COMPUTATION OF LR(0) ITEMS

EX. NO. 9

YUVRAJ SINGH CHAUHAN

RA1911027010058

AIM: To write a program for computing LR(0) items.

ALGORITHM:

- The LR Parser is a Shift-reduce Parser that makes use of a Deterministic Finite Automata, recognizing the Set Of All Viable Prefixes by reading the stack from Bottom To Top.
- If a Finite-State Machine that recognizes viable prefixes of the right sentential forms is constructed, it can be used to guide the handle selection in the Shift-reduce Parser.
- Handle: Handle is a substring that matches the body of a production.
- Handle is a Right Sentential Form + position where reduction can be performed + production used for reduction.

LR(0) Items

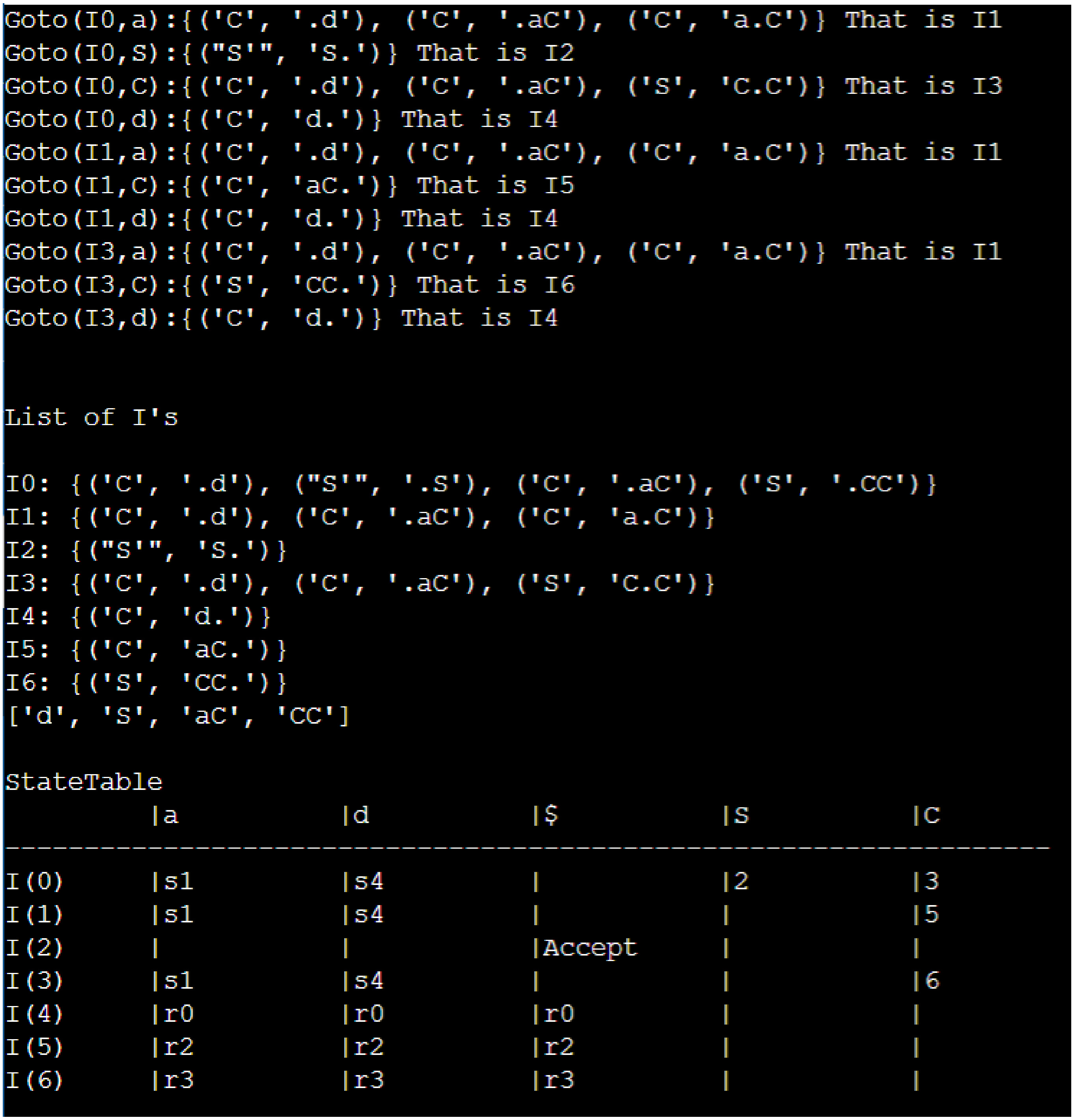
An LR(0) Item of a Grammar G is a Production of G with a Dot (.) at some position of the right side.
Production A → XYZ yields the Four items:
1. A→•XYZ We hope to see a string derivable from XYZ next on the input.
2. A→X•YZ We have just seen on the input a string derivable from X and that we hope next to see a string derivable from YZ next on the input.
3. A→XY•Z
4. A→XYZ•

- The production A→e generates only one item, A→•.
- Each of this item is a Viable prefixes.
- Closure Item : An Item created by the closure operation on a state.
- Complete Item : An Item where the Item Dot is at the end of the RHS.

PROGRAM:

```
gram = {
	"S":["CC"],
	"C":["aC","d"]
}
start = "S"
terms = ["a","d","$"]

non_terms = []
for i in gram:
    non_terms.append(i)
gram["S'"]= [start]

new_row = {}
for i in terms+non_terms:
    new_row[i]=""

non_terms += ["S'"]
stateTable = []

def Closure(term, I):
    if term in non_terms:
        for i in gram[term]:
            I+=[(term,"."+i)]
    I = list(set(I))
    for i in I:
        if "." != i[1][-1] and i[1][i[1].index(".")+1] in non_terms and i[1][i[1].index(".")+1] !=
term:
            I += Closure(i[1][i[1].index(".")+1], [])
    return I

Is = []
Is+=set(Closure("S'", []))

countI = 0
omegaList = [set(Is)]
while countI<len(omegaList):
    newrow = dict(new_row)
    vars_in_I = []
    Is = omegaList[countI]
    countI+=1
    for i in Is:
        if i[1][-1]!=".":
            indx = i[1].index(".")
            vars_in_I+=[i[1][indx+1]]
    vars_in_I = list(set(vars_in_I))
    for i in vars_in_I:
        In = []
        for j in Is:
            if "."+i in j[1]:
                rep = j[1].replace("."+i,i+".")
                In+=[(j[0],rep)]
        if (In[0][1][-1]!="."):
            temp = set(Closure(i,In))
            if temp not in omegaList:
                omegaList.append(temp)
            if i in non_terms:
                newrow[i] = str(omegaList.index(temp))
            else:
                newrow[i] = "s"+str(omegaList.index(temp))
            print(f'Goto(I{countI-1},{i}):{temp} That is I{omegaList.index(temp)}')
        else:
            temp = set(In)
            if temp not in omegaList:
                omegaList.append(temp)
            if i in non_terms:
                newrow[i] = str(omegaList.index(temp))
            else:
                newrow[i] = "s"+str(omegaList.index(temp))
            print(f'Goto(I{countI-1},{i}):{temp} That is I{omegaList.index(temp)}')

    stateTable.append(newrow)
print("\n\nList of I's\n")
for i in omegaList:
    print(f'I{omegaList.index(i)}: {i}')

I0 = []
for i in list(omegaList[0]):
    I0 += [i[1].replace(".","")]
print(I0)

for i in omegaList:
    for j in i:
        if "." in j[1][-1]:
            if j[1][-2]=="S":
                stateTable[omegaList.index(i)]["$"] = "Accept"
                break
            for k in terms:
                stateTable[omegaList.index(i)][k] = "r"+str(I0.index(j[1].replace(".","")))
print("\nStateTable")

print(f'{" ": <9}',end="")
for i in new_row:
    print(f'|{i: <11}',end="")

print(f'\n{"-":-<66}')
for i in stateTable:
    print(f'{"I("+str(stateTable.index(i))+")": <9}',end="")
    for j in i:
        print(f'|{i[j]: <10}',end=" ")
    print()
```

OUTPUT :



```
Goto(I0,a):{('C', '.d'), ('C', '.aC'), ('C', 'a.C')} That is I1
Goto(I0,S):{("S'", 'S.')} That is I2
Goto(I0,C):{('C', '.d'), ('C', '.aC'), ('S', 'C.C')} That is I3
Goto(I0,d):{('C', 'd.')} That is I4
Goto(I1,a):{('C', '.d'), ('C', '.aC'), ('C', 'a.C')} That is I1
Goto(I1,C):{('C', 'aC.')} That is I5
Goto(I1,d):{('C', 'd.')} That is I4
Goto(I3,a):{('C', '.d'), ('C', '.aC'), ('C', 'a.C')} That is I1
Goto(I3,C):{('S', 'CC.')} That is I6
Goto(I3,d):{('C', 'd.')} That is I4


List of I's

I0: {('C', '.d'), ("S'", '.S'), ('C', '.aC'), ('S', '.CC')}
I1: {('C', '.d'), ('C', '.aC'), ('C', 'a.C')}
I2: {("S'", 'S.')}
I3: {('C', '.d'), ('C', '.aC'), ('S', 'C.C')}
I4: {('C', 'd.')}
I5: {('C', 'aC.')}
I6: {('S', 'CC.')}
['d', 'S', 'aC', 'CC']

StateTable
         |a          |d          |$          |S          |C
------------------------------------------------------------------
I(0)     |s1         |s4         |           |2          |3
I(1)     |s1         |s4         |           |           |5
I(2)     |           |           |Accept     |           |
I(3)     |s1         |s4         |           |           |6
I(4)     |r0         |r0         |r0         |           |
I(5)     |r2         |r2         |r2         |           |
I(6)     |r3         |r3         |r3         |           |
```

RESULT :

LR(0) Items were computed successfully using python language.