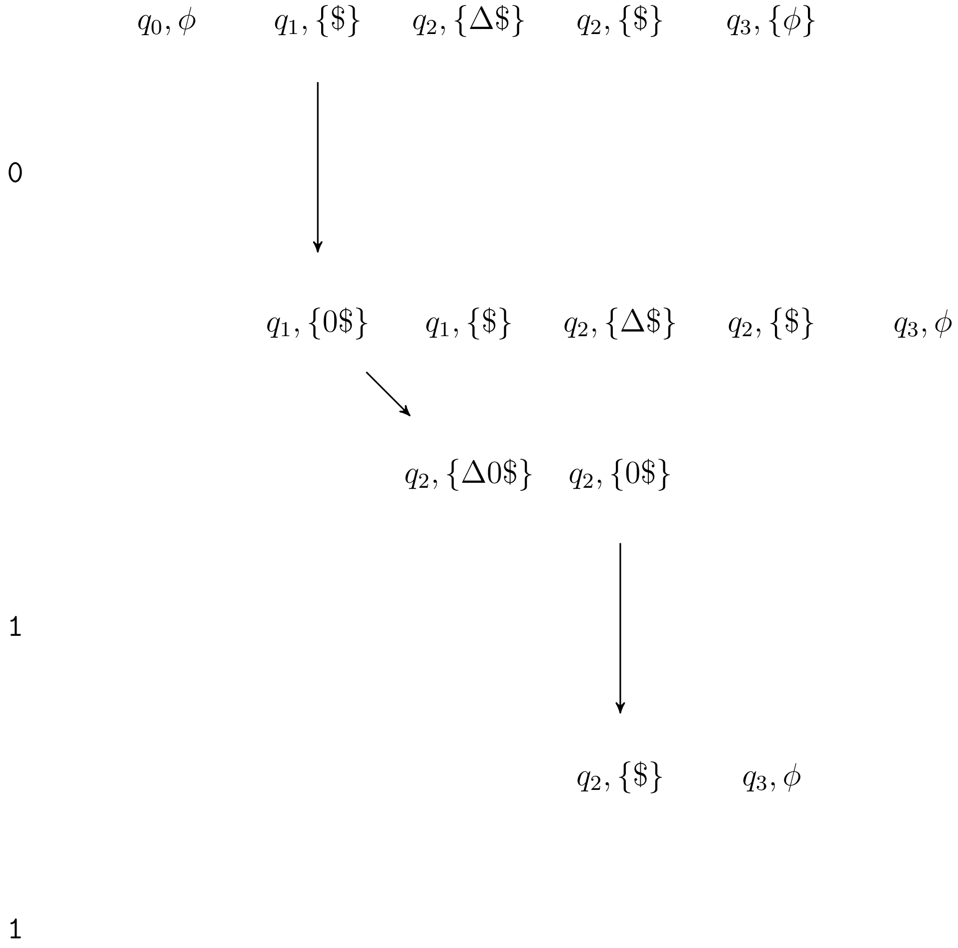
Finally, (9) becomes the same as (1)

(d) Let $s_1 = 001$ and $s_2 = 011$.

1. Simulate s_1:



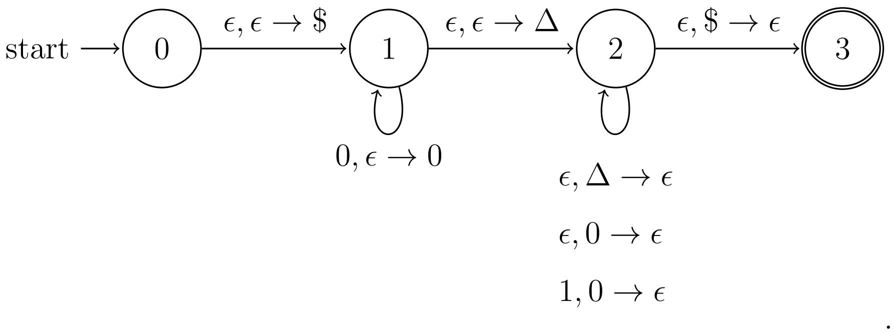
2. Simulate s_2:



Therefore, s_1 is accepted and s_2 is rejected.

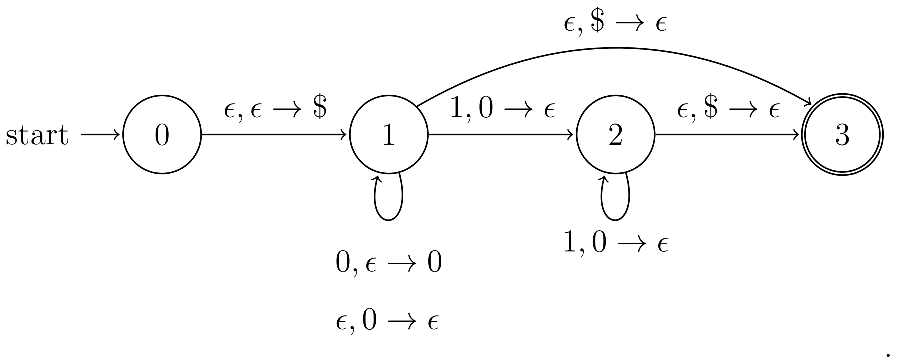
Alternative Solution for (a).

1. Here is the alternative PDA:

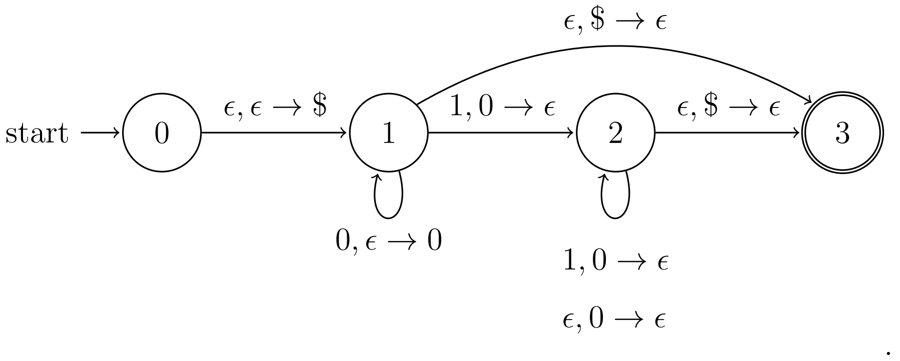


start
0
1
2
3
$\epsilon, \epsilon \to \$$
$\epsilon, \epsilon \to \Delta$
$\epsilon, \$ \to \epsilon$
$0, \epsilon \to 0$
$\epsilon, \Delta \to \epsilon$
$\epsilon, 0 \to \epsilon$
$1, 0 \to \epsilon$

2. Here is another alternative PDA:


$\epsilon, \$ \to \epsilon$
start
0
$\epsilon, \epsilon \to \$$
1
$1, 0 \to \epsilon$
2
$\epsilon, \$ \to \epsilon$
3
$0, \epsilon \to 0$
$\epsilon, 0 \to \epsilon$
$1, 0 \to \epsilon$

However, you cannot do


$\epsilon, \$ \to \epsilon$
start
0
$\epsilon, \epsilon \to \$$
1
$1, 0 \to \epsilon$
2
$\epsilon, \$ \to \epsilon$
3
$0, \epsilon \to 0$
$1, 0 \to \epsilon$
$\epsilon, 0 \to \epsilon$

because 0^+ cannot be accepted.

Common Mistakes.

1. For (c), you need to explain why the 64 rules of

$$A_{ij} \to A_{ik}A_{kj}$$

are redundant.

Problem 6 (20 pts)

Consider the language

$$\{0^{2^n} \mid n \geq 0\}$$

We would like to use a 2-tape TM to recognize the language by following procedure:
Repeat

1. Copy half of tape 1 to tape 2 and make tape 1 empty;

2. Copy tape 2 back to tape 1;

until tape 1 is empty.

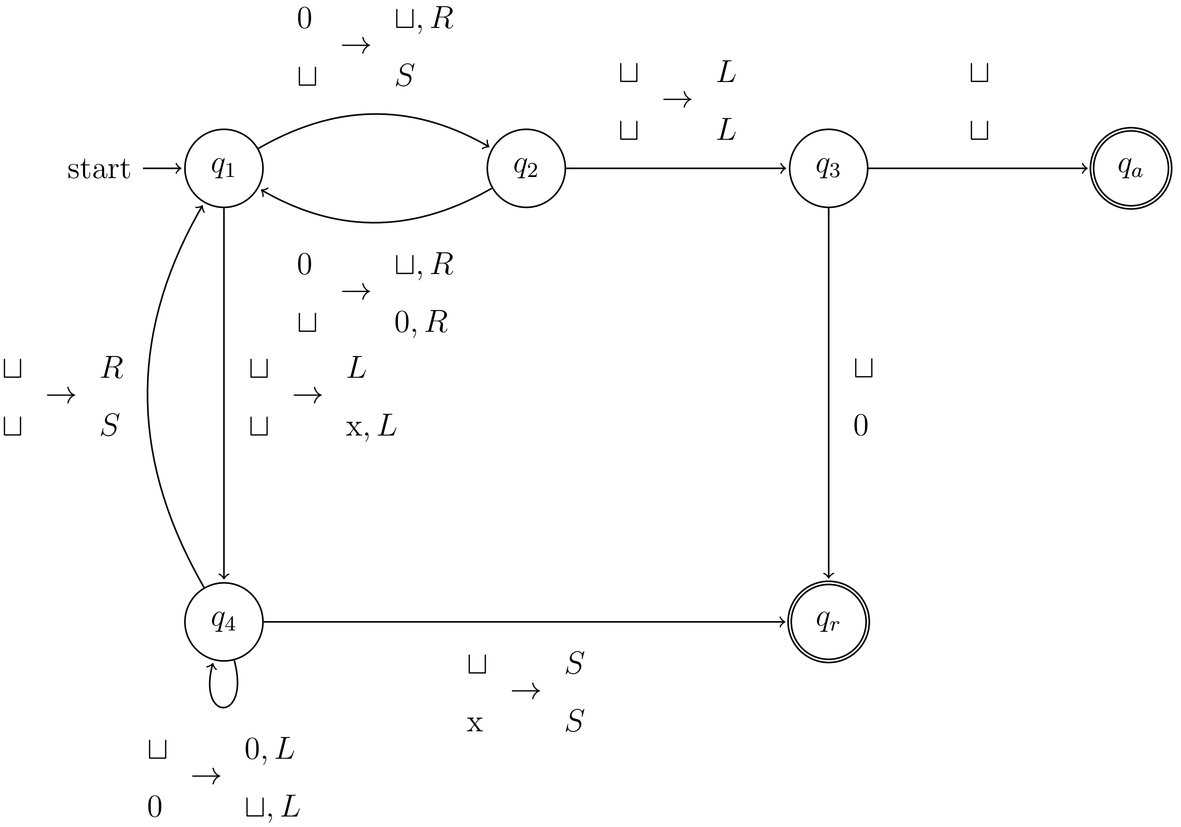
(a) Please give a state diagram for this TM. The number of states should be ≤ 6 (including q_a and q_r).

(b) Simulate the strings "ϵ" (i.e., empty string), "0000" and "000000".

Please note that the movement of the head can be L, R and S.

Answer

(a) The state diagram of this TM can be described as follows:



$\begin{smallmatrix}0 \\ \sqcup\end{smallmatrix} \rightarrow \begin{smallmatrix}\sqcup, R \\ S\end{smallmatrix}$
$\begin{smallmatrix}\sqcup \\ \sqcup\end{smallmatrix} \rightarrow \begin{smallmatrix}L \\ L\end{smallmatrix}$
$\begin{smallmatrix}\sqcup \\ \sqcup\end{smallmatrix}$
start
q_1
q_2
q_3
q_a
$\begin{smallmatrix}0 \\ \sqcup\end{smallmatrix} \rightarrow \begin{smallmatrix}\sqcup, R \\ 0, R\end{smallmatrix}$
$\begin{smallmatrix}\sqcup \\ \sqcup\end{smallmatrix} \rightarrow \begin{smallmatrix}R \\ S\end{smallmatrix}$
$\begin{smallmatrix}\sqcup \\ \sqcup\end{smallmatrix} \rightarrow \begin{smallmatrix}L \\ \text{x}, L\end{smallmatrix}$
$\begin{smallmatrix}\sqcup \\ 0\end{smallmatrix}$
q_4
q_r
$\begin{smallmatrix}\sqcup \\ \text{x}\end{smallmatrix} \rightarrow \begin{smallmatrix}S \\ S\end{smallmatrix}$
$\begin{smallmatrix}\sqcup \\ 0\end{smallmatrix} \rightarrow \begin{smallmatrix}0, L \\ \sqcup, L\end{smallmatrix}$

.

All other links go to q_r.

This TM does the following actions.

i) Copy half of tape 1 to tape 2 (Loop between q_1 and q_2)
If the length is even, the loop stops at q_1 and goes to q_4.
If the length is odd, stops at q_2 and goes to q_3.

ii) If the length of tape 1 is even, reject if it is empty. ($q_4 \to q_r$)

Otherwise, we should copy tape 2 to tape 1 (Loop on q_4). Then we should start from i) again, to check if it is 0^{2^n} for some n.

iii) If the length of tape 1 is odd, then we are at q_3 and tape 2 may or may not have 0s.

We then can check if tape 2 has 0s or not.

Accept if it is blank. That is, tape 2 has no 0s, and the length is 1. ($q_3 \to q_a$)

Reject otherwise. The length of tape 1 is an odd number larger than 1. ($q_3 \to q_r$)

The reason of putting an x on tape 2 in $q_1 \to q_4$ is to distinguish the empty input from other even length inputs on q_1. If the additional alphabet is not added, then an additional state is required to handle this.

(b) Simulate empty string, string "0000" and "000000" as follows (read down the columns and left to right):

1. Simulate ϵ:

$$\left\{\begin{array}{l} q_1\sqcup \\ q_1\sqcup \end{array}\right.$$

$$\left\{\begin{array}{l} q_4\sqcup \\ q_4\text{x} \end{array}\right.$$

Reject ϵ.

2. Simulate "0000":

$$\begin{array}{llll}
\left\{\begin{array}{l} q_1 0000 \\ q_1\sqcup \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup q_1\sqcup \\ 00q_1\sqcup \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup q_1 00 \\ q_1\sqcup\sqcup\text{x} \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup q_4\sqcup 0 \\ q_4\sqcup\text{xx} \end{array}\right. \\
\left\{\begin{array}{l} \sqcup q_2 000 \\ q_2\sqcup \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup\sqcup q_4\sqcup \\ 0q_4 0\text{x} \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup\sqcup q_2 0 \\ q_2\sqcup\sqcup\text{x} \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup\sqcup q_1 0 \\ q_1\sqcup\text{xx} \end{array}\right. \\
\left\{\begin{array}{l} \sqcup\sqcup q_1 00 \\ 0q_1\sqcup \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup q_4\sqcup 0 \\ q_4 0\sqcup\text{x} \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup q_1\sqcup \\ 0q_1\sqcup\text{x} \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup q_2\sqcup \\ q_2\sqcup\text{xx} \end{array}\right. \\
\left\{\begin{array}{l} \sqcup\sqcup\sqcup q_2 0 \\ 0q_2\sqcup \end{array}\right. &
\left\{\begin{array}{l} \sqcup q_4\sqcup 00 \\ q_4\sqcup\sqcup\text{x} \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup\sqcup q_4\sqcup \\ q_4 0\text{xx} \end{array}\right. &
\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup q_3\sqcup \\ q_3\sqcup\text{xx} \end{array}\right.
\end{array}$$

Accept 0000

3. Simulate “000000”:

$$\left\{\begin{array}{l} q_1000000 \\ q_1\sqcup \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup q_200000 \\ q_2\sqcup \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup q_10000 \\ 0q_1\sqcup \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup q_2000 \\ 0q_2\sqcup \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup q_100 \\ 00q_1\sqcup \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup\sqcup q_20 \\ 00q_2\sqcup \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup\sqcup\sqcup q_1\sqcup \\ 000q_1\sqcup \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup\sqcup q_4\sqcup \\ 00q_40\mathrm{x} \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup q_4\sqcup 0 \\ 0q_40\sqcup \mathrm{x} \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup q_4\sqcup 00 \\ q_40\sqcup\sqcup \mathrm{x} \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup q_4\sqcup 000 \\ q_4\sqcup\sqcup\sqcup \mathrm{x} \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup q_1000 \\ q_1\sqcup\sqcup\sqcup \mathrm{x} \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup q_200 \\ q_2\sqcup\sqcup\sqcup \mathrm{x} \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup\sqcup q_10 \\ 0q_1\sqcup\sqcup \mathrm{x} \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup\sqcup\sqcup q_2\sqcup \\ 0q_2\sqcup\sqcup \mathrm{x} \end{array}\right.$$

$$\left\{\begin{array}{l} \sqcup\sqcup\sqcup\sqcup\sqcup q_3\sqcup \\ q_30\sqcup\sqcup \mathrm{x} \end{array}\right.$$

Reject 000000

Common Mistakes

1. ϵ should be rejected rather than accepted.

2. By definition, you cannot assume that there is a $\sqcup$ before the input string. However some of you asked the TA during the exam and their answer was yes. So we still give credits if you made such an assumption.

3. You can validate your diagram by checking 0 and 00 first. Both should be accepted.