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

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

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 $P = < Q, \sum, \Gamma, \delta, q_0, Z, F >$ be any non-deterministic PDA. The language accepted by P is the set $L(P) = \{ w \in \sum^* \mid (q_0,\ w,\ Z) \vdash_P^* (q_f,\ \epsilon,\ s),\ q_f \in F,\ s \in \Gamma^* \}$, where ϵ 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 $(q_0,\ w,\ Z) \vdash_P (p_1,\ w_1,\ s_1) \vdash_P (p_2,\ w_2,\ s_2) \vdash_P \ldots \vdash_P (p_i,\ w_i,\ s_i) \vdash_P (p_{i+1},\ w_{i+1},\ s_{i+1}) \vdash_P \ldots \vdash_P (q_f,\ \epsilon,\ u)$, 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 δ 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 δ' 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 $y = y_1 y_2 \ldots y_k$ 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 δ_1' means that this is transition on first tape), and following sequence of transitions on the other tape

1. $\delta_2'(p_i,\ x) = (q1,\ y_k,\ R)$
2. $\delta_2'(q_1,\ B) = (q_2,\ B,\ L)$
3. $\delta_2'(q_2,\ y_t) = (m_t,\ y_t,\ R)$ for $t = k,\ k-1,\ \ldots,\ 2$
4. $\delta_2'(m_t,\ B) = (q_1,\ y_{t-1},\ R)$ for $t = k,\ k-1,\ \ldots,\ 2$
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 $M = <Q,\ \sum,\ \Gamma,\ \delta,\ q_0,\ B,\ F>$, 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 $F = \{q_9\}$.

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

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

$\delta(q_0,\ 0) = (q_1,\ x,\ R)$

$\delta(q_0,\ 1) = (q_1,\ y,\ R)$

$\delta(q_0,\ x) = (q_4,\ x,\ L)$

$\delta(q_0,\ y) = (q_4,\ y,\ L)$

$\delta(q_1,\ 0) = (q_1,\ 0,\ R)$

$\delta(q_1,\ 1) = (q_1,\ 1,\ R)$

$\delta(q_1,\ B) = (q_2,\ B,\ L)$

$\delta(q_1,\ x) = (q_2,\ x,\ L)$

$\delta(q_1,\ y) = (q_2,\ y,\ L)$

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.

$\delta(q_2,\ 0) = (q_3,\ x,\ L)$

$\delta(q_2,\ 1) = (q_3,\ y,\ L)$

$\delta(q_3,\ 0) = (q_3,\ 0,\ L)$

$\delta(q_3,\ 1) = (q_3,\ 1,\ L)$

$\delta(q_3,\ x) = (q_0,\ x,\ R)$

$\delta(q_3,\ y) = (q_0,\ y,\ R)$

$\delta(q_5,\ 0) = (q_6,\ x,\ R)$

$\delta(q_5,\ 1) = (q_7,\ y,\ R)$

$\delta(q_5,\ B) = (q_9,\ B,\ L)$

$\delta(q_4,\ x) = (q_4,\ 0,\ L)$

$\delta(q_4,\ y) = (q_4,\ 1,\ L)$

$\delta(q_4,\ B) = (q_5,\ B,\ R)$

$\delta(q_6,\ 0) = (q_6,\ 0,\ R)$

$\delta(q_6,\ 1) = (q_6,\ 1,\ R)$

$\delta(q_6,\ B) = (q_6,\ B,\ R)$

$\delta(q_6,\ x) = (q_8,\ B,\ L)$

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.

$\delta(q_7,\ 0) = (q_7,\ 0,\ R)$ $\quad$ $\delta(q_8,\ 0) = (q_8,\ 0,\ L)$

$\delta(q_7,\ 1) = (q_7,\ 1,\ R)$ $\quad$ $\delta(q_8,\ 1) = (q_8,\ 1,\ L)$

$\delta(q_7,\ B) = (q_7,\ B,\ R)$ $\quad$ $\delta(q_8,\ B) = (q_8,\ B,\ L)$

$\delta(q_7,\ y) = (q_8,\ B,\ L)$ $\quad$ $\delta(q_8,\ x) = (q_5,\ x,\ R)$

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

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^{th} square for a given integer B.

Solution :

a) We know that universal language, $L_u = \{< M,\ w > \mid M$ is a TM that accepts $w\}$ is undecidable.

Let $L = \{< M_1,\ w > \mid M_1$ is a TM that accepts w and visits all states $\}$.

Showing that $L_u \leq_f L$ will show that L is undecidable.

Let M have states $q_0, q_1, ..., q_n$ and a final state q_f and accepts string w, i.e., $< M,\ w > \in L_u$.

Let f be a Turing computable function such that $f(M,\ w) = (M_1,\ w)$. Then f produces M_1 such that -

- When M reaches q_f on w, M_1 goes to the right end of the tape and writes a new symbol m on the first square found with a blank symbol and changes state to a new state q'
- M' then makes the following transitions -

$$q' \to q_0 \to q_1 \to ... \to q_n \to q'_f$$

where q'_f is the final state of M'.

Thus, M' visits all states on w, only if M halts on w in q_f, i.e., $< M,\ w > \in L_u$.

Since $L_u \leq_f L$, therefore L is undecidable.

b) We prove this for a Turing machine with an semi-infinite tape (equivalent to one with an infinite tape)

For a Turing machine $M = < Q,\ \sum,\ \Gamma,\ \delta,\ q_0,\ B,\ F >$, and given an integer B, define the partial-snapshot of M as the 3-tuple - (position of head, state, tape contents upto B^{th} square)

Claim : $L = \{< M, w > \mid M$ is a TM that accepts w and visits the B^{th} square $\}$ is decidable

Proof : A machine M, given B can have at most $n = B|\Gamma|^{B-1}|Q|$ partial snapshots.

This allows us to formulate an algorithm for the problem as follows

for i = 0 to n + 1

Simulate M for one step

Record the current partial snapshot of M

if B^{th} square is reached **then**

return "Yes"

else if current snapshot has been encountered before **then**

return "No"

end if

end for

Clearly the above algorithm terminates in finite steps for all inputs. For correctness consider the following

1. To go beyond the B^{th} square, the head needs to reach the B^{th} square first, and if that happens at any point the algorithm will return "Yes".
2. In the other case, i.e. the head is restricted to the first $B-1$ squares, there can only be n different partial snapshots. Thus by PHP, in $n+1$ steps, at least one snapshot has to occur more than once.
3. When a snapshot occurs again, the entire sequence of snapshots from its first occurrence will also repeat (as the state, tape and head position will be the same) after it, implying that the machine will get stuck in an infinite loop. Here, the algorithm correctly outputs "No"