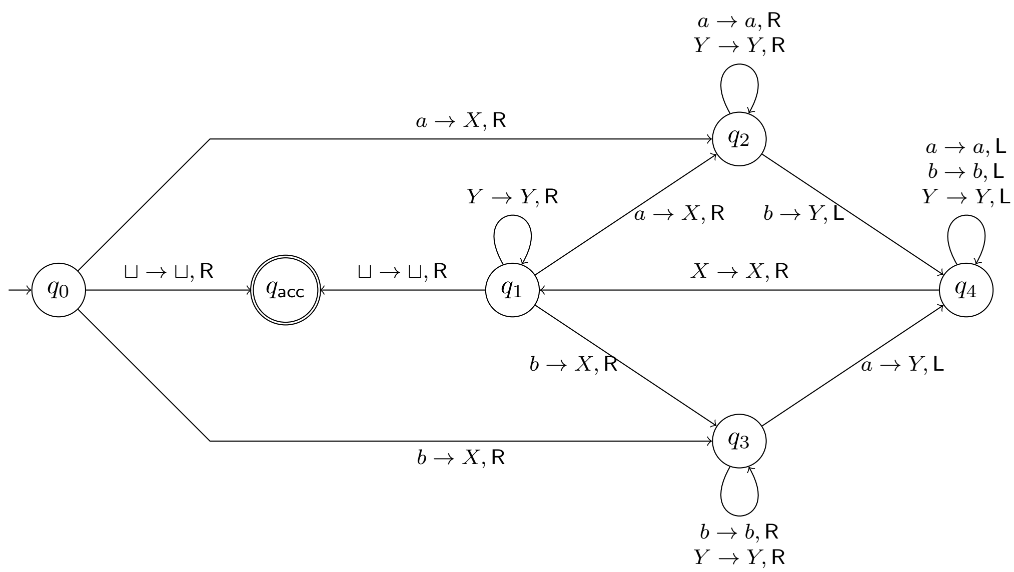
Solutions for Problem Set 6
CS 373: Theory of Computation

Assigned: October 12, 2010 Due on: October 19, 2010 at 10am

Homework Problems

Problem 1. [Category: Comprehension] Consider the following Turing Machine M with input alphabet $\Sigma = \{a, b\}$. The reject state q_{rej} is not shown, and if from a state there is no transition on some symbol then



as per our convention, we assume it goes to the reject state.

1. Give the formal definition of M as a tuple. **[3 points]**
2. Describe each step of the computation of M on the input *baabab* as a sequence of instantaneous descriptions. **[3 points]**
3. Describe the language recognized by M. Give an informal argument that outlines the intuition behind the algorithm used by M justifies your answer. **[4 points]**.

Solution:

1. The Turing Machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$ where
   - $Q = \{q_0, q_1, q_2, q_3, q_4, q_{\mathsf{acc}}, q_{\mathsf{rej}}),$
   - $\Sigma = \{a, b\},$
   - $\Gamma = \{a, b, \sqcup, X, Y\},$

- δ is given as follows

$$\begin{array}{lll}
\delta(q_0, \sqcup) = (q_{\mathsf{acc}}, \sqcup, \mathsf{R}) & \delta(q_0, a) = (q_2, X, \mathsf{R}) & \delta(q_0, b) = (q_3, X, \mathsf{R}) \\
\delta(q_1, Y) = (q_1, Y, \mathsf{R}) & \delta(q_1, a) = (q_2, X, \mathsf{R}) & \delta(q_1, b) = (q_3, X, \mathsf{R}) \\
 & \delta(q_1, \sqcup) = (q_{\mathsf{acc}}, \sqcup, \mathsf{R}) & \\
\delta(q_2, a) = (q_2, a, \mathsf{R}) & \delta(q_2, Y) = (q_2, Y, \mathsf{R}) & \delta(q_2, b) = (q_4, Y, \mathsf{L}) \\
\delta(q_3, b) = (q_3, b, \mathsf{R}) & \delta(q_3, Y) = (q_3, Y, \mathsf{R}) & \delta(q_3, a) = (q_4, Y, \mathsf{L}) \\
\delta(q_4, a) = (q_4, a, \mathsf{L}) & \delta(q_4, b) = (q_4, b, \mathsf{L}) & \delta(q_4, Y) = (q_4, Y, \mathsf{L}) \\
 & \delta(q_4, X) = (q_1, X, \mathsf{R}) &
\end{array}$$

In all other cases, $\delta(q, c) = (q_{\mathsf{rej}}, \sqcup, \mathsf{R})$.

2. The computation proceeds as follows.

$$\begin{aligned}
q_0 baabab &\vdash Xq_3aabab \vdash q_4XYabab \vdash Xq_1Yabab \vdash XYq_1abab \vdash XYXq_2bab \vdash XYq_4XYab \\
&\vdash XYXq_1Yab \vdash XYXYq_1ab \vdash XYXYXq_2b \vdash XYXYq_4XY \vdash XYXYXq_1Y \\
&\vdash XYXYXYq_1\sqcup \vdash XYXYXY\sqcup q_{\mathsf{acc}}\sqcup
\end{aligned}$$

3. The Turing machine recognizes the following language

$$L = \{w \in \{a, b\}^* \mid w \text{ has equal number of } a\text{s and } b\text{s}\}$$

The machine first marks the leftmost unmarked a (or b) as X and then scans right to find the left most unmarked b (or a). This matching b (or a) is marked as Y, and the machine scans back left to move to the rightmost X, and repeats the entire process.

■

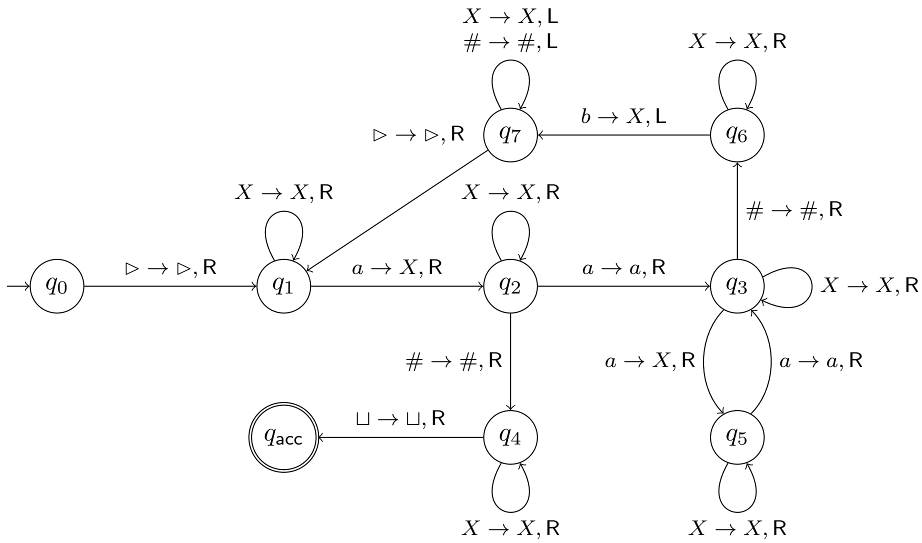
Problem 2. [Category: Design] For $\Sigma = \{\triangleright, \#, a, b\}$, design a Turing machine to recognize the language

$$L = \{\triangleright a^{2^n} \# b^n \mid n \geq 0\}$$

Note: By definition $\triangleright a\# \in L$. You need not prove that your construction is correct, but you must clearly explain the intuitions behind your construction. **[10 points]**

Solution: A very closely related problem is solved in example 3.7 of the textbook, where the machine checks if the number of 0s it sees is a power of 2. The solution there proceeds in stages. In each stage the machine, scans right and marks half the 0s (by marking alternate unmarked 0s), and returns back to the left end. In each stage it also checks that the number of unmarked 0s is either even or it is 1.

In this problem we need to check if the number of as we see is 2 raised to the power of the number of bs. We could just use a modified version of that solution — in each stage, when we mark-off half the unmarked as, we also mark off a b. We draw a TM implementing this solution; as always, missing transitions go to the reject state.



■

Problem 3. [Category: Comprehension+Proof] Solve problem 3.13. **[10 points]**

Solution: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$ be a Turing machine with stay put instead of left. We claim that such a machine can only recognize regular languages. The reason is though it can write on the tape, it cannot use what it writes once it move beyond that cell. We will show this by constructing an NFA N that recognizing the same language as M. The intuition behind this construction is that the NFA will store the symbol it reads in its control state, and whatever changes M makes to that cell, N will keep track of those changes in its finite control by making ϵ-steps. Once M moves right, N will read the next symbol.

Formally, $N = (Q', \Sigma, \delta', q_I, F)$ where

- $Q' = Q \times (\Gamma \cup \{\mathrm{rd}\})$. So the state is a pair (q, X) where q is the state of M that is being kept track of and X is the symbol in the cell that M is reading currently; if $X = \mathrm{rd}$ it means that M moved right in the previous step and N must read the next symbol from input.
- $q_I = (q_0, \mathrm{rd})$. Initially, M is in q_0, and we should read the first symbol.
- $F = \{(q_{\mathsf{acc}}, a) \mid (q_{\mathsf{acc}}, a) \in Q'\}$. So we accept whenever our simulation of M ends in an accept state.
- δ is given by

$$\delta'((q, X), a) = \begin{cases} \{(q', X')\} & \text{if } X \neq \mathrm{rd} \text{ and } a = \epsilon \text{ and } \delta(q, X) = (q', X', \mathrm{S}) \\ \{(q', \mathrm{rd})\} & \text{if } X \neq \mathrm{rd} \text{ and } a = \epsilon \text{ and } \delta(q, X) = (q', X', \mathsf{R}) \\ \{(q, a)\} & \text{if } X = \mathrm{rd} \text{ and } a \neq \epsilon \end{cases}$$

The correctness of this construction can be proved by observing that

$$q_0 w_1 w_2 \cdots w_n \vdash^* a_1 \cdots a_{k-1} q b w_{k+1} \cdots w_n \text{ iff } (q, b) \in \hat{\Delta}(q_0, w_1 w_2 \cdots w_k)$$

In other words, if M is scanning the kth cell which now contains b, and its control state is q then N will reach control state (q, b) after reading $w_1 \cdots w_k$; the converse also holds. As always this statement can be established by induction, but this time over the number of steps in the computation of M. ■