ArsDigita University
Month 8: Theory of Computation
Professor Shai Simonson

Problem Set 3

1. Construct non-deterministic pushdown automata to accept the following languages. Please comment and explain your machines.

a. $\{1^n 0^n \mid n > 0\}$
b. $\{0^n 1^{2n} \mid n >= 0\}$
c. $\{1^n 0^n \mid n > 0\} \cup \{0^n 1^{2n} \mid n >= 0\}$
d. (0+1)* - {ww | w in {0,1}*} (This is the complement of ww)

2. Construct Deterministic Pushdown Automata to accept the following languages. Again include comments.

a. $\{10^n 1^n \mid n > 0\} \cup \{110^n 1^{2n} \mid n > 0\}$.
b. Binary strings that contain an equal number of 0's and 1's.
c. Binary strings with twice as many ones as zeros.
d. Binary strings that start and end with the same symbol, and have the same number of zeros as ones.

3. Construct Context Free Grammars to accept the following Languages over {0,1}. Explain your grammars.

e. (Text: 2.4 b) The language {w | w starts and ends with the same symbol}.
f. (Text: 2.4 c) The language {w | the length of w is odd}.
g. (Text: 2.4 d) The language {w | the length of w is odd and its middle symbol is a zero}.
h. $\{0^n 1^n \mid n > 0\} \cup \{0^n 1^{2n} \mid n > 0\}$
i. $\{0^i 1^j 2^k \mid i \neq j \text{ or } j \neq k\}$.
j. Binary strings with twice as many ones as zeros.

4. Ambiguity

a. Explain why the grammar below that generates strings with an equal number of 0's and 1's is ambiguous.

$S \rightarrow 0A \mid 1B$
$A \rightarrow 0AA \mid 1S \mid 1$
$B \rightarrow 1BB \mid 0S \mid 0$

b. **Extra Credit:** Construct an unambiguous grammar that generates the same set of strings.

c. (Text: 2.21 a) Let G = (V, Sigma, R , <STMNT>) be the following grammar:

<STMT> → <ASSIGN> | <IF-THEN> | <IF-THEN-ELSE>|<BEGIN-END>
IF-THEN> → if condition then <STMT>
IF-THEN-ELSE> → if condition then <STMT> else <STMT>
<BEGIN-END> → begin <STMT-LIST> end
<STMT-LIST> → <STMT-LIST> <STMT> | <STMT>
<ASSIGN> → a := 1

Sigma = { if, condition, then, else, begin, end, a := 1 }
V = {<STMT>, <IF-THEN>, <IF-THEN-ELSE>, <BEGIN-END>,
<STMT-LIST>, <ASSIGN>}

G is a natural looking grammar for a fragment of a programming language, but G is ambiguous. Demonstrate that G is ambiguous.

d. **Extra Credit:** (Text: 2.21 b) Give a new unambiguous grammar for the same language.

5. Converting to Normal Form.

a. Put the following grammar into Chomsky Normal Form. Show all work.

S → A | AB0 | A1A
B → B1 | BC
A → A0 | ε
C → CB | CA | 1B

b. Convert the following grammar to an equivalent one with no unit productions and no useless symbols. Show that the original grammar had NO useless symbols. What useless symbols are there after getting rid of unit productions?

S → A | CB
A → C | D
B → 1B | 1
C → 0C | 0
D → 2D | 2

6. Derivations in Chomsky Normal Form

a. (Text: 2.19) Show that, if G is a CFG in Chomsky normal form, then for any string w in L(G) of length n >=1, exactly 2n-1 steps are required for any derivation of w.

b. (Text: 2.20) Let G be a CFG in Chomsky normal form that contains b variables. Show that, if G generates some string using a derivation with at least 2^b steps, then L(G) is infinite.

7. Two-Way PDA's

A 2-way PDA is a machine that is just like a deterministic PDA except that it can move either left or right on seeing a particular symbol in a particular state. It accepts if it moves off the right end of the input in a final state.

Show that the set $\{0^n1^n0^n \mid n > 0\}$ is accepted by a 2-way PDA. You may assume that there is a # symbol at the end of the string and a $ symbol at the beginning to mark the end points. To draw a 2-way PDA, just add R (right) or L (left) to each transition (so each arrow now will have an input symbol, a stack symbol, a push/pop command and an L or an R).

(Note that the set above is *not* a CFL).