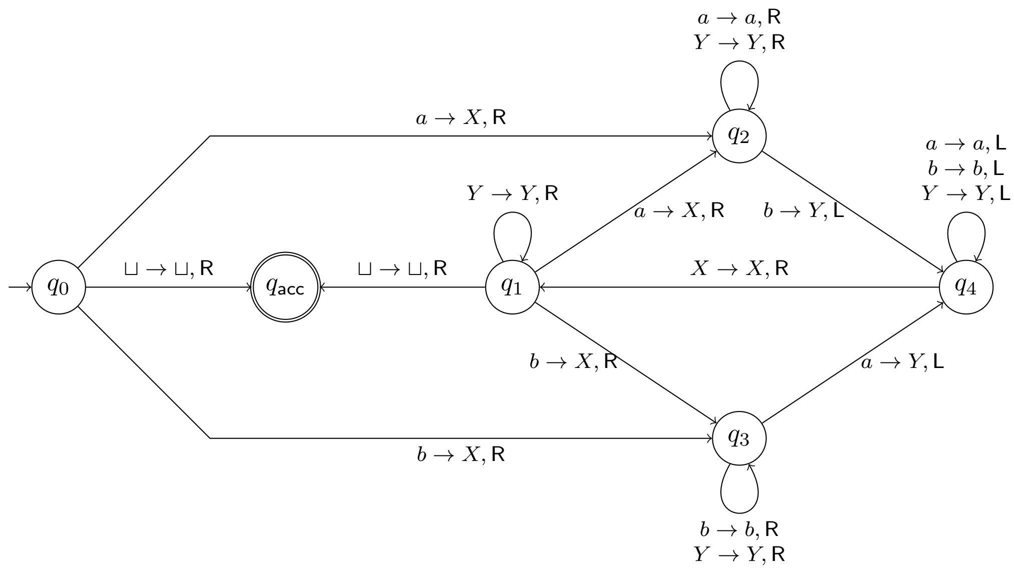
Solutions for Problem Set 8
CS 373: Theory of Computation

Assigned: April 4, 2013   Due on: April 11, 2013

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. **[5 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. **[2 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.

$$\begin{aligned} q_0baabab &\vdash Xq_3aabab \vdash q_4XYabab \vdash Xq_1Yabab \vdash XYq_1abab \vdash XYXq_2bab \vdash XYq_4XYab \\ &\vdash XYXq_1Yab \vdash XYXYq_1ab \vdash XYXYXq_2b \vdash XYXYq_4XY \vdash XYXYXq_1Y \\ &\vdash XYXYXYq_1\sqcup \vdash XYXYXY\sqcup q_{\mathsf{acc}}\sqcup \end{aligned}$$

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: Comprehension+Design+Proof] In this problem, we will show that for any Turing machine $M=(Q,\Sigma,\Gamma,\delta,q_0,q_{\mathsf{acc}},q_{\mathsf{rej}})$, there is a Type 0 grammar $G_M=(V,\Sigma,R,S)$ such that $\mathbf{L}(M)=\mathbf{L}(G_M)$. Thus, any problem that can be solved on a Turing machine, can be described using Type 0 grammars. The construction of the grammar G_M will be based on the intuition that the rules of G_M will "simulate M backwards". More precisely, derivations of G_M will have the form

$$S \overset{*}{\Rightarrow} \rhd u_1 q_{\mathsf{acc}} u_2 \lhd \overset{*}{\Rightarrow} \rhd \mathrm{C}_1 \lhd \overset{*}{\Rightarrow} \rhd \mathrm{C}_2 \lhd \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} \rhd \mathrm{C}_n \lhd \overset{*}{\Rightarrow} \rhd q_0 w \sqcup^i \lhd \overset{*}{\Rightarrow} w$$

where $q_0w \vdash \mathrm{C}_n \vdash \mathrm{C}_{n-1} \vdash \cdots \vdash \mathrm{C}_1 \vdash u_1q_{\mathsf{acc}}u_2$ is an accepting computation of M on the input w. $\rhd$ and $\lhd$ are special symbols used by the grammar G_M to "enclose" configurations. The variables of G_M will include $\{S,\rhd,\lhd\}\cup Q\cup(\Gamma\setminus\Sigma)$ and any additional variables that might be needed to describe the following types of derivations: first generate (any) accepting configuration from S; simulate (backwards) a single step of M; and produce the string w from the "initial configuration" $\rhd q_0 w\sqcup^i \lhd$ by erasing the unnecessary symbols.

1. Describe a set of rules that produce derivations of the form $S \overset{*}{\Rightarrow} \rhd\mathrm{C}\lhd$ where C is any accepting configuration of M. **[3 points]**

2. Describe a set of rules that simulate M backwards, i.e., have derivations $\rhd\mathrm{C}_1\lhd \overset{*}{\Rightarrow} \rhd\mathrm{C}_2\lhd$, where $config_1, \mathrm{C}_2$ are configurations of M such that $\mathrm{C}_2 \vdash \mathrm{C}_1$. **[4 points]**

3. Describe a set of rules that produce derivations of the form $\rhd q_0 w\sqcup^i\lhd \overset{*}{\Rightarrow} w$, by "erasing" the unnecessary symbols. **[3 points]**

Solution:

1. Recall that an accepting configuration has q_{acc} as the state, the head points somewhere on the tape, and the tape contains any sequence of symbols. We will have a variable A that will produce any string over the tape alphabet. Thus, we have the following rules

$$\{S \to \triangleright A q_{\mathsf{acc}} A \triangleleft\} \cup \{A \to aA \mid a \in \Gamma\} \cup \{A \to \epsilon\}$$

   Notice that all these rules are context-free.

2. To simulate M backwards, we will have rules to simulate each transition in M. For each transition of the form $\delta(q_1, a) = (q_2, b, R)$ (where $a, b \in \Gamma$), G_M has a rule $bq_2 \to q_1 a$. Next, for each transition of the form $\delta(q_1, a) = (q_2, b, L)$, we have a rule $\triangleright q_2 b \to \triangleright q_1 a$ (to handle the case when M executes this transition when the head is at the leftmost cell), and the rules $q_2 a' b \to a' q_1 a$ for each $a' \in \Gamma$. Thus, we have the following set of rules.

$$\{bq_2 \to q_1 a | \delta(q_1, a) = (q_2, b, R)\} \cup \{\triangleright q_2 b \to \triangleright q_1 a | \delta(q_1, a) = (q_2, b, L)\} \cup \{q_2 a' b \to a' q_1 a | \delta(q_1, a) = (q_2, b, L) \text{ and } a' \in \Gamma\}$$

3. We now describe the rules that will transform an initial configuration of M into the input string by "erasing" symbols.

$$\{\triangleright q_0 \to \epsilon,\ \sqcup\triangleleft \to \triangleleft,\ \triangleleft \to \epsilon\}$$

   Observe that we don't have a rule $\sqcup \to \epsilon$. This is because if we get to a string $\triangleright q_0 w \sqcup^i \triangleleft$, we only want to erase the $\sqcup$s at the right end; if there are any $\sqcup$ in the middle of w (or in fact, any symbol in $\Gamma \setminus \Sigma$) then "w" is not an input string, and after erasing $w \notin \Sigma^*$ and hence not in $\mathbf{L}(G_M)$.

■

Problem 3. [Category: Comprehension+Design+Proof] A *2-PDA* is a pushdown automaton that has *two stacks*. Thus, like a PDA, it is a nondeterministic machine. In each step, the current control state, the current input symbol read (which could be ϵ meaning nothing is read from the input), the symbol popped from the first stack (which could be ϵ, meaning that no symbol is popped), and the symbol popped from the second stack (which again could be ϵ), determine what the possible next state is, and the symbols to be pushed onto each of the two stacks (which could be ϵ meaning that no symbol is pushed) are. Additionally, like a PDA, the 2-PDA accepts an input if it reaches a accepting/final state after reading the entire input, irrespective of the contents of either of its two stacks.

1. Give the formal definition of a 2-PDA as a tuple, giving the domain and range of the transition function. **[2 points]**

2. Give the formal definitions of the instantaneous description of the 2-PDA, computation on a word, and the language accepted by the machine. **[3 points]**

3. Prove that if M is a (deterministic, single-tape) Turing machine then there is a 2-PDA P such that $\mathbf{L}(P) = \mathbf{L}(M)$. You only need to give the construction of the 2-PDA; you do not have to prove that your construction is correct. **[5 points]**

Solution:

1. A 2-PDA is $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q is a finite set of control states, Σ is the input alphabet, Γ is the stack alphabet, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final/accepting states. The transition function δ is given by

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \to 2^{Q \times (\Gamma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\})}$$

2. An instantaneous description of a 2-PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is a triple $\langle q, \sigma_1, \sigma_2\rangle$ where $q \in Q$, and $\sigma_1, \sigma_2 \in \Gamma^*$. Given input $w \in \Sigma^*$ and instantaneous descriptions $\langle q_1, \sigma_1, \tau_1\rangle$ and $\langle q_2, \sigma_2, \tau_2\rangle$, we say $\langle q_1, \sigma_1, \tau_1\rangle \xrightarrow{w}_P \langle q_2, \sigma_2, \tau_2\rangle$ iff there is a sequence of instantaneous descriptions $\langle r_0, s_0, t_0\rangle, \langle r_1, s_1, t_1\rangle, \ldots \langle r_k, s_k, t_k\rangle$ and $x_1, x_2, \ldots x_k$ where for each i $x_i \in \Sigma \cup \{\epsilon\}$, such that

   - $w = x_1 x_2 \cdots x_k$,
   - $r_0 = q_1$, $s_0 = \sigma_1$, and $t_0 = \tau_1$,
   - $r_k = q_2$, $s_k = \sigma_2$, and $t_k = \tau_k$, and
   - for every $i \in \{0, \ldots k-1\}$, there are $(r_{i+1}, b_1, b_2) \in \delta(r_i, x_{i+1}, a_1, a_2)$ and $s, t \in \Gamma^*$ such that $s_i = a_1 s$, $s_{i+1} = b_1 s$, $t_i = a_2 t$, and $t_{i+1} = a_2 t$, where $a_1, a_2, b_1, b_2 \in (\Gamma \cup \{\epsilon\})$.

   We say that P accepts $w \in \Sigma^*$ iff for some $q \in F$, $\sigma, \tau \in \Gamma^*$, $\langle q_0, \epsilon, \epsilon\rangle \xrightarrow{w}_P \langle q, \sigma, \tau\rangle$. The language recognized by a 2-PDA P is given by $\mathbf{L}(P) = \{w \in \Sigma^* \mid P \text{ accepts } w\}$.

3. Consider a Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$. To simulate M on a 2-PDA P, P will store the control state of M and the current symbol being read in its control state, the portion of M's tape to the left of the current head position in its first stack (in reverse order), and the portion of M's tape to the right of the head on the second stack. The two stacks will also have end-markers at the bottom of the stack so that P knows when it reaches the left and right end of M's tape. Thus, when M is in an instantaneous description $\alpha q a \beta$ (where $\alpha, \beta \in \Gamma^*$ and $a \in \Gamma$), P's configuration will be $\langle (q, a), \alpha^R\$, \beta\$\rangle$, where \$ is a bottom of stack symbol; we assume that the leftmost symbol on either stack is the top of the stack.

   The machine P will proceed in phases. First it will read the entire input and copy the input onto its stacks; it will nondeterministically guess when it has finished reading all the input symbols. All steps of P after this will be ϵ-transitions. Now, after the first phase, P will need to transfer symbols from its first stack to the second stack so that it can move to a configuration where it is simulating the initial instaneous description of M. Once it completes this second phase, it will start simulating the steps of M.

   Formally, $P = (Q', \Sigma, \Gamma', \delta', q'_0, F)$ where

   - $Q' = (Q \times \Gamma) \cup \{I, C, R\}$. I is the initial state, C is first copy phase state, R is the second reverse phase; otherwise, the state of P stores the current state of M and the current symbol being read.
   - $\Gamma' = \Gamma \cup \{\$\}$, where $\$ \notin \Gamma$; stack alphabet has a bottom of stack symbol \$ in addition to the symbols that M can write on its tape.
   - $q'_0 = I$
   - $F = \{(q_{\mathsf{acc}}, a) \mid a \in \Gamma\}$
   - δ' is given as follows.

   $$\begin{array}{ll}
   \delta'(I, \epsilon, \epsilon, \epsilon) = \{(C, \$, \$)\} & \\
   \delta'(C, a, \epsilon, \epsilon) = \{(C, a, \epsilon)\} & \text{if } a \in \Sigma \\
   \delta'(C, \epsilon, \epsilon, \epsilon) = \{(R, \epsilon, \epsilon)\} & \\
   \delta'(R, \epsilon, a, \epsilon) = \{(R, \epsilon, a)\} & \text{if } a \in \Sigma \\
   \delta'(R, \epsilon, \$, a) = \{((q_0, a), \$, \epsilon)\} & \text{if } a \in \Sigma \\
   \delta'(R, \epsilon, \$, \$) = \{((q_0, \sqcup), \$, \$)\} & \\
   \delta'((q, a), \epsilon, b, \epsilon) = \{((q', b), \epsilon, c)\} & \text{if } b \in \Gamma, \text{ and } \delta(q, a) = (q', c, \mathsf{L}) \\
   \delta'((q, a), \epsilon, \$, \epsilon) = \{((q', c), \$, \epsilon)\} & \text{if } \delta(q, a) = (q', c, \mathsf{L}) \\
   \delta'((q, a), \epsilon, \epsilon, b) = \{((q', b), c, \epsilon)\} & \text{if } b \in \Gamma \text{ and } \delta(q, a) = (q', c, \mathsf{R}) \\
   \delta'((q, a), \epsilon, \epsilon, \$) = \{((q', \sqcup), c, \$)\} & \text{if } \delta(q, a) = (q', c, \mathsf{R})
   \end{array}$$

   and $\delta'(p, a, b, c) = \emptyset$ in all cases not covered above.

■