Spring25A

Friday, August 29, 2025 9:31 AM

Problem 1 (CO5): Pumping Lemma (5 points)

Consider the following language. [Recall: For any string w, $|w|$ denotes the length of w.]

$$L = \{0^i 1w : w \in \{0,1\}^* \text{ and } i \geq |w|\}$$

Prove that L is not regular using the pumping lemma.

let,

$$\underbrace{0001}_{X}\underbrace{00}_{Y}\underbrace{1}_{Z}$$

$X = 0^3 1^1$; $i = 3$

$Y = 0^2$; $j = 2$

$Z = 1^1$; $k = 1$

$p = j + k$

$i \geq p$

now,

$XYYZ$

$= 0^i 1 0^j 0^j 1^k$

Here,

$i \geq 2j + k$

if $i = 3$, $j = 2$, $k = 1$,

LHS $= 3$ RHS $= 2 \cdot 2 + 1 = 5$

$\therefore i \ngeq 2j + k$

$\therefore$ L is non-regular

Problem 2 (CO3): Designing Context-Free Grammars (15 points)

Consider the following languages.

$$L_1 = \{w \in \{a,b\}^* : w \text{ is a palindrome}\}$$
$$L_2 = \{w \in \{a,b\}^* : w \text{ contains bb as a substring}\}$$
$$L_3 = \{w \in \{a,b\}^* : w \text{ does not contain any palindromic substring of length two}\}$$
$$L_4 = L_1 \cap L_3$$
$$L_5 = \{w_1\#w_2 : w_1, w_2 \in L_2 \text{ and } |w_1| = |w_2|\}$$

Now solve the following problems.

(a) **Give** a context-free grammar that generates L_1. (3 points)

(b) **Give** a context-free grammar that generates L_2. (3 points)

(c) **Give** a context-free grammar that generates L_3. [Note: A palindromic substring is defined as a substring that reads the same backward as forward. For example, in the string 'abab' – 'ε', 'a', 'b', 'aba', 'bab' are palindromic substrings.] (3 points)

(d) **Write** all five-letter strings in L_4. (1 point)

(e) **Give** a context-free grammar that generates L_4. (2 points)

(f) **Give** a context-free grammar that generates L_5. (3 points)

2.

a) $S \to aSa \mid bSb \mid a \mid b \mid \epsilon$

b) $S \to XbbX$

$X \to aX \mid bX \mid \epsilon$

c) $S \to A \mid B$

$A \to aB \mid \epsilon$

$B \to bA \mid \epsilon$

d) ababa

babab

e) $S \to A \mid B$

$A \to aBa \mid a$

$B \to bAb \mid b$

f) $S \to aSa \mid aSb \mid bSa \mid bSb \mid bbPbb$

$P \to aPa \mid aPb \mid bPa \mid bPb \mid \#$

Problem 3 (CO3): Derivations, Parse Trees and Ambiguity (10 points)

Consider the following context-free grammar.

$$S \to DQ$$
$$D \to Daa \mid Dab \mid Dba \mid Dbb \mid a \mid b$$
$$Q \to aQb \mid bQa \mid \varepsilon$$

(a) **Give** a leftmost derivation for the string 'abbababab'. (3 points)

(b) **Draw** the parse tree corresponding to the derivation you gave in (a). (2 points)

(c) **Show** that the given grammar is ambiguous by drawing two more parse trees (apart from the one that you have drawn in (b)) for the given string in (a). (4 points)

a) S

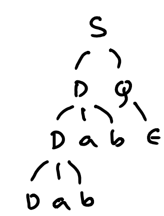
$\to DQ$

$\to DabQ$

$\to DababQ$

$\to DabababQ$

b)



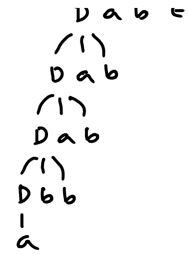
(a) **Give** a leftmost derivation for the string 'abbababab'. (3 points)

(b) **Draw** the parse tree corresponding to the derivation you gave in (a). (2 points)

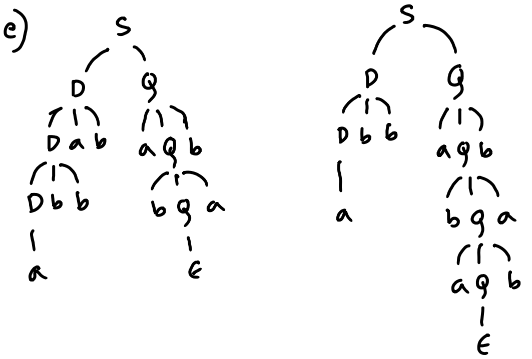
(c) **Show** that the given grammar is ambiguous by drawing two more parse trees (apart from the one that you have drawn in (b)) for the given string in (a). (4 points)

(d) **Write** one five-letter string that has exactly one parse tree in the given grammar. (1 point)

→ DababQ
→ DabababQ
→ DbbabababQ
→ abbabababQ
→ abbababab



c)



d) aaaaa

Problem 4 (CO3): Designing Pushdown Automata (15 points)

Consider the following languages.

$$L_1 = \{0^i 1^j 2^k : j = i + k \text{ or } i = j + k;\ \text{where } i, j, k \geq 0\}$$
$$L_2 = \{w_1 \# w_2 : w_1, w_2 \in \{0,1\}^* \text{ and } |w_1| \neq |w_2|\}$$
$$L_3 = \{w_1 \# w_2 : w_1, w_2 \in \{0,1\}^*, w_1 \text{ contains odd numbers of 0s and } |w_1| = |w_2|\}$$

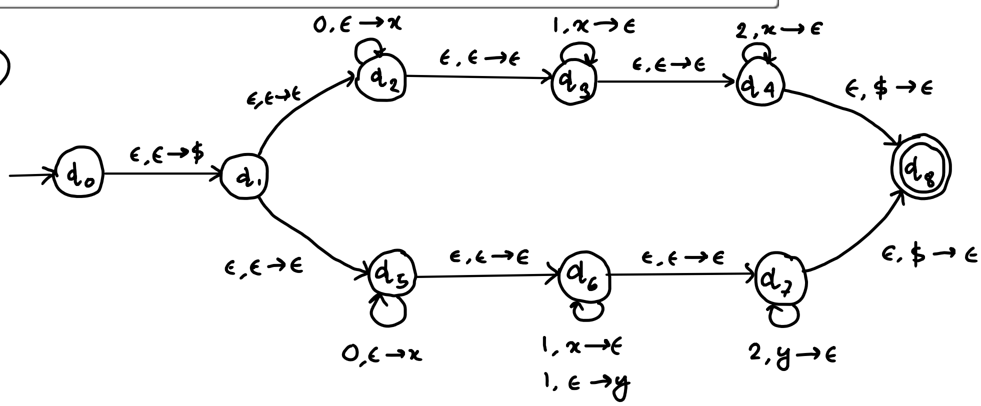
Now solve the following problems.

(a) **Give** the state diagram of a pushdown automaton that recognizes L_1. (5 points)

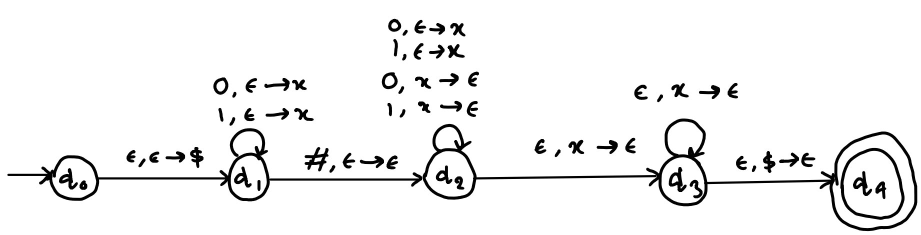
(b) **Give** the state diagram of a pushdown automaton that recognizes L_2. (5 points)

(c) **Give** the state diagram of a pushdown automaton that recognizes L_3. (5 points)

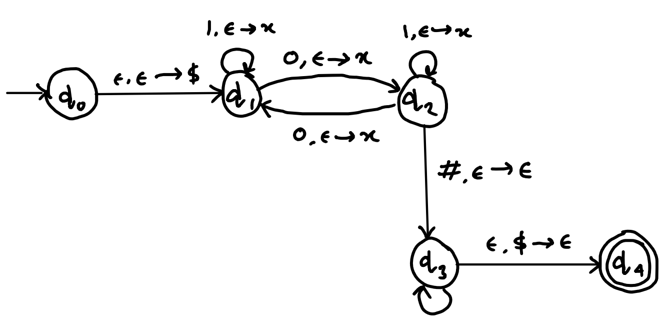
a)



b)



c)



$0, x \to \epsilon$
$1, x \to \epsilon$

Problem 5 (CO4): Turing Machines (5 points)

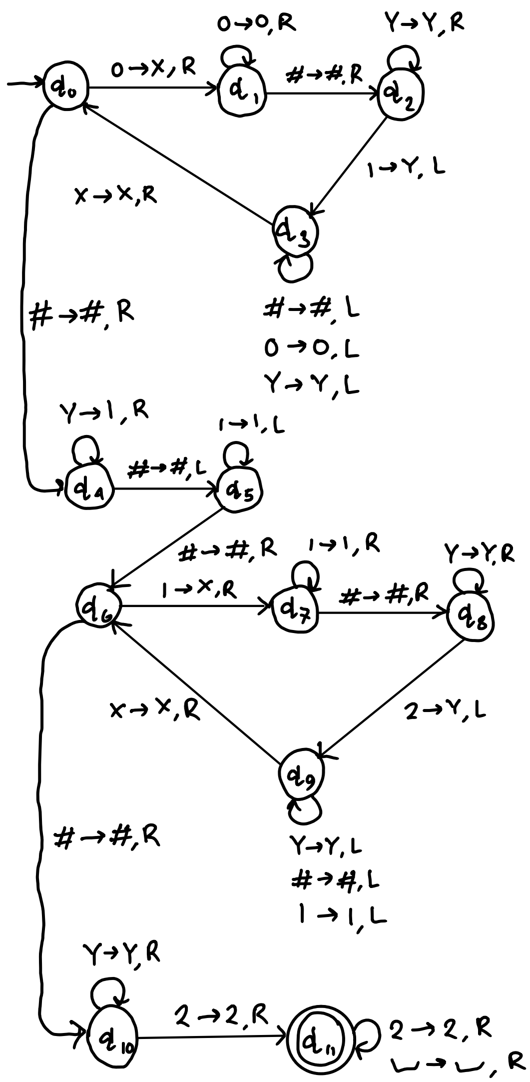
(a) For any Turing-recognizable language L, is it possible to build a recognizer for $\overline{L}$? [Recall: $\overline{L}$ denotes the complement of the language L] (1 point)

Consider the following language.

$$L_1 = \{0^n\#1^n\#2^m : n, m \geq 0 \text{ and } n < m\}$$

(b) **Give** the state diagram of a turing machine that decides L_1. (4 points)

b)

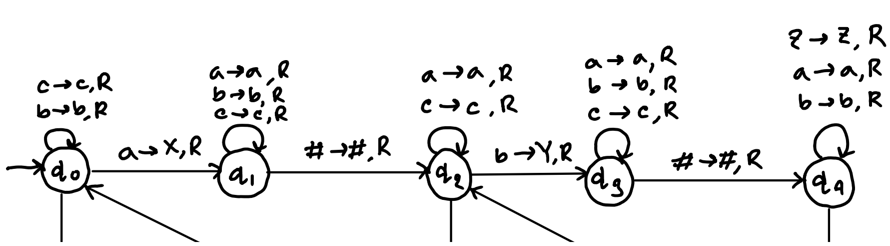


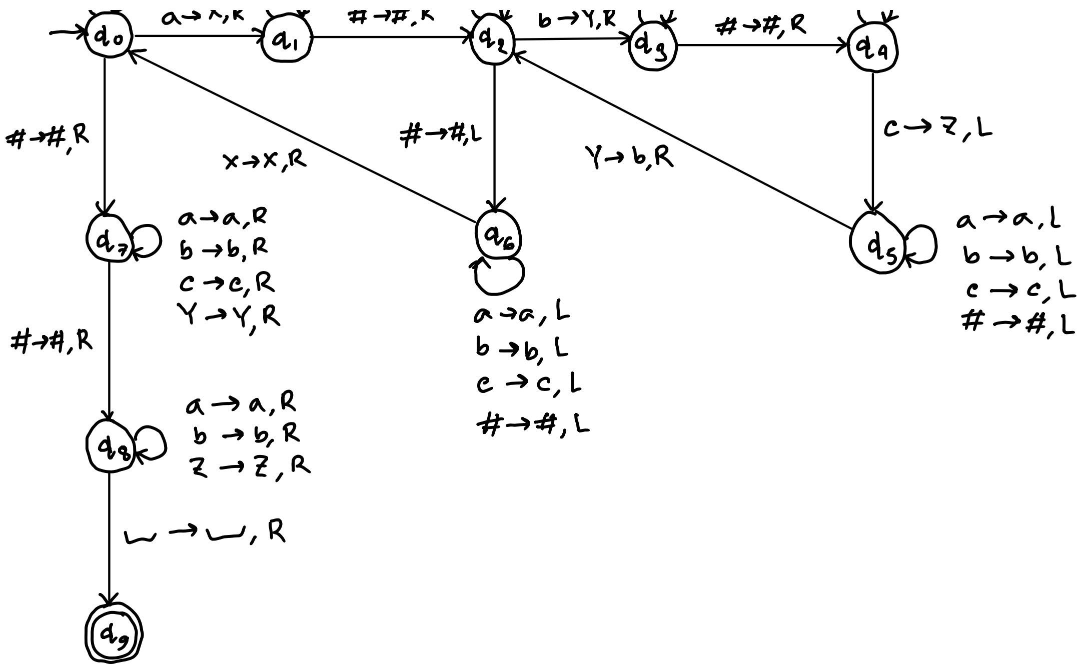
Problem 6 (Bonus): Designing Turing Machines (3 points)

Consider the following language. [Note: Count(w, x) is defined as the count of x in the string w. For example, Count(abba, b) = 2.]

$$L = \{w_1\#w_2\#w_3 : w_1, w_2, w_3 \in \{a, b, c\}^* \text{ and } Count(w_3, c) = Count(w_1, a) \times Count(w_2, b)\}$$

Give the state diagram of a turing machine that decides L.




a → X, R
→ #, R
b → Y, R
→ #, R
→ #, R
x → x, R
→ #, L
Y → b, R
c → Z, L
a → a, R
b → b, R
c → c, R
Y → Y, R
a → a, L
b → b, L
c → c, L
→ #, L
a → a, L
b → b, L
c → c, L
→ #, L
→ #, R
a → a, R
b → b, R
Z → Z, R
␣ → ␣, R