COT 4210 Quiz #2 Part A: Context Free Grammars 2/23/2021 Solution

1) (9 pts) Below is the formal description of a DFA which describes a language L. Use the algorithm shown in class to convert a DFA to an equivalent context free grammar to produce a CFG that accepts L.

$Q = \{a, b, c, d\}$ $\Sigma = \{0, 1\}$ $q_0 = a$ $F = \{b, d\}$

δ State\Character	0	1
a	b	d
b	a	c
c	d	b
d	c	a

Clearly designate your CFG formally, listing each of the components in the formal designation. Please make your variable names easy to follow such that the correspondence to the DFA is clear.

Solution

Here is the formal designation for the grammar:

$\Sigma = \{0, 1\}$ (terminals)
$V = \{A, B, C, D\}$ (variables)
$S = A$ (start symbol)

(Rules)
$A \rightarrow 0B \mid 1D$
$B \rightarrow 0A \mid 1C \mid \varepsilon$
$C \rightarrow 0D \mid 1B$
$D \rightarrow 0C \mid 1A \mid \varepsilon$

Grading: 2 pts for stating variables, 1 pt for clearly indicating the start symbol, 1 pt for the rules for A and C, 2 pts for the rules for B and D.

2) (9 pts) The following grammar over the alphabet {a, b} below is in the process of being converted into Chomsky Normal Form. A new start symbol, S' has been added. The next step is to remove all epsilon rules. Show the resulting grammar when all epsilon rules have been removed. To save time, feel free to copy and paste all of the old rules and then just add the new rules at the end. (Note: the variables in the grammar currently are S', S, T, U, V

S' → S
S → aS | Sb | TUV
T → TT | TU | VT | ε
U → bUb | bVVa | bVUVb
V → aab | bVa | ε

Solution
First let's deal with removing V going to epsilon:

S' → S
S → aS | Sb | TUV | TU
T → TT | TU | VT | ε
U → bUb | bVVa | bVUVb | bVa | ba | bUVb | bVUb | bUb
V → aab | bVa | ba

Now, let's remove the T to epsilon rule:

S' → S
S → aS | Sb | TUV | TU | UV | U
T → TT | TU | VT | U | V
U → bUb | bVVa | bVUVb | bVa | ba | bUVb | bVUb | bUb
V → aab | bVa | ba

Grading: 6 pts total for the addition of the yellow rules for V, 3 pts total for the addition of the rules for T. If extra rules are added that shouldn't be, just take 1 or 2 pts off based on the severity of the error. If not enough rules are added, take off an integer number of points proportionally.

3) (7 pts) Design a context grammar over the alphabet { (,) } representing the set of strings of matching parentheses. Use the usual notion of matching parentheses, namely, each open parenthesis has a matching close parenthesis and inside of each pair of matching parentheses any valid set of matching parentheses are allows. Examples of strings in the language are: (), ()(), (()), (()()(())) and ε. Examples of strings that aren't matching parentheses are: ())(, ((), and ()()())(). In the first example, the prefix of the first three characters has more close parentheses than open parentheses. In the second example, the full string has more open parentheses than close parentheses and in the last example, the prefix of the first seven characters has more close parentheses than open parentheses. Clearly designate your CFG formally, listing each of the components in the formal designation.

Solution

$\Sigma = \{(,)\}$
$V = \{T\}$
$S = T$

$T \rightarrow (T) \mid TT \mid \varepsilon$

Intuitively, we can have a starting and ending parenthesis and any valid expression inside of that, which we can express with T. Alternatively, we can have 2 separate expressions next to each other. (If there are several, we can just use the TT rule multiple times to generate as many Ts in a row as we need.) Finally, when there are no more parentheses to fill in we must use epsilon.

Grading: Full credit for any valid answer. Break down by points:

1 pt listing variables
1 pt stating start variable
5 pts rules

Give partial for each portion as necessary.