Homework 3

Student Generated Solutions

April 17, 2006

2.4

a. {w| w contains at least three 1s }

$S \rightarrow R1R1R1R$
$R \rightarrow 0R|1R|\epsilon$

b. {w| starts and ends with the same symbol }

$S \rightarrow 0S0|1S1|\epsilon$

c. {w| the length of w is odd }

$S \rightarrow 0|1|0S1|1S0|0S0|1S1$

d. {w| the length of w is odd and its middle symbol is a 0 }

$S \rightarrow 0|0S0|0S1|1S0|1S1$

e. {w = w^R, that is, w is a palindrome }

$S \rightarrow 0|1|0S0|1S1|\epsilon$

f. The empty set

$S \rightarrow S$

2.10

The PDA could immediately branch in two directions without consuming input and putting a $ on the stack so we know its top. Acceptance in either branch will make the machine be in the accept state.

1. Branch one

   Read the 'a' symbols and push them onto the stack. As b's are read pop one 'a' for each 'b' read. If reading the 'b's finishes exactly with a $ on the stack, then as long as the machine reads only c's accept the input.

2. Branch two
   Read the 'a' symbols but do not write them onto the stack. Write the 'b' symbols onto to the stack as they are read. Ignore b's. When the 'c' symbols are seen, pop one 'b' for each 'c' read. If the stack is the $ sign at the same time as the c's are done reading, then accept the string.

2.14

Convert the following CFG into an equivalent CFG in Chomsky normal form using the procedure given in Theorem 2.9.

$$A \to BAB|B|\epsilon$$
$$B \to 00|\epsilon$$

Solution
Step1: Add a new start variable S_0

$$S_0 \to A$$
$$A \to BAB|B|\epsilon$$
$$B \to 00|\epsilon$$

Step 2: Remove all the ϵ rules

1. Remove A $\rightarrow \epsilon$

$$S_0 \rightarrow A|\epsilon$$
$$A \rightarrow BAB|BB|B$$
$$B \rightarrow 00|\epsilon$$

2. Remove B $\rightarrow \epsilon$

$$S_0 \rightarrow A|\epsilon$$
$$A \rightarrow BAB|BA|AB|A|BB|B$$
$$B \rightarrow 00$$

Step 3-2: Remove unit rules

1. Remove A $\rightarrow$ A

$$S_0 \rightarrow A|\epsilon$$
$$A \rightarrow BAB|BA|AB|BB|B$$
$$B \rightarrow 00$$

2. Remove A $\rightarrow$ B

$$S_0 \rightarrow A|\epsilon$$
$$A \rightarrow BAB|BA|AB|BB|00$$
$$B \rightarrow 00$$

3. Remove $S_0 \rightarrow$ A

$$S_0 \rightarrow BAB|BA|AB|BB|00|\epsilon$$
$$A \rightarrow BAB|BA|AB|BB|00$$
$$B \rightarrow 00$$

Step4: Converting the remaining rules into the proper form by adding additional variables and rules, the grammar below is in Chomsky normal equivalent to the CFG given.

$$S_0 \to BA_1|BA|AB|BB|00|\epsilon$$
$$A \to BA_1|BA|AB|BB|00$$
$$A_1 \to AB$$
$$B \to 00$$

Assuming that 0 is the member of the terminal set rather than 00, the CFG is as following:

$$S_0 \to BA_1|BA|AB|BB|UU|\epsilon$$
$$A \to BA_1|BA|AB|BB|UU$$
$$A_1 \to AB$$
$$U \to 0$$
$$B \to UU$$

2.27

Let $G = (V, \Sigma, R\,\langle STMT\rangle)$ be the following grammer.

$$\langle STMT\rangle \to \langle ASSIGN\rangle \,|\, \langle IF-THEN\rangle \,|\, \langle IF-THEN-ELSE\rangle$$
$$\langle IF-THEN\rangle \to \text{if condition then } \langle STMT\rangle$$
$$\langle IF-THEN-ELSE\rangle \to \text{if condition then } \langle STMT\rangle\, else\, \langle STMT\rangle$$
$$\langle ASSIGN\rangle \to a := 1$$

G is a natural-looking grammer for a fragment of a programming language but G is ambiguous.

$$\Sigma = \{if, condition, then, else, a := 1\}$$
$$V = \{\langle STMT\rangle, \langle IF-THEN\rangle, \langle IF-THEN-ELSE\rangle, \langle ASSIGN\rangle\}$$

a. Show that G is ambiguous

G is ambigious if it can be shown that a string from this grammar is derived ambiguously - that is it can be derived by two or more different leftmost derivations(or parse trees).

The following are the two leftmost derivations for the string :

"if condition then if condition then $a := 1$ else $a := 1$ "

1. First leftmost derivation.

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

2. Second leftmost derivation.

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

b. Give a new unambiguous grammar for the same language.

$$\begin{aligned}
\langle STMT\rangle &\rightarrow \langle ASSIGN\rangle | \langle IF-THEN\rangle \\
\langle IF-THEN\rangle &\rightarrow \text{if condition then } \langle ASSIGN-ELSE\rangle \\
\langle IF-THEN\rangle &\rightarrow \text{if condition then } \langle IF-THEN\rangle
\end{aligned}$$

$$\begin{aligned} \langle ASSIGN - ELSE \rangle &\rightarrow \langle ASSIGN \rangle | \langle ASSIGN \rangle \text{ else } \langle IF - THEN \rangle \\ \langle ASSIGN \rangle &\rightarrow a := 1 \end{aligned}$$

This will be unambiguous because $\langle IF - THEN \rangle$ when it goes to $\langle ASSIGN - ELSE \rangle$ forces one to have the string $a := 1$ else ; whereas, when it goes to $\langle IF - THEN \rangle$ one gets immediately another copy of if condition then . No other rule was causing ambiguity in the original grammar. One can show by induction of the complexity of the derivations of strings generated by the two grammars that they have the same strings.

2.30

In each of the following cases, we assume that the specified language is a CFL and contradict the pumping lemma concept.

a. $\{0^n1^n0^n1^n | n \geq 0\}$
Let p be the pumping length and consider the string $0^p1^p0^p1^p$. By the pumping lemma there should be $uvxyz$ with $|vy| > 0$, $|vxy| \leq p$, such that for all i, uv^ixy^i z belongs to A. If vxy involves two characters then it must be of the form 0^k1^j since vxy has length less than p. But if this is the case, then pumping down would produce a string not in the language. If vxy involves only 0's or $1'$, then again since $|vxy| \leq p$, these 0's or 1's come from the same block, so pumping down produces a string not in the language.

b. and c. Solutions to these are in the book.

d. $\{t_1\#t_2\#.....t_k | k \geq 0$, each $t_i \in \{a, b\}^*$ and $t_i = t_j$ for some $i \neq j$ $\}$
let p be the pumping length and s= $a^pb^p\#a^pb^p$
We show that s = uvxyz cannot be pumped.
cannot be a part of v and y because it then does uv^0xy^0z is not in the language.
If both v and y are nonempty and occur both on the left-hand side or the right-hand side of of the #, then uv^2xy^2z will not be in the language because it is longer on one side . .

If both v and y were non-empty and straddled the #, then by the third condition $v = b^j$ and $y = a^k$ for some j and k less than p but then pumping down produces a string not in the language.

3.2

Sequence of configurations that the Turing Machine M_1 enetrs from Example 3.9

a. 11
$q_1 11, Xq_3 1, X1q_3 \sqcup, X1 \sqcup q_{reject}$

b. 1#1
$q_1 1\#1$
$Xq_3\#1$
$X\#q_5 1$
$X\#Xq_6$
$X\#q_6 X$
$Xq_7\#X$
$Xq_1\#X$
$X\#q_8 X$
$X\#Xq_8 \sqcup$
$X\#X \sqcup q_{accept}$

c. 1##1
$q_1 1\#\#1$
$Xq_3\#\#1$
$X\#q_5\#1$
Xq_{reject}
note:(In state q_5 reading a # results in a or goes to a reject state because in q_5 no outgoing transition arrow with # is present.

d. 10#11
$q_1 10\#11$

$Xq_3 0\#11$
$X0q_3\#11$
$X0\#q_5 11$
$X0q_6\#X1$
$Xq_7 0\#X1$
$q_7 X0\#X1$
$Xq_1 0\#X1$
$XXq_2\#X1$
$XX\#q_4 X1$
$XX\#q_4 1$
$XX\#Xq_{reject}$
note:(In state q_4 and upoon reading a 1 results in a reject state because there are no outgoing transition arrow with 1 in q_4

e. $10\#10$
$q_1 10\#10$
$X0q_3\#10$
$X0\#q_5 10$
$X0q_6\#X0$
$Xq_7 0\#X0$
$q_7 X0\#X0$
$Xq_1 0\#X0$
$XXq_2\#0$
$XX\#q_4 0$
$XXq_6\#X \sqcup$
$Xq_7 X\#X \sqcup$
$XXq_1\#X \sqcup$
$XX\#Xq_8 X \sqcup$
$XX\#Xq_8 \sqcup$
$XX\#X \sqcup q_{accept}$