$S \to Aa \mid b$

$A \to Ac \mid Sd \mid \varepsilon$

$S \Rightarrow_{lm} Aa \Rightarrow_{lm} Sda$

Elimination of left recursion

Create an order between non-terminals

1) S

2) A

$i = 1$

the symbol S does not have immediate left recursion

skip

$i = 2$

$A \to \overset{\alpha_1}{A c} \mid \overset{\alpha_2}{A ad} \mid \overset{\beta_1}{bd} \mid \overset{\beta_2}{\varepsilon}$

$\mapsto$

$A \to bdA' \mid A'$

$A' \to cA' \mid adA' \mid \varepsilon$

end

Left factoring $\qquad \Sigma = \{i, t, e, a, b\}$

$$S \to \underbrace{iEtS} \mid \underbrace{iEtSeS} \mid a$$

$$E \to b$$

$\leadsto$

$$S \to iEtS\,S'$$

$$S' \to \varepsilon \mid eS$$

$$E \to b$$

not $\underline{LL(1)}$

n° of lookahead symbols

Left-recursive

Left-to Right

stmt → expr ;
| if (expr) stmt
| for (optexpr ; optexpr ; optexpr) stmt
| other

optexpr → expr | ε

for (; expr ; expr) other $

stmt

0) call stmt()

1) chose stmt → for (optexpr ; optexpr ; optexpr) stm

2) match for → lookahead (

3) match (→ lookohed ;

4) call optexpr()

↳ 1) choose optexpr → ε

2) return

5) match ; → lookahead expr

6) call optexpr()

↳ 1) optexpr → expr

2) match expr → lookahead ;

3) return

7) match ; → lookohed expr → 1) choose stmt → other

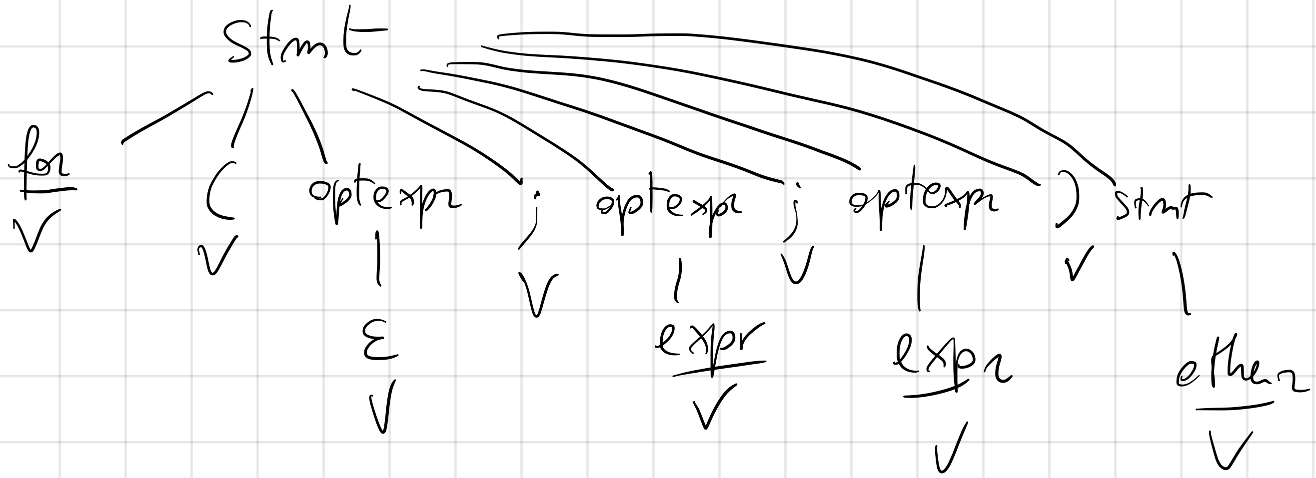
2) match other

3) return

8) match)

9) call stmt()

10) lookohed is $ and production is completed OK



$$\text{stmt} \underset{lm}{\Rightarrow} \underline{\text{for}}\ (\ \text{optexpr}\ ;\ \text{optexpr}\ ;\ \text{optexpr}\ )\ \text{stmt} \underset{lm}{\Rightarrow}$$

$$\underline{\text{for}}\ (\ ;\ \text{optexpr}\ ;\ \text{optexpr}\ )\ \text{stmt} \underset{lm}{\Rightarrow} \underline{\text{for}}\ (\ ;\ \underline{\text{expr}}\ ;\ \text{optexpr}\ )\ \text{stm}$$

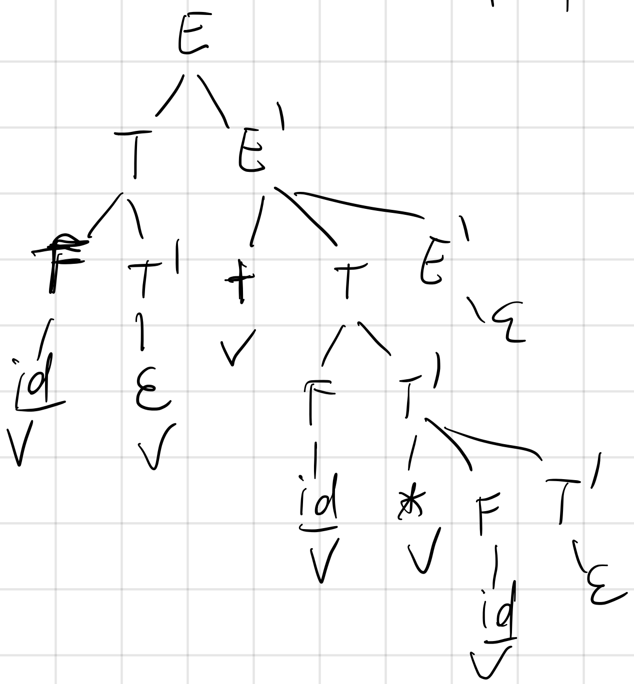
$$\underset{lm}{\Rightarrow} \underline{\text{for}}\ (\ ;\ \underline{\text{expr}}\ ;\ \underline{\text{expr}}\ )\ \text{stmt} \underset{lm}{\Rightarrow} \underline{\text{for}}\ (\ ;\ \underline{\text{expr}}\ ;\ \underline{\text{expr}}\ )\ \underline{\text{othe}}$$

$E \to E + T \mid T$

$T \to T * F \mid F$

$F \to id \mid (E)$

$id + id * id \ \$$



$E \to TE'$

$E' \to +TE' \mid \varepsilon$

$T \to FT'$

$T' \to *FT' \mid \varepsilon$

$F \to id \mid (E)$

$\text{FIRST}(E) = \{ id, (\}$

$\text{FIRST}(E') = \{ \varepsilon, + \}$

$\text{FIRST}(T) = \{ id, (\}$

$\text{FIRST}(T') = \{ \varepsilon, * \}$

$\text{FIRST}(F) = \{ id, (\}$

e.g. $\text{FIRST}(T' + id) = \{ *, + \}$

$E \to TE'$

$E' \to +TE' \mid \varepsilon$

$T \to FT'$

$T' \to *FT' \mid \varepsilon$

$F \to (E) \mid id$

$\text{Follow}(E) = \{ \$,) \}$

$\text{Follow}(E') = \{ \$,) \}$

$\text{Follow}(T) = \{ +, \$,) \}$

$\text{Follow}(T') = \{ +, \$,) \}$

$\text{Follow}(F) = \{ *, +, \$,) \}$

	id	+	*	(	)	\$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \varepsilon$	$T' \to *FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to id$			$F \to (E)$		

Parsing table for recursive descent parser. The parser can be predictive. The grammar is LL(1).