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 <Instruction> → <Call> as <Call> is an unproductive nonterminal.
- Separate rules for <ExprArith> according to operators precedence. The <Op> nonterminal has been removed in the process. Then transform these to remove left-recursion.
- Separate rules for <Cond> according to operators precedence (in this case there is only grouping between |s and ->), then remove left-recursion. We also ensured that -> is right-associative with rule 26: in the produced parsing tree, the right hand-side of an implication will be in a <Cond> 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 $\alpha \to \beta$ the set $\text{First}^1(\beta \cdot \text{Follow}^1(\alpha))$ (see Table **??**). Then for each pair of rules with the same left handside $\alpha \to \beta$ and $\alpha \to \beta'$ 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 LL(1) action table of Table **??**.

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

Table 1: The YALCC modified grammar.

$$\text{First}^1(\texttt{Prog [ProgName] Is<Code>End Follow}^1(\texttt{<Program>})) = \{\texttt{Prog}\} \quad (1)$$

$$\text{First}^1(\texttt{<Instruction>;<Code> Follow}^1(\texttt{<Code>})) = \{\texttt{[VarName]}, \texttt{If}, \texttt{While}, \texttt{Print}, \texttt{Input}\} \quad (2)$$

$$\text{First}^1(\varepsilon \text{ Follow}^1(\texttt{<Code>})) = \{\texttt{End}, \texttt{Else}\} \quad (3)$$

$$\text{First}^1(\texttt{<Assign>} \text{ Follow}^1(\texttt{<Instruction>})) = \{\texttt{[VarName]}\} \quad (4)$$

$$\text{First}^1(\texttt{<If>} \text{ Follow}^1(\texttt{<Instruction>})) = \{\texttt{If}\} \quad (5)$$

$$\text{First}^1(\texttt{<While>} \text{ Follow}^1(\texttt{<Instruction>})) = \{\texttt{While}\} \quad (6)$$

$$\text{First}^1(\texttt{<Output>} \text{ Follow}^1(\texttt{<Instruction>})) = \{\texttt{Print}\} \quad (7)$$

$$\text{First}^1(\texttt{<Input>} \text{ Follow}^1(\texttt{<Instruction>})) = \{\texttt{Input}\} \quad (8)$$

$$\text{First}^1(\texttt{[VarName]=<ExprArith>} \text{ Follow}^1(\texttt{<Assign>})) = \{\texttt{[VarName]}\} \quad (9)$$

$$\text{First}^1(\texttt{<Prod><ExprArith'>} \text{ Follow}^1(\texttt{<ExprArith>})) = \{\texttt{[VarName]}, \texttt{[Number]}, \texttt{(}, \texttt{-}\} \quad (10)$$

$$\text{First}^1(\texttt{+<Prod><ExprArith'>} \text{ Follow}^1(\texttt{<ExprArith'>})) = \{\texttt{+}\} \quad (11)$$

$$\text{First}^1(\texttt{-<Prod><ExprArith'>} \text{ Follow}^1(\texttt{<ExprArith'>})) = \{\texttt{-}\} \quad (12)$$

$$\text{First}^1(\varepsilon \text{ Follow}^1(\texttt{<ExprArith'>})) = \{\texttt{;}, \texttt{)}, \texttt{==}, \texttt{<=}, \texttt{<}, \texttt{|}, \texttt{->}, \texttt{}\} \quad (13)$$

$$\text{First}^1(\texttt{<Atom><Prod'>} \text{ Follow}^1(\texttt{<Prod>})) = \{\texttt{[VarName]}, \texttt{[Number]}, \texttt{(}, \texttt{-}\} \quad (14)$$

$$\text{First}^1(\texttt{*<Atom><Prod'>} \text{ Follow}^1(\texttt{<Prod'>})) = \{\texttt{*}\} \quad (15)$$

$$\text{First}^1(\texttt{/<Atom><Prod'>} \text{ Follow}^1(\texttt{<Prod'>})) = \{\texttt{/}\} \quad (16)$$

$$\text{First}^1(\varepsilon \text{ Follow}^1(\texttt{<Prod'>})) = \{\texttt{+}, \texttt{-}, \texttt{;}, \texttt{)}, \texttt{==}, \texttt{<=}, \texttt{<}, \texttt{|}, \texttt{->}, \texttt{}\} \quad (17)$$

$$\text{First}^1(\texttt{[VarName]} \text{ Follow}^1(\texttt{<Atom>})) = \{\texttt{[VarName]}\} \quad (18)$$

$$\text{First}^1(\texttt{[Number]} \text{ Follow}^1(\texttt{<Atom>})) = \{\texttt{[Number]}\} \quad (19)$$

$$\text{First}^1(\texttt{(<ExprArith>)} \text{ Follow}^1(\texttt{<Atom>})) = \{\texttt{(}\} \quad (20)$$

$$\text{First}^1(\texttt{-<Atom>} \text{ Follow}^1(\texttt{<Atom>})) = \{\texttt{-}\} \quad (21)$$

$$\text{First}^1(\texttt{If\{<Cond>\}Then<Code><IfTail>} \text{ Follow}^1(\texttt{<If>})) = \{\texttt{If}\} \quad (22)$$

$$\text{First}^1(\texttt{End} \text{ Follow}^1(\texttt{<IfTail>})) = \{\texttt{End}\} \quad (23)$$

$$\text{First}^1(\texttt{Else<Code>End} \text{ Follow}^1(\texttt{<IfTail>})) = \{\texttt{Else}\} \quad (24)$$

$$\text{First}^1(\texttt{<SimpleCond><Cond'>} \text{ Follow}^1(\texttt{<Cond>})) = \{\texttt{|}, \texttt{[VarName]}, \texttt{[Number]}, \texttt{(}, \texttt{-}\} \quad (25)$$

$$\text{First}^1(\texttt{-><Cond>} \text{ Follow}^1(\texttt{<Cond'>})) = \{\texttt{->}\} \quad (26)$$

$$\text{First}^1(\varepsilon \text{ Follow}^1(\texttt{<Cond'>})) = \{\texttt{}, \texttt{|}\} \quad (27)$$

$$\text{First}^1(\texttt{|<Cond>|} \text{ Follow}^1(\texttt{<SimpleCond>})) = \{\texttt{|}\} \quad (28)$$

$$\text{First}^1(\texttt{<ExprArith><Comp><ExprArith>} \text{ Follow}^1(\texttt{<SimpleCond>})) = \{\texttt{[VarName]}, \texttt{[Number]}, \texttt{(}, \texttt{-}\} \quad (29)$$

$$\text{First}^1(\texttt{==} \text{ Follow}^1(\texttt{<Comp>})) = \{\texttt{==}\} \quad (30)$$

$$\text{First}^1(\texttt{<=} \text{ Follow}^1(\texttt{<Comp>})) = \{\texttt{<=}\} \quad (31)$$

$$\text{First}^1(\texttt{<} \text{ Follow}^1(\texttt{<Comp>})) = \{\texttt{<}\} \quad (32)$$

$$\text{First}^1(\texttt{While\{<Cond>\}Do<Code>End} \text{ Follow}^1(\texttt{<While>})) = \{\texttt{While}\} \quad (33)$$

$$\text{First}^1(\texttt{Print([VarName])} \text{Follow}^1(\texttt{<Output>})) = \{\texttt{Print}\} \quad (34)$$

$$\text{First}^1(\texttt{Input([VarName])} \text{Follow}^1(\texttt{<Output>})) = \{\texttt{Input}\} \quad (35)$$

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

	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	13			
<Prod>														14			14										14	14
<Prod'>											17		17	17	15	16		17		17	17	17	17	17	17			
<Atom>														21			20										18	19
<If>				22																								
<IfTail>			23			24																						
<Cond>														25			25								25		25	25
<Cond'>																				27				26	27			
<SimpleCond>														29			29								28		29	29
<Comp>																					30	31	32					
<While>							33																					
<Output>									34																			
<Input>										35																		

Table 3: Action table