Automata Theory

Theory Assignment 2

Moida Praneeth Jain, 2022010193

Question 1

Pumping lemma for context free languages states that if L is a context-free language then $\exists p$ (pumping length) such that $\forall s \in L$, s can be divided into 5 substrings $s = uvxyz$ such that

- $uv^txy^tz \in L \ \forall t \geq 0$
- $|vy| > 0$
- $|vxy| \leq p$

(a)

Given: $L = \{a^i b^j c^k \mid 0 \leq i \leq j \leq k\}$

To Prove: L is not a context-free language

Proof: Assume L is a context-free language.

$\therefore$ L satisfies the pumping lemma for context-free languages. Let p be its pumping length.

Consider the string $s = a^p b^p c^p$. $|s| = 3p > p$, $\therefore s = uvxyz$

Define a function $f : L \to \mathbb{N}$ that takes in a string from the language and outputs the number of unique characters in it. For example, $f(\text{aabc}) = 3, f(\text{aaa}) = 1, f(\varepsilon) = 0$.

- Case 1: $f(v) \leq 1 \wedge f(y) \leq 1$

  The maximum number of unique symbols in both v and y combined is $\max(f(v) + f(y)) = 2$ in this case. Since we have 3 symbols (a, b, c), atleast one symbol never occurs in both v and y. Let this symbol be m.

  - Subcase 1: $m = a$

    We pump down with $t = 0$, i.e, $s' = uv^0xy^0z = uxz$.

    v and y contained either b or c (or both). Before pumping, all three characters had equal count p. After pumping, the count of either b or c (or both) is lower than the count of a.

    $\therefore s' \notin L$
  - Subcase 2: $m = b$

    If a is present in u or v, then we pump up with $t = 2$, i.e, $s' = uv^2xy^2z$. The number of b remains same as p while the number of a has increased.

    Otherweise, if c is present in u or v, then we pump down with $t = 0$, i.e, $s' = uv^0xy^0z$. The number of b remains same as p while the number of c has decreased.

    $\therefore s' \notin L$
  - Subcase 3: $m = c$

    We pump up with $t = 2$, i.e, $s' = uv^2xy^2z$. The number of c remains the same as p while the number of a or b (or both) increases.

    $\therefore s' \notin L$

  In all the subcases, the pumping lemma does not hold.

- Case 2: $f(v) > 1 \vee f(y) > 1$

  We pump up with $t = 2$, i.e, $s' = uv^2xy^2z$.

  WLOG assume $f(v) > 1$, i.e, $v = l_1l_2l_3$ in the correct order (l_3 may be ε).

  $v^2 = l_1l_2l_3l_1l_2l_3$. But, l_1 cannot appear after l_2. This is a contradiction.

  Pumping lemma does not hold.

In all possible cases, pumping lemma does not hold. This is a contradiction.

$\therefore L$ is not a context free language.

QED

(b)

Given: $L = \{ww \mid w \in \{0,1\}^*\}$

To Prove: L is not a context-free language

Proof: Assume L is a context-free language.

$\therefore$ L satisfies the pumping lemma for context-free languages. Let p be its pumping length.

Consider the string $s = 0^p1^p0^p1^p$. $|s| = 4p > p$, $\therefore s = uvxyz$

- Case 1: vxy occurs in the left half of the string, i.e, in the first w.

  We pump up with $t = 2$, i.e, $s' = uv^2xy^2z$. This pushes a 1 onto the first position of the second half, while the first position of the first half has a 0. Therefore, this string is not of the form ww.

  $\therefore s' \notin L$
- Case 2: vxy occurs in the right half of the string, i.e, in the second w.

  We pump up with $t = 2$, i.e, $s' = uv^2xy^2z$. This pushes a 0 onto the last position of the first half, while the last position of the second half has a 1. Therefore, this string is not of the form ww.

  $\therefore s' \notin L$
- Case 3: vxy contains the midpoint of the string

  We pump down with t = 0, i.e, $s' = uv^0xy^0z = uxz$.

  Clearly, $s' = 0^p1^{k_1}0^{k_2}1^p$. Note that the initial 0^p and final 1^p remain unaffected because $|vxy| \leq p$. For s' to belong in L, $k_1 = p \wedge k_2 = p$. This is not possible as the string has been pumped down and its length can no longer be $4p$. This is a contradiction.

  $s' \notin L$

In all possible cases, pumping lemma does not hold. This is a contradiction.

$\therefore L$ is not a context free language.

QED

(c)

Given: $L = \{a^{n!} \mid n \geq 0\}$

To Prove: L is not a context-free language

Proof: Assume L is a context-free language.

$\therefore$ L satisfies the pumping lemma for context-free languages. Let p be its pumping length.

Consider the string $s = a^{p!}$. $|s| = p! \geq p$, $\therefore s = uvxyz$

On pumping the string, we get $s_i = uv^ixy^iz$

Clearly, $|s_i| = p! + (i-1)|vy|$ (Note that $|vy| > 0$ according to pumping lemma).

$\forall i \geq 0$ $s_i \in L$

$\forall i \geq 0 |s_i|$ is a factorial

$\forall i \geq 0$ $p! + (i-1)|vy|$ is a factorial

This implies that there exists an arithmetic progression of factorials with common difference $|vy|$.

Since Γ (The gamma function) is convex for positive inputs, no infinite sequence with linear non zero slope can fit it. So, there can exist no infinite arithmetic progression of factorials with non zero common difference,

This is a contradiction. L does not satisfy the pumping lemma.

$\therefore$ L is not a context free language.

QED

Question 2

Given: $F(a, b) = a_0b_0a_1b_1...a_nb_n$

Yes, recursively enumerable languages are closed under this operation.

To Prove: RE languages are closed under F

Proof: Consider two arbitrary RE languages L_1 and L_2.

Let the turing machines that recognize them be M_1 and M_2 respectively.

Let $L = \{F(a, b) \mid a \in L_1, b \in L_2\}$

Now, we construct a turing machine M that recognizes L.

Let this turing machine have five tapes, with the original input on the first tape. Note that multi-tape turing machines are equivalent to regular turing machines as they can be concatenated with a special delimiter, and multiple virtual heads can be used to simulate them on a single tape.

M = " On input string w:

1. Measure the length of w and store it in it's state (Let the length be p).
2. Loop over i from 0 to p, both inclusive.
3. Copy over characters at even indices upto i (inclusive) to tape 2 and tape 4.
4. Copy over characters at odd indices upto i (inclusive) to tape 3 and tape 5.
5. Copy the remaining characters to the end of tape 4 as well as tape 5.
7. Perform steps 8 and 9 in BFS order. (One transition in each, then next iteration)
8. Simulate M_1 on tape 2 and M_2 on tape 5. If both accept, then accept w.
9. Simulate M_1 on tape 4 and M_2 on tape 3. If both accept, then accept w.
10. After done looping, Reject w.

"

The idea is to split the string into two parts, delimiting at the index i. The part to the left is the interleaved part, and it is copied over to its respective tapes based on parity. The part to the right is the remaining part, which is appended to both the strings in 4 and 5. Now, both the combinations are checked (length of b greater than equal to a or length of a greater than b). If either of these cases

is accepted, then the string is accepted. Therefore, $\forall c \in L$, M accepts c. For any string not in L, it will either be rejected or not halt in M_1 or M_2.

QED

Question 3

No restrictions
The computational power of a PDA that has a binary tree instead of the stack and allowing all four operations is equal to that of a turing machine. To prove this equivalence, we can model a binary tree as an array (assume 1-indexed)

For node present at index n:

$\text{left child} \rightarrow 2n, \text{right child} \rightarrow 2n+1, \text{parent} \rightarrow \left\lfloor \frac{n}{2} \right\rfloor$

Since the PDA can go to any child and parent node, it has access to the entire tree. ∴ Given an index i, the PDA can traverse to $i-1$ and $i+1$, thus simulating the left and right operations of a standard turing machine.

The read and write operations and the states of the PDA are same as that of the turing machine.

With this, we have simulated all the operations of a standard turing machines in a PDA with a binary tree. Therefore, such a PDA is as powerful as a turing machine.

Cannot traverse back to parent

TO DO
In this case

Question 4
Given: Binary string $w \in \{0,1\}^n$

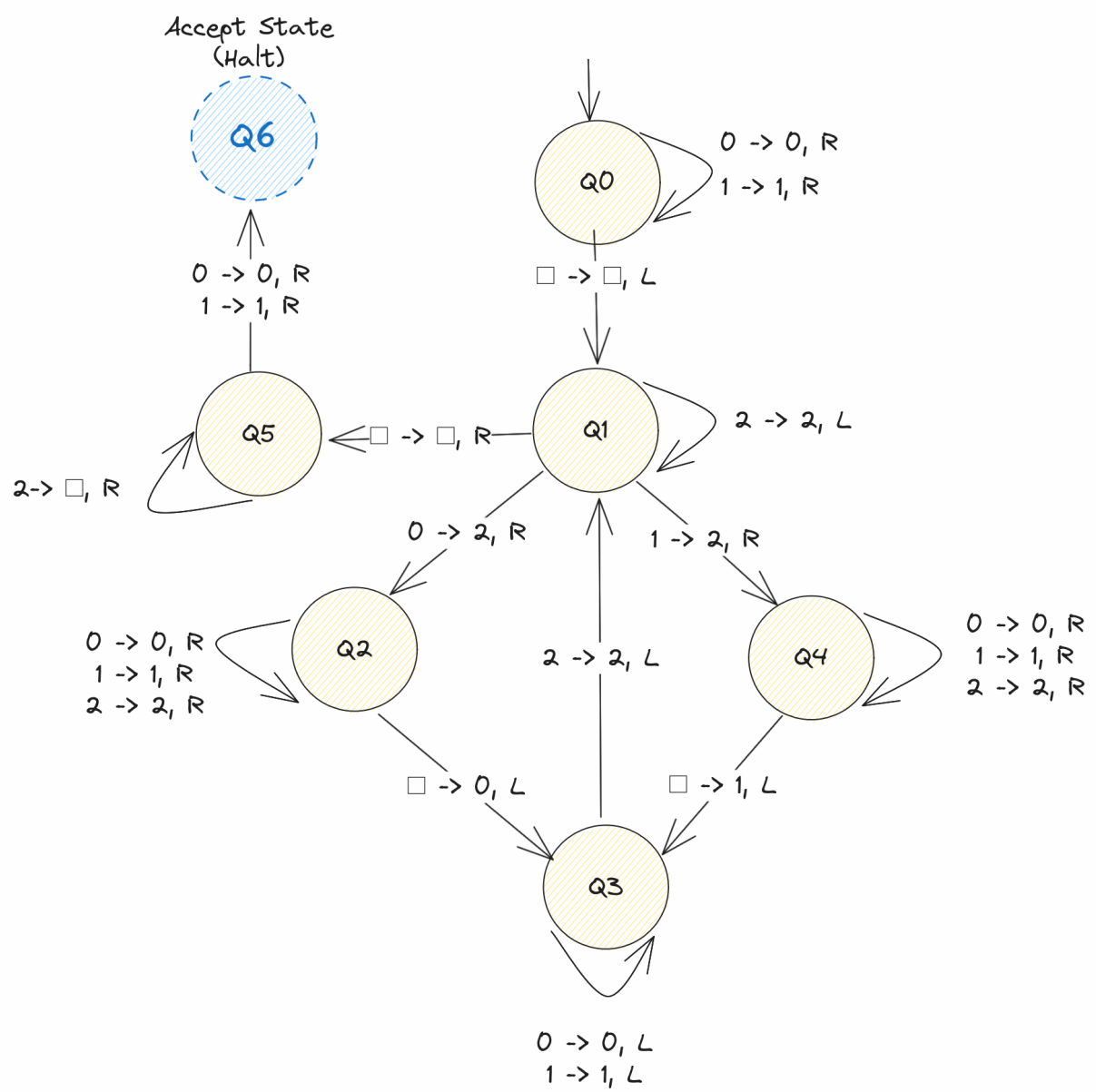
To construct: Turing machine that halts with w^R on its tape.

Construction:

M = " On input string w:
1. Traverse to the end of the string.
2. Start moving left until you encounter a 0 or 1. If none found, go to step 6.
3. Replace the symbol with a new symbol (say 2)
4. Start moving right until you encounter the first blank, and replace it with the respective 0 or 1.
5. Go back to step 2.
6. Start moving right until you find a 0 or 1. Replace every 2 with blank.
7. On reading a 0 or 1, halt.

"

We go the end of the string. Then, we move left, find a character, mark it as read (by writing a 2) and then copy it over to the rightmost end. We do this until nothing is left to be read, then replace all 2's with blanks.


Accept State
(Halt)
Q6
Q0
0 -> 0, R
1 -> 1, R
□ -> □, L
Q5
□ -> □, R
Q1
2 -> 2, L
2-> □, R
0 -> 2, R
1 -> 2, R
Q2
0 -> 0, R
1 -> 1, R
2 -> 2, R
2 -> 2, L
Q4
0 -> 0, R
1 -> 1, R
2 -> 2, R
□ -> 0, L
□ -> 1, L
Q3
0 -> 0, L
1 -> 1, L