**Session 6: Predictive Parsing**

1. **OBJECTIVES**:

Manual implementation of LL(1) and LR(1) parsing algorithms.

1. **Demonstration of Useful Resources:**
2. Computation of the FIRST and FOLLOW functions.

* To Compute FIRST(X) for all grammar symbols X, apply the following rules until no more terminals or ε can be added to any FIRST set.

1. If X is a terminal, then FIRST(X) is {X}.
2. If X is a non-terminal and X → Y1Y2 … Yk is a production for some k ≥ 1, then place b in FIRST(X) if for some i, b is in FIRST(Yi), and ε is in all of FIRST(Y1), …, FIRST(Yi-1);

that is, Y1…Yi-1 derives ε.

1. If ε is in FIRST(Yj ) for all j = 1, 2, …, k then add ε to FIRST(X).

**Sample input and corresponding output:**

FIRST(E) = {(, id}

FIRST(T) = {(, id}

FIRST(E') = {+, ε}

FIRST(T') = {\*, ε}

FIRST(F) = {(, id}

E → TE'

E' → +TE' | ε

T → FT'

T' → \*FT' | ε

F → (E) | id

* To compute FOLLOW(A) for all non-terminals A, apply the following rules until nothing can be added to any FOLLOW set.

1. Place $ in FOLLOW(S), where S is the start symbol and $ is the right end-marker of an input.
2. If there is a production A →αBβ, then everything in FIRST(β) except ε is in FOLLOW(B).
3. If there is a production A →αB, then everything in FOLLOW(A) is in FOLLOW(B).
4. If there is a production A →αBβ where FIRST(β) contains ε, then everything in FOLLOW(A) is in FOLLOW(B).

**Sample input and corresponding output:**

FOLLOW(E) = {$, )}

FOLLOW(T) = {+, $, )}

FOLLOW(E') = {), $}

FOLLOW(T') = {+, ),$}

FOLLOW(F) = {\*, +, $, )}

FIRST(E) = {(, id}

FIRST(T) = {(, id}

FIRST(E') = {+, ε}

FIRST(T') = {\*, ε}

FIRST(F) = {(, id}

E → TE'

E' → +TE' | ε

T → FT'

T' → \*FT' | ε

F → (E) | id

1. Table for LL(1) non-recursive predictive parsing with the given grammar:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Non-terminal | Input symbols | | | | | |
| id | + | \* | ( | ) | $ |
| E | E → TE*'* |  |  | E → TE*'* |  |  |
| E' |  | E*'*→+TE*'* |  |  | E*'*→ε | E*'*→ε |
| T | T → FT*'* |  |  | T → FT*'* |  |  |
| T' |  | T*'*→ε | T*'*→ \*F T*'* |  | T*'*→ε | T*'*→ε |
| F | F →id |  |  | F → (E) |  |  |

1. An example of construction of tools for LR(1) parsing

A

a

accept

S

$

A

a

b

A

a

1. S →AbA
2. A →aA
3. A → a

S' → S

S →AbA

A→aA

A → a

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **State** | **ACTION** | | | **GOTO**   1. S →AbA 2. A →aA 3. A →a | |
| **a** | **b** | **$** | **S** | **A**   1. S →AbA 2. A →aA 3. A →a |
| 0 | s3 |  |  | 1 | 2 |
| 1 |  |  | acc |  |  |
| 2 |  | s4 |  |  |  |
| 3 | s3 | r3 | r3 |  | 5 |
| 4 | s3 |  |  |  | 6 |
| 5 |  | r2 | r2 |  |  |
| 6 |  |  | r1 |  |  |

FOLLOW(S) = {$}

FOLLOW(A) = {b, $}

FOLLOW(E) = {$, )}

FOLLOW(T) = {+, $, )}

FOLLOW(E') = {), $}

FOLLOW(T') = {+, ),$}

FOLLOW(F) = {\*, +, $, )}

FIRST(S) = FIRST(A) ={a}

If A →α• is in Ii, then set ACTION(i, a) to rj (“Reduce by rule j”, where j is the serial no. of A →α) for all a in FOLLOW(A).

I3 contains A→ a•; I5 contains A → aA•; I6 contains S → AbA•.

1. **Lab Exercise:**

Perform the tasks 1, 2, and 3 of the Assignment #6 which is described below.

1. **Assignment #6:**

Suppose, you are given the following grammar and the input string ***abcd***:

S → aXd

X → YZ

Y → b

Y → ε

Z → cX

Z → ε

* You are required to perform the following tasks manually:

1. Find the FIRST and FOLLOW sets of each of the non-terminals.
2. Construct the predictive parsing table for LL(1) method.
3. Demonstrate the moves of the LL(1) parser on the given input.
4. Construct the LR(0) automaton for the grammar.
5. Construct the parsing table for LR(1) parsing with the grammar.
6. Demonstrate the moves of the LR(1) parser on the given input.