902 43500 HOMEWORK 4

due Tuesday, December 7, 2010

All problems are in the book: Michael Sipser, *Introduction to the Theory of Computation*, 2nd Edition, PAWS Publishing Company, 2005.

Solution by Cheng-Chung Li

Problem 1 Give implementation-level descriptions of Turing Machines that decide the language $\{w|w$ does not contain twice as many 0s as 1s$\}$ over the alphabet $\{0, 1\}$.
Ans.

"On input string w:

1. Scan the tape and mark the first 0 which has not been marked. If there is no unmarked 0, go to stage 5.

2. Move on and mark the next unmarked 0. If there is not any on the tape, *accept*. Otherwise, move the head back to the front of the tape.

3. Scan the tape and mark the first 1 which has not been marked. If there is no unmarked 1, *accept*.

4. Move the head back to the front of the tape and repeat stage 1.

5. Move the head back to the front of the tape. Scan the tape to see if there are any unmarked 1s. If there is not, *reject*. Otherwise, *accept*."

□

Problem 2 Let a k-PDA be a pushdown automaton that has k stacks. Thus a 0-PDA is an NFA and a 1-PDA is a conventional PDA. You already know that 1-PDAs are more powerful (recognize a larger class of languages) than 0-PDAs.

a. Show that 2-PDAs are more powerful than 1-PDAs.

b. Show that 3-PDAs are not more powerful than 2-PDAs.
(Hint: Simulate a Turing Machine tape with two stacks.)

Ans.

a. Since the example 2.36 in the textbook has proved that the language $L = \{a^n b^n c^n | n \geq 0\}$ is not context-free, building a 1-PDA to recognize L is impossible. However, you can easily build a 2-PDA to recognize L. According to the discussion, 2-PDAs are more powerful than 1-PDAs since they can recognize more languages than 1-PDAs do.

b. The first step for solving this problem is to show that the power of the 2-PDAs is equivalence to the standard Turing Machines. We split the tape of a TM into two stacks. Stack 1 (Stack 2, respectively) stores the characters on the left (right, respectively) of the head, with the bottom of stack storing the leftmost (rightmost, respectively) character of the tape in the TM. Moreover, for each transition $\delta(q_i, c_i) = (q_j, c_j, L)$ in the TM, the corresponding PDA transition pops c_i off stack 2, pushes c_j into stack 2, pops stack 1 and then pushes the popped character into stack 2, and goes from state q_i to q_j. For any transition $\delta(q_i, c_i) = (q_j, c_j, R)$ in the TM, the corresponding PDA transition pops c_i off stack 2 and takes away c_i, push c_j into stack 1, and goes from state q_i to q_j. Please also check the slides I made for the course on November 30, 2010 on the blog. Actually, the power of the k-tape Turing Machines is greater than the k-PDAs. Also, the k-PDAs are more powerful than the 2-PDAs. However, we know that the 2-PDAs are equivalence to both the standard and the k-tape Turing Machines. Thus all machines listed above have the same power.

□

Problem 3 Show that the collection of Turing-recognizable languages is closed under the operations of

a. concatenation.

b. star.

c. intersection.

Ans.

a. For any two Turing-recognizable languages L_1 and L_2, let M_1 and M_2 be the TMs that recognize them. We construct a NTM M' that recognizes the concatenation of L_1 and L_2:
"On input w:

1. Nondeterministically cut w into two parts w_1w_2.

2. Run M_1 on w_1. If it halts and rejects, *reject.* If it accepts, go to stage 3.

3. Run M_2 on w_2. If it accepts, *accept.* If it halts and rejects, *reject.*"

If there is a way to cut w into two substrings such that M_1 accepts the first part and M_2 accepts the second part, w belongs to the concatenation of L_1 and L_2 and M' will accept w after a finite number of steps.

b. For any Turing-recognizable language L, let M be the TM that recognizes it. We construct a NTM M' that recognizes the star of L:
"On input w:

1. Nondeterministically cut w into parts so that $w = w_1w_2 \dots w_n$.

2. Run M on w_i for $i = 1, 2, \dots, n$. If M accepts all of them, *accept.* If it halts and rejects any of them, *reject.*"

If there is a way to cut w into substrings such that M accepts all the substrings, w belongs to the star of L and thus M' will accept w after a finite number of steps.

c. For any two Turing-recognizable languages L_1 and L_2, let M_1 and M_2 be the TMs that recognize them. We construct a TM M' that recognizes the intersection of L_1 and L_2:
"On input w:

1. Run M_1 on w, if it halts and rejects, *reject.* If it accepts, go to stage 2.

2. Run M_2 on w, if it halts and rejects, *reject.* If it accepts *accept.*"

If both of M_1 and M_2 accept w, w belongs to the intersection of L_1 and L_2 and M' will accept w after a finite number of steps.

$\square$