Compiler Design

Exp–6 Predictive Parsing

Name:- K. DUSHYANT REDDY Reg No.:- RA1911033010029

Branch:- CSE-SE

Date:- 08/03/22

AIM: To implement Predictive Parsing

CODE:

#Predictive Parsing

import re

import json

import pandas as pd

from tabulate import tabulate

rules = {}

printer = []

callStack = []

rules = {}

print("Enter the productions(Note- For Epsilon Use @ Symbol):")

for i in input().split(' '):

    NT = ""

    rule = re.split('->|\|', i)

    length = len(rule)

    NonTerminal = {'RHS': [],

                   'first': [],

                   'follow': []}

    for index, j in enumerate(rule):

        if index == 0:

            NT = j

            continue

        NonTerminal['RHS'].append(j)

        if index == length-1:

            rules[NT] = NonTerminal

start = input("Enter the start symbol:")

*def* first(*NT*):

    f = []

    for i in rules[*NT*]['RHS']:

        if not i[0].isupper():

            f.append(i[0])

        elif i[0] == '@':

            f.append("@")

        elif i[0].isupper():

            for j in first(i[0]):

                if j == '@':

                    count = 1

                    Break = True

                    while True:

                        try:

                            if not i[count].isupper():

                                f.append(i[count])

                                break

                            for k in first(i[count]):

                                if(k == '@'):

                                    count += 1

                                    Break = False

                                    break

                                f.append(k)

                            if(Break):

                                break

                            Break = True

                        except:

                            f.append("@")

                            break

                else:

                    f.append(j)

    return f

*def* follow(*NT*):

    f = []

    if *NT* == start:

        f.append('$')

    for LHS in rules.keys():

        for j in rules[LHS]['RHS']:

            if *NT* in j:

                try:

                    index = j.find(*NT*)

                    if not j[index+1].isupper():

                        f.append(j[index+1])

                    elif j[index+1].isupper():

                        for k in first(j[index+1]):

                            if k == "@":

                                count = 1

                                Break = True

                                while True:

                                    if not j[index+1+count].isupper():

                                        f.append(j[index+1+count])

                                        break

                                    for ele in first(j[index+1+count]):

                                        if(k == '@'):

                                            count += 1

                                            Break = False

                                            break

                                        f.append(ele)

                                if(Break):

                                    break

                                Break = True

                            else:

                                f.append(k)

                except:

                    if not LHS in callStack:

                        callStack.append(LHS)

                        for k in follow(LHS):

                            f.append(k)

    return f

for NT in rules.keys():

    rules[NT]["first"] = list(set(first(NT)))

    printer.append("FIRST(%s)=%s" % (NT, list(set(first(NT)))))

for NT in rules.keys():

    callStack.clear()

    rules[NT]["follow"] = list(set(follow(NT)))

    callStack.clear()

    printer.append("FOLLOW(%s)=%s" % (NT, list(set(follow(NT)))))

for i in printer:

    print(i)

print("\n")

terminals = []

nonTerminals = list(rules.keys())

for i in rules.keys():

    for k in rules[i]["first"]:

        terminals.append(k)

    for l in rules[i]["follow"]:

        terminals.append(l)

terminals = list(set(terminals))

terminals.remove('@')

data = []

for NT in nonTerminals:

    temp = {}

    for T in rules[NT]["first"]:

        if T == "@":

            for l in rules[NT]["follow"]:

                if l in temp.keys():

                    temp[l] = temp[l] + "\n" + NT+"->@"

                else:

                    temp[l] = NT+"->@"

        for k in rules[NT]["RHS"]:

            if not k[0].isupper():

                if k[0] == T:

                    """  if T in temp.keys():

                         temp[T] = temp[T] + "\n" + NT+"->"+k

                     else: """

                    temp[T] = NT+"->"+k

            elif k[0].isupper():

                if T in first(k[0]):

                    if T in temp.keys():

                        temp[T] = temp[T] + "\n" + NT+"->"+k

                    else:

                        temp[T] = NT+"->"+k

                if '@' in rules[k[0]]["RHS"]:

                    count = 1

                    Break = True

                    while True:

                        if k[count].isupper():

                            if T in first(k[count]):

                                if T in temp.keys():

                                    temp[T] = temp[T] + \

                                        "\n" + NT+"->"+k

                                else:

                                    temp[T] = NT+"->"+k

                                break

                            if "@" in rules[k[count]]["RHS"]:

                                count += 1

                                Break = False

                        elif k[count] == T:

                            if T in temp.keys():

                                temp[T] = temp[T] + "\n" + NT+"->"+k

                            else:

                                temp[T] = NT+"->"+k

                        if(Break):

                            break

                        Break = True

    data.append(temp)

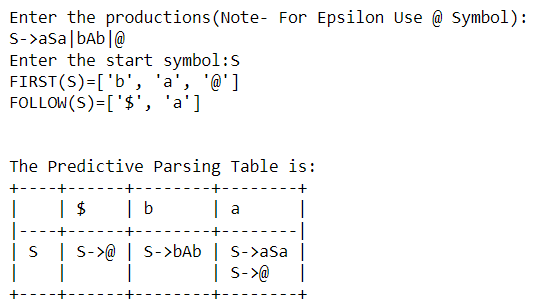
df = pd.DataFrame(data, *index*=nonTerminals, *columns*=terminals)

df.fillna(" ", *inplace*=True)

print("The Predictive Parsing Table is:")

print(tabulate(df, *headers*='keys', *tablefmt*='psql'))

OUTPUT:



RESULT:

The code is successfully implemented in Python language.