PTC (Fall 2018) – Assignment 3

徐翔哲 161250170

2018 年 12 月 19 日

Problem 1

Consider the (deterministic) Turing machine M given by

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, B\}, \delta, q_0, B, \{q_2\})$$

which has exactly four transitions defined in it, as described below.

1. $\delta(q_0, a) = (q_0, B, R)$
2. $\delta(q_0, b) = (q_1, B, R)$
3. $\delta(q_1, b) = (q_1, B, R)$
4. $\delta(q_1, B) = (q_2, B, R)$

Please answer the following questions:

a. Specify the execution trace of M on the input string abb.

b. Provide a regular expression for the language of the Turing machine.

c. Suppose we added the transition $\delta(q_1, a) = (q_0, B, R)$ to the above machine, provide a regular expression for the language of the resulting Turing machine.

a

$q_0aab \vdash q_0ab \vdash q_0b \vdash q_1B \vdash q_2B$

b

a^*b^+

c

$a^*b^+(a^+b^+)^*$

Problem 2

Please design TM′s to decide following languages:

a. $L_1 = \{1^m \times 1^n = 1^{mn} \mid m, n \in \mathbb{N}^+\}$ (e.g. $11 \times 111 = 111111 \in L_1$, but $1 \times 1 = 11 \notin L_1$)

b. $L_2 = \{ww \mid w \in \{a, b\}^*\}$

a

First, let's define a TM M_1 which will convert $\times 1^n = 1^{n'}$ to $\times 1^n = 1^{n'-n}$ where $n' \geq n > 0$

$$M_1 = (\{p_0, p_R, p_a, p_{a'}, p_d, p_L, p_f, p_e\}, \{1, 0, \times, =\}, \{1, 0, \times, =, B, a\}, \delta, p_0, B, \{p_e\})$$

At the beginning , the head should at $\times$ Then we will enter p_a, which will change the first 1 to a when the head goes right.
$\delta(p_0, \times) = (p_a, \times, R)$
$\delta(p_a, a) = (p_a, a, R)$
$\delta(p_a, 1) = (p_R, a, R)$

The state p_R moves the head to the first blank at the right of the input, and then switches to state p_d, which will delete a 1.
$\delta(p_R, X) = (p_R, X, R)$, where $X = \{=, 1\}$
$\delta(p_R, B) = (p_d, B, L)$
$\delta(p_d, 1) = (p_L, B, L)$
The state p_L will moves to $\times$ and then enters p'_a
$\delta(p_L, X) = (p_L, X, L)$, where $X = \{1, =, a\}$
$\delta(p_L, \times) = (p_a, \times, R)$

Then the state $p_{a'}$ will act as p_a, change a 1 to a, then move to the right and delete a 1 ...... except that $p_{a'}$ will go to the state p_f if it sees no 1 before the =, under which condition p_a would halt without accepting.
$\delta(p_{a'}, a) = (p_{a'}, a, R)$
$\delta(p_{a'}, 1) = (p_R, a, R)$
$\delta(p_{a'}, =) = (p_f, =, L)$

The state p_f will change all the a back to 1.
$\delta(p_f, a) = (p_f, 1, L)$
$\delta(p_f, \times) = (p_e, \times, R)$
Then we construct the TM M to describe the language L_1

$$M = (M_1.states \cup \{q_0, q_R, q_{call}, q_L, q_1, q_e, q_{check}, q_f\}, \{1, 0, \times, =\}, \{1, 0, \times, =, B, a\}, M_1.\delta \cup \delta, q_0, B, \{q_f\})$$

The state q_0 will remove the first 1, and goes to state q_R
$\delta(q_0, 1) = (q_R, B, R)$
The state q_R will move to the $\times$ and then enters q_{call}
$\delta(q_R, 1) = (q_R, 1, R)$
$\delta(q_R, \times) = (q_{call}, \times, L)$
The state q_{call} will call the M_1
$\delta(q_call, 1) = (p_0, 1, R)$

When the M returns from M_1, state q_L will move the head to the blank at the left end of the input and then enters q_1

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

$\delta(q_L, \times) = (q_L, \times, L)$

$\delta(q_L, B) = (q_1, B, R)$

q_1 acts just like q_0 except that when it finds no 1 before $\times$, it will switch to q_e

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

$\delta(q_1, \times) = (q_e, \times, R)$

q_e and q_{check} will check whether the RHS of the = is blank.

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

$\delta(q_e, =) = (q_{check}, =, R)$

$\delta(q_{check}, B) = (q_f, B, L)$

b

We will use multi-track TMs in this problem.

First, define a TM M_1 which will compare whether two strings are equal to each other. The beginning of the second string will be marked as <x,c>, the beginning before the first string is marked as <z,B>, where c $\in$ { a,b }. Other related cells on the second track is initialized to *.

$$M_1(\{q_0, q_c, q_a, q_b, q_{da}, q_{db}, q_L, q_f, q_e\}, \{a, b\}, \{a, b, x, *, z, B\}, \delta, q_0, B, \{q_e\})$$

q_0 will move the head to the end of the first string. $\delta(q_0, < *, X >) = (q_0, < *, X >, R)$, where $X \in \{a, b\}$

$\delta(q_0, < x, X >) = (q_c, < x, X >, L)$, where $X \in \{a, b\}$

q_c will remove a character from the tail of the first string and remember this string in its state.

$\delta(q_c, < *, a >) = (q_a, < *, B >, R)$

$\delta(q_c, < *, b >) = (q_b, < *, B >, R)$

$\delta(q_a, < *, X >) = (q_a, < *, X >, R)$

$\delta(q_b, < *, X >) = (q_b, < *, X >, R)$

q_a and q_b will move the head to the end of the second string and remember to delete a or b respectively.

$\delta(q_a, < *, B >) = (q_{da}, < *, B >, L)$

$\delta(q_b, < *, B >) = (q_{db}, < *, B >, B)$

q_{da} and q_{db} will delete one a or one b respectively.

$\delta(q_{da}, < *, a >) = (q_L, < *, B >, L)$

$\delta(q_{da}, < x, a >) = (q_L, < x, B >, L)$

$\delta(q_{db}, < *, b >) = (q_L, < *, B >, L)$

$\delta(q_{db}, < x, b >) = (q_L, < x, B >, L)$

q_L will move the head to the ending blanks of the first string.

$\delta(q_L, < *, X >) = (q_L, < *, X >, L)$, where $X \in \{a, b\}$

$\delta(q_L, < x, X >) = (q_c, < x, X >, L)$, where $X \in \{a, b\}$

q_c will find the next char to remove, when it meets z on the second track, the TM goes to a new state q_f

$\delta(q_c, < *, B >) = (q_c, < *, B >, L)$

$\delta(q_c, < z, B >) = (q_f, < z, B >, R)$

q_f will check whether the second string has been reduced to blanks, if so, it will change to the accepting stateq_e.

$\delta(q_f, < *, B >) = (q_f, < *, B >, R)$

$\delta(q_f, < x, B >) = (q_e, < x, B >, R)$

Then we define the TM M for language L_2.

$M(M_1.states \cup \{p_0, p_1, p_{odd}, p_{even}, p_{back}, p_{fwd}, p_R, p_{last}, p_{next}, p_{markZ}\}, \{a, b\}, \{a, b, x, *, t, z, B\}, M_1.\delta \cup \delta, p_0, B, \{q_e\})$

M will accept the empty strings.

$\delta(p_0, < B, B >) = (q_e, < B, B >, R)$

Firstly, marked the first blank at the left of the input as x.

$\delta(p_0, < B, X >) = (p_1, < B, X >, L)$

$\delta(p_1, < B, B >) = (p_{odd}, < x, B >, R)$

If this is the odd indexed(start from 1) char from the input, then the state will change to even and vice versa.

$\delta(p_{odd}, < B, X >) = (p_{even}, < *, X >, R)$, where $X \in \{a, b\}$

$\delta(p_{even}, < B, X >) = (p_{back}, < t, X >, L)$, where $X \in \{a, b\}$

When M makes every two steps right, it will move x forward for one step. Then when the first blank at right is met, the x will be moved to the end of the first string.

$\delta(p_{back}, < *, X >) = (p_{back}, < *, X >, L)$, where $X \in \{a, b\}$

$\delta(p_{back}, < x, X >) = (p_{fwd}, < *, X >, R)$, where $X \in \{a, b, B\}$

$\delta(p_{fwd}, < *, X >) = (p_R, < x, X >, R)$, where $X \in \{a, b\}$

$\delta(p_R, < *, X >) = (p_R, < *, X >, R)$, where $X \in \{a, b\}$

$\delta(p_R, < t, X >) = (p_{odd}, < *, X >, R)$, where $X \in \{a, b\}$

Then we move x forward for one more step.

$\delta(p_{odd}, < B, B >) = (p_{last}, < *, B >, R)$

$\delta(p_{last}, < *, X >) = (p_{last}, < *, X >, L)$, where $X \in \{a, b\}$

$\delta(p_{last}, < x, X >) = (p_{next}, < *, X >, R)$, where $X \in \{a, b\}$

$\delta(p_{next}, < *, X >) = (p_{markZ}, < x, X >, L)$, where $X \in \{a, b\}$

$\delta(p_{markZ}, < *, X >) = (p_{markZ}, < *, X >, L)$, where $X \in \{a, b\}$

Finally, we marked the blank at the left of the input as z, and call M_1.

$\delta(p_{markZ}, < *, B >) = (q_0, < z, B >, R)$

Problem 3

A *useless state* in a Turing machine is one that is never entered on any input string. Consider the problem of determining whether a state in a Turing machine is useless. Formulate this problem as a language and show it is decidable or undecidable. (**Hint**: consider the language E_{TM})

Solution.

Problem 4

Show that the following questions are decidable:

a. The set L of codes for TM's M such that, when started with the blank tape will eventually write some nonblank symbol on its tape. (**Hint**: If M has m states, consider the first m transitions that it makes)

b. The set L of codes for TM's that never make a move left on any input.

c. The set L of pairs (M, w) such that TM M, started with input w, never scans any tape cell more than once.

Proof.

Problem 5

If a pushdown automaton has k stacks, we call it $k-$PDA. Clearly, $0-$PDA is NFA, $1-$PDA is PDA, and $1-$PDA is more powerful than $0-$PDA.

1. What is the difference between the express ability of 2-PDA and 1-PDA. Please clarify your argument. Prove the (un)equivalence.

2. How about 3-PDA and 2-PDA.

Solution.

Problem 6

Suppose we have an encoding of context-free grammars using some finite alphabet. Consider the following two languages:

1. $L_1 = \{(G, A, B) \mid G$ is a (coded) CFG, A and B are (coded) varibles of G, and the sets of terminal strings derived from A and B are the same$\}$.

2. $L_2 = \{(G_1, G_2) \mid G_1$ and G_2 are (coded) CFG′s, and $L(G_1) = L(G_2)\}$.

Answer the following questions:

a. Show that L_1 is polynomial-time reducible to L_2.

b. Show that L_2 is polynomial-time reducible to L_1.

Proof.

Problem 7

As classes of languages, $\mathcal{P}$ and $\mathcal{NP}$ each have certain closure properties. Prove or disprove that $\mathcal{P}$ and $\mathcal{NP}$ are closed under each of the following operations:

a. Union.

b. Concatenation.

c. Complementation.

Proof.