Lecture 11

Monday February 10

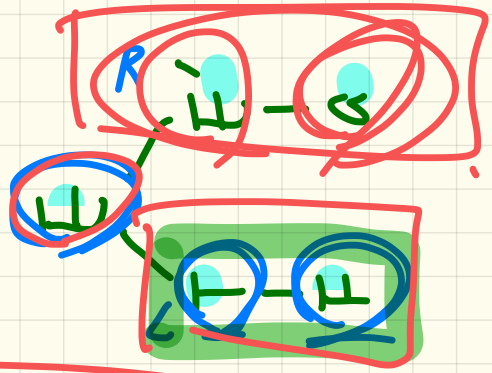
varargs variable argument

```
public ASTNode(String label, ASTNode ...children) {
    /* Your Task */
}
```

ASTNode[] children

children.length

children[i]



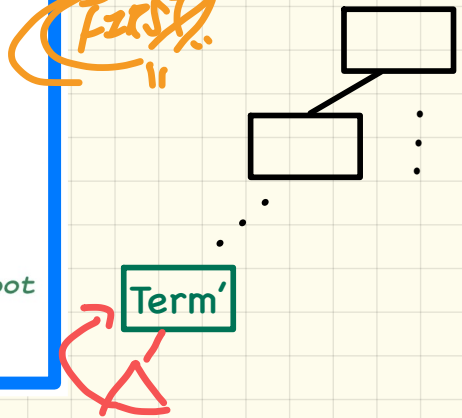

```
new ASTNode("E",
    new ASTNode("T",
        new ASTNode("F")
    ),
    new ASTNode("E", new ASTNode("a"))
)
```

Top-Down Parsing: Backtrack

backtrack ≜ pop *focus*.siblings; *focus* := *focus*.parent; *focus*.resetChildren

```
ALGORITHM: TDParse
  INPUT: CFG G = (V, Σ, R, S)
  OUTPUT: Root of a Parse Tree or Syntax Error
PROCEDURE:
  root := a new node for the start symbol S
  focus := root
  initialize an empty stack trace
  trace.push(null)
  word := NextWord()
  while (true):
    if focus ∈ V then
      if ∃ unvisited rule focus → β1 β2 ... βn ∈ R then
        create β1, β2 ... βn as children of focus
        trace.push(βn βn−1 ... β2)
        focus := β1
      else
        if focus = S then report syntax error
        else backtrack
        end
      end
    elseif word matches focus then
      word := NextWord()
      focus := trace.pop()
    elseif word = EOF ∧ focus = null then return root
    else backtrack
    end
```

0	*Goal*	→	*Expr*
1	*Expr*	→	*Term Expr′*
2	*Expr′*	→	+ *Term Expr′*
3		|	- *Term Expr′*
4		|	ϵ
5	*Term*	→	*Factor Term′*
6	*Term′*	→	× *Factor Term′*
7		|	÷ *Factor Term′*
8		|	ϵ
9	*Factor*	→	(*Expr*)
10		|	num
11		|	name



FIRST Set

$$\textsc{First}(\alpha) = \begin{cases} \{\alpha\} & \text{if } \alpha \in T \\ \{w \mid w \in \Sigma^* \wedge \alpha \Rightarrow^* w\beta \wedge \beta \in (V \cup \Sigma)^*\} & \text{if } \alpha \in V \end{cases}$$

Right-Recursive CFG:

0	*Goal*	→	*Expr*	6	*Term′*	→	× *Factor Term′*
1	*Expr*	→	*Term Expr′*	7		|	÷ *Factor Term′*
2	*Expr′*	→	+ *Term Expr′*	8		|	ϵ
3		|	- *Term Expr′*	9	*Factor*	→	(*Expr*)
4		|	ϵ	10		|	num
5	*Term*	→	*Factor Term′*	11		|	name

	num	name	+	-	×	÷	(	)	eof	ϵ
FIRST	num	name	+	-	x	÷	(	)	eof	ϵ

	Expr	*Expr′*	*Term*	*Term′*	*Factor*
FIRST	(, name, num	+, -, ϵ	(, name, num	x, ÷, ϵ	(, name, num

FIRST Set: Algorithm

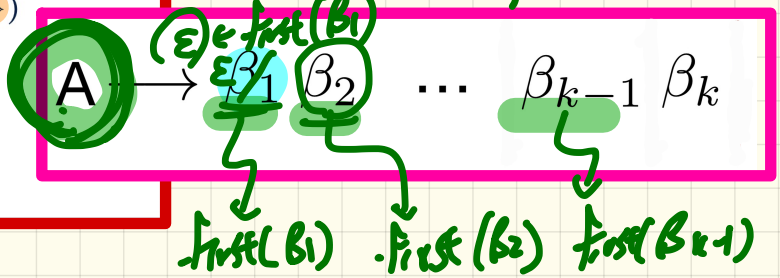
$$\textsc{First}(\alpha) = \begin{cases} \{\alpha\} & \text{if } \alpha \in T \\ \{w \mid w \in \Sigma^* \wedge \alpha \overset{*}{\Rightarrow} w\beta \wedge \beta \in (V \cup \Sigma)^*\} & \text{if } \alpha \in V \end{cases}$$

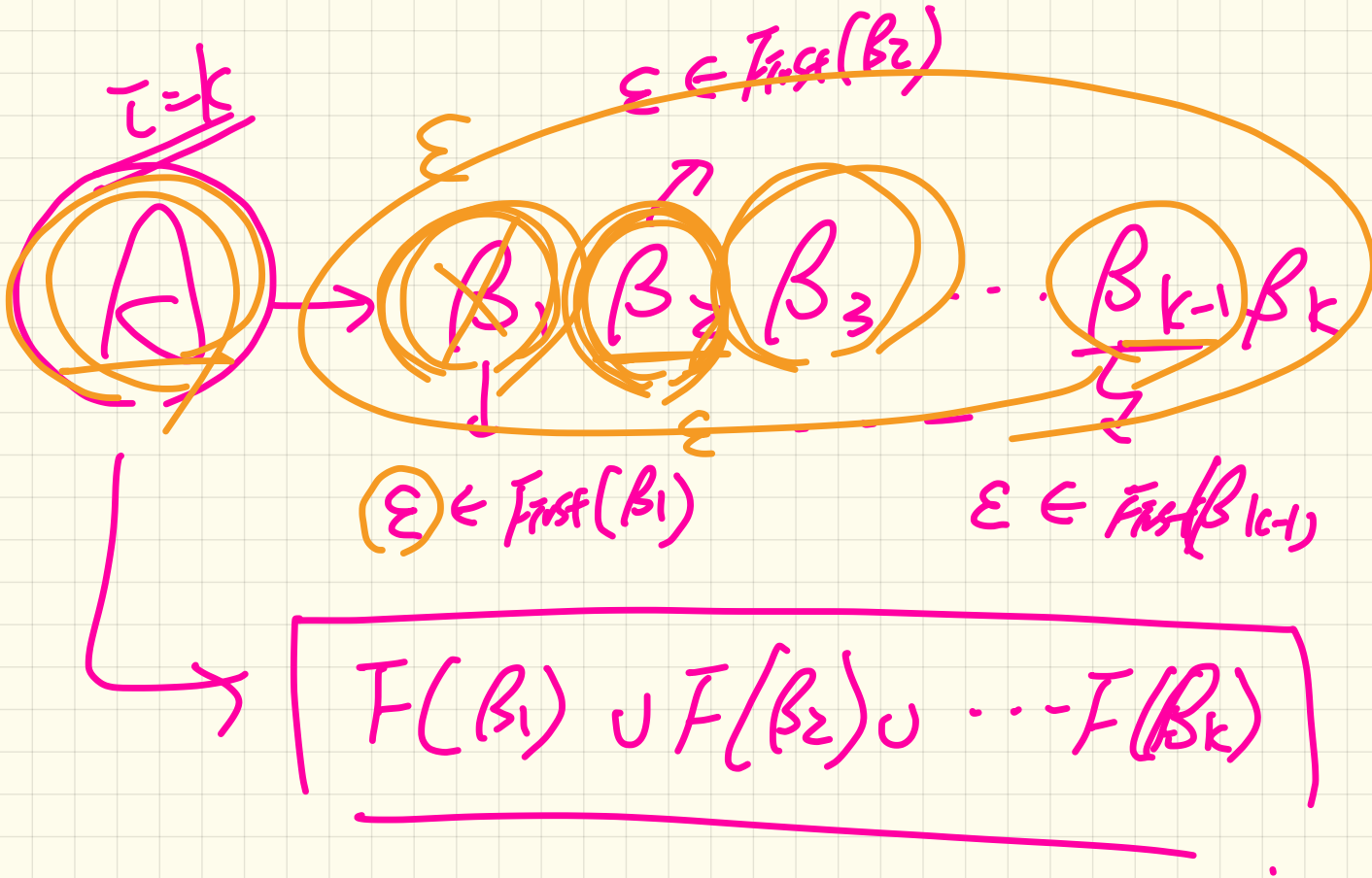
ALGORITHM: *GetFirst*
INPUT: *CFG* $G = (V, \Sigma, R, S)$
$T \subset \Sigma^*$ *denotes valid terminals*
OUTPUT: $\textsc{First} : V \cup T \cup \{\epsilon, eof\} \longrightarrow \mathbb{P}(T \cup \{\epsilon, eof\})$
PROCEDURE:
for $\alpha \in (T \cup \{eof, \epsilon\})$: $\textsc{First}(\alpha) := \{\alpha\}$
for $A \in V$: $\textsc{First}(A) := \varnothing$
lastFirst := FIRST
while (*lastFirst* ≠ FIRST) :
  for $A \to \beta_1 \beta_2 \ldots \beta_k \in R$ s.t. $\forall \beta_j : \beta_j \in (T \cup V)$:
    rhs := $\textsc{First}(\beta_1) - \{\epsilon\}$
    for $i := 1$; $\epsilon \in \textsc{First}(\beta_i) \wedge i < k$; $i{+}{+}$:
      rhs := rhs ∪ ($\textsc{First}(\beta_{i+1}) - \{\epsilon\}$)
    if $i = k \wedge \epsilon \in \textsc{First}(\beta_k)$ **then**
      rhs := rhs ∪ {ϵ}
    end
    $\textsc{First}(A) := \textsc{First}(A) \cup rhs$
  lastFirst := FIRST

β_i is nullable

$\beta_1 \ldots \beta_{k-1}$ are nullable

$\textsc{First}(A) = \textsc{First}(\beta_1) \cup \textsc{First}(\beta_2)$




i=k
A
ε
ε ∈ First(B2)
B1
B2
B3
Bk-1
Bk
ε ∈ First(B1)
ε ∈ First(Bk-1)
F(B1) ∪ F(B2) ∪ ··· F(Bk)

Right-Recursive CFG:

0	*Goal*	→	*Expr*	6	*Term′*	→	× *Factor Term′*
1	*Expr*	→	*Term Expr′*	7		|	÷ *Factor Term′*
2	*Expr′*	→	\+ *Term Expr′*	8		|	ϵ
3		|	\- *Term Expr′*	9	*Factor*	→	(*Expr*)
4		|	ϵ	10		|	num
5	*Term*	→	*Factor Term′*	11		|	name

F, E′, T′, T, E

FIRST Set: Tracing

First choose rules whose RHS starts with a terminal

ALGORITHM: *GetFirst*
INPUT: *CFG* $G = (V, \Sigma, R, S)$
$T \subset \Sigma^*$ *denotes valid terminals*
OUTPUT: FIRST : $V \cup T \cup \{\epsilon, eof\} \longrightarrow \mathbb{P}(T \cup \{\epsilon, eof\})$
PROCEDURE:
for $\alpha \in (T \cup \{eof, \epsilon\})$: FIRST := $\{\alpha\}$
for $A \in V$: FIRST := $\varnothing$
lastFirst := FIRST
while (*lastFirst* ≠ FIRST):
for $A \to \beta_1 \beta_2 \ldots \beta_k \in R$ s.t. $\forall \beta_j : \beta_j \in (T \cup V)$:
rhs := FIRST(β_1) − {ϵ}
for (*i* := 1; ϵ ∈ FIRST(β_i) ∧ *i* < *k*; *i*++):
rhs := *rhs* ∪ (FIRST(β_{i+1}) − {ϵ})
if *i* = *k* ∧ ϵ ∈ FIRST(β_k) then
rhs := *rhs* ∪ {ϵ}
end
FIRST(A) := FIRST(A) ∪ *rhs*
lastFirst := FIRST

ϵ ∈ F(β₂)

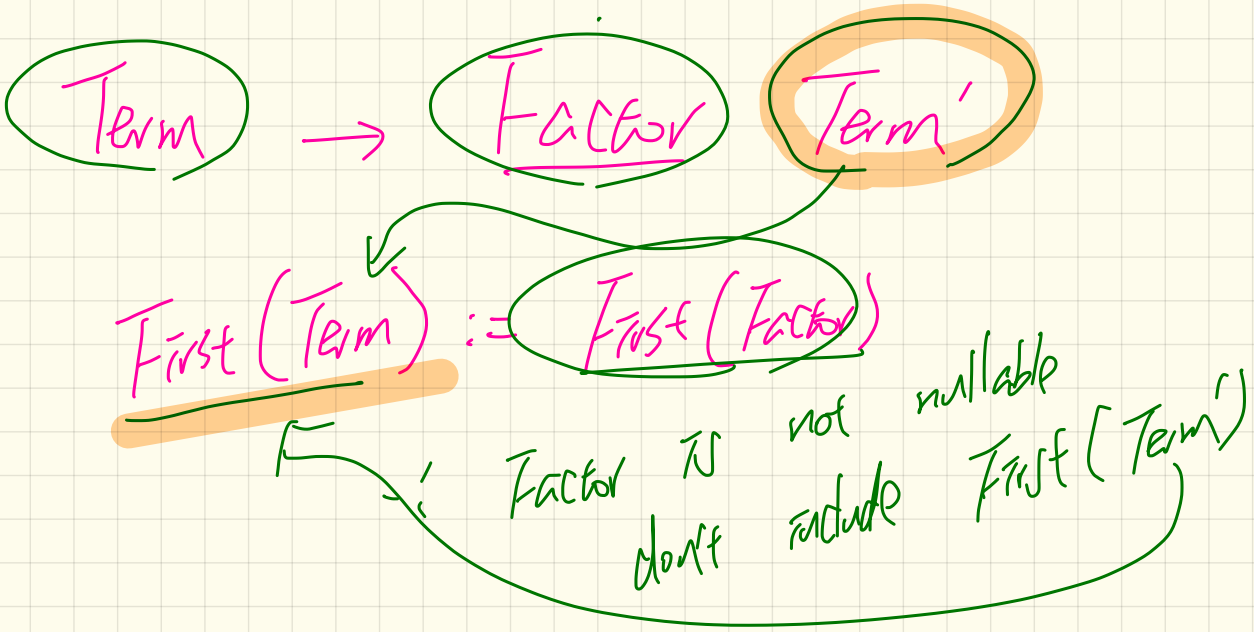
A → ... β_i β_{i+1}

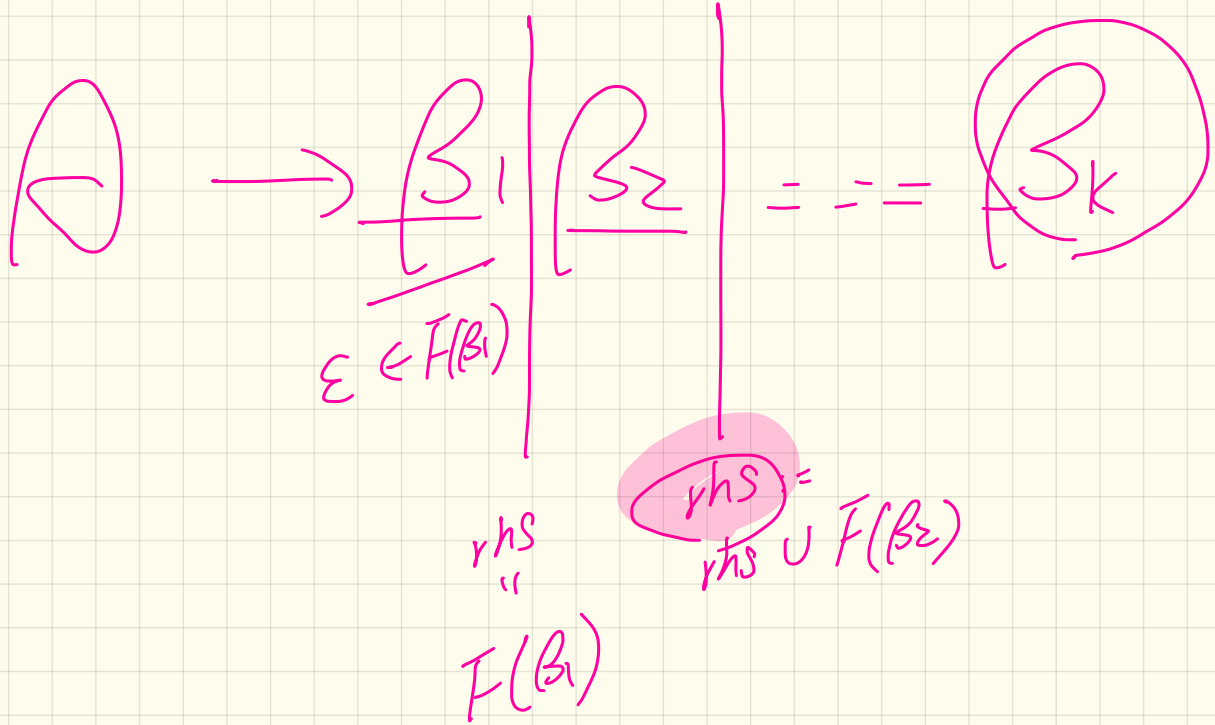
F(β_{i+1})

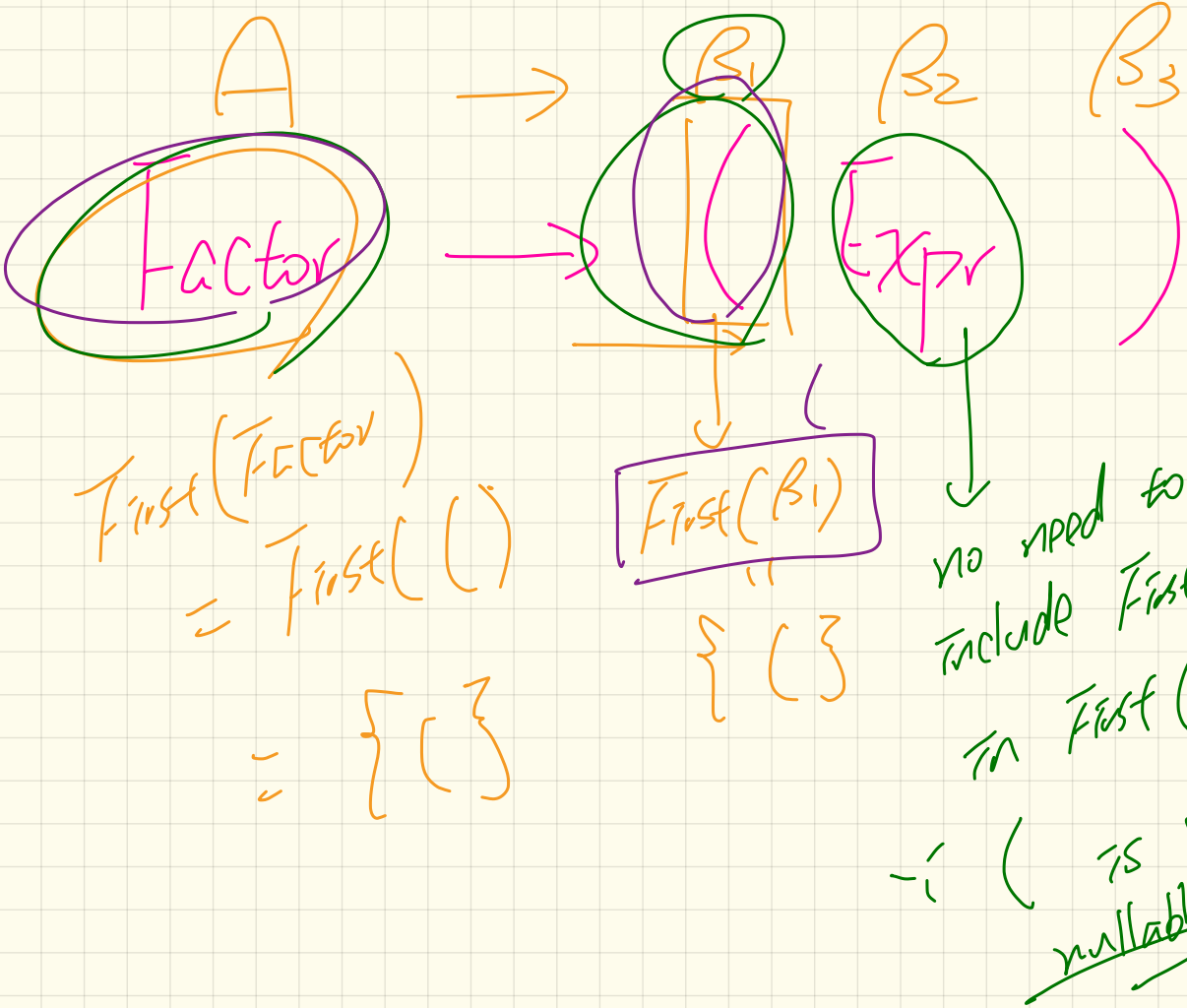
num	name	+	-	×	÷	(	)	eof	ϵ
num	name	+	−	×	÷	(	)	eof	ϵ

Expr	Expr′	Term	Term′	Factor
(	+	(	×	(
num	−	num	÷	num
name	ϵ	name	ϵ	name

0	*Goal*	→	*Expr*	6	*Term′*	→	× *Factor Term′*
1	*Expr*	→	*Term Expr′*	7		|	÷ *Factor Term′*
2	*Expr′*	→	+ *Term Expr′*	8		|	ϵ
3		|	- *Term Expr′*	9	*Factor*	→	(*Expr*)
4		|	ϵ	10		|	num
5	*Term*	→	*Factor Term′*	11		|	name




A
β_1
β_2
β_k
$\varepsilon \in F(\beta_1)$
rhs
$F(\beta_1)$
rhs
rhs $\cup$ $F(\beta_2)$

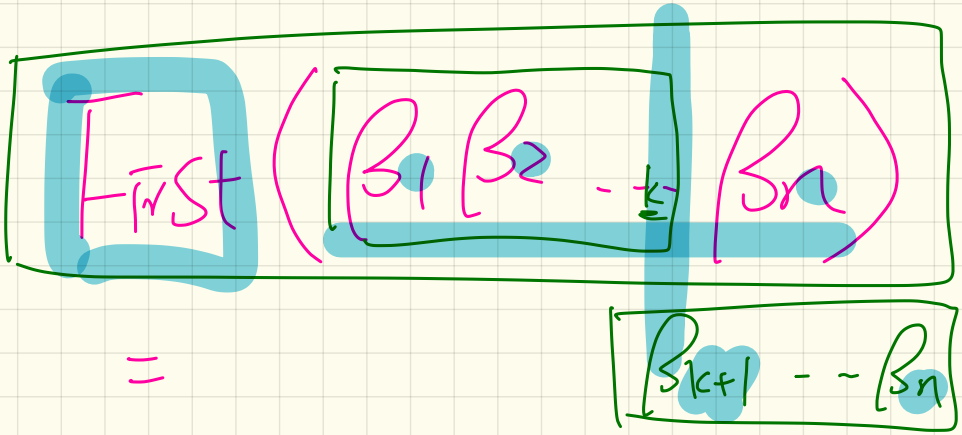

A
B1
B2
B3
Factor
Expr
First(Factor)
= First(( )
= { (}
First(B1)
{ (}
no need to include First(E) in First(F)
-(is not nullable

0	*Goal*	→	*Expr*	6	*Term′*	→	× *Factor Term′*
1	*Expr*	→	*Term Expr′*	7		|	÷ *Factor Term′*
2	*Expr′*	→	+ *Term Expr′*	8		|	ϵ
3		|	- *Term Expr′*	9	*Factor*	→	(*Expr*)
4		|	ϵ	10		|	num
5	*Term*	→	*Factor Term′*	11		|	name

V → ~~Term~~ Term' Expr' Factor

First(V) := First(Term) * ± ε

First(V) = First(Term') − {ε} ∪ First(Expr') − {ε} ∪ First(Factor)



=

$$First(\beta_1) \cup First(\beta_2) \cup \cdots \cup First(\beta_k)$$

β_1 null, β_2 null, ... β_{k-1} null, β_k not null

Extended First Set

	num	name	+	-	×	÷	(	)	eof	ϵ
FIRST	num	name	+	-	x	÷	(	)	eof	ϵ

	Expr	*Expr′*	*Term*	*Term′*	*Factor*
FIRST	(, name, num	+, -, ϵ	(, name, num	x, ÷, ϵ	(, name, num

$$\textsc{First}(\beta_1\beta_2\ldots\beta_n) = \left\{ \textsc{First}(\beta_1) \cup \textsc{First}(\beta_2) \cup \ldots \beta_k \;\middle|\; \begin{array}{l} \forall i : 1 \le i < k \bullet \epsilon \in \textsc{First}(\beta_i) \\ \wedge \\ \epsilon \notin \textsc{First}(\beta_k) \end{array} \right\}$$

Right-Recursive CFG:

0	*Goal*	→	*Expr*	6	*Term′*	→	x *Factor Term′*
1	*Expr*	→	*Term Expr′*	7		|	÷ *Factor Term′*
2	*Expr′*	→	+ *Term Expr′*	8		|	ϵ
3		|	- *Term Expr′*	9	*Factor*	→	(*Expr*)
4		|	ϵ	10		|	num
5	*Term*	→	*Factor Term′*	11		|	name

Is the FIRST Set Sufficient?

Expr′	→	+	Term Term′	(1)
	|	−	Term Term′	(2)
	|	ϵ		(3)

Expr′ → ε

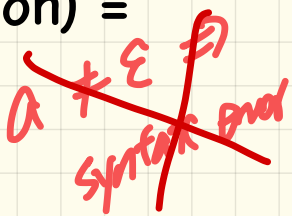
↓

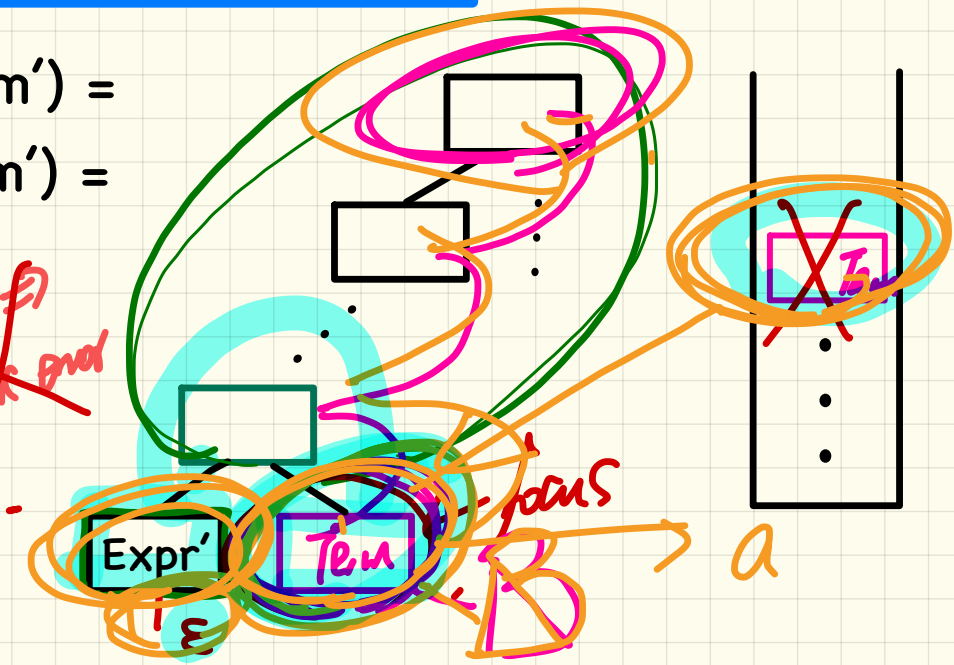
Follow(Expr′)

FIRST(+ Term Term′) =

FIRST(- Term Term′) =

FIRST(epsilon) =





FOLLOW Set

variable

$V \Rightarrow \cdots \Rightarrow x$

$$\text{FOLLOW}(V) = \{ W \mid W, X, Y \in \Sigma^* \wedge V \overset{*}{\Rightarrow} X \wedge S \overset{*}{\Rightarrow} XWY \}$$

derivable from (V)

Right-Recursive CFG:

0	*Goal*	→	*Expr*	6	*Term′*	→	x *Factor Term′*
1	*Expr*	→	*Term Expr′*	7		|	÷ *Factor Term′*
2	*Expr′*	→	+ *Term Expr′*	8		|	ϵ
3		|	- *Term Expr′*	9	*Factor*	→	(*Expr*)
4		|	ϵ	10		|	num
5	*Term*	→	*Factor Term′*	11		|	name

	Expr	Expr′	Term	Term′	Factor
FIRST	(, name, num	+, -, ϵ	(, name, num	x, ÷, ϵ	(, name, num

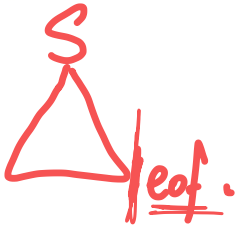
	Expr	Expr′	Term	Term′	Factor
FOLLOW	eof,)	eof,)	eof, +, -,)	eof, +, -,)	eof, +, -, x, ÷,)

Right-Recursive CFG:

0	*Goal*	→	*Expr*	6	*Term′*	→	x *Factor Term′*
1	*Expr*	→	*Term Expr′*	7		|	÷ *Factor Term′*
2	*Expr′*	→	+ *Term Expr′*	8		|	ϵ
3		|	- *Term Expr′*	9	*Factor*	→	(*Expr*)
4		|	ϵ	10		|	num
5	*Term*	→	*Factor Term′*	11		|	name

G, F, E, T, T'

```
ALGORITHM: GetFollow
  INPUT: CFG G = (V, Σ, R, S)
  OUTPUT: FOLLOW : V ⟶ ℙ(T ∪ {eof})
PROCEDURE:
  for A ∈ V: FOLLOW := ∅
  FOLLOW(S) := {eof}
  lastFollow := FOLLOW
  while (lastFirst ≠ FIRST):
    for A → β1 β2 ... βk ∈ R:
      trailer := FOLLOW(A)
      for i: k .. 1:
        if βi ∈ V then
          FOLLOW(βi) := FOLLOW(βi) ∪ trailer
          if ϵ ∈ FIRST(βi)
            then trailer := trailer ∪ (FIRST(βi) − ϵ)
            else trailer := FIRST(βi)
        else
          trailer := FIRST(βi)
    lastFollow := FOLLOW
```

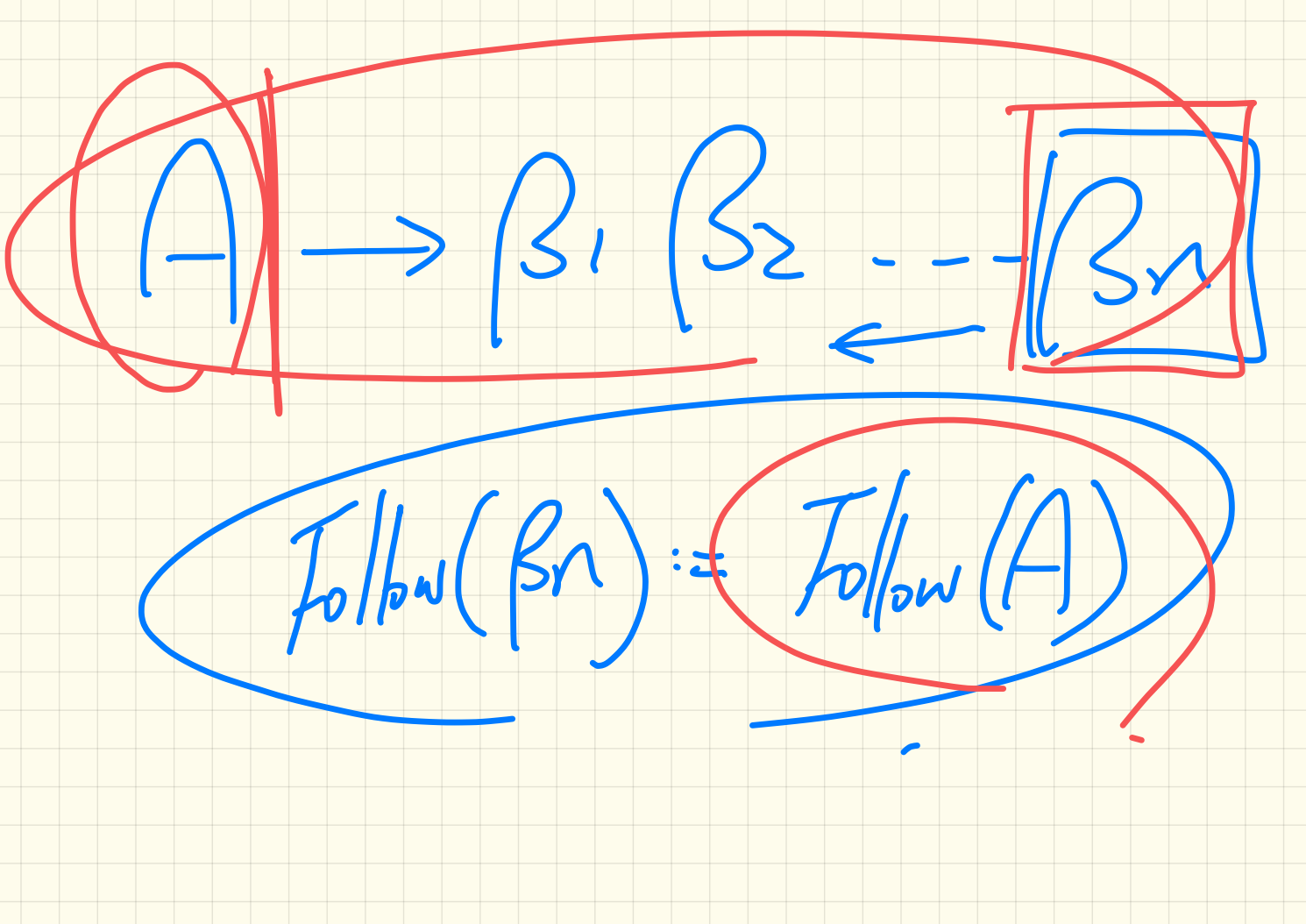


FOLLOW Set: Tracing

First choose rules whose LHS is processed.
Then rules whose RHS ends with a terminal.

	Expr	Expr′	Term	Term′	Factor
FIRST	(, name, num	+, -, ϵ	(, name, num	x, ÷, ϵ	(, name, num

Expr	Expr′	Term	Term′	Factor
eof.				


A → B1 B2 ... Bn
Follow(Bn) := Follow(A)