Solutions to Homework 7

March 7, 2015

Exercise 1 (Ex 7.4.1, page 301)**.** *Give algorithms to decide the following:*

a) Is $L(G)$ finite, for a given CFG G? Hint: Use the pumping lemma.

b) Does $L(G)$ contain at least 100 strings, for a given CFG G?

Proof. (a) Convert G to Greibach normal form (without useless symbols), that is, all productions are of the form $A \to a\gamma$, where $\gamma \in V^*$. Build a directed graph with vertices V (= variables), and add an edge $A \to B$ if B appears in γ. $L(G)$ is finite if and only if the graph does not contain cycles.

(b) Apply (a) first, and assume without loss of generality G is already in Greibach normal form. If $L(G)$ is infinite, return YES. Otherwise, enumeration all possible derivations from S; if there are at least 100 different strings, return YES; otherwise, return NO.

□

Exercise 2 (Ex 7.4.2, page 308)**.** *Develop linear-time algorithms for the following questions about CFG's:*

a) Which symbols appear in some sentential form?

b) Which symbols are nullable (derive ϵ)?

Proof. (a) Fix symbol $a \in \Sigma$. We want to know whether or not a appears in some sentential form $\gamma \in (V \cup T)^*$. If suffices to know whether or not a appears in each variable. Construct a graph with vertices $V \cup \{a\}$. If there is a production rule $A \to \gamma$, where γ contains terminal a, then add an edge $A \to a$; If there is a production rule $A \to \gamma$, where γ contains variable B, then add an edge $A \to B$. It is easy to check that a can appear in variable A if and only if there is a path from A to a. Both the construction and reachability test can be done in linear time.

(b) Initially, mark all variables A with production $A \to \epsilon$. If there is a production $A \to B_1 B_2 \ldots B_m$, where all B_i's all nullable, then mark A as nullable. This can be implemented in linear time (for each production of the form $A \to B_1 B_2 \ldots B_m$, assign an integer denoting the number of nullable variable on the right hand side, which is 0 initially; for each variable A, store a list of *pointers to all productions* whose right hand side containing A, and when A becomes nullable, update the corresponding counters). □

Exercise 3 (Ex 8.2.2, page 335-336)**.** *Design Turing machines for the following languages:*

a) The set of strings with an equal number of 0*'s and* 1*'s.*

b) $\{a^n b^n c^n : n \geq 1\}$.

c) $\{ww^R : w$ *is any string of* 0*'s and* 1*'s*$\}$.

Proof. (a)

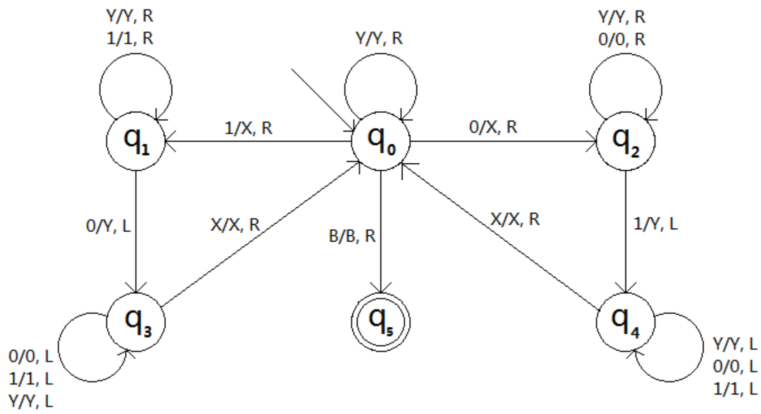

Figure 1: TM accepting strings with an equal number of 0's and 1's

(b) Please see lecture notes 13.
(c) See Figure 3. □

Exercise 4 (Ex 8.2.3, page 336)**.** *Design a Turing machine that takes as input a number* N *and adds* 1 *to it in binary. To be precise, the tape initially contains a \$ followed by* N *in binary. The tape head is initially scanning the \$ in state* q_0. *Your TM should halt with* $N+1$, *in binary, on its tape, scanning the leftmost symbol of* $N+1$, *in state* q_f. *You may destroy the \$*

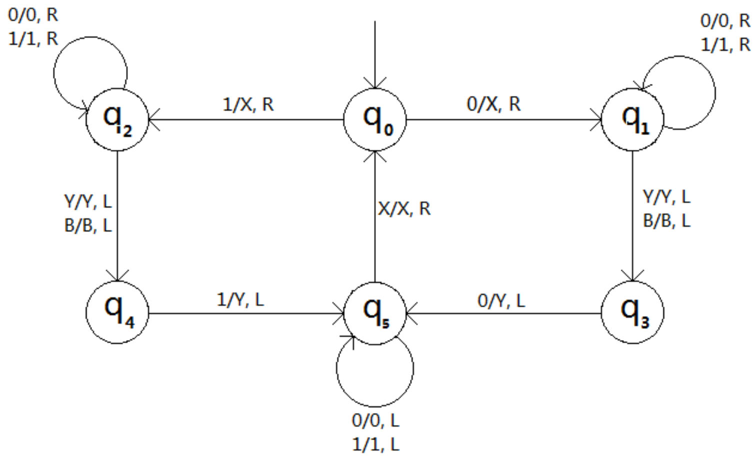

Figure 2: TM accepting $\{ww^R : w \text{ is any string of 0's and 1's}\}$

in creating $N + 1$*, if necessary. For instance,* $q_0\$10011 \vdash^* \$q_f10100$*, and* $q_0\$11111 \vdash^* q_f100000$.

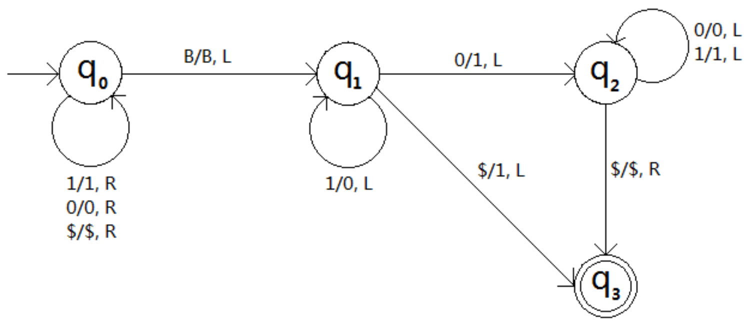

Figure 3: TM performing adding by one

Exercises are from the book "Automata Theory, Language, and Computation", 3rd edition, by John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman, published by Addison-Wesley.