SLR PARSER

https://www.geeksforgeeks.org/slr-parser-with-examples/

L stands for the left to right scanning  
R stands for rightmost  derivation in reverse  
0 stands for no. of input symbols of lookahead

PROBLEM

1. Given the set of grammar

E → E+T/T

T → T\*F / F

F → (E) / id

**Steps to solve-SLR Parser**

1. Preliminary Steps
2. Split into separate production rules
3. Find FOLLOW ( Elimination of left recursion not required)
4. Augmented Grammar ( to indicate to the parser when it should stop parsing and announce acceptance of input•)
5. Canonical LR (0) Collections-Closure operation –Goto
6. Transition Diagram (DFA) of Goto Function
7. SLR Parser Table
8. Stack Implementation with the given input

**Step 1 : Preliminary Steps**

1. **Split into separate production rules**

**E** → **E+T**

**E** → **T**

**T** → **T\*F**

**T** → **F**

**F** → **(E)**

**F** → **id**

1. **To Find Follow**

**( Video Link : https://www•youtube•com/watch?v=SBnjVW8dUqo)**

**Follow Functions-**

* Follow(E) = { $ , ) }
* Follow(T) = { \*,+ , $ , ) }
* Follow(F) = { \* , + , $ , ) }

**( Video Link : https://www•youtube•com/watch?v=ZfB4JU2YZ\_0)**

E’ → E  
E → E + T  
E → T  
T → T \* F  
T → F

F🡪(E)  
F → id

**Step 3 : Canonical LR (0) Collections – Using Closure and Goto**

**( Video Link : https://www•youtube•com/watch?v=ZfB4JU2YZ\_0)**

**I0=** E’ → •E  
       E → •E + T  
       E → •T  
       T → •T \* F  
       T → •F

F🡪• (E)  
       F → •id

**I1: GOTO (I0,E)**

E’ → E•

E🡪E• + T

**I2: GOTO (I0,T)**

E → T•

T → T•\*F

**I3: GOTO (I0,F)**

T → F•

**I4:GOTO (I0,( )**

F → (•E)

E → •E+T

E → •T

T → •T\*F

T → •F

F → •(E)

F → •id

**I5:GOTO (I0,id)**

F → id•

**I6:GOTO (I1,+)**

E → E+•T

T → •T\*F

T → •F

F → •(E)

F → •id

**I7:GOTO ( I2,\*)**

T → T\*•F

F → •(E)

F → •id

**I8:GOTO (I4,E)**

F → (E•)

E → E•+T

**I9:GOTO (I6,T)**

E🡪E+T•

T🡪T•\*F

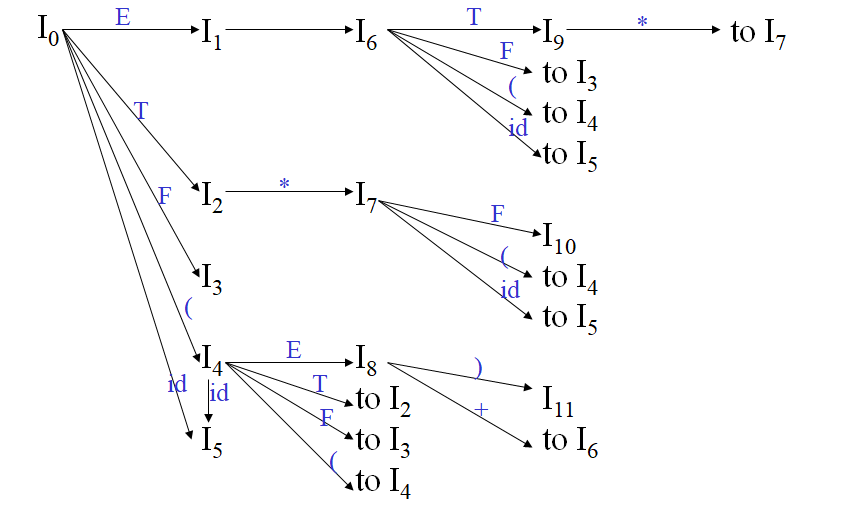
**I10:GOTO(I7,F)**

T → T\*F•

**I11:GOTO(I8,) )**

F🡪(E)•

**Step 4 : Transition Diagram (DFA) of Goto Function**



**Step 5 : SLR Parser Table-Construction**

**(Video Link :** [**https://www.youtube.com/watch?v=8Cq3EIgXOec**](https://www.youtube.com/watch?v=8Cq3EIgXOec)**)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ACTION | | | | | |  | GOTO | | |
| **state** | **id** | **+** | **\*** | **(** | **)** | **$** |  | **E** | **T** | **F** |
| 0 | s5 |  |  | s4 |  |  |  | 1 | 2 | 3 |
| 1 |  | s6 |  |  |  | acc |  |  |  |  |
| 2 |  | r2 | s7 |  | r2 | r2 |  |  |  |  |
| 3 |  | r4 | r4 |  | r4 | r4 |  |  |  |  |
| 4 | s5 |  |  | s4 |  |  |  | 8 | 2 | 3 |
| 5 |  | r6 | r6 |  | r6 | r6 |  |  |  |  |
| 6 | s5 |  |  | s4 |  |  |  |  | 9 | 3 |
| 7 | s5 |  |  | s4 |  |  |  |  |  | 10 |
| 8 |  | s6 |  |  | s11 |  |  |  |  |  |
| 9 |  | r1 | s7 |  | r1 | r1 |  |  |  |  |
| 10 |  | r3 | r3 |  | r3 | r3 |  |  |  |  |
| 11 |  | r5 | r5 |  | r5 | r5 |  |  |  |  |

**Action : .before terminal ( Shift Operation)**

**Goto: .before Non-Terminal**

**E’🡪always accepting state at $**

**(Calculation for reduce operation)**

**E** → **E+T (r1)**

**E** → **T (r2)**

**T** → **T\*F (r3)**

**T** → **F (r4)**

**F** → **(E) (r5)**

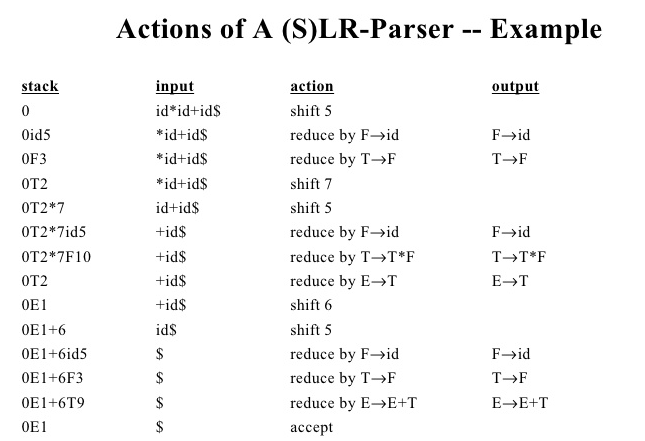
**F** → **id (r6)**

**( Explanation:**

**E🡪T. – it is achieved in I2, in I2, we read the non terminal symbol T so follow (T) = { +,$,)}, Mark r2 in state 2 under +,),$)**

1. **Step 6 :** Stack Implementation with the given input

**( Video Link :** [**https://www.youtube.com/watch?v=xJHMuVUyTWE**](https://www.youtube.com/watch?v=xJHMuVUyTWE)**)**

****