CS 39: Theory of Computation **Winter '23**

PSET 5 — 02/20/2023

Prof. Chakrabarti ***Student: Amittai Siavava***

Credit Statement

All the work on this PSET is my own. I usually discuss ideas with Paul Shin, but this weekend we were both busy and couldn't meet to discuss ideas. We only chatted about all the different things we had tried out on question 2.2. (and none of them had worked yet).

I also referred to the following books:

(a) **Introduction to the Theory of Computation** by **Michael Sipser**.

(b) **A Mathematical Introduction to Logic** by **Herbert Enderton**.

Problem 1.

For each of the following languages, say whether or not it is a CFL and prove your answer, either by designing an appropriate CFG or PDA or by using closure properties and/or the pumping lemma. If designing a CFG/PDA, please explain your construction in brief so the grader can understand your design.

Lemma 1.1. *(Pumping Lemma for CFLs)*

If a language is a CFL, then the language has a pumping length p such that any string s in the language that has length at least *p can be written as $s = uvwxy$ where*

(i) $|vwx| \leq p$.

(ii) $|vx| > 0$.

(iii) $uv^k wx^k y \in L$ *for all* $k \geq 0$.

(a) $L = \{a^n b^n c^m : n \leq m \leq 2n\}$

The language is not a CFL.

Suppose it is, then take p to be the pumping length and $s = a^p b^p c^{2p}$, then s is a string in the language.

Since $|vwx| \leq p$, vwx must not span three distinct letters. In particular, it must be of one of the following forms:

(i) $vwx = a^i b^j$ for some $i, j \geq 0, 1 < i + j \leq p$.

Since $|vx| > 0$, at least one of v and x must be nonempty, so vx contains some number of a's and some number of b's (but no c's). There are two scenarios here.

- If vx contains the same number of a's as the number of b's, then pumping down the string does not break the equality of a's and b's, but ir reduces the number of a's and the number of b's by some non-zero l_1 and l_2, respectively. So $a^{p-l_1} b^{p-l_2} c^2 p \in L$, suggesting that $2p \leq 2p-(l_1+l_2)$, which can only happen since $l_1 + l_2 > 0$ (following from $|vx| > 0$).
- If vx contains more a's than b's, then pumping down the string breaks the equality of a's and b's the resulting string cannot be in L. The same scenario applies for when vx contains more b's than a's.

(ii) $vwx = b^i c^j$ for some $i, j \geq 0, 1 < i + j \leq p$.

If $i > 0$ (so vwx starts with a sequence of b's).

- If v is nonempty, then v contains a number of b's so pumping down the string reduces the number of b's but does not reduce the number of a's, breaking the equality of the number of a's and the number of b's in the string. The resulting string must not be in L, contradicting the pumping lemma.
- If v is empty, then x must be nonempty (since $|vx| > 0$). In particular, x must contain a number of c's. It may also contain a number of b's. If x contains only c's (say, $x = c^l$ for some $0 < l \leq j$), pumping up the string tells us that $a^p b^p c^{2p+l}$ is in L, but that implies that $2p + l \leq 2p$, so $l = 0$, but if $l = |x| = 0$ and $|v| = 0$ then we would have that $|vx| = 0$, contradicting the pumping lemma. If x contains both c's and b's, the pumping up will increase the number of b's in the string, breaking the equality between the number of b's and the number of a's. So the resulting string must not be in L, contradicting the pumping lemma.

If $i = 0$, then $j > 0$ and, since $|vx| > 0$, $vx = c^l$ for some $0 < l < j$. Pumping up the string increases the number of c's in the string by some amount l. For the string to still be a valid member of L, then $2p + l \leq 2p$, implying that $l = 0$ so $|vx| = 0$, contradicting the pumping lemma.

Therefore, L must not be context-free since the pumping lemma is not satisfied for some strings in L of length greater than p.

(b) $\left\{a^n b^{n^2} : n \geq 0\right\}$

The language is not a CFL.

Suppose it is, then, following the pumping lemma (1.1), take p to be the pumping length and $s = a^p b^{p^2}$ to be a string in the language.

Write $s = uvwxy$, the pumping lemma tells us that $|vwx| \leq 0$, $|vx| > 0$, and $uv^k wx^k y \in L$ for all $k \geq 0$.

There are two scenarios;

(i) $vwx = a^i$ for some $0 < i \leq p$. Since $|vx| > 0$, $vx\ vx = a^l$ for some $0 < l < i$. Pumping down tells us that $a^{p-l}b^{p^2} \in L$, so $p^1 = (p-l)^2$, which implies that $l = 0$, but that contradicts the pumping lemma's condition that $|vx| > 0$.

(ii) $vwx = a^i b^j$ for some $0 < i$, $0 < j$, and $i + j \leq p$. Since $|vx| > 0$, $vx = a^{l_1} b^{l_2}$ with the condition that $0 < l_1 + l_2 < i + j$

- If $l_1 = 0$, then pumping up the string increases the number of b's but does not change the number of a's, breaking the condition that the number of b's equal to the square of the number of a's.
- If $l_1 > 0$ and $l_2 = 0$, then pumping up the string increases the number of a's but does not change the number of b's, breaking the condition that the number of b's equal to the square of the number of a's.
- If $l_1 > 0$ and $l_2 > 0$, then pumping up the string tells us that $a^{p+l_1} b^{p^2+l_2} \in L$, so $p^2 + l_2 = (p + l_1)^2$.

  By expanding the equation, we see that:

$$
\begin{aligned}
p^2 + l_2 &= (p + l_1)^2 \\
&= p^2 + 2pl_1 + l_1^2 \\
\therefore l_2 &= 2pl_1 + l_1^2 \\
\therefore l_2 &> p \qquad \text{since } l_1 \text{ (string length) must be positive!} \\
\therefore l_1 + l_2 &= |vx| > p \qquad \text{(contradiction)}
\end{aligned}
$$

Therefore, L must not be context-free since the pumping lemma is not satisfied for some strings in L of length greater than p.

Problem 2.

Do the same for each of the following languages:

(a) $L = \{b_i \# b_{i+1} : i \geq 1\}$, where b is the binary representation of i with no leading 0's.

The language is not context-free. Suppose it was and p as the pumping length. Take $s = 10^p 1^p \# 10^{p-1} 10^p$. Since $10^p 1^p + 1 = 10^{p-1} 10^p$ for all p, the string is guaranteed to be in the language.

Write $s = uvwxy$ where, per the pumping lemma (1.1), $|vwx| \leq p$, $|vx| > 0$, and $uv^k wx^k y \in L$ for all $k \geq 0$.

There are three possibilities:

(i) If vwx is a substring to the left of the $\#$ symbol in s, then, since $|vx| > 0$, pumping down the string changes the left part of the string (previously equivalent to b_i) by removing *at least* one symbol. Therefore, the resulting string cannot be equivalent to b_i. Since the right side of the string (b_{i+1}) remains unaltered, the new string is of the form $s' = b_j \# b_{i+1}$ for some $j \neq i$, so it is not in L, contradicting the pumping lemma.

(ii) Similarly, if vwx is a substring to the right of the $\#$ symbol, then pumping down the string changes the right part of the string (previously equivalent to b_{i+1}) by removing *at least* one symbol. Therefore, the resulting string cannot be equivalent to b_{i+1}. Since the left side of the string (b_i) remains unaltered, the new string is of the form $s' = b_i \# b_j$ for some $j \neq i + 1$, so it is not in L, contradicting the pumping lemma.

(iii) If vwx is a substring in the middle of the string, then:

- If $\#$ is contained in $|vx|$, pumping down results in a string without a $\#$ symbol, so it is not in L.
- If $\#$ is contained in $|w|$, then:
  - If the 1 immediately after the $\#$ is not in w, then it must be in x so pumping down removes it (but does not remove all the other 0's) resulting in a string to the left that has leading 0's, hence cannot be equivalent to b_i.

- If the 1 immediately after the # in the original string is in w, then vx reads 1^a0^b for some $0 < a + b < p - 2$. Pumping down the string suggests that $10^p1^{p-a}\#10^{p-1-b}10^p$ is in L, so

$$10^p1^{p-a} + 1 = 10^{p-1-b}10^p$$

$$\therefore 10^{p-1}10^{p-a} = 10^{p-1-b}10^p$$

$$\therefore p - a = p \implies a = 0$$

$$\therefore p - 1 - b = p - 1 \implies b = 0$$

  But if $|v| = a = 0$ and $|x| = b = 0$, then $|vx| = 0$, which contradicts the pumping lemma.

Therefore, L must not be context-free since the pumping lemma is not satisfied for some strings in L of length greater than p.

(b) $L = (a \cup b)^* - \{(a^nb^n)^n : n \geq 1\}$

I got stuck on this problem.

Problem 3.

Formally describe a two-tape deterministic TM that decides the language $w \in \{0,1\}^* : w = w^R$. Provide a diagram, plus explanations and comments, so that your grader can understand how your TM works. You may assume that in a particular move one (or both) of the heads is allowed to remain stationary. Thus, the transition function for such a TM would look like

$$\delta : Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{L, R, S\}^2 .$$

Indicate $\delta(q, a, b) = (r, c, d, L, S)$ by drawing an arrow from q to r and labeling it "$(a, b) \to (c, d), (L, S)$." Appreciate the ease of programming with two tapes for this language.

Strategy: Let tape A and tape B be the two tapes in the TM. Note that at the start, the input string is on tape A (plus infinite blanks on the right end), and tape B is blank.

(i) Shift the input string on tape A to the right by 1, and insert a $\#$ symbol at the left end of tape A to mark the beginning. We also insert a $\#$ at the left-most position on tape B to mark the beginning.

(ii) Move the head on tape A to the right end of the string and the head on tape B to the left second position (just after the $\#$). Traverse backwards on tape A and forwards on tape B, copying the string (in reverse) to tape B.

(iii) Move both heads to beginning of each tape. Traverse forwards on tape A and tape B, comparing the corresponding elements of tape A and tape B. If they are not equal, then $w \neq w^R$. Reject the input string. If they are equal, then $w = w^R$. Accept the input string.

Note: In the turing machine diagram, when we denote (x, y) to as readings from the two tapes such that $x, y \in \{0,1\}$ and $x \neq y$ (otherwise, we denote the reading as (x, x) to specify equality). We denote $\sqcup$ when the reading from the tape is blank.

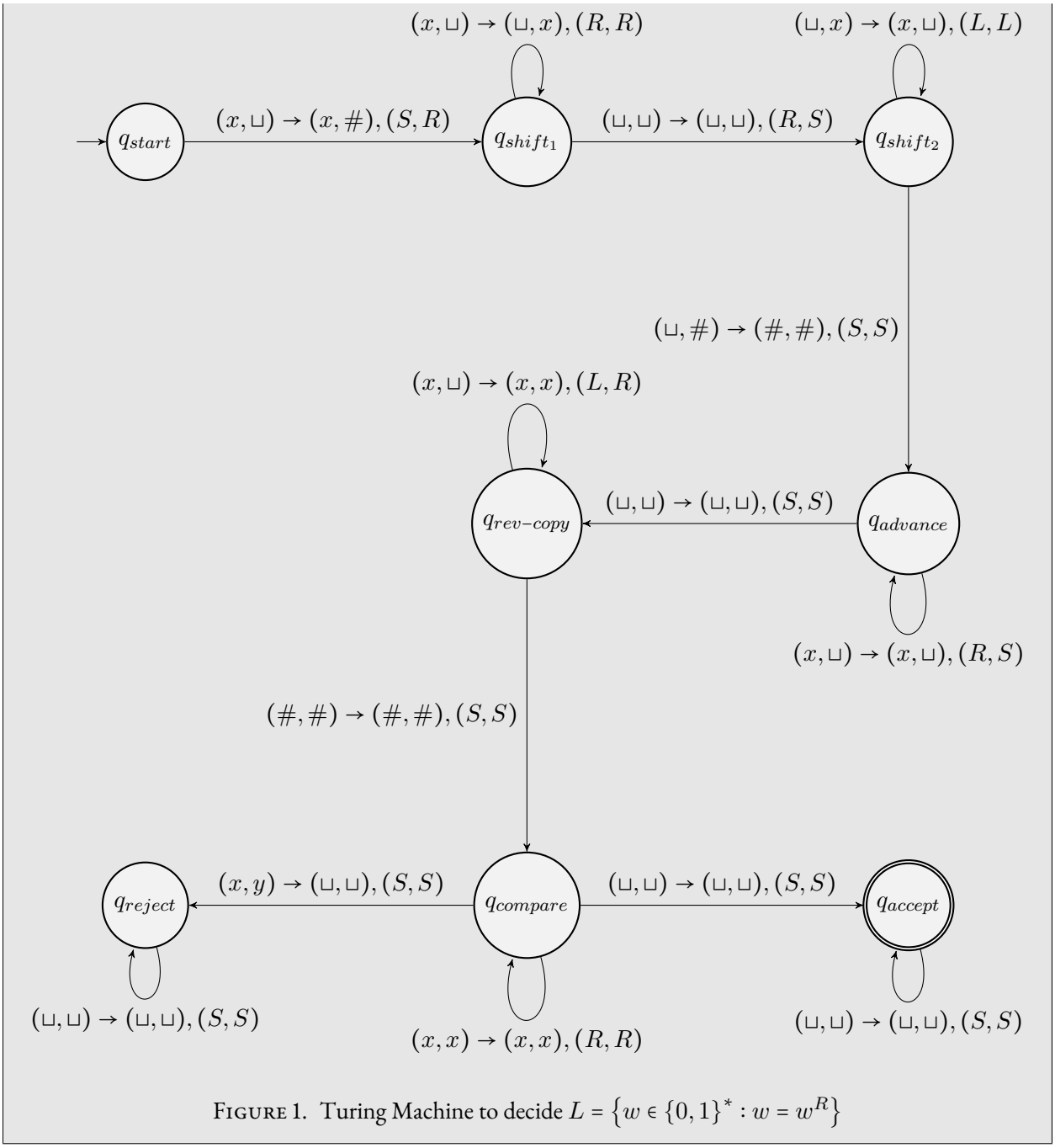

Figure 1. Turing Machine to decide $L = \left\{w \in \{0,1\}^* : w = w^R\right\}$

Problem 4.

Formally describe a two-tape NDTM for the language $\{ww : w \in \{0,1\}^*\}$. Again, provide a diagram, plus explanations and comments.

Appreciate the ease of programming resulting from nondeterminism and the availability of two tapes.

Strategy: Let tape A and tape B be the two tapes in the TM. Note that at the start, the input string is on tape A (plus infinite blanks on the right end), and tape B is blank.

(i) Add a left-end marker ($\#$) to tape B and move the head on tape B one step right.

(ii) Traverse tape A from left to right, copying the contents of tape A onto tape B.

(iii) Non-deterministically, stop copying the contents of tape A onto tape B, move the head on tape B to the left-most end (but maintain the head on tape A at its current position) and compare the remaining contents on tape A with those copied onto tape B.

(iv) Accept if there is such a computational path that copies some (possibly empty) part of the input string onto tape B, compares it with the remaining part of the string on tape A, and successfully reaches the end of the string.

Note: In the turing machine diagram, when we denote (x, y) to as readings from the two tapes such that $x, y \in \{0,1\}$ and $x \neq y$ (otherwise, we denote the reading as (x, x) to specify equality). We denote $\sqcup$ when the reading from the tape is blank.

Note: similar to NFAs, I allowed the NDTM to crash on unexpected inputs rather than explicitly handling them / rejecting.

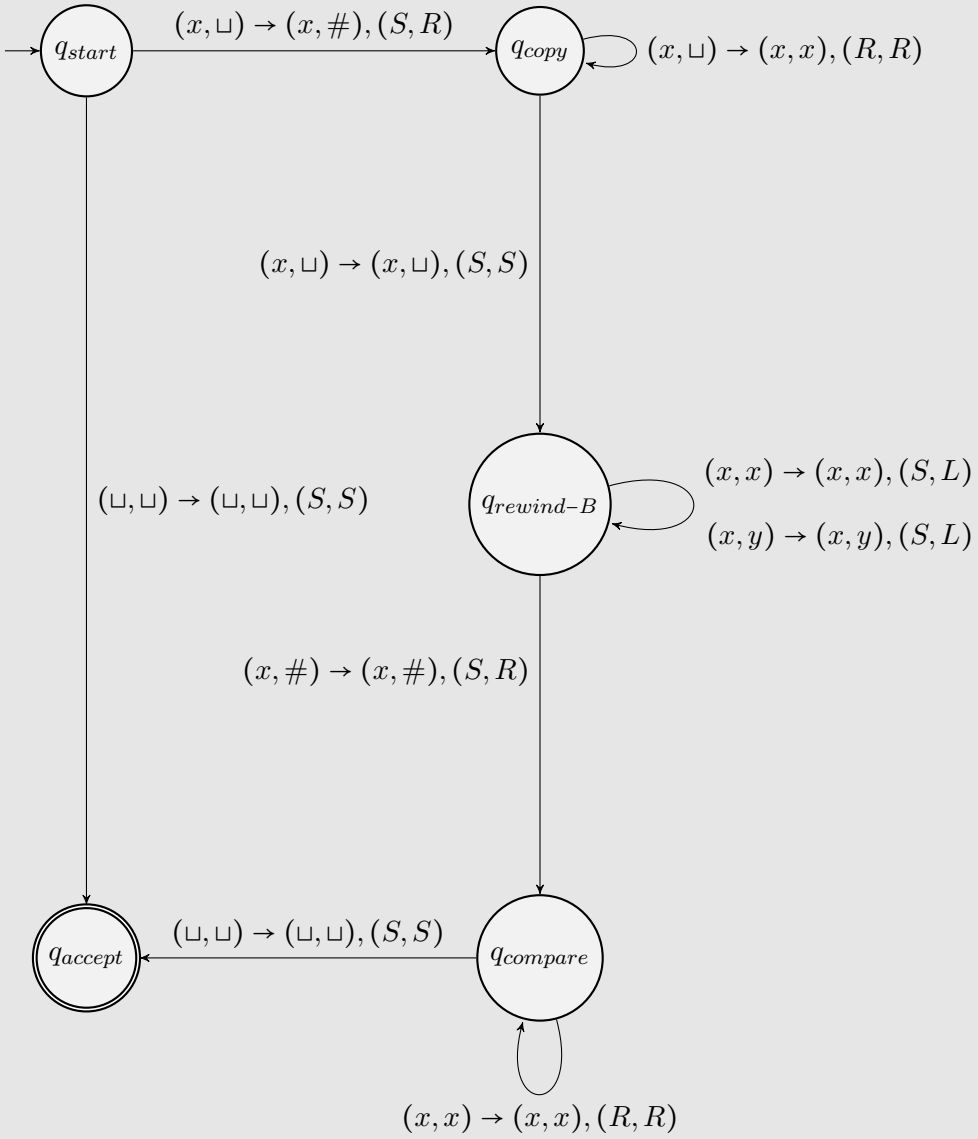

Figure 2. Turing Machine to decide $L = \{ww : w \in \{0, 1\}^*\}$

Problem 5.

Show that a k-tape TM M can be simulated by a single-tape TM M' in such a way that a computation that takes time t (i.e., t steps of one configuration yielding another) on M takes time $O(t^2)$ on M'. The big-O notation may hide a constant that depends on k. You may assume that $t \geq |x|$, where x is the input to M.

Let M be a k-tape TM over some alphabet Σ.

A single-tape TM M' can simulate M as follows:

(i) M' copies the contains of all the tapes of M onto its single tape, with a separator between the contents of each tape. For instance, if the contents of the k tapes of M are

$$t_1 = \sigma_{(1,1)}, \sigma_{(1,2)} \dots \sigma_{(1,n_1)}$$

$$t_2 = \sigma_{(2,1)}, \sigma_{(2,2)} \dots \sigma_{(2,n_2)}$$

$$\vdots$$

$$t_k = \sigma_{(k,1)}, \sigma_{(k,2)}, \dots, \sigma_{(k,n_k)}$$

with each $\sigma_{(i,j)} \in \Sigma$, then the corresponding configuration of M' is

$$\#, \sigma_{(1,1)}, \sigma_{(1,2)} \dots \sigma_{(1,n)}, \#, \sigma_{(2,1)}, \sigma_{(2,2)} \dots \sigma_{(2,n)}, \#, \dots, \#, \sigma_{(k,1)}, \sigma_{(k,2)}, \dots, \sigma_{(k,n_k)}\#$$

where $\#$ is a special symbol that is not in the alphabet of M.

(ii) M' tracks multiple head positions (within each sub-tape) for each of the k tapes of M. For instance, in the above example, if the heads of M are at the beginning of each tape, then the configuration n M' (using H_i to denote the head position on tape i) would be:

$$\#, H_1, \sigma_{(1,1)}, \sigma_{(1,2)} \dots \sigma_{(1,n)}, \#, H_2, \sigma_{(2,1)}, \sigma_{(2,2)} \dots \sigma_{(2,n)}, \#, \dots, \#, H_k, \sigma_{(k,1)}, \sigma_{(k,2)}, \dots, \sigma_{(k,n_k)}\#$$

(iii) On each transition, M reads some k-tuple of inputs from the k tapes. transitions to some new state q, and write some k-tuple of updates to the k tapes. in turn, M' simulates a sequence of transitions that updates each of the k positions representing the head k head positions on its single tape.

For a computation that runs in t steps on M;

- M' copies the contents of the k tapes of M onto its single tape and sets the head positions to the beginning of each sub-tape in $k \cdot |x|$ steps.
- For each of the subsequent $k - 1$ steps of M, M' simulates a computation that updates each sub-tape. In the worst case, this requires traversing traversing the entire tape of M', which which takes $k \cdot |x|$ steps. This is repeated $t - 1$ times for a total of $(t - 1) \cdot k \cdot |x|$ steps.
- In total, M' takes $k \cdot |x| + (t - 1) \cdot (k \cdot |x|) = t \cdot k \cdot |x|$ steps to simulate the computation of M. Assuming $t \geq |x|$, the simulation runs in time $t' \leq kt^2$ steps, which is $O(t^2)$.