Université Libre de Bruxelles

MA1 computer science & engineering

2015-2016

Introduction to language theory and compiling : project part 2

Author:
Benjamin Akboka
Nazar Filipchuk

Professor:
Gilles Geeraerts

November 27, 2015

1 Introduction

In this part of the project, a parser as to be created. In order to do so, it will first be needed to modify the grammar. That means delete every unproductive and/or unreachable variables. Then the left recursion has to be managed and finally the grammar need to take into account the priority of operators.

The next step in the creation of the parser is the creation of the parsing table. For which, the $first^1$ and the $follow^1$ needs to be calculated. The final step will be the implementation into a java program.

2 The Modified grammar

2.1 Productivity

All variables are productive. Since there are a LOT of rules for each subsets represented underneath, will only written the difference between the last one. Here is the proof :
$V_0 = \phi$
$V_1 = \{V_0, < ExprArith >, < Op >, < BinOp >, < Comp >, < Print >, < Read >\}$
$V_2 = \{V_1, < Assign >, < SimpleCond >, < Instruction >\}$
$V_3 = \{V_2, < Cond >, < InstList >\}$
$V_4 = \{V_3, < Code >\}$
$V_5 = \{V_4, < Program >, < While >, < For >\}$

2.2 Reachable

All variables are reachable. Same thing as before, it only written the difference between the last set. Here is the proof :
$V_0 = \{S\}$
$V_1 = \{V_0, begin, < Code >, end\}$
$V_2 = \{V_1, < InstList >\}$
$V_3 = \{V_2, < Instruction >\}$
$V_4 = \{V_3, < Assign >, < if >, < While >, < For >, < Print >, < Read >\}$
$V_5 = \{V_4, < ExprArith >, :=, < Cond >\}$
$V_6 = \{V_5, < Op >, < BinOp >, < SimpleCond >\}$
$V_7 = \{V_6, < Comp >\}$

2.3 Removing left recursion

The first recursion to remove is on the rule 17. Here is the modified rule :

<ExprArith>	→ <A_1><A_2>
<A_1 >	→ [VarName]
	→ [Number]
	→ (<ExprArith>)
	→ - <ExprArith>
<A_2>	→ <Op><ExprArith>
	→ ϵ

The next one to remove is on the rule 24 :

<Cond>	→ $< B_1 >< B_2 >$
<B_1 >	→ not <SimpleCond>
	→ <SimpleCond>
<B_2 >	→ <BinOp><Cond>
	→ ϵ

2.4 Factorisation

In this step, the grammar will be factorised. Only two rule needs to be changed. If and InstList :

<If>	→ if <Cond> then <Code> <EndIf>
<EndIf>	→ fi
	→ else <Code> fi
<InstList>	→ <Instruction> <Z>
<Z>	→ ; <InstList>
	→ ϵ

2.5 Priorities and association

Now, it is time to take into consideration the priorities of the associations. Those are described in the project assignment. The rules will be redefined beginning by the lower priorities. Here are the modifications :

<ExprArith>	$\rightarrow$ <factor> <terms>
<terms>	$\rightarrow \epsilon$
	<AddSub> <ExprArith>
<factor>	$\rightarrow$ <A1> <factors>
<factors>	$\rightarrow \epsilon$
	$\rightarrow$ <MultiDiv> <factor>
	$\rightarrow$ <A1> <factors>
<A_1>	$\rightarrow$ [VarName]
	$\rightarrow$ [Number]
	$\rightarrow$ (<ExprArith>)
	$\rightarrow$ -<A_1>
<AddSub>	$\rightarrow$ +
	$\rightarrow$ -
<MultiDiv>	$\rightarrow$ *
	$\rightarrow$ /
<Cond>	$\rightarrow$ <B_1><B_2>
<B_1>	$\rightarrow$ <SimpleCond>
	$\rightarrow$ not <SimpleCond>
<B_2>	$\rightarrow$ or <Not>
	$\rightarrow$ <And>
<And>	$\rightarrow$ and<Not>
	$\rightarrow \epsilon$
<Not>	$\rightarrow$ not<Cond>
	$\rightarrow$ <Cond>
<SimpleCond>	$\rightarrow$ <ExprArith><Comp><ExprArith>

2.6 Final grammar

1.)	<Goal>	→ <Program>$
2.)	<Program>	→ begin <Code> end
3.)	<Code>	→ ϵ
4.)		→ <InstList>
5.)	<InstList>	→ <Instruction><Z>
6.)	<Z>	→ ; <InstList>
7.)		→ ϵ
8.)	<Instruction>	→ <Assign>
9.)		→ <If>
10.)		→ <While>
11.)		→ <For>
12.)		→ <Print>
13.)		→ <Read>
14.)	<Assign>	→ [VarName] := <ExprArith>
15.)	<ExprArith>	→ <factor> <terms>
16.)	<terms>	→ ϵ
17.)		<AddSub> <ExprArith>
18.)	<factor>	→ <A1> <factors>
19.)	<factors>	→ ϵ
20.)		<MultiDiv> <factor>
21.)		<A1> <factors>
22.)	<A_1>	→ [VarName]
23.)		→ [Number]
24.)		→ (<ExprArith>)
25.)		→ -<A_1>
26.)	<AddSub>	→ +
27.)		→ -
28.)	<MultiDiv>	→ *
29.)		→ /
30.)	<Cond>	→ <B_1><B_2>
31.)	<B_1>	→ <SimpleCond>
32.)		→ not <SimpleCond>
33.)	<B_2>	→ or <Cond>
34.)		→ <And>
35.)	<And>	→ and<Cond>
36.)		→ ϵ
37.)	<SimpleCond>	→ <ExprArith><Comp><ExprArith>

38.)	<If>	→ if <Cond> then <Code> <EndIf>
39.)	<EndIf>	→ fi
40.)		→ else <Code> fi
41.)	<Comp>	→ =
42.)		→ >=
43.)		→ >
44.)		→ <=
45.)		→ <
46.)		→ /=
47.)	<While>	→ while <Cond> do <Code> od
48.)	<For>	→ for [VarName] from <ExprArith> by <ExprArith> to
49.)		<ExprArith> do <Code> od
50.)	<Print>	→ print([VarName])
51.)	<Read>	→ read([VarName])

3 The parsing table

3.1 First and Follow

Symbol	$First^1(A)$	$Follow^1(A)$
<Goal>	begin	$
<Program>	begin	$
<Code>	ϵ, [VarName], if, while, for, print, read,	end, od, fi, else
<InstList>	[VarName], if, while, for, print, read	end, od, fi, else
<EndList>	;, ϵ	end, od, fi, else
<Instruction>	[VarName], if, while, for, print, read	end, od, fi, else, ;
<Assign>	[VarName]	end, od, fi, else ;
<ExprArith>	[VarName], [Number], (, -,	by, to, do, =, <=, <, >=, >, /=,), ;, od, fi, else, end, or, and, then
<terms>	ϵ, +, -	by, to, do, =, <=, <, >=, >, /=,), ;, od, fi, else, end, or, and, then
<factors>	ϵ , *, /, [VarName], [Number], (, -	+, -, by, to, do, =, <=, <, >=, >, /=,), ;, od, fi, else, end, or, and, then
<factor>	[VarName], [Number], (, -	+, -, by, to, do, =, <=, <, >=, >, /=,), ;, od, fi, else, end, or, and, then
$<A_1>$	[VarName], [Number], (, -	*, /, [VarName], [Number], (, -, +, by, to, do, =, <=, <, >=, >, /=,), ;, od, fi, else, end, or, and, then
<AddSub>	+, -	[VarName], [Number], (, -
<MultiDiv>	*, /	[VarName], [Number], (, -
<Cond>	[VarName], [Number], (, -, not,	do, then
$<B_1>$	not, [VarName], [Number], (, -	or, and, do, then
$<B_2>$	or, and, ϵ	do, then
<And>	and, ϵ	do, then
<SimpleCond>	[VarName], [Number], (, -	and, or, do, then
<If>	if	;, od, fi, else, end
<EndIf>	fi, else	end, fi, else, od ;
<Comp>	=, <=, <, >=, >, /=	[VarName], [Number], (, -
<While>	while	end, od, fi, else, od, ;
<For>	for	end, od, fi, else, od, ;
<Print>	print	end, od, fi, od, else, ;
<Read>	read	end, od, fi, od, else, ;

3.2 Predict set

1	<Goal> → <Program>$	begin
2	<Program> → begin <Code> end	begin
3	<Code> → ϵ	od, fi, else, end
4	<Code> → <InstList>	VarName, while, print, if, read, for
5	<InstList> → <Instruction><EndList>	VarName while, print, if, read, for
6	<EndList> → ; <InstList>	;
7	<EndList> → ϵ	od, fi, else, end
8	<Instruction> → <Assign>	VarName
9	<Instruction> → <If>	if
10	<Instruction> → <While>	while
11	<Instruction> → <For>	for
12	<Instruction> → <Print>	print
13	<Instruction> → <Read>	read
14	<Assign> → [VarName] := <ExprArith>	VarName
15	<ExprArith> → <A_1><A_2>	VarName, Number, (, -
16	<terms> → ϵ	by, to, do, =, <=, <, >=, >, /=,), ;, od, fi, else, end, or, and, then
17	<terms> → <AddSub><ExprErith>	+,-
18	<factor> → <A1> <factors>	VarName, Number, (, -
19	<factors> → ϵ	+, -, by, to, do, =, <=, <, >=, >, /=,), ;, od, fi, else, end, or, and, then
20	<factors> → <MultiDiv> <factor>	*, /
21	<factors> → <A1> <factors>	VarName, Number, (, -
22	<A_1> → [VarName]	VarName
23	<A_1> → [Number]	Number
24	<A_1> → (<ExprArith>)	(
25	<A_1> → -<A_1>	-
26	<AddSub> → +	+
27	<AddSub> → -	-
28	<MultiDiv> → *	*
29	<MultiDiv> → /	/
30	<Cond> → <B_1><B_2>	VarName, Number, (, -
31	<B_2> → or <Cond>	or
32	<B_2> → <And>	and
33	<B_1> → <SimpleCond>	VarName, Number, (, -
34	<B_1> → not <SimpleCond>	not

35	<And> → and<Not>	and
36	<And> → ϵ	do, then, VarName, Number, (, -
37	<SimpleCond> → <ExprArith><Comp><ExprArith>	VarName, Number, (, -
38	<If> → if <Cond> then <Code> <EndIf>	if
39	<EndIf> → fi	fi
40	<EndIf> → else <Code> fi	else
41	<Comp> → =	=
42	<Comp> → >=	>=
43	<Comp> → >	>
44	<Comp> → <=	<=
45	<Comp> → <	<
46	<Comp> → /=	/=
47	<While> → while <Cond> do <Code> od	while
48	<For> → for [VarName] from <ExprArith> by <ExprArith> to	for
49	<ExprArith> do <Code> od	
50	<Print> → print([VarName])	print
51	<Read> → read([VarName])	read