

Project - Part 2

Parser

Aldar Saranov, Przemyslaw Gasinski

Aldar.Saranov@ulb.ac.be
Przemyslaw.Gasinski@ulb.ac.be

Initial grammar:

```
<Program>
  -> PROGRAM [ProgName] [EndLine] <Vars> <Code> END
<Vars>
  -> INTEGER <VarList> [EndLine]
  ->  $\epsilon$ 
<VarList>
  -> [VarName], <VarList>
  -> [VarName]
<Code>
  -> <Instruction> [EndLine] <Code>
  ->  $\epsilon$ 
<Instruction>
  -> <Assign>
  -> <If>
  -> <Do>
  -> <Print>
  -> <Read>
<Assign>
  -> [VarName] = <ExprArith>
<ExprArith>
  -> [VarName]
  -> [Number]
  -> (<ExprArith>)
  -> -<ExprArith>
  -> <ExprArith> <Op> <ExprArith>
<Op>
  -> +
  -> -
  -> *
  -> /
<If>
  -> IF (<Cond>) THEN [EndLine] <Code> ENDIF
  -> IF (<Cond>) THEN [EndLine] <Code> ELSE [EndLine] <Code>
    ENDIF
<Cond>
  -> <Cond> <BinOp> <Cond>
  -> .NOT. <SimpleCond>
  -> <SimpleCond>
<SimpleCond>
  -> <ExprArith> <Comp> <ExprArith>
<BinOp>
  -> .AND.
  -> .OR.
<Comp>
  -> .EQ.
  -> .GE.
  -> .GT.
  -> .LE.
  -> .LT.
  -> .NE.
<Do>
  -> DO [VarName] = [Number], [Number] [EndLine] <Code>
    ENDDO
<Print>
  -> PRINT*, <ExpList>
```

```

<Read>
    -> READ*, <VarList>
<ExpList>
    -> <ExprArith>, <ExpList>
    -> <ExprArith>

```

No unproductive or inaccessible symbols found.

Removing left-recursion:

```

<Program>
    -> PROGRAM [ProgName] [EndLine] <Vars> <Code> END
<Vars>
    -> INTEGER <VarList> [EndLine]
    ->  $\epsilon$ 
<VarList>
    -> [VarName], <VarList>
    -> [VarName]
<Code>
    -> <Instruction> [EndLine] <Code>
    ->  $\epsilon$ 
<Instruction>
    -> <Assign>
    -> <If>
    -> <Do>
    -> <Print>
    -> <Read>
<Assign>
    -> [VarName] = <ExprArith>
<ExprArith>
    -> [VarName] <ExprArithRec>
    -> [Number] <ExprArithRec>
    -> (<ExprArith>) <ExprArithRec>
    -> -<ExprArith> <ExprArithRec>
<ExprArithRec>
    -> <Op> <ExprArith> <ExprArithRec>
    ->  $\epsilon$ 
<Op>
    -> +
    -> -
    -> *
    -> /
<If>
    -> IF (<Cond>) THEN [EndLine] <Code> ENDIF
    -> IF (<Cond>) THEN [EndLine] <Code> ELSE [EndLine] <Code>
        ENDIF
<Cond>
    -> .NOT. <SimpleCond> <CondRec>
    -> <SimpleCond> <CondRec>
<CondRec>
    -> <BinOp> <Cond> <CondRec>
    ->  $\epsilon$ 
<SimpleCond>

```

```

-> <ExprArith> <Comp> <ExprArith>
<BinOp>
-> .AND.
-> .OR.
<Comp>
-> .EQ.
-> .GE.
-> .GT.
-> .LE.
-> .LT.
-> .NE.
<Do>
-> DO [VarName] = [Number], [Number] [EndLine] <Code>
ENDDO
<Print>
-> PRINT*, <ExpList>
<Read>
-> READ*, <VarList>
<ExpList>
-> <ExprArith>, <ExpList>
-> <ExprArith>

```

Applying factorization:

```

<Program>
-> PROGRAM [ProgName] [EndLine] <Vars> <Code> END
<Vars>
-> INTEGER <VarList> [EndLine]
->  $\epsilon$ 
<VarList>
-> [VarName], <FactVarList>
<FactVarList>
-> <VarList>
->  $\epsilon$ 
<Code>
-> <Instruction> [EndLine] <Code>
->  $\epsilon$ 
<Instruction>
-> <Assign>
-> <If>
-> <Do>
-> <Print>
-> <Read>
<Assign>
-> [VarName] = <ExprArith>
<ExprArith>
-> <FactExprArith> <ExprArithRec>
<FactExprArith>
-> [VarName]
-> [Number]
-> (<ExprArith>)
-> -<ExprArith>
<ExprArithRec>
-> <Op> <ExprArith> <ExprArithRec>
->  $\epsilon$ 
<Op>
-> +

```

```

-> -
-> *
-> /
<If>
-> IF (<Cond>) THEN [EndLine] <Code> <FactIf>
<FactIf>
-> ENDIF
-> ELSE [EndLine] <Code> ENDIF
<Cond>
-> <CondPrefix> <SimpleCond> <CondRec>
<CondPrefix>
-> .NOT.
->  $\epsilon$ 
<CondRec>
-> <BinOp> <Cond> <CondRec>
->  $\epsilon$ 
<SimpleCond>
-> <ExprArith> <Comp> <ExprArith>
<BinOp>
-> .AND.
-> .OR.
<Comp>
-> .EQ.
-> .GE.
-> .GT.
-> .LE.
-> .LT.
-> .NE.
<Do>
-> DO [VarName] = [Number], [Number] [EndLine] <Code>
ENDDO
<Print>
-> PRINT*, <ExpList>
<Read>
-> READ*, <VarList>
<ExpList>
-> <ExprArith> <FactExprArith>
<FactExprArith>
-> , <ExpList>
->  $\epsilon$ 

```

Making non-ambiguous

```

<Program>
-> PROGRAM [ProgName] [EndLine] <Vars> <Code> END
<Vars>
-> INTEGER <VarList> [EndLine]
->  $\epsilon$ 
<VarList>
-> [VarName], <FactVarList>
<FactVarList>
-> <VarList>
->  $\epsilon$ 
<Code>
-> <Instruction> [EndLine] <Code>

```

```

-> ε
<Instruction>
-> <Assign>
-> <If>
-> <Do>
-> <Print>
-> <Read>
<Assign>
-> [VarName] = <ExprArith>
<ExprArith>
-> <ArithT> <RecArithE>
<RecArithE>
-> <Op1> <ArithT> <RecArithE>
-> ε
<Op1>
-> +
-> -
<ArithT>
-> <ArithF> <RecArithT>
<RecArithT>
-> <Op2> <ArithF> <RecArithT>
-> ε
<Op2>
-> *
-> /
<ArithF>
-> [VarName]
-> [Number]
-> (<ExprArith>)
-> -<ExprArith>
<If>
-> IF (<Cond>) THEN [EndLine] <Code> <FactIf>
<FactIf>
-> ENDIF
-> ELSE [EndLine] <Code> ENDIF
<CondPrefix>
-> .NOT.
-> ε
<Cond>
-> <CondT> <CondRecE>
<CondRecE>
-> .OR. <CondT> <CondRecE>
-> ε
<CondT>
-> <CondPrefix> <SimpleCond> <CondRecT>
<CondRecT>
-> .AND. <CondPrefix> <CondF> <CondRecT>
-> ε
<CondF>
-> <ExprArith> <Comp> <ExprArith>
<Comp>
-> .EQ.
-> .GE.
-> .GT.
-> .LE.
-> .LT.
-> .NE.

```

```

<Do>
    -> DO [VarName] = [Number], [Number] [EndLine] <Code>
    ENDDO
<Print>
    -> PRINT*, <ExpList>
<Read>
    -> READ*, <VarList>
<ExpList>
    -> <ExprArith> <FactExprArith>
<FactExprArith>
    -> , <ExpList>
    -> ε

```

Obtained grammar:

Number	Left side	Right side
0.	<All>	<Program> \$
1.	<Program>	PROGRAM [ProgName] [EndLine] <Vars> <Code>
2.	<Vars>	INTEGER <VarList> [EndLine]
3.		ε
4.	<VarList>	[VarName], <FactVarList>
5.	<FactVarList>	<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.	<Op1>	+
16.		-
17.	<Op2>	*
18.		/
19.	<ExprArith>	<ArithT> <RecArithE>
20.	<RecArithE>	<Op1> <ArithT> <RecArithE>
21.		ε
22.	<ArithT>	<ArithF> <RecArithT>
23.	<RecArithT>	<Op2> <ArithF> <RecArithT>
24.		ε
25.	<ArithF>	[VarName]
26.		<Number>
27.		(ExprArith)
28.		-<ArithF>
29.	<If>	IF (<Cond>) THEN [EndLine] <Code> <FactIf>
30.	<FactIf>	ENDIF
31.		ELSE [EndLine] <Code> ENDIF

32.	<CondPrefix>	.NOT.
33.		ϵ
34.	<Cond>	<CondT> <CondRecE>
35.	<CondRecE>	.OR. <CondT> <CondRecE>
36.		ϵ
37.	<CondT>	<CondPrefix> <CondF> <CondRecT>
38.	<CondRecT>	.AND. <CondPrefix> <CondF> <CondRecT>
39.		ϵ
40.	<CondF>	<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> <FactExprArith>
51.	<FactExprArith>	, <ExpList>
52.		ϵ

First input	First output
<Program> \$	PROGRAM
PROGRAM [ProgName] [EndLine]	PROGRAM
<Vars> <Code>	
INTEGER <VarList> [EndLine]	INTEGER
[VarName], <FactVarList>	VARNAME
<VarList>	VARNAME
<Instruction> [EndLine] <Code>	VARNAME, IF, DO, PRINT, READ
<Assign>	VARNAME
<If>	IF
<Do>	DO
<Print>	PRINT
<Read>	READ
[VarName] = <ExprArith>	VARNAME
<ArithT> <RecArithE>	VARNAME, NUMBER, (, -
<Op1> <ArithT> <RecArithE>	+, -
+	+
-	-
<ArithF> <RecArithT>	VARNAME, NUMBER, (, -
<Op2> <ArithF> <RecArithT>	*, /
*	*
/	/
[VarName]	VARNAME
[Number]	NUMBER
(ExprArith)	(
-<ExprArith>	-
IF (<Cond>) THEN [EndLine]	IF

<Code> <FactIf>	
ENDIF	ENDIF
ELSE [EndLine] <Code> ENDIF	ELSE
.NOT.	.NOT.
<CondT> <CondRecE>	.NOT., VARNAME, NUMBER, (, -
.AND. <CondPrefix> <SimpleCond>	.AND.
<CondRecT>	
<ExprArith> <Comp> <ExprArith>	VARNAME, NUMBER, (, -
.EQ.	.EQ.
.GE.	.GE.
.GT.	.GT.
.LE.	.LE.
.LT.	.LT.
.NE.	.NE.
DO [VarName] = [Number], [Number]	DO
[EndLine] <Code> ENDDO	
PRINT*, <ExpList>	PRINT*,
READ*, <VarList>	READ*,
<ExprArith> <FactExprArith>	VARNAME, NUMBER, (, -
, <ExpList>	,

Input	First	Follow
<All>	PROGRAM	ϵ
<Program>	PROGRAM	\$
<Vars>	ϵ , INTEGER	\$, VARNAME, IF, DO, PRINT, READ
<VarList>	VARNAME	ENDLINE
<FactVarList>	ϵ , VARNAME	ENDLINE
<Code>	ϵ , VARNAME, IF, DO, PRINT, READ	\$, ENDIF, ELSE, ENDDO
<Instruction>	VARNAME, IF, DO, PRINT, READ	ENDLINE
<Assign>	VARNAME,	ENDLINE
<Op1>	+, -	VARNAME, NUMBER, (, -
<Op2>	*, /	VARNAME, NUMBER, (, -
<ExprArith>	VARNAME, NUMBER, (, -	ENDLINE, .EQ., .GE., .GT., .LE., .LT., .NE., .AND.,), .OR., COMMA
<RecArithE>	ϵ , +, -	ENDLINE, .EQ., .GE., .GT., .LE., .LT., .NE., .AND.,), .OR., COMMA
<ArithT>	VARNAME, NUMBER, (, -	+, -, ENDLINE, .EQ., .GE., .GT., .LE., .LT., .NE., .AND.,), .OR., COMMA
<RecArithT>	ϵ , *, /	+, -, ENDLINE, .EQ., .GE., .GT., .LE., .LT., .NE., .AND.,), .OR., COMMA
<ArithF>	VARNAME, NUMBER, (, -	*, /, +, -, ENDLINE, .EQ., .GE., .GT., .LE., .LT.,

		.NE., .AND.,), .OR., COMMA
<If>	IF	ENDLINE
<FactIf>	ENDIF, ELSE	ENDLINE
<CondPrefix>	ε , .NOT.	VARNAME, NUMBER, (, -
<Cond>	.NOT., VARNAME, NUMBER, (, -)
<CondRecE>	ε , .OR.)
<CondT>	.NOT., VARNAME, NUMBER, (, -), .OR.
<CondRecT>	ε , .AND.), .OR.
<CondF>	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
<FactExprArith>	ε , COMMA	ENDLINE