

# Assignment 4

COL 352

Introduction to Automata & Theory of Computation

## Problem 1

For any arbitrary TM  $M$ , show that it can be simulated by another with tape alphabet restricted to  $\{0, 1, B\}$ . Discuss if it is possible to restrict to only two tape symbols.

**Solution:** The problem reduces to determine an encoding and transition of any Turing machine to a new environment, specifically representing alphabets of the new Turing machine over the original alphabet.

One of the possible encoding is a binary encoding of alphabets. Clearly this encoding function is one-one. Let  $k$  be the size of the original alphabet. Then the total bits required to encode it are no more than  $b = \log(k)$ .

The symbol  $B$  can be used to separate the encoded alphabets. Note that now  $BB$  will denote the end of the input (and not simply  $B$ ). The state space will have to expand to recognize the original alphabet.

For a transition in the new TM,  $b$  bits are read before taking any action. Once  $B$  is encountered, the encoded symbol is interpreted. If action is to write on the tape then move the tape  $k$  symbols left and write the encoded symbol in reverse direction. Moving left/right is just traversing the tape  $k$  times.

It is also possible to do the same with just two alphabets by using the an encoding like the following

$$1 \rightarrow 11 \quad 0 \rightarrow 1B \quad B \rightarrow BB$$

## Problem 2

Prove that a TM can recognize any Context Free Language. Moreover prove that it is strictly more powerful by showing that the language  $\{ww \mid w \in (0+1)^*\}$  is recursive. Recall that this language is not a CFL

**Solution :** Since the non-deterministic Turing machine is equivalent to a standard Turing machine. So to prove the result we will prove that a non-deterministic push-down automaton can be simulated by a non-deterministic TM with two tapes - one for input (read-only), another to simulate the stack of the PDA.

Let  $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z, F \rangle$  be any non-deterministic PDA. The language accepted by  $P$  is the set  $L(P) = \{ w \in \Sigma^* \mid (q_0, w, Z) \vdash_P^* (q_f, \epsilon, s), q_f \in F, s \in \Gamma^* \}$ , where  $\epsilon$  is empty string and  $(q, u, v)$  means  $q$  is the state of the control unit,  $u$  is the unread part of the input string,  $v$  are the stack contents.

Let  $w \in L(P)$  be a string, then  $(q_0, w, Z) \vdash_P^* (q_f, \epsilon, u)$ ,  $q_f \in F$ ,  $u \in \Gamma^*$ . If  $(q_0, w, Z) \vdash_P^* (q_f, \epsilon, u)$  denotes the sequence of moves  $(q_0, w, Z) \vdash_P (p_1, w_1, s_1) \vdash_P (p_2, w_2, s_2) \vdash_P \dots \vdash_P (p_i, w_i, s_i) \vdash_P (p_{i+1}, w_{i+1}, s_{i+1}) \vdash_P \dots \vdash_P (q_f, \epsilon, u)$ , then we will explain how the general move  $(p_i, w_i, s_i) \vdash_P (p_{i+1}, w_{i+1}, s_{i+1})$  is handled by the Turing machine  $M$  described above.

Let the PDA transition  $(p_i, w_i, s_i) \vdash_P (p_{i+1}, w_{i+1}, s_{i+1})$  be  $\delta(p_i, a, x) = (p_{i+1}, y)$  where  $\delta$  is the transition function and stack symbol  $x$  is replaced by string  $y \in \Gamma^*$  on reading input symbol  $a$ . If  $y$  is empty string, then the corresponding transition of  $M$  is  $\delta'(p_i, [a, x]) = (p_{i+1}, [a, B], [R, L])$ , where  $\delta'$  is the transition function of  $M$ , the head on the input tape moves towards right after replacing symbol  $a$  by the same symbol  $a$ , the head on the other tape moves towards left after replacing symbol  $x$  by blank symbol  $B$ .

If  $y = y_1 y_2 \dots y_k$  is a non-empty string where  $y_1$  is topmost symbol of stack after the transition, then  $M$  makes the following transitions - a single transition  $\delta'_1(p_i, a) = (p_{i+1}, a, R)$  on the input tape (1 in  $\delta'_1$  means that this is transition on first tape), and following sequence of transitions on the other tape

1.  $\delta'_2(p_i, x) = (q_1, y_k, R)$
2.  $\delta'_2(q_1, B) = (q_2, B, L)$
3.  $\delta'_2(q_2, y_t) = (m_t, y_t, R)$  for  $t = k, k-1, \dots, 2$
4.  $\delta'_2(m_t, B) = (q_1, y_{t-1}, R)$  for  $t = k, k-1, \dots, 2$
5.  $\delta'_2(q_2, y_1) = (q_3, y_1, R)$
6.  $\delta'_2(q_3, B) = (p_{i+1}, B, L)$

Now, to demonstrate that a TM is strictly more powerful than a PDA

Claim : Language  $\{ ww \mid w \in (0+1)^* \}$  is recursive.

Proof : This language is accepted by the Turing machine  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle$ , where

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}$ , with the initial state  $q_0$  and the set of final states  $F = \{q_9\}$ .

$\Sigma = \{0, 1\}$ , the input alphabet.  $\Gamma = \{0, 1, x, y, B\}$ , the tape alphabet.  $B$  is the 'blank' symbol.

The transition function  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ , where  $L$  denotes left move and  $R$  denotes right move, is defined as follows -

$$\begin{aligned}
\delta(q_0, 0) &= (q_1, x, R) & \delta(q_1, 0) &= (q_1, 0, R) \\
\delta(q_0, 1) &= (q_1, y, R) & \delta(q_1, 1) &= (q_1, 1, R) \\
\delta(q_0, x) &= (q_4, x, L) & \delta(q_1, B) &= (q_2, B, L) \\
\delta(q_0, y) &= (q_4, y, L) & \delta(q_1, x) &= (q_2, x, L) \\
& & \delta(q_1, y) &= (q_2, y, L)
\end{aligned}$$

The state  $q_0$  converts 0 or 1 to  $x$  or  $y$  respectively, one at a time, from the left side. The head is taken by state  $q_1$  to the right until it finds some  $x$  or  $y$  or  $B$  and then control is given to state  $q_2$ . The state  $q_2$  converts a single bit 0 or 1 to  $x$  or  $y$  respectively, one at a time, from the right side. The head is taken by state  $q_3$  to the left until it finds some  $x$  or  $y$  and then control is given to state  $q_1$ . The process repeats until all the 0's and 1's are converted to  $x$  and  $y$ .

Thus, 0 and 1 are converted to  $x$  and  $y$  from both the ends simultaneously towards the middle of the string. If the given string is of even length then in the end,  $q_0$  finds no 0 or 1 for conversion and hence control is given to state  $q_4$  at the midpoint of the string. Then in state  $q_4$ , the machine  $M$  converts all  $x$ 's and  $y$ 's to 0 and 1 respectively in the left half string and then control is given to state  $q_5$  at the beginning of the string.

$$\begin{aligned}
\delta(q_2, 0) &= (q_3, x, L) & \delta(q_4, x) &= (q_4, 0, L) \\
\delta(q_2, 1) &= (q_3, y, L) & \delta(q_4, y) &= (q_4, 1, L) \\
\delta(q_3, 0) &= (q_3, 0, L) & \delta(q_4, B) &= (q_5, B, R) \\
\delta(q_3, 1) &= (q_3, 1, L) & \delta(q_6, 0) &= (q_6, 0, R) \\
\delta(q_3, x) &= (q_0, x, R) & \delta(q_6, 1) &= (q_6, 1, R) \\
\delta(q_3, y) &= (q_0, y, R) & \delta(q_6, B) &= (q_6, B, R) \\
\delta(q_5, 0) &= (q_6, x, R) & \delta(q_6, x) &= (q_8, B, L) \\
\delta(q_5, 1) &= (q_7, y, R) & & \\
\delta(q_5, B) &= (q_9, B, L) & &
\end{aligned}$$

The state  $q_5$  converts 0, 1 to  $x, y$  respectively and gives control to states  $q_6, q_7$  respectively which then go right until they find some  $x$  or  $y$ . If  $q_6$  finds  $x$  then it converts it to  $B$  and gives control to state  $q_8$  which then moves left until it finds some  $x$  or  $y$  and then gives control to state  $q_5$ . If  $q_7$  finds  $y$  then it converts it to  $B$  and gives control to state  $q_8$  which then moves left until it finds some  $x$  or  $y$  and then gives control to state  $q_5$ . The process repeats until the first half has been again converted to  $x, y$  form and second half is converted to  $B$ 's.

Note that the second half bits are converted to  $B$ , if they match the corresponding bit in first half.

If this process completes successfully then in the end, state  $q_5$  finds no 0 or 1 for conversion and hence the string is of the form  $ww$ . Therefore state  $q_5$  gives the control to final state  $q_9$ .

$$\delta(q_7, 0) = (q_7, 0, R)$$

$$\delta(q_8, 0) = (q_8, 0, L)$$

$$\delta(q_7, 1) = (q_7, 1, R)$$

$$\delta(q_8, 1) = (q_8, 1, L)$$

$$\delta(q_7, B) = (q_7, B, R)$$

$$\delta(q_8, B) = (q_8, B, L)$$

$$\delta(q_7, y) = (q_8, B, L)$$

$$\delta(q_8, x) = (q_5, x, R)$$

$$\delta(q_8, y) = (q_5, y, R)$$

### Problem 3

Are the following problems decidable ?

- (a) Given a TM  $M$ , whether there is a  $w$  such that  $M$  enters each of its states during the computation on  $w$ .
- (b) Given a TM  $M$  and an input  $w$ , does the head ever visit the  $B^{\text{th}}$  square for a given integer  $B$ .

**Solution :**

a)

b)