CSE340 FALL 2020 - HOMEWORK 2

Due: Monday September 28, 2020 by 11:59 PM on Canvas

All submissions should be typed, no exceptions.

1. Consider the grammar.

S → C B b	(1)
A → a A	(2)
A → ε	(3)
B → d B e	(4)
B → & A B C !	(5)
B→ C	(6)
C → c C a	(7)
C → ε	(8)

where S, A, B, C and D are the non-terminals, S is the start symbol, and a, b , c , f , g , '&' and '!' are the terminals.

1.1. **FIRST sets**. Do the following

1.1.1. Do an initialization pass by applying FIRST sets rules I and II. Show the resulting FIRST sets

1.1.2. Do one pass on the grammar rules in the order in which they are listed as follows. For each grammar rules, apply FIRST set rule III, then apply FIRST set rule IV then apply FIRST set rule V. Show the resulting FIRST sets after this one pass.

1.1.3. Show the final result of the calculation of the FIRST sets

1.2. **FOLLOW sets**. Do the following

1.2.1. Do an initialization pass by applying FOLLOW set rules I. Then do one pass by applying FOLLOW sets rules IV, and V. Show the resulting FOLLOW sets

1.2.2. Do one pass on the grammar rules in the order in which they are listed as follows. For each grammar rules, apply FOLLOW set rule II, then apply FOLLOW set rule III. Show the resulting FOLLOW sets after this one pass.

1.2.3. Show the final result of the calculation of the FOLLOW sets

2. Consider the grammar

   S → A B | C D

   A → a A b | ε

   B → D C | A

   C → b B c | ε

   D → b | ε

   Show that this grammar does not have a recursive descent predictive parser. To get full credit you should show all the conditions of predictive parsing that are not satisfied by this grammar. You do not need to show the conditions of predictive parsing that are satisfied. You will need to calculate FIRST and FOLLOW sets but you do not need to show how you calculated them . You will need to explicitly show which conditions of predictive parsing are not satisfied.

3. Consider the grammar

   S → A c B | e C A

   A → a | B D

   B → b B | ε

   C → c C | ε

   D→ d D | ε

   3.1. Show that the grammar has a predictive recursive descent parser. You should show that the conditions of predictive parsing apply for every non-terminal.

   3.2. Write parse_input()

   3.3. Write parse_S()

   3.4. Write parse_A().

   3.5. Give a **full execution** trace for your parser from part 3.2 above on input c b d c

   Your parser should follow the general model of predictive parser that we saw in class. In particular, for non-terminals that can generate ε, the parser should check the FOLLOW set before choosing to parse the righthand side that generates ε.

4. We say that a symbol of a grammar is useless if it does not appear in the derivation of a string (possibly empty) of terminals. Formally, *A is useful* if there exists a derivation $S \Rightarrow^* x\ A\ y \Rightarrow^* w$ where w is a string of terminals. *A is useless* if it is not useful. Note that a useful symbol can be either a terminal or a non-terminal. Also, a useless symbol can be a terminal or a non-terminal.

   Consider the grammar

   S → A b B | D c A

   A → a B

   B → b A | A E

   E → e D | ε

$C \rightarrow c\ A \mid E$

$D \rightarrow d\ D \mid A\ C\ D$

Which are the useful symbols of this grammar? For every useful symbol, give a derivation that shows that the symbol is useful.