Université libre de Bruxelles

Project - Part 2

Parser

Aldar Saranov, Przemyslaw Gasinski

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

INFO-F403 Introduction to language theory and compiling (M-INFOS/F277)

Gilles Geeraerts

November 2016

Initial grammar:

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

Making non-ambiguous

```
<Program>
     -> PROGRAM [ProgName] [EndLine] <Vars> <Code> END
<Vars>
     -> INTEGER <VarList> [EndLine]
     -> ε
<VarList>
     -> [VarName], <FactVarList>
<FactVarList>
     -> <VarList>
     -> ε
<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> END
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

String	First(String)
<Program> $	PROGRAM
PROGRAM [ProgName] [EndLine] <Vars> <Code>	PROGRAM
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] <Code> <FactIf>	IF
ENDIF	ENDIF
ELSE [EndLine] <Code> ENDIF	ELSE
.NOT.	.NOT.
<CondT> <CondRecE>	.NOT., VARNAME, NUMBER, (, -
.OR. <CondT> <CondRecE>	.OR.
.AND. <CondPrefix> <SimpleCond> <CondRecT>	.AND.
<ExprArith> <Comp> <ExprArith>	VARNAME, NUMBER, (, -
.EQ.	.EQ.
.GE.	.GE.
.GT.	.GT.
.LE.	.LE.
.LT.	.LT.
.NE.	.NE.
DO [VarName] = [Number], [Number] [EndLine] <Code> ENDDO	DO
PRINT*, <ExpList>	PRINT*,
READ*, <VarList>	READ*,
<ExprArith> <FactExprArith>	VARNAME, NUMBER, (, -
, <ExpList>	COMMA

Input	First	Follow
<All>	PROGRAM	ε
<Program>	PROGRAM	$
<Vars>	ε, INTEGER	END, VARNAME, IF,DO, PRINT, READ
<VarList>	VARNAME	ENDLINE
<FactVarList>	ε, VARNAME	ENDLINE
<Code>	ε, VARNAME, IF,DO, PRINT, READ	END, 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

Action table part 1

	$	VARNAME	INTEGER	NUMBER	PROGRAM	END	COMMA	EQUAL
<All>					0			
<Program>					1			
<Vars>		3	2			3		
<VarList>		4						
<FactVarList>		5						
<Code>		7				8		
<Instruction>		9						
<Assign>		14						
<Op1>								
<Op2>								
<ExprArith>		19		19				
<RecArithE>							21	
<ArithT>		22		22				
<RecArithT>							24	
<ArithF>		25		26				
<If>								
<FactIf>								
<CondPrefix>		33		33				
<Cond>		34		34				
<CondRecE>								
<CondT>		37		37				
<CondRecT>								
<CondF>								
<Comp>		40		40				
<Do>								
<Print>								
<Read>								
<ExpList>		50		50				
<FactExprArith							51	

Action table part 2

	(	)	MINUS	PLUS	TIMES	DIVIDE	IF	THEN
<All>								
<Program>								
<Vars>							3	
<VarList>								
<FactVarList>								
<Code>							7	
<Instruction>							10	
<Assign>								
<Op1>			16	15				
<Op2>					17	18		
<ExprArith>	19		19					
<RecArithE>		21	20	20				
<ArithT>	22		22					
<RecArithT>		24	24	24	23	23		
<ArithF>	27		28					
<If>							29	
<FactIf>								
<CondPrefix>	33		33					
<Cond>	34		34					
<CondRecE>		36						
<CondT>	37		37					
<CondRecT>		39						
<CondF>								
<Comp>	40		40					
<Do>								
<Print>								
<Read>								
<ExpList>	50		50					
<FactExprArith>								

Action table part 3

	ENDIF	ELSE	NOT	AND	OR	.EQ.	.GE.	.GR.	.LE.
<All>									
<Program>									
<Vars>									
<VarList>									
<FactVarList>									
<Code>	8	8							
<Instruction>									
<Assign>									
<Op1>									
<Op2>									
<ExprArith>									
<RecArithE>				21	21	21	21	21	21
<ArithT>									
<RecArithT>				24	24	24	24	24	24
<ArithF>									
<If>									
<FactIf>	30	31							
<CondPrefix>			32						
<Cond>			34						
<CondRecE>					35				

<CondT>			37						
<CondRecT>				38	39				
<CondF>									
<Comp>						41	42	43	44
<Do>									
<Print>									
<Read>									
<ExpList>									
<FactExprArith>									

Action table part 4

	.LT.	.NE.	DO	ENDDO	PRINT	READ	ENDLINE
<All>							
<Program>							
<Vars>			3		3	3	
<VarList>							
<FactVarList>							6
<Code>			7	8	7	7	
<Instruction>			11		12	13	
<Assign>							
<Op1>							
<Op2>							
<ExprArith>							
<RecArithE>	21	21					21
<ArithT>							
<RecArithT>	24	24					24
<ArithF>							
<If>							
<FactIf>							
<CondPrefix>							
<Cond>							
<CondRecE>							
<CondT>							
<CondRecT>							
<CondF>							
<Comp>	45	46					
<Do>			47				
<Print>					48		
<Read>						49	
<ExpList>							
<FactExprArith>							52