

Python CODE

```
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"]
# each row in state table will be dictionary {nonterms ,term,$}
stateTable = []
# l = [(terminal, closure)]
# l = [("S","A.A")]

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

ls = []
ls+=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))
    # print(vars_in_I)
    for i in vars_in_I:
        In = []
        for j in Is:
            if "."+i in j[1]:
                rep = j[1].replace(".",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(l{countI-1},{i}):{temp} That is l{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(l{countI-1},{i}):{temp} That is l{omegaList.index(temp)}')

```

```

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

#populate replace elements in state Table
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()

```

IMPLEMENTATION

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/Shivam/Desktop/STUDY MATERIAL/Compiler Design/Lab/LR(0).py
Goto(I0,C):({'C', 'a.C'), ('S', 'C.C'), ('C', 'd')} That is I1
Goto(I0,d):({'C', 'd'}) That is I2
Goto(I0,a):({'C', 'a.C'), ('C', 'a.C'), ('C', 'd')} That is I3
Goto(I0,S):({'S', 'S'}) That is I4
Goto(I1,C):({'S', 'CC'}) That is I5
Goto(I1,d):({'C', 'd'}) That is I2
Goto(I1,a):({'C', 'a.C'), ('C', 'a.C'), ('C', 'd')} That is I3
Goto(I3,C):({'C', 'a.C'}) That is I6
Goto(I3,d):({'C', 'd'}) That is I2
Goto(I3,a):({'C', 'a.C'), ('C', 'a.C'), ('C', 'd')} That is I3

List of I's
I0: ({'C', 'a.C'), ('S', 'CC'), ('S', 'S'), ('C', 'd')})
I1: ({'C', 'a.C'), ('S', 'C.C'), ('C', 'd')})
I2: ({'C', 'd'})
I3: ({'C', 'a.C'), ('C', 'a.C'), ('C', 'd')})
I4: ({'S', 'S'})
I5: ({'S', 'CC'})
I6: ({'C', 'a.C'})
['a.C', 'CC', 'S', 'd']

StateTable
-----
|a      |d      |$      |S      |C
-----
I(0)    |s3     |s2     |       |4     |1
I(1)    |s3     |s2     |       |      |5
I(2)    |s3     |r3     |       |      |
I(3)    |s3     |s2     |       |      |6
I(4)    |       |       |Accept |      |
I(5)    |r1     |r1     |       |      |
I(6)    |r0     |r0     |r0     |      |
>>> |
```

RESULT

Code was successfully implemented and the output was verified.