Introduction to the Theory of Computation 2022 — Midterm 2

Solutions

Problem 1 (25 pts). Suppose that we operate a burger shop and use the following rules to produce hamburgers.

$$\begin{aligned} H &\to bCb \\ C &\to mC \mid CbC \mid \epsilon \end{aligned} \tag{1}$$

where we denote

$$\begin{aligned} H &\coloneqq \text{hamburger}, \\ C &\coloneqq \text{content}, \\ b &\coloneqq \text{bread}, \\ m &\coloneqq \text{meat}. \end{aligned}$$

If we collect the rules (1) as a set R, then we can use a context-free grammar

$$(V = \{H, C\}, \Sigma = \{b, m\}, R, S = H) \tag{2}$$

to describe a hamburger language.

(a) (10 pts) Convert the grammar to a PDA by the procedure in Lemma 2.21 of the textbook (in our slides chap2_PDA3.pdf.)

(b) (10 pts) Convert the grammar to CNF by the procedure in Theorem 2.9 of the textbook (in our slides chap2_CNF2.pdf.)

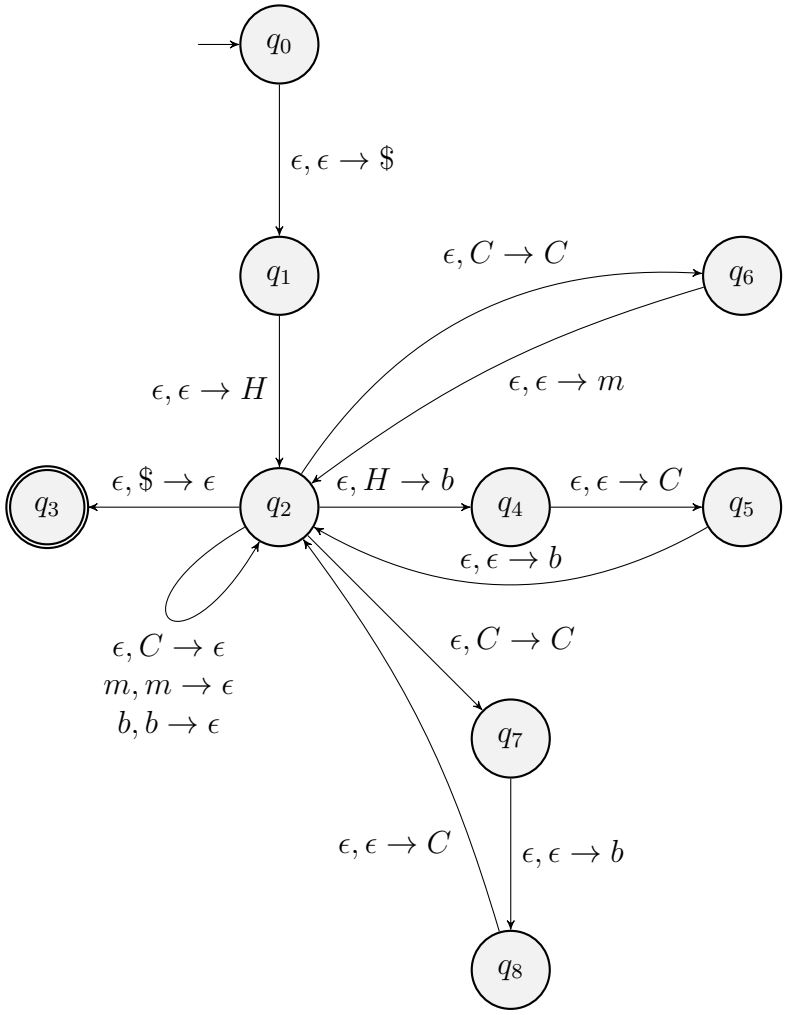
(c) (5 pts) Is a Big Mac

$$bmbmb$$

ambiguous under our hamburger context-free grammars (2)?

Solution.

(a) Please see the following diagram.



(b) Add $S_0 \to H$

$$\begin{aligned} S_0 &\to H \\ H &\to bCb \\ C &\to mC \mid CbC \mid \epsilon \end{aligned}$$

Remove $C \to \epsilon$

$$\begin{aligned} S_0 &\to H \\ H &\to bCb \mid bb \\ C &\to mC \mid m \mid CbC \mid Cb \mid bC \mid b \end{aligned}$$

Remove $S_0 \to H$

$$\begin{aligned} S_0 &\to bCb \mid bb \\ C &\to mC \mid m \mid CbC \mid Cb \mid bC \mid b \end{aligned}$$

Add $B \to b$

$$\begin{aligned} S_0 &\to BCB \mid BB \\ C &\to mC \mid m \mid CBC \mid CB \mid BC \mid b \\ B &\to b \end{aligned}$$

Add $M \to m$

$$\begin{aligned} S_0 &\to BCB \mid BB \\ C &\to MC \mid m \mid CBC \mid CB \mid BC \mid b \\ B &\to b \\ M &\to m \end{aligned}$$

Remove BCB

$$\begin{aligned} S_0 &\to UB \mid BB \\ C &\to MC \mid m \mid CBC \mid CB \mid BC \mid b \\ B &\to b \\ M &\to m \\ U &\to BC \end{aligned}$$

Remove CBC

$$\begin{aligned} S_0 &\to UB \mid BB \\ C &\to MC \mid m \mid CU \mid CB \mid BC \mid b \\ B &\to b \\ M &\to m \\ U &\to BC \end{aligned}$$

(c) Here is the leftmost derivation.

$$\begin{aligned} H &\Rightarrow bCb \\ &\Rightarrow bmCb \\ &\Rightarrow bmCbCb \\ &\Rightarrow bmbCb \\ &\Rightarrow bmbmCb \\ &\Rightarrow bmbmb \end{aligned}$$

However, there exists another leftmost derivation.

$$\begin{aligned} H &\Rightarrow bCb \\ &\Rightarrow bCbCb \\ &\Rightarrow bmCbCb \\ &\Rightarrow bmbCb \\ &\Rightarrow bmbmCb \\ &\Rightarrow bmbmb \end{aligned}$$

Therefore, the Big Mac

$$bmbmb$$

is ambiguous under our hamburger context-free grammar.

Problem 2 (20 pts). Consider the alphabet $\Sigma = \{0, 1\}$ and the language

$$A = \{0^m 1^n \mid 0.5n \leq m \leq 1.5n\}$$

(a) (5 pts) Let m, n be nonnegative integers and $c_1, \ldots, c_n$ be integers with $c_i \in \{1, 2, 3\}$ for each $1 \leq i \leq n$. Show that

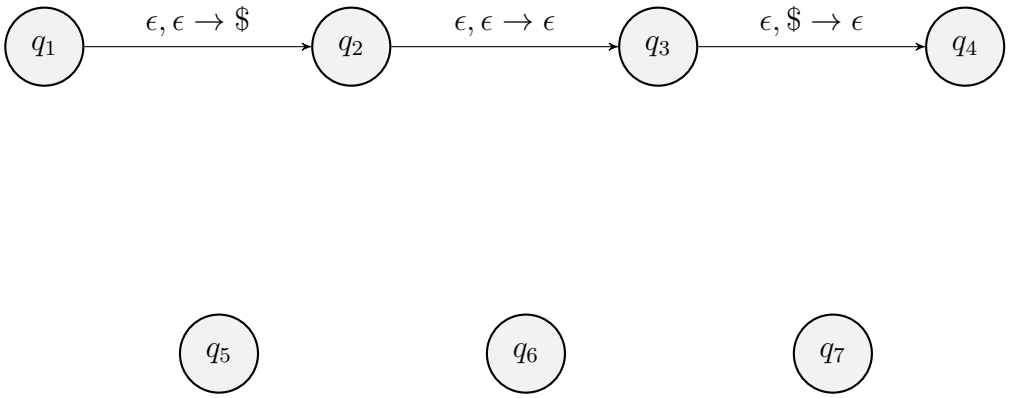
$$0.5n \leq m \leq 1.5n \text{ if and only if there exist } c_1, c_2, \ldots, c_n \text{ such that } c_1 + c_2 + \ldots + c_n = 2m.$$

You need to prove both the "if" and "only if" directions.

(b) (10 pts) Based on the idea from (a), design a PDA with ≤ 7 states to recognize A. The stack alphabet is restricted to be

$$\Gamma = \{0, \$\}.$$

Please finish the diagram based on the following one to get full points.



You must briefly explain the design of your diagram.

(c) (5 pts) Following page 5 of slide "chap2_PDA2.pdf", please simulate your PDA in the previous subproblem on the two strings "001" and "011" by drawing the corresponding simulation trees. Then determine whether the PDA accepts the two strings according to your simulation.

Solution.

(a) For the "if" direction, since $c_i \in \{1, 2, 3\}$, we have

$$n \leq c_1 + \cdots + c_n = 2m \leq 3n,$$

which gives $0.5n \leq m \leq 1.5n$.

As for the "only if" direction, we show the existence by construction. For $n \leq 2m < 2n$, we may let $(c_1, \ldots, c_n)$ to be any combination of $(2n - 2m)$ 1's and $(2m - n)$ 2's. For $2n \leq 2m \leq 3n$, we may let $(c_1, \ldots, c_n)$ to be any combination of $(3n - 2m)$ 2's and $(2m - 2n)$ 3's.

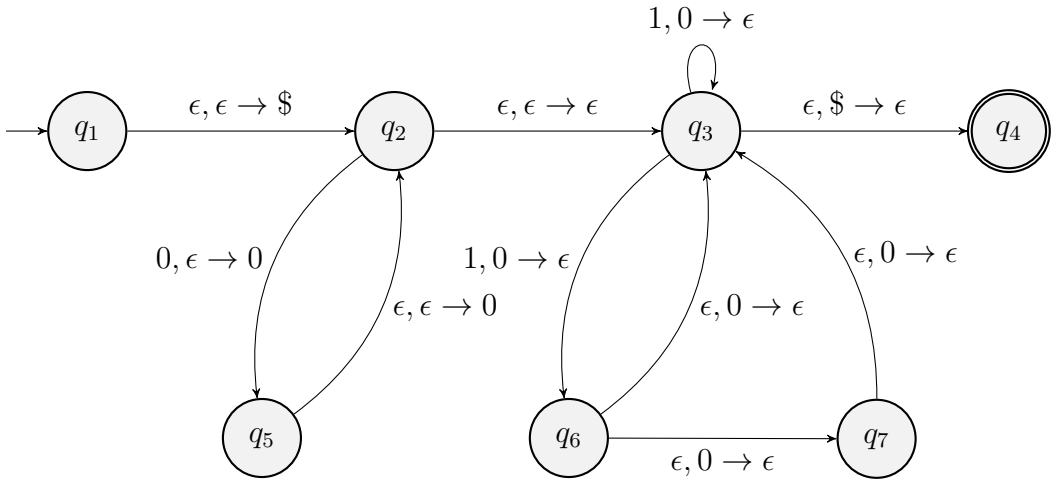
Common mistake: For the "only if" direction, we need to prove the existence of $c_1, \ldots, c_n$.

Some simply say that $c_1 + \cdots c_n$ can be any integer between n and $3n$, but you need a proof on that. Some try to prove the opposite statement. The opposite statement of the right-hand side is

$$\forall c_1, \ldots, c_n,\ c_1 + \cdots + c_n \neq 2m.$$

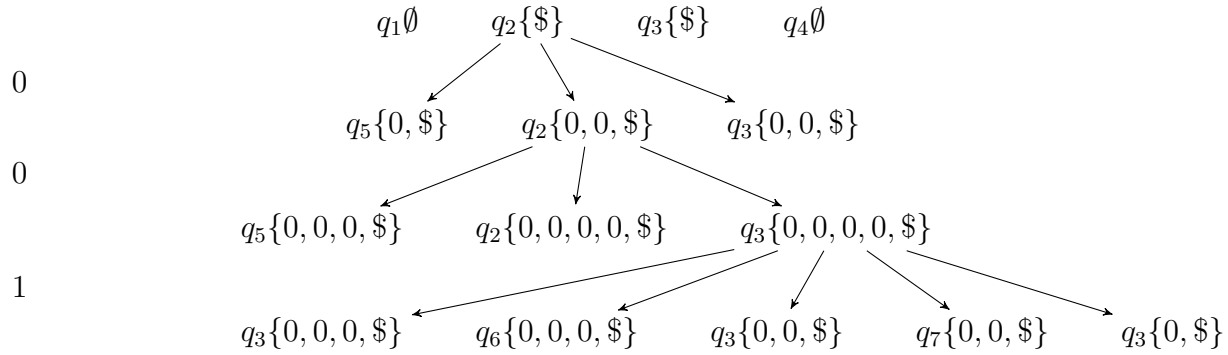
But then it is unclear how to prove that $0.5n \leq m \leq 1.5n$ is false.

(b) The diagram is shown as the following:



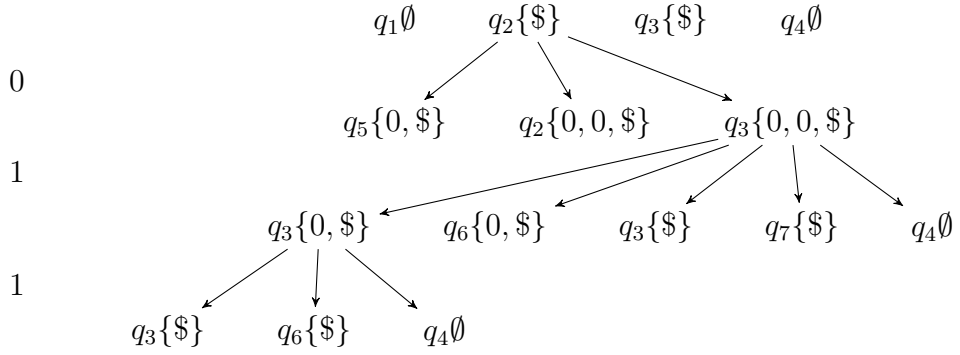
For q_2 and q_5, the PDA pushes two 0's to the stack upon reading an input 0. Then the PDA nondeterministically guess whether the input has completed reading 0's. Then for q_3, q_6, q_7, upon receiving an input 1, the PDA nondeterministically pops one, two, or three 0's from the stack via the path $q_3 \to q_3$, $q_3 \to q_6 \to q_3$, or $q_3 \to q_6 \to q_7 \to q_3$, respectively. In the end, we check whether the top of the stack is the \$ pushed in the beginning.

(c) For "001":



Since no branch stops at an accepting state after processing the input strings, "001" is rejected.

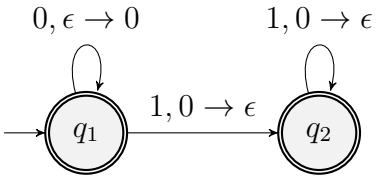
For "011":



After processing the last input character, there's a branch stopping at q_4, so "011" is accepted.

Common mistake: You cannot just show the path that leads to the acceptance or rejection.

Problem 3 (15 pts). Consider the following PDA P with $\Sigma = \{0, 1\}$ and $\Gamma = \{0\}$:



(a) (5 pts) What's the language recognized by P?

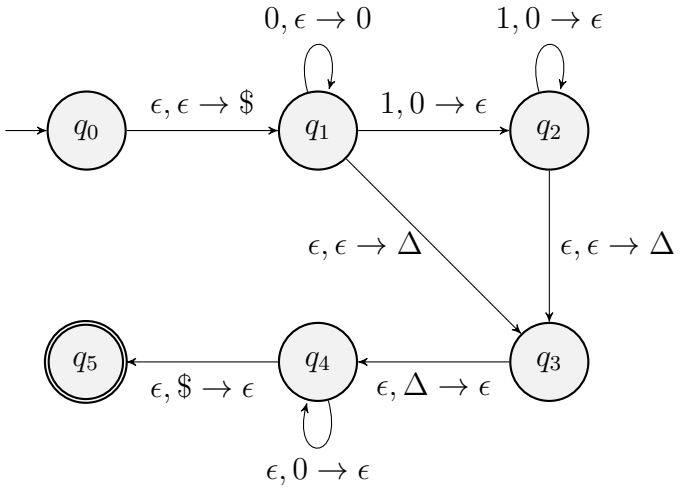
(b) (5 pts) Convert P to P' satisfing the three conditions mentioned in page 2 of "chap2_PDA4.pdf" **using the corrected procedure taught in class**. You are allowed to extend Γ up to a 3-member set, e.g, $\{0, \$, \Delta\}$. Besides, the number of states should be no more than 6 after the conversion. (Your states should be called $q_0, q_1, \ldots, q_5$ to make the notation simpler.)

(c) (5 pts) Based on (b), convert P' to a CFG by using the procedure in Lemma 2.27 of the textbook. For simplicity, you only need to write each $A_{pq} \to aA_{rs}b$ rule. The rules $A_{pq} \to A_{pr}A_{rq}$ and $A_{pp} \to \epsilon$ are not needed. In order to prepare for $A_{pq} \to aA_{rs}b$ rules, please give table(s) for each stack alphabet t pushed/popped, similar to what we had in slides.

Solution.

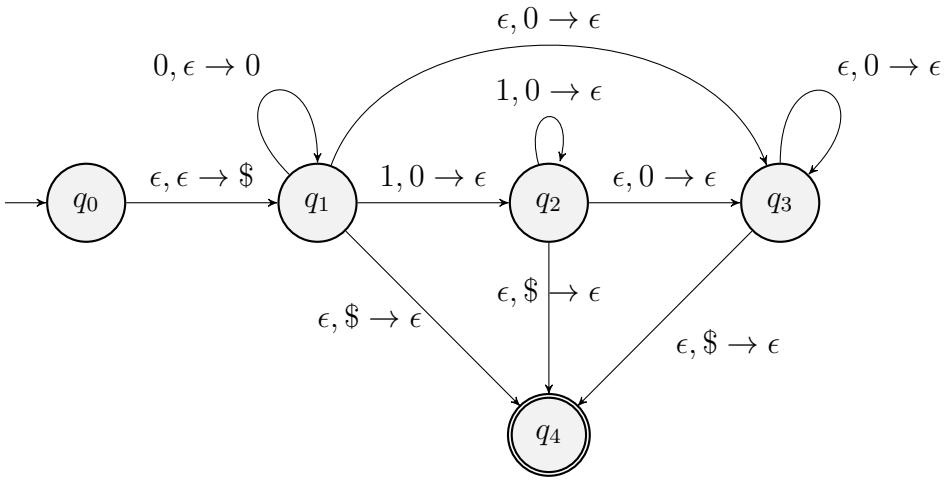
(a) $L(P) = \{0^m 1^n \mid m \geq n \geq 0\}$.

(b) The diagram is like the following:



States q_0, q_4 and q_5 are used to satisfy the first and the second conditions: single accept state and empty stack before accepting. Note that when adding q_4 (which corresponds to q_{pop} in the slide), additional $\epsilon \to \epsilon$ link would be generated, so we need q_3 to satisfy the third condition: either push or pop in each transition.

Alternative Solution:



(c) The rules are as the following:

- $t = \Delta$:

p	r	s	q	a	b	rules
1	3	3	4	ϵ	ϵ	$A_{14} \to A_{33}$
2	3	3	4	ϵ	ϵ	$A_{24} \to A_{33}$

- $t = \$$:

p	r	s	q	a	b	rules
0	1	4	5	ϵ	ϵ	$A_{05} \to A_{14}$

- $t = 0$:

p	r	s	q	a	b	rules
1	1	1	2	0	1	$A_{12} \to 0A_{11}1$
1	1	2	2	0	1	$A_{12} \to 0A_{12}1$
1	1	4	4	0	ϵ	$A_{14} \to 0A_{14}$

Alternative Solution:

- $u = 0$:

p	r	s	q	a	b	rules
1	1	1	2	0	1	$A_{12} \to 0A_{11}1$
1	1	2	2	0	1	$A_{12} \to 0A_{12}1$
1	1	1	3	0	ϵ	$A_{13} \to 0A_{11}$
1	1	2	3	0	ϵ	$A_{13} \to 0A_{12}$
1	1	3	3	0	ϵ	$A_{13} \to 0A_{13}$

- $u = \$$:

p	r	s	q	a	b	rules
0	1	1	4	ϵ	ϵ	$A_{04} \to A_{11}$
0	1	2	4	ϵ	ϵ	$A_{04} \to A_{12}$
0	1	3	4	ϵ	ϵ	$A_{04} \to A_{13}$

Problem 4 (20 pts). Consider the language

$$L = \{w \in \{0,1\}^* \mid w \text{ contains } 01101\}.$$

(a) (10 pts) Give the diagram of a PDA that recognizes L with at most four states. You should use $\Sigma = \Gamma = \{0, 1\}$ for your PDA.

(b) (10 pts) Intuitively, the language L can easily also be recognized by an NFA. Here we discuss the relation of PDA and NFA. Suppose

$$P = (Q, \Sigma, \Gamma, \delta, q_0, F)$$

is a PDA that

uses only a finite amount of stack cells, say m cells, in processing any input string. (3)

In other words, P is still a normal PDA with infinite stack space, but P is assumed to use only a finite amount of stack cells due to its design. For example, the PDA in subproblem (a) can be designed to use a finite number of stack cells.

We would like to formally define an NFA

$$N = (Q', \Sigma, \delta', q_0', F')$$

in terms of $Q, \Sigma, \Gamma, \delta, q_0$ such that N recognizes the same language as P.

To formally represent the stack contents, we use strings over Γ where the left most character is the top of stack. For example, if the machine pushed 0 first and 1 later, the stack content is represented with the string 10. Using this notation, all possible stack contents that uses at most m cells can be expressed as

$$S = \{s \in \Gamma^* \mid |s| \leq m\}.$$

We can then define the set of states for the NFA to be

$$Q' = Q \times S.$$

Please finish our construction by **giving the definition of δ', q_0' and F'** (you are not required to prove the equivalence formally).

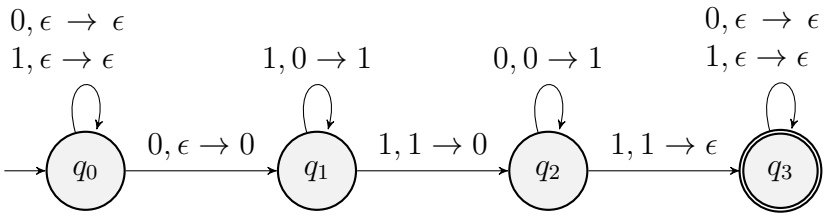
Specifically, the transition function of the NFA is now

$$\delta' : Q' \times \Sigma_\epsilon \to P(Q').$$

Thus for any $(q, s) \in Q'$ and $a \in \Sigma_\epsilon$, what is $\delta'((q, s), a)$?

Solution.

(a) This problem can easily be recognized by an NFA which non-deterministically start to match for 01101 and then stay accepted. However, that would require 6 states. Since PDA has an extra stack that can be used to store "states", this allows us to do it with only four states:



We use the following combination of state and stack to record the matching process:

state	stack	match process
q_0	{}	no match yet
q_1	{0}	0
q_1	{1}	01
q_2	{0}	011
q_2	{1}	0110
q_3	{}	01101

(b) Let s denote the whole stack content and s_i for the ith cell of the stack. Then, the transition function of the NFA transitions according to the state and stack content:

$$\forall (q, s) \in Q', a \in \Sigma_\epsilon$$
$$\delta'((q, s), a) = \begin{cases} A(q, a, s) \cup B(q, a, s) & \text{if } |s| > 0 \\ A(q, a, s) & \text{otherwise} \end{cases}$$

where

$$A(q, a, s) = \{(q', xs) \mid (q', x) \in \delta(q, a, \epsilon) \text{ and } |xs| \leq m\}$$
$$B(q, a, s) = \{(q', xs_2 \dots s_{|s|}) \mid (q', x) \in \delta(q, a, s_1)\}$$

In the definition above, A is the set of states reached without consuming stack symbols, while B is the states reached by consuming the top of the stack. Note that in the definition of A, we do not include those transitions that makes $|xs| > m$ since we need (q', xs) to be in Q'. By assumption (3), the PDA would never have $|xs| > m$ so these transitions can indeed be safely excluded from our δ'.

The starting state of the NFA should be the starting state of the PDA with an empty stack:

$$q_0' = (q_0, \epsilon)$$

Finally, the accept state of the NFA should be the accept state of the PDA with any stack content. Therefore, we have

$$F' = \{(q, s) \mid q \in F, s \in S\}.$$

This result tell us that the reason why PDA is more powerful than NFA (and DFA) is not that it has stack but the stack space is unlimited.

Problem 5 (20 pts). Let x and y be two string such that

$$x = x_1 x_2 \dots x_n \text{ and } y = y_1 y_2 \dots y_m, \text{ where } x_i, y_i \in \{0, 1\}.$$

We define the Kronecker product of x and y, denoted as $x \otimes y$, to be the string

$$(x_1 \cdot y) \circ (x_2 \cdot y) \circ \dots \circ (x_n \cdot y).$$

In the above definition, $\circ$ denotes concatenation and $x_i \cdot y$ is defined as multiplying each character in y by x_i. When either x or y is ϵ, we define $x \otimes y = \epsilon$.

For example, we have

$$01 \otimes 101 = (0 \cdot 101) \circ (1 \cdot 101) = 000101.$$

(a) (15 pts) In this problem, you are required to design a two-tape Turing machine that calculates the Kronecker product. The machine starts with the tape content:

$$\text{tape1 } : x\#y \sqcup \dots$$
$$\text{tape2 } : \sqcup \dots$$

and it should end with the tape content:

$$\text{tape1 } : x\#y \sqcup \dots$$
$$\text{tape2 } : x \otimes y \sqcup \dots$$

Following the textbook, we define the two-tape TM's tape to be only infinite on the right side. Also, the tape heads are allowed to move right, move left or stay. The machine can assume the input string is properly in the form $x\#y$ where $x, y \in \{0,1\}^*$ (i.e., the machine does not have to check the format), and the heads are initialized at the start of the tapes as usual.

You only have to give the diagram. For specifying the transitions, please follow the notation used in lecture slides "chap3_multitapeTM1.pdf". Moreover, you can simplify the diagram by using notations like

$$\begin{cases} \{0,1,\#\} \to 1, R \\ \sqcup \to S \end{cases}$$

if the first tape reads any character in $\{0,1,\#\}$, writes 1 and goes right, while the second tape reads $\sqcup$ and stays.

The state q_{reject} and the edges to q_{reject} can be ignored. The machine should accept after $x \otimes y$ is constructed. The machine should use

$$\Sigma = \{0,1,\#\} \text{ and } \Gamma = \{0,1,\#,\sqcup\}$$

and the number of states should be less than or equal to 7 (including q_{accept} and q_{reject}). (Hint: Make good use of the second tape to reduce the number of states.)

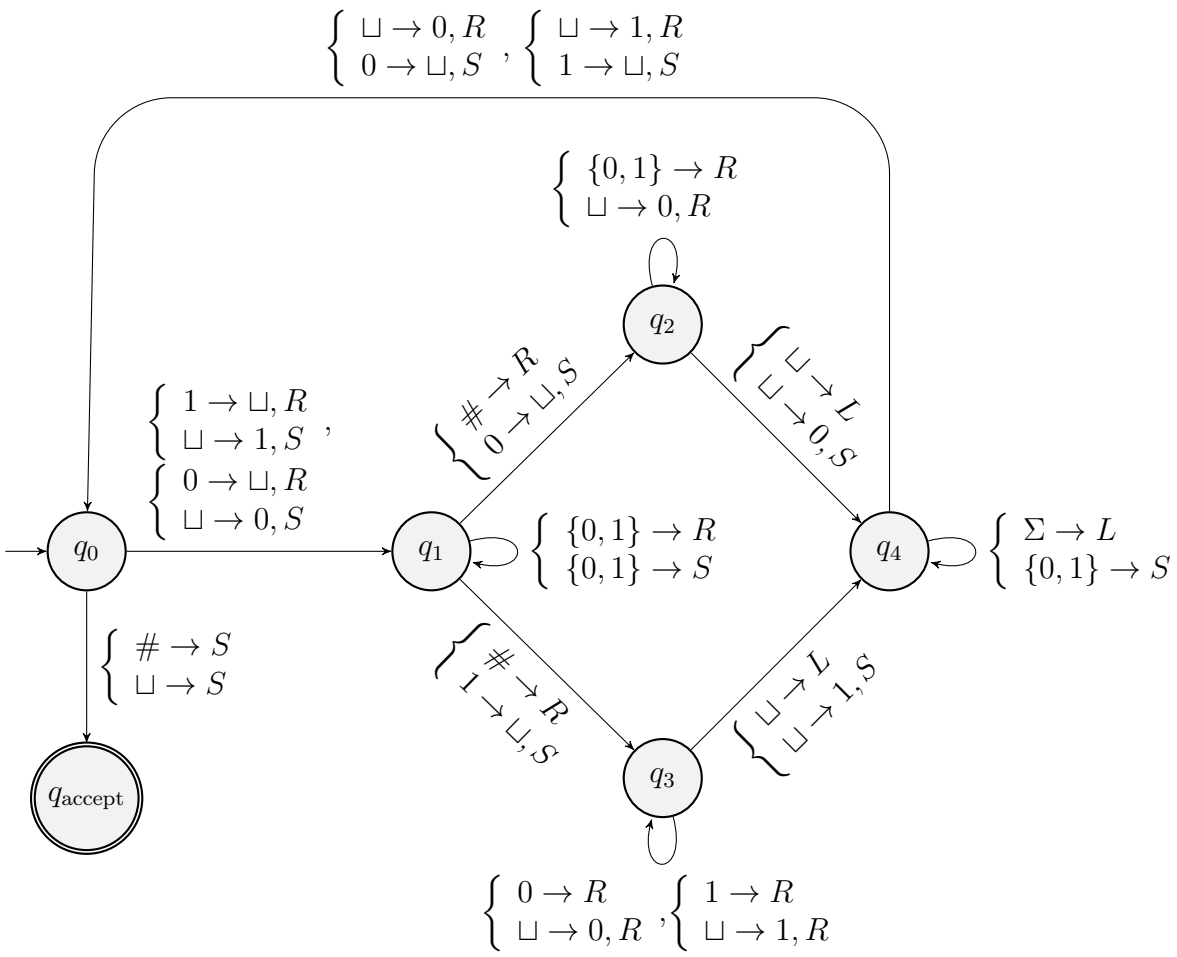
(b) (5 pts) Simulate the machine on the string

$$01\#10$$

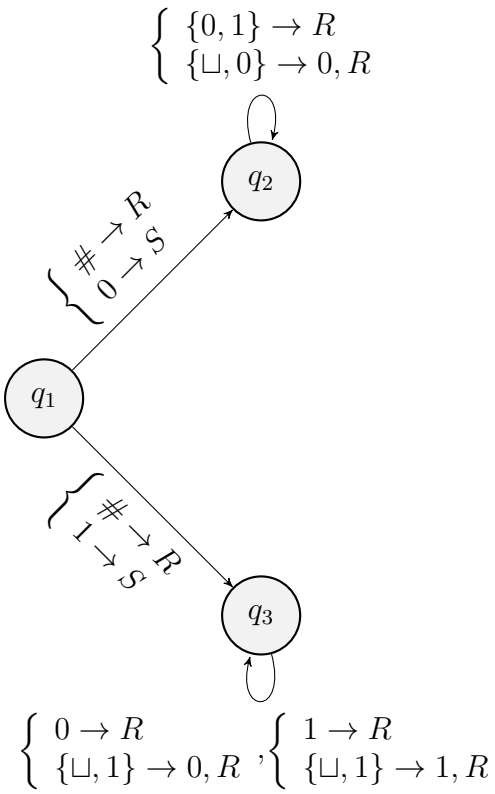
and show the configuration at each step. Multiple steps that involve only head movements (no state or tape change) can be condensed into one step.

Solution.

(a) We provide the diagram with two different notations. The first one specify the rule for tape 1 and tape 2 in the first and second line:



For the design of q_1, q_2 and q_4, we can also do:



At q_0, we copy the current character in x to the second tape and mark its position with $\sqcup$. At q_1, we move the head of first tape to the # in the middle. Then, according to the character we copied earlier, we transition to either q_2 or q_3, where we scan through y and calculate $x_i \cdot y$ on the second tape. When y ends, the machine goes to q_4 while storing x_i onto the second tape again. At q_4, the machine restores the head of the first tape back to the position of x_i, which was marked with $\sqcup$. Then, we restore x_i from the second tape and move to the next character of x. If there are more characters of x, the machine repeats the steps above. Otherwise, it accepts.

Common mistake: We require that contents in tape 1 are not changed.

(b)

$$\begin{array}{l} q_001\#10\sqcup \\ q_0\sqcup \end{array} \Longrightarrow \begin{array}{l} \sqcup q_1 1\#10\sqcup \\ q_1 0\sqcup \end{array} \Longrightarrow \begin{array}{l} \sqcup 1 q_1\#10\sqcup \\ q_1 0\sqcup \end{array} \Longrightarrow \begin{array}{l} \sqcup 1\# q_2 10\sqcup \\ q_2\sqcup \end{array} \Longrightarrow \begin{array}{l} \sqcup 1\#1 q_2 0\sqcup \\ 0 q_2\sqcup \end{array}$$

$$\Longrightarrow \begin{array}{l} \sqcup 1\#10 q_2\sqcup \\ 00 q_2\sqcup \end{array} \Longrightarrow \begin{array}{l} \sqcup 1\#1 q_4 0\sqcup \\ 00 q_4 0\sqcup \end{array} \Longrightarrow \quad \cdots \quad \Longrightarrow \begin{array}{l} q_4 \sqcup 1\#10\sqcup \\ 00 q_4 0\sqcup \end{array} \Longrightarrow \begin{array}{l} 0 q_0 1\#10\sqcup \\ 00 q_0\sqcup \end{array}$$

$$\Longrightarrow \begin{array}{l} 0\sqcup q_1\#10\sqcup \\ 00 q_1 1\sqcup \end{array} \Longrightarrow \begin{array}{l} 0\sqcup \#q_3 10\sqcup \\ 00 q_3\sqcup \end{array} \Longrightarrow \begin{array}{l} 0\sqcup \#1 q_3 0\sqcup \\ 001 q_3\sqcup \end{array} \Longrightarrow \begin{array}{l} 0\sqcup \#10 q_3\sqcup \\ 0010 q_3\sqcup \end{array} \Longrightarrow \begin{array}{l} 0\sqcup \#1 q_4 0\sqcup \\ 0010 q_4 1\sqcup \end{array}$$

$$\Longrightarrow \quad \cdots \quad \Longrightarrow \begin{array}{l} 0 q_4 \sqcup \#10\sqcup \\ 0010 q_4 1\sqcup \end{array} \Longrightarrow \begin{array}{l} 01 q_0\#10\sqcup \\ 0010 q_0\sqcup \end{array} \Longrightarrow \text{accept}$$