**FIRST AND FOLLOW**

from collections import defaultdict

productions = defaultdict(list)

first = defaultdict(set)

follow = defaultdict(set)

def get\_first(X):

if not X.isupper(): return {X}

if first[X]: return first[X]

for prod in productions[X]:

for symbol in prod:

f = get\_first(symbol)

first[X] |= (f - {'ε'})

if 'ε' not in f: break

else:

first[X].add('ε')

return first[X]

def compute\_follow(start):

follow[start].add('$')

while True:

updated = False

for lhs, prods in productions.items():

for prod in prods:

trailer = follow[lhs].copy()

for symbol in reversed(prod):

if symbol.isupper():

if not trailer.issubset(follow[symbol]):

follow[symbol] |= trailer

updated = True

if 'ε' in first[symbol]:

trailer |= (first[symbol] - {'ε'})

else:

trailer = first[symbol]

else:

trailer = get\_first(symbol)

if not updated: break

# Input

n = int(input("Number of productions: "))

for \_ in range(n):

lhs, rhs = input().replace(" ", "").split("->")

productions[lhs] += rhs.split('|')

start\_symbol = list(productions.keys())[0]

non\_terminals = list(productions.keys())

for non\_term in non\_terminals:

get\_first(non\_term)

compute\_follow(start\_symbol)

print("\nFIRST sets:")

for nt in productions:

print(f"FIRST({nt}) = {{ {', '.join(first[nt])} }}")

print("\nFOLLOW sets:")

for nt in productions:

print(f"FOLLOW({nt}) = {{ {', '.join(follow[nt])} }}")

**LL(1) PARSER**

from collections import defaultdict

productions = defaultdict(list)

first = defaultdict(set)

follow = defaultdict(set)

table = defaultdict(dict)

def get\_first(X):

if not X.isupper(): return {X}

if first[X]: return first[X]

for prod in productions[X]:

for sym in prod:

f = get\_first(sym)

first[X] |= (f - {'ε'})

if 'ε' not in f: break

else: first[X].add('ε')

return first[X]

def compute\_follow(start):

follow[start].add('$')

while True:

updated = False

for lhs, prods in productions.items():

for prod in prods:

trailer = follow[lhs].copy()

for sym in reversed(prod):

if sym.isupper():

if not trailer.issubset(follow[sym]):

follow[sym] |= trailer

updated = True

if 'ε' in first[sym]:

trailer |= (first[sym] - {'ε'})

else:

trailer = first[sym]

else:

trailer = get\_first(sym)

if not updated: break

def build\_table():

for lhs, prods in productions.items():

for prod in prods:

first\_set = set()

for sym in prod:

f = get\_first(sym)

first\_set |= (f - {'ε'})

if 'ε' not in f: break

else:

first\_set.add('ε')

for term in first\_set:

table[lhs][term] = prod

if 'ε' in first\_set:

for term in follow[lhs]:

table[lhs][term] = prod

def parse(input\_str, start):

stack = ['$' , start]

input\_str += '$'

ip = list(input\_str)

steps = []

while stack:

top = stack.pop()

curr = ip[0]

stack\_content = ''.join(stack[::-1]) + top

input\_content = ''.join(ip)

if top == curr == '$':

steps.append((stack\_content, input\_content, "Accept"))

break

elif not top.isupper():

if top == curr:

ip.pop(0)

steps.append((stack\_content, input\_content, f"Match {top}"))

else:

steps.append((stack\_content, input\_content, "Error"))

break

elif curr in table[top]:

prod = table[top][curr]

steps.append((stack\_content, input\_content, f"{top} → {prod}"))

if prod != 'ε':

stack += reversed(prod)

else:

steps.append((stack\_content, input\_content, "Error"))

break

# Print trace table

print("\nParsing Trace:")

print(f"{'Stack':<20} {'Input':<20} {'Action'}")

print("-" \* 60)

for s, i, a in steps:

print(f"{s:<20} {i:<20} {a}")

# Input

n = int(input("No. of productions: "))

for \_ in range(n):

lhs, rhs = input("Production: ").replace(" ", "").split("->")

productions[lhs] += rhs.split('|')

start = list(productions.keys())[0]

for nt in productions: get\_first(nt)

compute\_follow(start)

build\_table()

# Print FIRST & FOLLOW

print("\nFIRST sets:")

for nt in productions: print(f"FIRST({nt}) = {{ {', '.join(first[nt])} }}")

print("\nFOLLOW sets:")

for nt in productions: print(f"FOLLOW({nt}) = {{ {', '.join(follow[nt])} }}")

# Terminals

terminals = set()

for row in table.values(): terminals.update(row)

terminals = sorted(terminals - {'ε'}) + ['$']

# Print Parse Table

print("\nLL(1) Parse Table:")

header = f"{'NT/T':<10}" + "".join(f"{t:<10}" for t in terminals)

print(header)

print("-" \* len(header))

for nt in productions:

row = f"{nt:<10}"

for t in terminals:

prod = table[nt].get(t, "")

cell = f"{nt}->{prod}" if prod else ""

row += f"{cell:<10}"

print(row)

# Parse input

inp = input("\nEnter input string: ")

parse(inp, start)

**OUTPUT**

No. of productions: 3

Production: E->TA

Production: A->+TA|ε

Production: T->id

FIRST sets:

FIRST(E) = { i }

FIRST(A) = { ε, + }

FIRST(T) = { i }

FOLLOW sets:

FOLLOW(E) = { $ }

FOLLOW(A) = { $ }

FOLLOW(T) = { $, + }

LL(1) Parse Table:

NT/T $ + i $

--------------------------------------------------

E E->TA

A A->ε A->+TA A->ε

T T->id

Enter input string: id+id

Parsing Trace:

Stack Input Action

------------------------------------------------------------

$E id+id$ E → TA

A$T id+id$ T → id

dA$i id+id$ Match i

A$d d+id$ Match d

$A +id$ A → +TA

TA$+ +id$ Match +

A$T id$ T → id

dA$i id$ Match i

A$d d$ Match d

$A $ A → ε

$ $ Accept

=== Code Execution Successful ===

**SLR PARSER**