CSE340 FALL 2021 - HOMEWORK 2

Due: Friday October 1, 2021 by 11:59 PM on Canvas

All submissions should be typed, no exceptions.

1. Consider the grammar.

S → A B C D (1)

A → A B & (2)

A → C D (3)

C → A c (4)

C → ε (5)

B → b B ! (6)

B → C (7)

D → d D (8)

D → ε (9)

where S, A, B, C and D are the non-terminals, S is the start symbol, and b, c , d ‘&’ and ‘!’ are the terminals.

* 1. **FIRST sets**. Do the following
     1. Do an initialization pass by applying FIRST sets rules I and II. Show the resulting FIRST sets
     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.
     3. Do a second 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.
     4. Show the final result of the calculation of the FIRST sets
  2. **FOLLOW sets**. Do the following
     1. Do an initialization pass by applying FOLLOW set rules I. Then do one pass by applying FOLLOW sets rules IV, and V (adding FIRST to FOLLOW). Show the resulting FOLLOW sets
     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.
     3. Do a second 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.
     4. Show the final result of the calculation of the FOLLOW sets

See homework 2 from last semester for a similar problem

1. Consider the grammar

S → a S b | c

A → C S | C S E

C → D B | B

B → a C a | ε

D → d | ε

Show that this grammar does not have a recursive descent predictive parser. To get full credit you should show, **for each non-terminal**, all the conditions of predictive parsing that are not satisfied by the rules of the non-terminal. It is not enough to show that for some non-terminal the rules are not satisfied.

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.

1. Consider the grammar

S → s S t | A k B C

A → f A g | E D

B → b B | g

C → c | E

D → d D | ε

E → e E | ε

* 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.
  2. Write parse\_input()
  3. Write parse\_S()
  4. Write parse\_A().
  5. Give a **full execution** trace for your parser from part 3.2 above on input s s f g k g t t

trace should show the correct sequence of calls including calls to functions that you were not asked to write in parts 3.2, 3.3 and 3.4 above. The trace should show the sequence of function calls like parse\_X() where X is a non-terminal and expect(ttype), where ttype is a token type.

Your parse functions 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.

1. 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 ⇒\* x A y ⇒\* 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.

**Example**. S → S A B | a B C

A → A S | B E

B → a B | A | b

C → c | E

F → c | E

E → e E

1. F is useless because there is no derivation starting with S in which F appears.
2. E is useless because it cannot derive a string of terminals, so there can be no derivation of a string of terminals starting from S in which E appears.
3. e is useless because there is no derivation of a string of terminals starting from S in which e appears
4. A is uselss because it cannot derive a string of terminals. One rule is recursive (A → A S), so we cannot hope of getting a string of terminals from A by only using the rule A → A S. The other rule for A also cannot derive a string of terminals because it contains E.
5. S is not useless because it appears in the derivation S ⇒ a B C ⇒ a b C ⇒ a b c
6. B is not useless because it appears in the derivation S ⇒ a B C ⇒ a b C ⇒ a b c
7. C is not useless because it appears in the derivation S ⇒ a B C ⇒ a b C ⇒ a b c

Consider the grammar

S → A B G | D E F

A → a B | B D

B → b D | A E

E → e E | C E

C → c A | E | c

D → d | F g

F → f F | G f

G → g G | F g

Which are the useful symbols of this grammar? For every useful symbol, give a derivation that shows that the symbol is useful. If a derivation shows multiple symbols to be useful, you can give the derivation and state which symbols are useful according to the derivation.