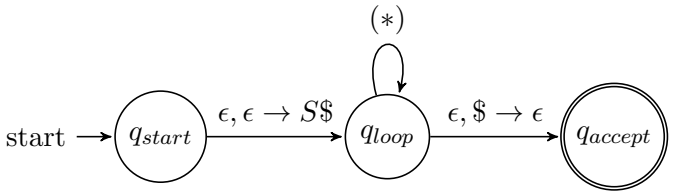
U Vic CSC320 W2020 Homework 3 Solutions

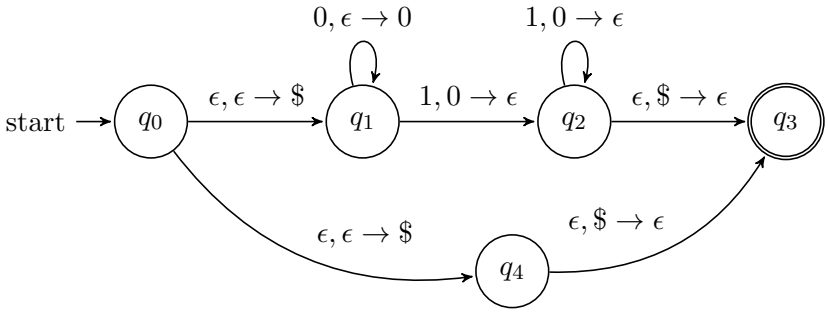
1. The automaton has the following form (10 marks)



   Where $(*)$ consists of the following transitions (10 marks):

$$\begin{aligned}
\epsilon, S &\to aAbS \\
\epsilon, S &\to bBaS \\
\epsilon, S &\to \epsilon \\
\epsilon, A &\to aAbA \\
\epsilon, A &\to \epsilon \\
\epsilon, B &\to bBaB \\
\epsilon, B &\to \epsilon \\
a, a &\to \epsilon \\
b, b &\to \epsilon
\end{aligned}$$

2. First we must add an extra state q_4, and transitions that take us from q_0 to q_3 via q_4, consuming no input and going from an empty stack to an empty stack. We also make q_0 non-final. This gives the following PDA (5 marks)



   (10 marks) We need to match the (1) push \$ going from q_0 to q_1 with the pop \$ going from q_2 to q_3, the (2) push \$ going from q_0 to q_4 with the pop \$ going from q_4 to q_3, the push 0 going

from q_1 to q_1 to the pop 0 going from (3) q_1 to q_2 and (4) q_2 to q_2. (1) gives us $A_{03} \to \epsilon A_{12} \epsilon$, (2) gives us $A_{03} \to \epsilon A_{44} \epsilon$, (3) gives us $A_{12} \to 0A_{11}1$, and (4) gives us $A_{12} \to 0A_{12}1$. Since we have $A_{44} \to \epsilon$ and $A_{11} \to \epsilon$ we simplify to get (5 marks)

$$A_{03} \to \epsilon \mid A_{12}$$
$$A_{12} \to 01 \mid 0A_{12}1$$

Note that with the full translation, there are many more productions, e.g., every production of the form $A_{ii} \to \epsilon$, $0 \leq i \leq 4$, and $A_{ij} \to A_{ik}A_{kj}$, $0 \leq i, j, k \leq 4$. (You do not need to include these to get full marks, but it is OK to include them.)

3. Baisc idea (10 marks): This is almost like just scanning the input from left to right, which is is similar to a DFA. However, we can use S to simulate ϵ-moves, so the fact that a TM has only one accept state (unlike a DFA) is not a problem – we can just take an ϵ-move to the single accepting state to simulate many accepting states. Rejecting can be handled the same way. So clearly all regular languages are recognized by such a machine.

   Some subtle points (5 marks):

   - We can assume without loss of generality that we only ever write back the symbol that is already on the cell we are scanning. This is obvious if we move right (R). If we stay put (S), we could encode the symbol we would have written in the state, so we don't actually have to write a different symbol.
   - This model has some apparent extra power, since it can accept without reading the whole input. However, we can use the fact that regular languages are closed under prefix to conclude that the languages recognized by this model are exactly the regular languages.

4. For (a) $L = L_1 \cap L_2$, were L_1 is decided by M_1 and L_2 by M_2. We just need to run the two deciding machines in parallel, and accept iff both of them accept (6 marks). For (b) $L = L_1L_2$, we can try all the ways of writing an input w as a concatentation w_1w_2 . If for any of these, the decider for L_1 accepts w_1 and the decider for L_2 accepts w_2 we accept w. Otherwise we reject w. Or we could use nondeterminism (6 marks for either). (c) is given in the lecture notes – just switch accepting and rejecting states (3 marks).

5. We simulate M for m steps (4). If it halts without writing a nonblank, we reject (2). If it ever writes a nonblank symbol during the simulation, we immediately accept (2). Otherwise we reject – this is because after m we will have entered a state we've already been in, scanning the same symbol (i.e. blank.) At this point we know we will just do the same thing we've already done repeatedly, and never write a nonblank (7).

6. (Only if. 5 marks) Suppose that L is decided by TM M. We can enumerate L as follows: for each w in the standard ordering $\epsilon \prec 0 \prec 1 \prec 00 \prec 01 \ldots$, run M on w. If M accepts output w. If M rejects go to the next string. Note this works since M always either accepts or rejects.

   (If. 10 marks) Now suppose that L is enumerated by an enumerator E which respects the standard order. To decide whether w is in L, we run E, and for each v it enumerates, if $v = w$ then we halt and accept. Otherwise, if E halts before outputting w (L is finite), or E outputs v such that $w \prec v$, we reject. Note that if L is infinite, and $w \notin L$, then eventually E must output such a v.