1.a) First, we check if there is any unreachable and unproductive variable.

…

Belong to the previous algorithm, no variable has to be removed.

1.b) Next, we check if the grammar is ambiguous by this example :

…

So, we have to modify the grammar in order to remove its ambiguousness.

Rules 14-22 are replaced by :

|  |  |  |
| --- | --- | --- |
| P |  |  |
| <ExprArith> | → | <ExprArith> <Op+-> <ExprArith\_b> |
|  | → | <ExprArith\_b> |
| <ExprArith\_b> | → | <ExprArith\_b> <Op\*/> <ExprArith\_c> |
|  | → | <ExprArith\_c> |
| <ExprArith\_c> | → | - <ExprArith\_c> |
|  | → | [VarName] |
|  | → | [Number] |
|  | → | ( <ExprArith> ) |
| <Op+-> | → | + |
|  | → | - |
| <Op\*/> | → | \* |
|  | → | / |
|  |  |  |

Rules 25-27 are replaced by :

|  |  |  |
| --- | --- | --- |
| P |  |  |
| <Cond> | → | <Cond> .OR. <Cond\_b> |
|  | → | <Cond\_b> |
| <Cond\_b> | → | <Cond\_b> .AND. <Cond\_c> |
|  | → | <Cond\_c> |
| <Cond\_c> | → | .NOT. <SimpleCond> |
|  | → | <SimpleCond> |

1.c) Here is the final grammar after removing left-recursion and applying factorisation.

|  |  |  |  |
| --- | --- | --- | --- |
| # | P |  |  |
| [1] | <Program> | → | PROGRAM [ProgName] [EndLine] <Vars> <Code> END |
| [2] | <Vars> | → | INTEGER <VarList> [EndLine] |
| [3] |  | → | ε |
| [4] | <VarList> | → | [VarName] <VarList\_next> |
| [5] | <VarList\_next> | → | , <VarList> |
| [6] |  | → | ε |
| [7] | <Code> | → | <Instruction> [EndLine] <Code> |
| [8] |  | → | ε |
| [9] | <Instruction> | → | <Assign> |
| [10] |  | → | <If> |
| [11] |  | → | <Do> |
| [12] |  | → | <Print> |
| [13] |  | → | <Read> |
| [14] | <Assign> | → | [VarName] = <ExprArith> |
| [15] | <ExprArith> | → | <ExprArith\_b> <ExprArith'> |
| [16] | <ExprArith'> | → | <Op+-> <ExprArith\_b> <ExprArith'> |
| [17] |  | → | ε |
| [18] | <ExprArith\_b> | → | <ExprArith\_c> <ExprArith\_b'> |
| [19] | <ExprArith\_b’> | → | <Op\*/> <ExprArith\_c> <ExprArith\_b'> |
| [20] |  | → | ε |
| [21] | <ExprArith\_c> | → | - <ExprArith\_c> |
| [22] |  | → | [VarName] |
| [23] |  | → | [Number] |
| [24] |  | → | ( <ExprArith> ) |
| [25] | <Op+-> | → | + |
| [26] |  | → | - |
| [27] | <Op\*/> | → | \* |
| [28] |  | → | / |
| [29] | <If> | → | IF (<Cond>) THEN [EndLine] <Code> <If-next> |
| [30] | <If-next> | → | ENDIF |
| [31] |  | → | ELSE [EndLine] <Code> ENDIF |
| [32] | <Cond> | → | <Cond\_b> <Cond'> |
| [33] | <Cond'> | → | .OR. <Cond\_b> <Cond'> |
| [34] |  | → | ε |
| [35] | <Cond\_b> | → | <Cond\_c> <Cond\_b'> |
| [36] | <Cond\_b'> | → | .AND. <Cond\_c> <Cond\_b'> |
| [37] |  | → | ε |
| [38] | <Cond\_c> | → | .NOT. <SimpleCond> |
| [39] |  | → | <SimpleCond> |
| [40] | <SimpleCond> | → | <ExprArith> <Comp> <ExprArith> |
| [41] | <Comp> | → | .EQ. |
| [42] |  | → | .GE. |
| [43] |  | → | .GT. |
| [44] |  | → | .LE. |
| [45] |  | → | .LT. |
| [46] |  | → | .NE. |
| [47] | <Do> | → | DO [VarName] = [Number], [Number] [EndLine] <Code> ENDDO |
| [48] | <Print> | → | PRINT\*, <ExpList> |
| [49] | <Read> | → | READ\*, <VarList> |
| [50] | <ExpList> | → | <ExprArith> <ExpList\_next> |
| [51] | <ExpList\_next> | → | , <ExpList> |
| [52] |  | → | ε |

2) Next, we have to calculate the First of all the variables and the Follow of the variables for which the First contains ε.

|  |  |  |
| --- | --- | --- |
| V | First(V) | Follow(V) |
| <Program> | PROGRAM |  |
| <Vars> | INTEGER ε | [VarName] IF DO PRINT\* READ\* END |
| <VarList> | [VarName] | [EndLine] |
| <VarList\_next> | , ε | [EndLine] |
| <Code> | [VarName] IF DO PRINT\* READ\* ε | END ENDIF ELSE ENDDO |
| <Instruction> | [VarName] IF DO PRINT\* READ\* | [EndLine] |
| <Assign> | [VarName] | [EndLine] |
| <ExprArith> | - [VarName] [Number] ( | [EndLine] ) .AND. .OR. , .EQ. .GE. .GT. .LE. .LT. .NE. **?** |
| <ExprArith’> | + - ε | [EndLine] ) .AND. .OR. , .EQ. .GE. .GT. .LE. .LT. .NE. |
| <ExprArith\_b> | - [VarName] [Number] ( | + - [EndLine] ) .AND. .OR. , .EQ. .GE. .GT. .LE. .LT. .NE. |
| <ExprArith\_b’> | \* / ε | + - [EndLine] ) .AND. .OR. , .EQ. .GE. .GT. .LE. .LT. .NE. |
| <ExprArith\_c> | - [VarName] [Number] ( | \* / + - [EndLine] ) .AND. .OR. , .EQ. .GE. .GT. .LE. .LT. .NE. |
| <Op+-> | + - | - [VarName] [Number] ( |
| <Op\*/> | \* / | - [VarName] [Number] ( |
| <If> | IF | [EndLine] |
| <If-next> | ENDIF ELSE | [EndLine] |
| <Cond> | .NOT. - [VarName] [Number] ( | ) |
| <Cond’> | .OR. ε | ) |
| <Cond\_b> | .NOT. - [VarName] [Number] ( | .OR. ) |
| <Cond\_b’> | .AND. ε | .OR. ) |
| <Cond\_c> | .NOT. - [VarName] [Number] ( | .AND. .OR. ) |
| <SimpleCond> | - [VarName] [Number] ( | .AND. .OR. ) |
| <Comp> | .EQ. .GE. .GT. .LE. .LT. .NE. | - [VarName] [Number] ( |
| <Do> | DO | [EndLine] |
| <Print> | PRINT\* | [EndLine] |
| <Read> | READ\* | [EndLine] |
| <ExpList> | - [VarName] [Number] ( | [EndLine] |
| <ExpList\_next> | , ε | [EndLine] |

Therefore, we have the action table (excel).

We can see that there is not conflict : LL(1) parser.