Solutions for Problem Set 6
CS 373: Theory of Computation

Assigned: February 28, 2013 Due on: March 7, 2013

Problem 1. [Category: Design+Proof] Design a context-free grammar for the language $L = \{a^i b^j \mid 2i \leq j \leq 3i,\ i, j \in \mathbb{N}\}$. Provide a formal proof that your construction is correct. *Hint:* Build a grammar for the case when $j = 2i$ and $j = 3i$, and think of a way to fuse the two together. **[10 points]**

Solution: Consider the grammar $G = (\{S\}, \{a, b\}, R, S)$, where the rules R are given as follows.

$$S \to aSbb \mid aSbbb \mid \epsilon$$

Let us now prove that this grammar is correct, i.e., $\mathbf{L}(G) = L$.

$(L \subseteq \mathbf{L}(G))$: Consider $w \in L$. We will show that $S \stackrel{*}{\Rightarrow} w$, by induction on the length of w. For the base case, when $|w| = 0$, observe that since $S \to \epsilon \in R$, $S \Rightarrow \epsilon = w$. Assume that for any $w \in L$ such that $|w| < n$, we have that $S \stackrel{*}{\Rightarrow} w$. Consider $w = a^i b^j \in L$ of length $i + j = n$. Observe that since $2i \leq j \leq 3i$, we have that either $2(i-1) \leq j - 2 \leq 3(i-1)$ or $2(i-1) \leq j - 3 \leq 3(i-1)$. We will consider these two cases in order.

Suppose $2(i-1) \leq j-2 \leq 3(i-1)$. Then, $w' = a^{i-1}b^{j-2} \in L$ and by induction hypothesis, we have $S \stackrel{*}{\Rightarrow} w'$. Then we have the following derivation of w from S: $S \Rightarrow aSbb \stackrel{*}{\Rightarrow} aw'bb = w$.

Suppose $2(i-1) \leq j-3 \leq 3(i-1)$. Then $w' = a^{i-1}b^{j-3} \in L$ and by induction hyposthesis $S \stackrel{*}{\Rightarrow} w'$. The derivation of w then is given by $S \Rightarrow aSbbb \stackrel{*}{\Rightarrow} aw'bbb = w$.

$(\mathbf{L}(G) \subseteq L)$: Suppose $S \stackrel{*}{\Rightarrow} w$ then we will show that $w \in L$ by induction on the number of steps in the derivation of w. For the base case observe that if $S \Rightarrow w$ (in one step) then w must be ϵ and $w \in L$. Assume that if w is derived in $n-1$ steps from S then $w \in L$. Consider a string w that is derived from S in n steps. Now the derivation of w is in one of two forms: either $S \Rightarrow aSbb \stackrel{*}{\Rightarrow} aw'bb = w$ or $S \Rightarrow aSbbb \stackrel{*}{\Rightarrow} aw'bbb = w$. In either case, observe that $S \stackrel{*}{\Rightarrow} w'$ in $n-1$ steps and hence by induction hypothesis $w' = a^i b^j$, where $2i \leq j \leq 3i$. Notice that $aw'bb = a^{i+1}b^{j+2}$, while $aw'bbb = a^{i+1}b^{j+3}$. Moreover, since $2i \leq j \leq 3i$, $2(i+1) \leq j+2 \leq 3(i+1)$ and $2(i+1) \leq j+3 \leq 3(i+1)$, and so in either case $w \in L$. ■

Problem 2. [Category: Comprehension+Design] Let $G = (V, \Sigma, R, \langle \text{STMT} \rangle)$ be the following grammar

$$\begin{aligned}
\langle \text{STMT} \rangle &\longrightarrow \langle \text{ASSIGN} \rangle \mid \langle \text{IF-THEN} \rangle \mid \langle \text{IF-THEN-ELSE} \rangle \\
\langle \text{IF-THEN} \rangle &\longrightarrow \texttt{if condition then } \langle \text{STMT} \rangle \\
\langle \text{IF-THEN-ELSE} \rangle &\longrightarrow \texttt{ if condition then } \langle \text{STMT} \rangle \texttt{ else } \langle \text{STMT} \rangle \\
\langle \text{assign} \rangle &\longrightarrow \texttt{a := 1}
\end{aligned}$$

where $\Sigma = \{\texttt{if}, \texttt{then}, \texttt{else}, \texttt{condition}, \texttt{a := 1}\}$ and $V = \{\langle \text{STMT} \rangle, \langle \text{IF-THEN} \rangle, \langle \text{IF-THEN-ELSE} \rangle, \langle \text{ASSIGN} \rangle\}$. G is a natural looking grammar for a fragment of a programming language, but G is ambiguous.

1. Show that G is ambiguous. **[5 points]**
2. Give a new unambiguous grammar for the same language. You need not prove that your grammar is correct but explain your construction. You may want to look at examples in Lecture 12. **[5 points]**

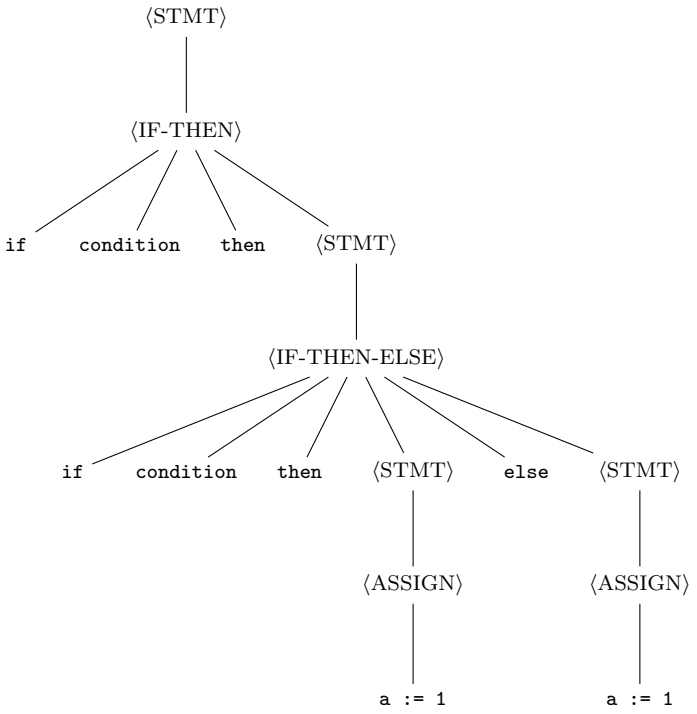


Figure 1: First parse tree

Solution:

1. The grammar is ambiguous because the string "`if condition then if condition then a := 1 else a := 1`" has the two parse trees shown in Figures 1 and 2.

2. The grammar is ambiguous because it is not clear whether the `else` should bind to the innermost `if` or the outermost one. One natural semantics that is often taken in most programming languages is that the `else` should bind to the closest `if`. So for the string "`if condition then if condition then a := 1 else a := 1`", our new grammar will only allow the parse tree given in Figure 1. We will ensure this by making sure that the statement in the body of the `then` is always "matched". The new grammar is $G' = (V', \Sigma, R', \langle \text{STMT} \rangle)$, where $V' = \{\langle \text{STMT} \rangle, \langle \text{MTCH} \rangle, \langle \text{UNMTCH} \rangle, \langle \text{ASSIGN} \rangle\}$, $\Sigma = \{\texttt{if}, \texttt{condition}, \texttt{then}, \texttt{else}, \texttt{a := 1}\}$, and the rules R' are given by

$$\begin{aligned}
\langle \text{STMT} \rangle &\to \langle \text{ASSIGN} \rangle \mid \langle \text{MTCH} \rangle \mid \langle \text{UNMTCH} \rangle \\
\langle \text{MTCH} \rangle &\to \langle \text{ASSIGN} \rangle \mid \texttt{if condition then } \langle \text{MTCH} \rangle \texttt{ else } \langle \text{MTCH} \rangle \\
\langle \text{UNMTCH} \rangle &\to \texttt{if condition then } \langle STMT \rangle \mid \texttt{if condition then } \langle \text{MTCH} \rangle \texttt{ else } \langle \text{UNMTCH} \rangle \\
\langle \text{ASSIGN} \rangle &\to \texttt{a := 1}
\end{aligned}$$

■

Problem 3. [Category: Comprehension] Consider the PDA P over the input alphabet $\{0, 1, \#\}$ shown in the figure below; a, in the transitions below, is either 0 or 1.

1. Write the formal description of the PDA P listing the states, stack alphabet, transition function, initial state and final states. **[5 points]**

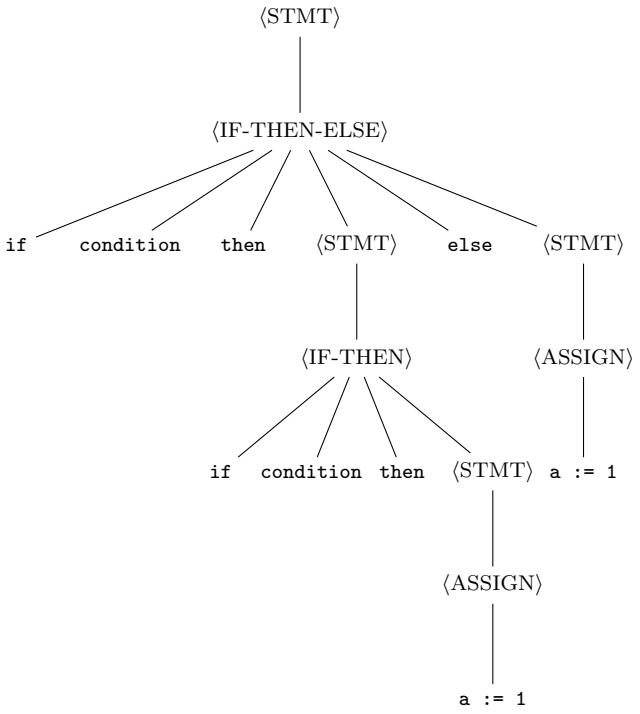
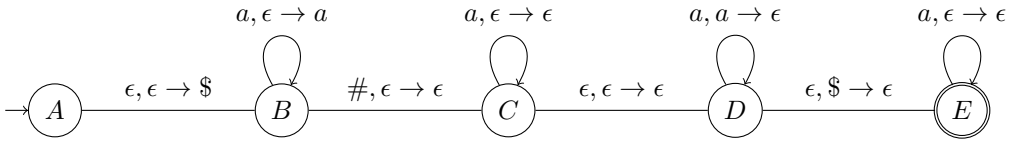


Figure 2: Second Parse Tree



2. For each of the following strings either show that they are accepted by P by describing an accepting computation, or show that they are not accepted by showing the *entire* computation tree on the input: 01#10, 01#01, 01#111000. **[3 points]**

3. Describe the language recognized by the PDA P. Give an informal justification for your answer, by explaining how the PDA works. **[2 points]**

Solution:

1. The PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ where

   - $Q = \{A, B, C, D, E\}$
   - $\Sigma = \{0, 1\}$
   - $\Gamma = \{\$, 0, 1\}$
   - $q_0 = A$
   - $F = \{E\}$

- And δ is given by

$$\begin{array}{llll}\delta(A,\epsilon,\epsilon)=\{(B,\$)\} & \delta(B,0,\epsilon)=\{(B,0)\} & \delta(B,1,\epsilon)=\{(B,1)\} & \delta(B,\#,\epsilon)=\{(C,\epsilon)\}\\ \delta(C,0,\epsilon)=\{(C,\epsilon)\} & \delta(C,1,\epsilon)=\{(C,\epsilon)\} & \delta(C,\epsilon,\epsilon)=\{(D,\epsilon)\} & \delta(D,0,0)=\{(D,\epsilon)\}\\ \delta(D,1,1)=\{(D,\epsilon)\} & \delta(D,\epsilon,\$)=\{(E,\epsilon)\} & \delta(E,0,\epsilon)=\{(E,\epsilon)\} & \delta(E,1,\epsilon)=\{(E,\epsilon)\}\end{array}$$

  and in all other cases, $\delta(q,a,x)=\emptyset$.

2. 01#10 is accepted by the following computation

$$(A,\epsilon)\xrightarrow{\epsilon}(B,\$)\xrightarrow{0}(B,0\$)\xrightarrow{1}(B,10\$)\xrightarrow{\#}(C,10\$)\xrightarrow{\epsilon}(D,10\$)\xrightarrow{1}(D,0\$)\xrightarrow{0}(D,\$)\xrightarrow{\epsilon}(E,\epsilon)$$

   01#01 is not accepted by P as can be seen from the following computation tree.

$$\begin{array}{lcl}(A,\epsilon) & \xrightarrow{\epsilon} & (B,\$)\\ 0\big\downarrow & & \big\downarrow 0\\ \text{X} & & (B,0\$)\\ & & \big\downarrow 1\\ & & (B,10\$)\\ & & \big\downarrow \#\\ (D,10\$) & \xleftarrow{\epsilon} & (C,10\$)\\ 0\big\downarrow & & \big\downarrow 0\\ \text{X} & & (C,10\$)\xrightarrow{\epsilon}(D,10\$)\\ & & \big\downarrow 1 \qquad\quad \big\downarrow 1\\ (D,10\$) & \xleftarrow{\epsilon} & (C,10\$) \qquad (D,0\$)\end{array}$$

   01#111000 is accepted by the following computation

$$(A,\epsilon)\xrightarrow{\epsilon}(B,\$)\xrightarrow{0}(B,0\$)\xrightarrow{1}(B,10\$)\xrightarrow{\#}(C,10\$)\xrightarrow{1}(C,10\$)\xrightarrow{1}(C,10\$)\xrightarrow{\epsilon}(D,10\$)\xrightarrow{1}$$
$$(D,0\$)\xrightarrow{0}(D,\$)\xrightarrow{\epsilon}(E,\epsilon)\xrightarrow{0}(E,\epsilon)\xrightarrow{0}(E,\epsilon)$$

3. $\mathbf{L}(P)=\{x\#ux^Rv \mid x,u,v\in\{0,1\}^*\}$. This can be seen as follows. P first pushes a bottom of stack symbol \$, and then pushes the symbols of x as it reads them. When it reads # it moves to state C, where it just reads symbols and non-deterministically guesses when "x^R" is going to start by moving to state D. In state D, it reads symbols while matching the symbols popped from the stack to make sure that x^R is seen, and then when the bottom of stack symbol \$ is seen, it moves to state E, where it reads the rest of the input and accepts.

■