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:

Assume that the tape is 0 indexed.

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. 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.

Assuming strings are of equal length

Let this turing machine have 3 times, with the original input on the first tape.

M = " On input string w:

1. Copy over even indices of w onto tape 2 until a blank is read.
2. Copy over odd indices of w onto tape 3 until a blank is read.
3. Run M_1 on tape 2 and M_2 on tape 3.
4. If both accept, then accept w.
5. Otherwise, reject w.

"

Assuming otherwise

Let this turing machine have five tapes, with the original input on the first 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. Increment i by 2 each time.
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:

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

Since the PDA can go to any child and parent node, it has access to the entire tree. $\therefore$ 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 machine in a PDA with a binary tree. Therefore, such a PDA is as powerful as a turing machine.

Cannot traverse back to parent

In this case, since we cannot traverse back to the parent, traversing to the left index and right index of the array representation of the binary tree is not possible.

To perform a general operation of a turing machine using this restricted PDA with a binary tree, we can copy over the entire conents of the tree (as an array) while marking the head virtually with a special symbol (say with a x^o where x is the value at that position). This is possible because we can directly read the character at any node of the binary trie (i.e, at any index of the array). In case of the write operation, the PDA will copy over the modified value instead of copying the original value and then modifying it, because modifying after copying may require traversing to the parent.

As we have simulated all the operations of a standard turing machine in this restricted PDA, such a PDA is as powerful as a turing machine.

Question 4

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

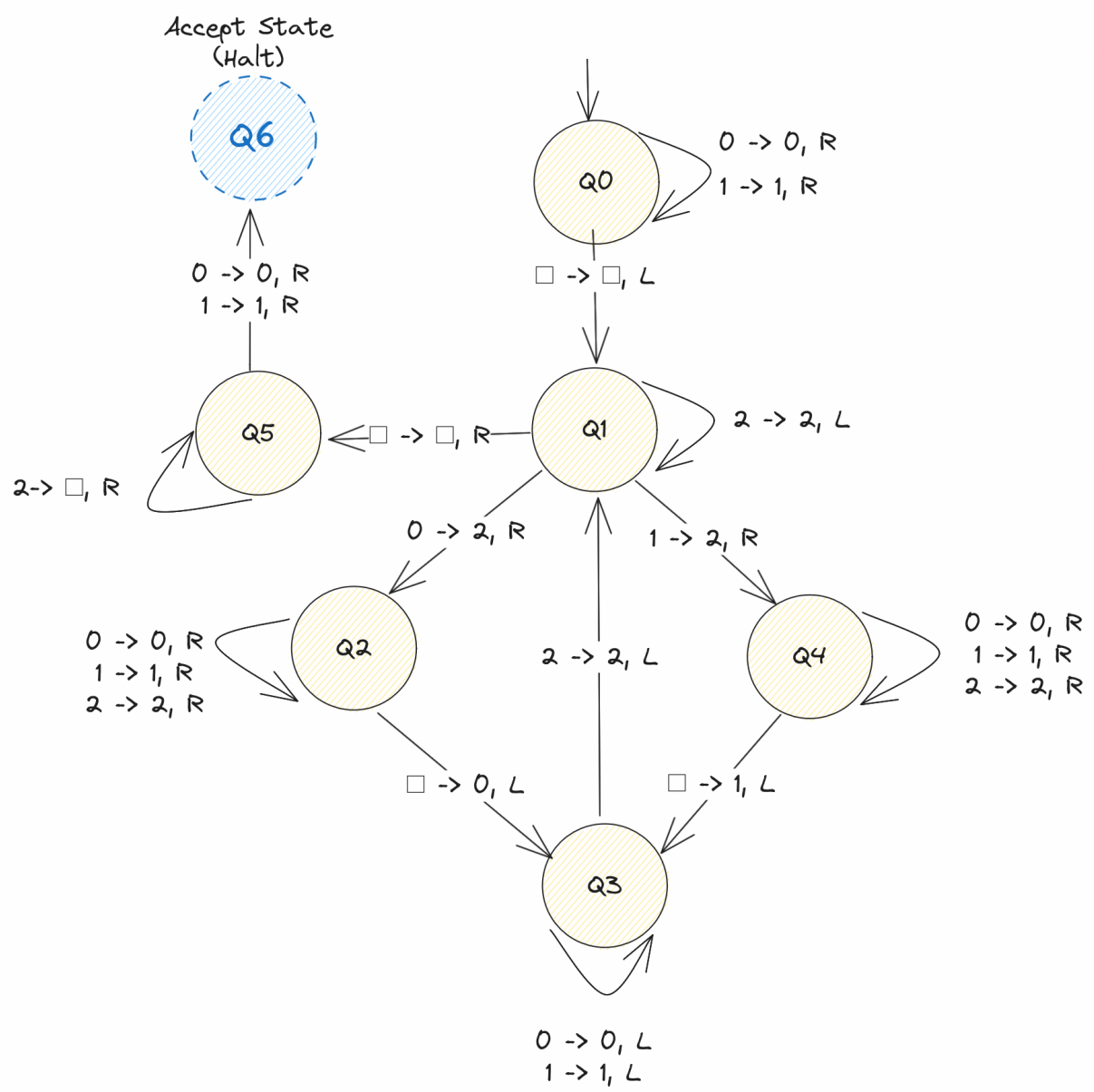
To construct: Turing machine that halts with w^R on its tape.

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.



Question 5

Given: S is the set of all **finite** binary strings, L is the set of all languages.

To Prove: S is countably infinite. L is uncountably infinite.

Proof:

Construct a set S' such that

$$S' = \{1w \mid w \in S\}$$

i.e, prepend a 1 to all the strings in S. Note that the cardinality of S and S' is equal.

Clearly, every string in S' does not contain any leading zeroes. Now, we will show a bijection between S' and $\mathbb{N}$.

Consider any string $s \in S'$. It has an equivalent decimal representation in $\mathbb{N}$. Therefore, there exists an injection from S' to $\mathbb{N}$.

Consider any $n \in \mathbb{N}'$. Since S contains all finite binary strings, it also contains the the binary string representing n without the leading 1. $\therefore S'$ contains the binary string representing n. Therefore, there exists an injection from N to S'.

Since there exists an injection from S' to $\mathbb{N}$ and from $\mathbb{N}$ to S', there exists a bijection between S' and $\mathbb{N}$. Hence, S' is countably infinite.

Since S and S' have the same cardinality, S is also countably infinite.

Let $P(S)$ denote the power set of S. Every element in the power set of S is a language.

$$\forall l \in P(S)\ l \in L$$

$\therefore$ The cardinality of L is atleast that of the power set of the natural numbers, which is uncountably infinite according to cantor's theorem.

$\therefore$ L is uncountable infinite.

QED

Question 6

To Prove: Recursively Enumerable Languages are closed under the star operation.

Proof: Let L be an arbitrary RE language.

$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 ... \quad (L^0 = \{\varepsilon\})$

$L^* = \cup_{i=0}^{\infty} L^i$

Since L is recursively enumerable, $\exists$ a turing machine M that recognizes L.

To show that L^* is also recursively enumerable, we construct a turing machine M^* that recognizes L^*. This machine will partition the input string into all possible partitions. Then, for each partition, we will simulate M on all substrings of the partition. If for any partition, all the substrings are accepted by M, then the input string will be accepted by M. Otherwise, the string will be rejected.

M^* = "On input string w:
1. if w is ε, accept w.

(Simulate each partition via dovetailing)
2. Generate next partition of w. If none left, reject w.
3. Let the current partition be $w_1, w_2, ...w_k$.
4. Simualte M on $w_i \forall i$. If M accepts all, then accept w.
5. Otherwise, go to step 2.

"

Now, we prove the correctness of this turing machine.

Consider any string $s \in L^*$. s is a concatenation of substrings of L. $\therefore$ it will be partitioned into those substrings in step 2, and each of these will be accepted in step 4, hence resulting in s being accepted.

Consider any string $s \notin L^*$. s is not a concatention of substrings of L. $\therefore$ there exists no partition for which all substrings will be accepted in step 4. So, after looping through all partitions, s will be either be rejected or not halt.

$\therefore$ M^* recognizes L^*. L^* is a recursively enumerable language.

$\therefore$ Recursively Enumerable Languages are closed under the star operation.

QED

Question 7

Given: Lilliputian Turing Machine (LTM) with transition function

$$\delta : \{Q \times \Gamma^k\} \to \{Q \times \Gamma^k \times \mathbb{Z}_0^+\}$$

$$\delta(q_i, (a_1, a_2, a_3, ...a_k)) = \big(q_j, (b_1, b_2, ...b_k), r\big)$$

To Prove: The Lilliputian Turing Machine is as powerful as a standard Turing Machine in terms of the languages it can recognize.

Proof:

First, we will simulate a standard TM using an LTM.

Consider a general transition in a standard TM:

$$\delta(q_i, a) = \big(q_j, b, L \text{ or } R\big)$$

1. If the turing machine moves left (L)

$$\delta(q_i, (a)) = \big(q_j, (b), -1\big)$$

2. If the turing machine moves right (R)

$$\delta(q_i, (a)) = \big(q_j, (b), +1\big)$$

Hence, a standard TM can be simulated by an LTM.

Now, we will simulate an LTM using a standard TM. Since a standard turing machine is equivalent two a turing machine with k tapes, as we can simply add a delimeter at the end of the first tape and then concatenate the following tape onto it (delimiter separated) and maintain k virtual heads, we will use a k tape turing machine to simulate the LTM.

Consider a general transition in an LTM

$$\delta(q_i, (a_1, a_2, a_3, ...a_k)) = \big(q_j, (b_1, b_2, ...b_k), r\big)$$

1. Copy the characters at the next k indices (including the current one from the head) onto tape 2.
2. Start from the first index of tape 2. Iterate through this tape from left to right and check if character at index i matches a_i. If does not match, halt. If all are matched, stop at the blank at the end of the tape.
3. Now, in tape 1, if head of tape 2 is not on a blank, do nothing.
4. If head of tape 2 is on a blank, then perform the following transitions

   $$\delta\big(q_i', a_j\big) = \big(q_j', b_j, R\big) \quad \forall j \in \{1, 2, 3, ...k\}$$

   Now we move the head back to where it was by performing the following transition

   $$\delta(q_i', x) = \big(q_j', x, R\big)$$

   k times.
5. Now, we need to simulate the movement to address $p + r$.

   If r is positive, perform the below operation r times

$$\delta(q_i', x) = \left(q_j', x, R\right) \quad \forall x \in \Gamma$$

If r is negative, perform the below operation r times

$$\delta(q_i', x) = \left(q_j', x, L\right) \quad \forall x \in \Gamma$$

If r is zero, do nothing.

Hence, an LTM can be simulated by a k-tape turing machine, which is equivalent to a standard turing machine. $\therefore$ An LTM can be simulated by a standard turing machine.

Since an LTM can be simulated by a standard TM and a standard TM can be simulated by an LTM, these are equivalent to each other. Hence, the Lilliputian Turing Machine is as powerful as a standard Turing Machine in terms of the languages it can recognize.

QED