Turing Machines

8 Apr 2019

1. Write the algorithm for a Turing Machine which, when given a list of strings over $\{0,1\}$ separated by #, determines if all strings are different i.e. accepts the language $\{\#x_1\#x_2\#...\#x_k | x_i \in \{0,1\}^*, x_i \neq x_j \text{ for } i \neq j\}$

   Solution Solution Compare x_1 with $x_2, \cdots, x_k$, then compare x_2 with $x_3, \cdots, x_k$,and so on. $M =$ On input w
   (a) Place a mark on top of the leftmost tape symbol. If that symbol was a blank, accept. If that symbol was a # continue with the next stage. Otherwise reject.
   (b) Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so accept.
   (c) By zig-zagging, compare the two strings to the right of the marked #-s. If they are equal, reject
   (d) Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. If no # is available for the rightmost mark, all strings have been compared so accept

2. Prove that the problem of determining whether a CFG generates any strings at all is decidable. The Emptiness testing problem for CFG can be expressed as a language, E_{CFG}.

$$E_{CFG} = \{\langle G \rangle \mid G \text{ is a CFG and } L(G) = \phi\}$$

   Solution

   [Refer to the proof in Sipser book]

E_{CFG} is a decidable language.

PROOF IDEA To find an algorithm for this problem, we might attempt to use TM S from Theorem 4.7. It states that we can test whether a CFG generates some particular string w. To determine whether $L(G) = \emptyset$, the algorithm might try going through all possible w's, one by one. But there are infinitely many w's to try, so this method could end up running forever. We need to take a different approach.

In order to determine whether the language of a grammar is empty, we need to test whether the start variable can generate a string of terminals. The algorithm does so by solving a more general problem. It determines *for each variable* whether that variable is capable of generating a string of terminals. When the algorithm has determined that a variable can generate some string of terminals, the algorithm keeps track of this information by placing a mark on that variable.

First, the algorithm marks all the terminal symbols in the grammar. Then, it scans all the rules of the grammar. If it ever finds a rule that permits some variable to be replaced by some string of symbols, all of which are already marked, the algorithm knows that this variable can be marked, too. The algorithm continues in this way until it cannot mark any additional variables. The TM R implements this algorithm.

PROOF

$R =$ "On input $\langle G \rangle$, where G is a CFG:

1. Mark all terminal symbols in G.
2. Repeat until no new variables get marked:
3. Mark any variable A where G has a rule $A \to U_1 U_2 \cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.
4. If the start variable is not marked, *accept*; otherwise, *reject*."

Turing Machines

8 Apr 2019

Intruction : Write the answers to the problems marked as (to submit) neatly in loose sheets with your name and roll number. Submit to the TA after the class.

1. Describe a Turing Machine for accepting $\{a^i b^j c^k | i, j, k \geq 1, i = j + k\}$. (To submit)

2. Define a two-headed finite automaton (2DFA) to be a deterministic finite automaton that has two read-only, bidirectional heads that start at the left-hand end of the input tape and can be independently controlled to move in either direction. The tape of a 2DFA is finite and is just large enough to contain the input plus two additional blank tape cells, one on the left-hand end and one on the right-hand end, that serve as delimiters. A 2DFA accepts its input by entering a special accept state. (to submit)

   (a) Describe how a 2DFA can recognize $\{a^n b^n c^n \mid n \geq 0\}$.

   (b) Let $A_{2DFA} = \{\langle M, x\rangle | M \text{ is a 2DFA and M accepts } x\}$. Show that A_{2DFA} is decidable. (Hint: if the 2DFA has s states and input size is x, compute an upper bound on the number of steps to simulate for acceptance.)

3. Prove that EQ_{DFA} is decidable by testing the two DFAs on a finite number of strings. Calculate a number that works. (To submit)

4. Let $C = \{\langle G, x\rangle | G \text{ is a CFG}, x \text{ is a substring of some } y \in L(G)\}$. Show that C is decidable. (To submit) (Hint : The Turing Machine for this problem can use the techniques for deciding emptiness of CFGs. However, you will also need to apply closure properties of CFGs)

5. A useless state in a pushdown automaton is never entered on any input string. Consider the problem of determining whether a pushdown automaton has any useless states. Formulate this problem as a language and show that it is decidable. (Home)

6. Let M be a Turing Machine with one semi-infinite tape and two read/write heads. A transition of M is of the form -

$\delta(q, a, b) = (q', c, d, D, D_2)$ where $D_1, D_2 \in L, R$.
If both the heads point to the same tape cell, the symbol is replaced by c. The first head moves in the direction D_1 and the second head moves in the direction D_2. Show that this two-head Turing machine can be simulated by a standard Turing machine M_1 with a semi-infinite tape with a single track and only one read/write head. (Home)