

Yet Another Language for the Compiler Course
Introduction to language theory and compiling
Project – Part 2

Gilles GEERAERTS

Arnaud LEPONCE

2025-2026

The grammar is transformed as follows:

- Eliminate rule $\langle \text{Instruction} \rangle \rightarrow \langle \text{Call} \rangle$ as $\langle \text{Call} \rangle$ is an unproductive nonterminal.
- Separate rules for $\langle \text{ExprArith} \rangle$ according to operators precedence. The $\langle \text{Op} \rangle$ nonterminal has been removed in the process. Then transform these to remove left-recursion.
- Separate rules for $\langle \text{Cond} \rangle$ according to operators precedence (in this case there is only grouping between $|$ s and \rightarrow), then remove left-recursion. We also ensured that \rightarrow is right-associative with rule 26: in the produced parsing tree, the right hand-side of an implication will be in a $\langle \text{Cond} \rangle$ subtree.

The resulting grammar is given in Figure ???. We can show that this grammar is *LL1* by showing it is *Strong LL1*. To do this we compute for each rule $\alpha \rightarrow \beta$ the set $\text{First}^1(\beta \cdot \text{Follow}^1(\alpha))$ (see Table ??). Then for each pair of rules with the same left handside $\alpha \rightarrow \beta$ and $\alpha \rightarrow \beta'$ we check that

$$\text{First}^1(\beta \cdot \text{Follow}^1(\alpha)) \cap \text{First}^1(\beta' \cdot \text{Follow}^1(\alpha)) = \emptyset.$$

As we see, there is no conflict. Indeed, we can use these to fill up the $\text{LL}(1)$ action table of Table ??.

[1]	$\langle \text{Program} \rangle$	$\rightarrow \text{Prog} [\text{ProgName}] \text{ Is } \langle \text{Code} \rangle \text{ End}$
[2]	$\langle \text{Code} \rangle$	$\rightarrow \langle \text{Instruction} \rangle ; \langle \text{Code} \rangle$
[3]		$\rightarrow \varepsilon$
[4]	$\langle \text{Instruction} \rangle$	$\rightarrow \langle \text{Assign} \rangle$
[5]		$\rightarrow \langle \text{If} \rangle$
[6]		$\rightarrow \langle \text{While} \rangle$
[7]		$\rightarrow \langle \text{Output} \rangle$
[8]		$\rightarrow \langle \text{Input} \rangle$
[9]	$\langle \text{Assign} \rangle$	$\rightarrow [\text{VarName}] = \langle \text{ExprArith} \rangle$
[10]	$\langle \text{ExprArith} \rangle$	$\rightarrow \langle \text{Prod} \rangle \langle \text{ExprArith}' \rangle$
[11]	$\langle \text{ExprArith}' \rangle$	$\rightarrow + \langle \text{Prod} \rangle \langle \text{ExprArith}' \rangle$
[12]		$\rightarrow - \langle \text{Prod} \rangle \langle \text{ExprArith}' \rangle$
[13]		$\rightarrow \varepsilon$
[14]	$\langle \text{Prod} \rangle$	$\rightarrow \langle \text{Atom} \rangle \langle \text{Prod}' \rangle$
[15]	$\langle \text{Prod}' \rangle$	$\rightarrow * \langle \text{Atom} \rangle \langle \text{Prod}' \rangle$
[16]		$\rightarrow / \langle \text{Atom} \rangle \langle \text{Prod}' \rangle$
[17]		$\rightarrow \varepsilon$
[18]	$\langle \text{Atom} \rangle$	$\rightarrow [\text{VarName}]$
[19]		$\rightarrow [\text{Number}]$
[20]		$\rightarrow (\langle \text{ExprArith} \rangle)$
[21]		$\rightarrow - \langle \text{Atom} \rangle$
[22]	$\langle \text{If} \rangle$	$\rightarrow \text{If } \{ \langle \text{Cond} \rangle \} \text{ Then } \langle \text{Code} \rangle \langle \text{IfTail} \rangle$
[23]	$\langle \text{IfTail} \rangle$	$\rightarrow \text{End}$
[24]		$\rightarrow \text{Else } \langle \text{Code} \rangle \text{ End}$
[25]	$\langle \text{Cond} \rangle$	$\rightarrow \langle \text{SimpleCond} \rangle \langle \text{Cond}' \rangle$
[26]	$\langle \text{Cond}' \rangle$	$\rightarrow \rightarrow \langle \text{Cond} \rangle$
[27]		$\rightarrow \varepsilon$
[28]	$\langle \text{SimpleCond} \rangle$	$\rightarrow \langle \text{Cond} \rangle $
[29]		$\rightarrow \langle \text{ExprArith} \rangle \langle \text{Comp} \rangle \langle \text{ExprArith} \rangle$
[30]	$\langle \text{Comp} \rangle$	$\rightarrow ==$
[31]		$\rightarrow <=$
[32]		$\rightarrow <$
[33]	$\langle \text{While} \rangle$	$\rightarrow \text{While } \{ \langle \text{Cond} \rangle \} \text{ Do } \langle \text{Code} \rangle \text{ End}$
[34]	$\langle \text{Output} \rangle$	$\rightarrow \text{Print}([\text{VarName}])$
[35]	$\langle \text{Input} \rangle$	$\rightarrow \text{Input}([\text{VarName}])$

Table 1: The YALCC modified grammar.

$\text{First}^1(\text{Prog} \ [\text{ProgName}] \ \text{Is} < \text{Code} > \text{End} \ \text{Follow}^1(< \text{Program} >)) = \{\text{Prog}\}$	(1)
$\text{First}^1(< \text{Instruction} >; < \text{Code} > \ \text{Follow}^1(< \text{Code} >)) = \{[\text{VarName}], \text{If}, \text{While}, \text{Print}, \text{Input}\}$	(2)
$\text{First}^1(\varepsilon \ \text{Follow}^1(< \text{Code} >)) = \{\text{End}, \text{Else}\}$	(3)
$\text{First}^1(< \text{Assign} > \ \text{Follow}^1(< \text{Instruction} >)) = \{[\text{VarName}]\}$	(4)
$\text{First}^1(< \text{If} > \ \text{Follow}^1(< \text{Instruction} >)) = \{\text{If}\}$	(5)
$\text{First}^1(< \text{While} > \ \text{Follow}^1(< \text{Instruction} >)) = \{\text{While}\}$	(6)
$\text{First}^1(< \text{Output} > \ \text{Follow}^1(< \text{Instruction} >)) = \{\text{Print}\}$	(7)
$\text{First}^1(< \text{Input} > \ \text{Follow}^1(< \text{Instruction} >)) = \{\text{Input}\}$	(8)
$\text{First}^1([\text{VarName}] = < \text{ExprArith} > \ \text{Follow}^1(< \text{Assign} >)) = \{[\text{VarName}]\}$	(9)
$\text{First}^1(< \text{Prod} > < \text{ExprArith}' > \ \text{Follow}^1(< \text{ExprArith} >)) = \{[\text{VarName}], [\text{Number}], (, -)\}$	(10)
$\text{First}^1(+ < \text{Prod} > < \text{ExprArith}' > \ \text{Follow}^1(< \text{ExprArith}' >)) = \{+\}$	(11)
$\text{First}^1(- < \text{Prod} > < \text{ExprArith}' > \ \text{Follow}^1(< \text{ExprArith}' >)) = \{-\}$	(12)
$\text{First}^1(\varepsilon \ \text{Follow}^1(< \text{ExprArith}' >)) = \{;, , ==, <=, <, , ->, \}\}$	(13)
$\text{First}^1(< \text{Atom} > < \text{Prod}' > \ \text{Follow}^1(< \text{Prod} >)) = \{[\text{VarName}], [\text{Number}], (, -)\}$	(14)
$\text{First}^1(* < \text{Atom} > < \text{Prod}' > \ \text{Follow}^1(< \text{Prod}' >)) = \{*\}$	(15)
$\text{First}^1(/ < \text{Atom} > < \text{Prod}' > \ \text{Follow}^1(< \text{Prod}' >)) = \{/ \}$	(16)
$\text{First}^1(\varepsilon \ \text{Follow}^1(< \text{Prod}' >)) = \{+, -, ;, , ==, <=, <, , ->, \}\}$	(17)
$\text{First}^1([\text{VarName}] \ \text{Follow}^1(< \text{Atom} >)) = \{[\text{VarName}]\}$	(18)
$\text{First}^1([\text{Number}] \ \text{Follow}^1(< \text{Atom} >)) = \{[\text{Number}]\}$	(19)
$\text{First}^1((< \text{ExprArith} >) \ \text{Follow}^1(< \text{Atom} >)) = \{\{\}\}$	(20)
$\text{First}^1(- < \text{Atom} > \ \text{Follow}^1(< \text{Atom} >)) = \{-\}$	(21)
$\text{First}^1(\text{If} \{ < \text{Cond} > \} \text{Then} < \text{Code} > < \text{IfTail} > \ \text{Follow}^1(< \text{If} >)) = \{\text{If}\}$	(22)
$\text{First}^1(\text{End} \ \text{Follow}^1(< \text{IfTail} >)) = \{\text{End}\}$	(23)
$\text{First}^1(\text{Else} < \text{Code} > \text{End} \ \text{Follow}^1(< \text{IfTail} >)) = \{\text{Else}\}$	(24)
$\text{First}^1(< \text{SimpleCond} > < \text{Cond}' > \ \text{Follow}^1(< \text{Cond} >)) = \{ , [\text{VarName}], [\text{Number}], (, -)\}$	(25)
$\text{First}^1(- > < \text{Cond} > \ \text{Follow}^1(< \text{Cond}' >)) = \{->\}$	(26)
$\text{First}^1(\varepsilon \ \text{Follow}^1(< \text{Cond}' >)) = \{\}, \}$	(27)
$\text{First}^1(< \text{Cond} > \ \text{Follow}^1(< \text{SimpleCond} >)) = \{ \}$	(28)
$\text{First}^1(< \text{ExprArith} > < \text{Comp} > < \text{ExprArith} > \ \text{Follow}^1(< \text{SimpleCond} >)) = \{[\text{VarName}], [\text{Number}], (, -)\}$	(29)
$\text{First}^1(== \ \text{Follow}^1(< \text{Comp} >)) = \{==\}$	(30)
$\text{First}^1(<= \ \text{Follow}^1(< \text{Comp} >)) = \{<=\}$	(31)
$\text{First}^1(< \ \text{Follow}^1(< \text{Comp} >)) = \{<\}$	(32)
$\text{First}^1(\text{While} \{ < \text{Cond} > \} \text{Do} < \text{Code} > \text{End} \ \text{Follow}^1(< \text{While} >)) = \{\text{While}\}$	(33)
$\text{First}^1(\text{Print}([\text{VarName}]) \ \text{Follow}^1(< \text{Output} >)) = \{\text{Print}\}$	(34)
$\text{First}^1(\text{Input}([\text{VarName}]) \ \text{Follow}^1(< \text{Output} >)) = \{\text{Input}\}$	(35)

Table 2: Sets $\text{First}^1(\beta \cdot \text{Follow}^1(\alpha))$ for all rules $\alpha \rightarrow \beta$.

	Prog	Is	End	If	Then	Else	While	Do	Print	Input	;	=	+	-	*	/	()	{	}	==	<=	<	->		[ProgName]	[VarName]	[Number]		
<Program>	1																													
<Code>		3	2		3	2		2	2																		2			
<Instruction>			5			6		7	8																		4			
<Assign>																												9		
<ExprArith>															10			10									10	10		
<ExprArith'>												13		11	12					13		13	13	13	13	13				
<Prod>																14			14								14	14		
<Prod'>												17		17	17	15	16			17		17	17	17	17	17				
<Atom>																	21			20							18	19		
<If>			22																											
<IfTail>		23			24																									
<Cond>																25			25								25	25	25	
<Cond'>																												26	27	
<SimpleCond>																	29			29							28	29	29	
<Comp>																														
<While>							33																	30	31	32				
<Output>								34																						
<Input>									35																					

Table 3: Action table