MUJTABA SHAHID FAIZI

BSCS-5A

#131818

Lab 9 of Compiler Construction

**Code:**

**def** first\_and\_follow(grammar):  
 *# first & follow sets, epsilon-productions* first = {i: set() **for** i **in** grammar.nonterminals}  
 first.update((i, {i}) **for** i **in** grammar.terminals)  
 follow = {i: set() **for** i **in** grammar.nonterminals}  
 epsilon = set()  
 **while True**:  
 updated = **False  
   
 for** nt, expression **in** grammar.rules:  
 *# FIRST set w.r.t epsilon-productions* **for** symbol **in** expression:  
 updated |= union(first[nt], first[symbol])  
  
 **if** symbol **not in** epsilon:  
 **break  
 else**:  
 updated |= union(epsilon, {nt})  
   
 *# FOLLOW set w.r.t epsilon-productions* aux = follow[nt]  
 **for** symbol **in** reversed(expression):  
 **if** symbol **in** follow:  
 updated |= union(follow[symbol], aux)  
 **if** symbol **in** epsilon:  
 aux = aux.union(first[symbol])  
 **else**:  
 aux = first[symbol]  
   
 **if not** updated:  
 **return** first, follow, epsilon  
  
**def** union(first, begins):  
 n = len(first)  
 first |= begins  
 **return** len(first) != n  
  
**class** Grammar:  
   
 **def** \_\_init\_\_(self, \*rules):  
 self.rules = tuple(self.\_parse(rule) **for** rule **in** rules)  
  
 **def** \_parse(self, rule):  
 **return** tuple(rule.replace(**' '**, **''**).split(**'::='**))  
   
 **def** \_\_getitem\_\_(self, nonterminal):  
 **yield from** [rule **for** rule **in** self.rules   
 **if** rule[0] == nonterminal]  
   
 @staticmethod  
 **def** is\_nonterminal(symbol):  
 **return** symbol.isalpha() **and** symbol.isupper()  
   
 @property  
 **def** nonterminals(self):  
 **return** set(nt **for** nt, \_ **in** self.rules)  
   
 @property  
 **def** terminals(self):  
 **return** set(  
 symbol  
 **for** \_, expression **in** self.rules  
 **for** symbol **in** expression  
 **if not** self.is\_nonterminal(symbol)  
 )  
  
grammar=Grammar(  
 **'E::=TS'**,  
 **'S::=+TS'**,  
 **'S::=-TS'**,  
 **'S::='**,  
 **'T::=FR'**,  
 **'R::=\*FR'**,  
 **'R::=/FT'**,  
 **'R::='**,  
 **'F::=(E)'**,  
 **'F::=i'**,  
 **'F::=n'**,  
)  
first, follow, epsilon = first\_and\_follow(grammar)  
  
**for** a **in** follow:  
 follow[a].add(**'$'**)  
  
b=[]  
**for** a **in** first: *#for removing first sets for terminals; only non-terminal first sets are needed* **if** a.isalpha() **and** a.isupper():  
 **continue  
 else**:  
 b.extend(a)  
**for** c **in** b:  
 first.pop(c)  
  
print(**"PRODUCTIONS:"**)  
print(grammar.rules)  
  
print(**"\nFIRST SET:"**)  
print(first)  
  
print(**"\nFOLLOW SET:"**)  
print(follow)  
  
  
**def** ifcontains(nt,b): *#checks if the future productions will yield the required terminal/token* **for** rule **in** grammar.rules:  
 **if** rule[0] == nt:  
 **for** a **in** first[nt]: *#means that nested non-terminal also has that terminal in its first-set* **if** b == a:  
 **return True  
 return False**print(**"\nPARSE TABLE:"**)  
**for** a **in** grammar.nonterminals:  
 **if** a **in** epsilon:  
 print(a, **" , "**, **'$'**, **" :: "**, **"epsilon"**)  
 **for** b **in** grammar.terminals: *#running for each non-terminal for each terminal to get parse table entry* **if** b **in** first[a]:  
 **for** rule **in** grammar.rules: *#checking each production for getting the specific non-terminal* **if** rule[0] == a:  
 RHS=rule[1]  
 **if** RHS != **''**: *#for avoiding index error* **if** RHS[0] == b: *#if the r.h.s of production's first terminal matches* print(a, **" , "**, b, **" :: "**, rule[1])  
 **break  
 elif** RHS[0].isalpha() **and** RHS[0].isupper() **and** ifcontains(RHS[0],b)==**True**: *#if the nested non-terminal contains the terminal in its first set* print(a, **" , "**, b, **" :: "**, rule[1])  
 **break  
 elif** a **in** epsilon: *#follow set entry in the parse table if non-terminal contains epsilon in its production* **if** b **in** follow[a]:  
 print(a, **" , "**, b, **" :: "**, **"epsilon"**)

**Screenshot:**



