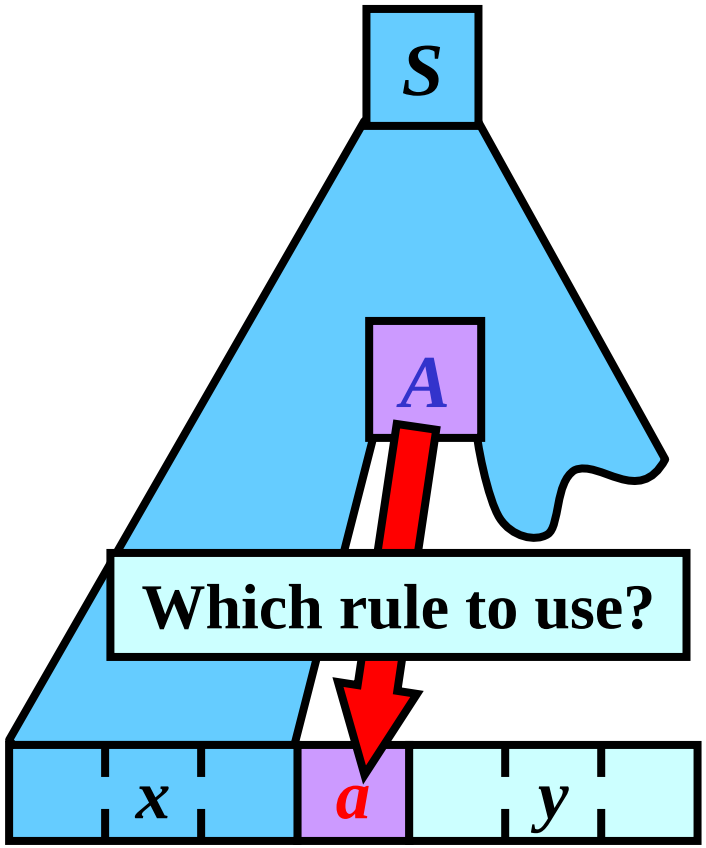
Part V.
Top-Down Parsing

Top-Down Parsing: Introduction

Problem:



Basic idea:

Table:

α	...	*a*	...
...			
A		α(*A*, *a*)	
...			

Use rule ***r***: $A \rightarrow x$

Question: Could you construct this table for **any** CFG?

Answer: **NO**

A Table-Based Selection of a Rule

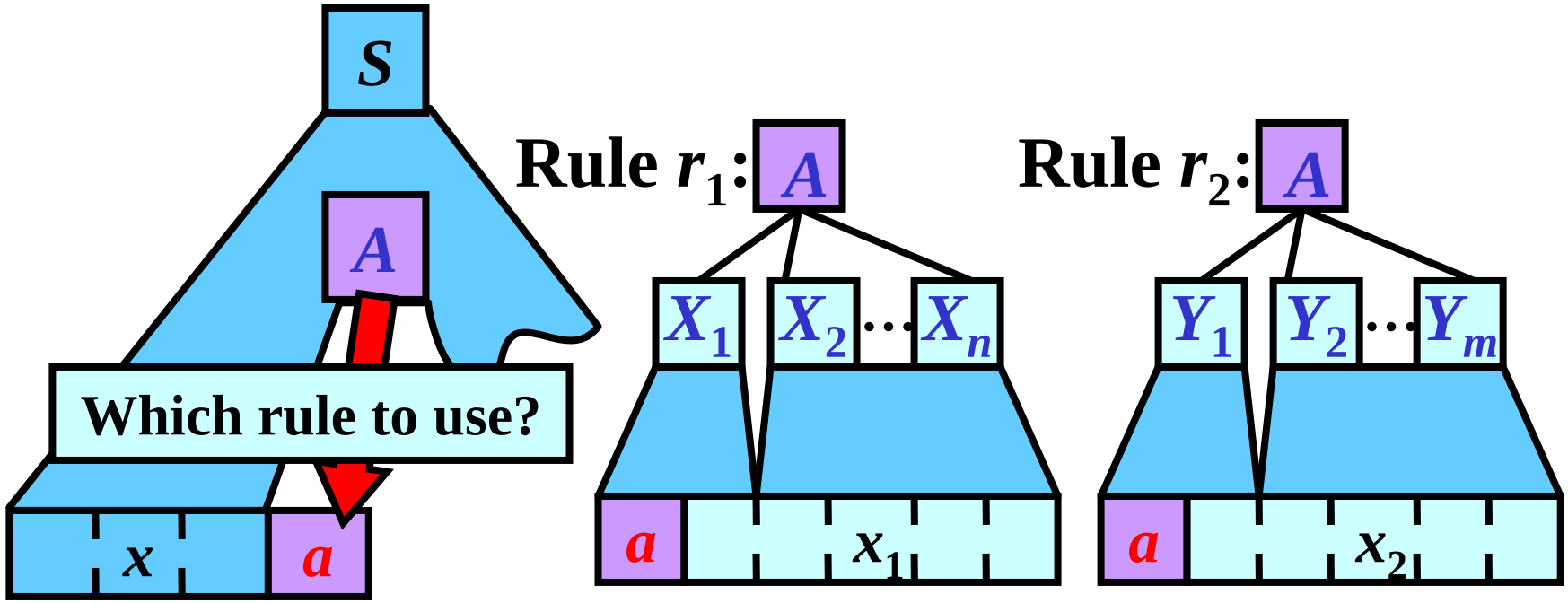


Table:

α	…	a	…
…			
A		α(A, a)	
…			

?

Use rule r_1: $A \rightarrow X_1 X_2 \ldots X_n$

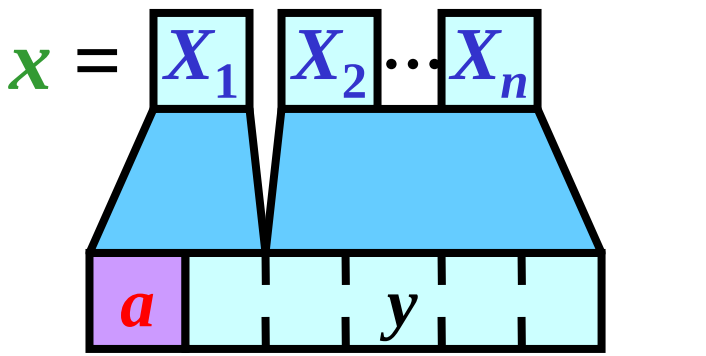
Use rule r_2: $A \rightarrow Y_1 Y_2 \ldots Y_m$

Set *First*

Gist: *First*(*x*) is the set of all terminals that can begin a sentential form derivable from *x*.

Definition: Let $G = (N, T, P, S)$ be a CFG. For every $x \in (N \cup T)^*$, we define the set *First*(x) as $First(x) = \{a: a \in T, x \Rightarrow^* ay; y \in (N \cup T)^*\}$.

Illustration:



$x = X_1X_2 \ldots X_n \Rightarrow^* ay$

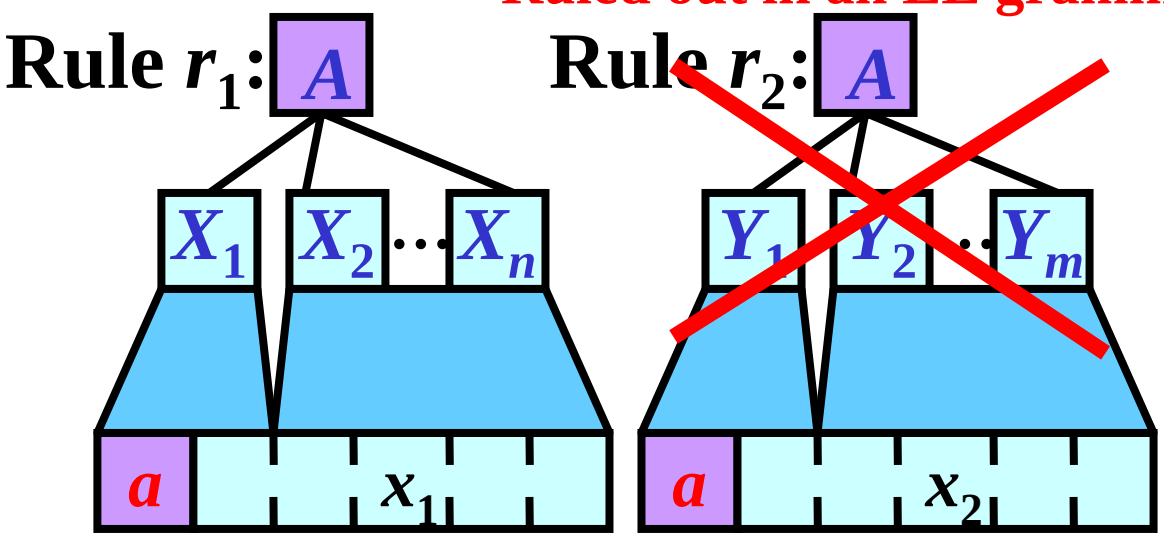
$a \in First(x)$

LL Grammars without ε-rules

Definition: Let $G = (N, T, P, S)$ be a CFG without ε-rules. G is an *LL grammar* if for every $a \in T$ and every $A \in N$ there is **no more than one** rule $A \to X_1X_2...X_n \in P$ such that $a \in First(X_1X_2...X_n)$

Illustration:

Ruled out in an LL grammar



$a \in First(X_1X_2...X_n)$ $a \in First(Y_1Y_2...Y_m)$

Table:

α	...	a	...
...			
A		α(A, a)	
...			

Only rule r_1:

$A \to X_1X_2...X_n$

*S*imple *P*rogramming *L*anguage (SPL)

1: <prog> → begin <st-list>
2: <st-list> → <stat> ; <st-list>
3: <st-list> → end
4: <stat> → read id
5: <stat> → write <item>
6: <stat> → id := add (<item> <it-list>
7: <it-list> → , <item> <it-list>
8: <it-list> →)
9: <item> → int
10: <item> → id

Note: G_{SPL} is LL grammar

Example:

```
begin
   read i;
   j := add(i, 1);
   write j;
end
```

$\in$ SPL

Algorithm: *First*(*X*)

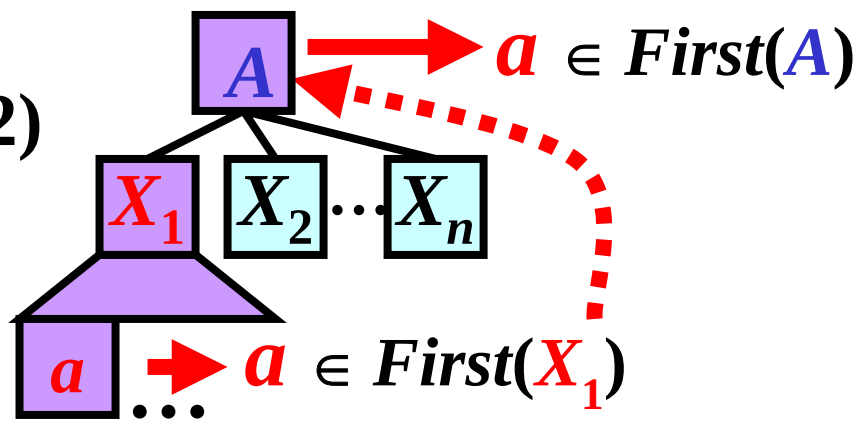
- **Input:** $G = (N, T, P, S)$ **<u>without ε-rules</u>**
- **Output:** *First*(*X*) for every $X \in N \cup T$

- **Method:**
- for each $a \in T$: *First*(a) := {a}
- **Apply the following rule until no *First* set can be changed:**
- if $A \to X_1X_2\ldots X_n \in P$, then add *First*($X_1$) to *First*($A$)

Illustration:

1) for each $a \in T$:
***First*(a) := {a}**
because $a \Rightarrow^0 a$

2)



First(*X*) for SPL: Example

First(**begin**) := {**begin**} *First*(**id**) := {**id**} *First*(**,**) := {**,**}
First(**end**) := {**end**} *First*(**int**) := {**int**} *First*(**(**) := {**(**}
First(**read**) := {**read**} *First*(**:=**) := {**:=**} *First*(**)**) := {**)**}
First(**write**) := {**write**} *First*(**add**) := {**add**} *First*(**;**) := {**;**}

<item> → **id** ∈ *P*: **add** *First*(**id**) **to** *First*(**<item>**)
<item> → **int** ∈ *P*: **add** *First*(**int**) **to** *First*(**<item>**)
Summary: *First*(**<item>**) = {**id**, **int**}

<it-list> → **)** ∈ *P*: **add** *First*(**)**) **to** *First*(**<it-list>**)
<it-list> → **,** … ∈ *P*: **add** *First*(**,**) **to** *First*(**<it-list>**)
Summary: *First*(**<it-list>**) = {**)**, **,**}

<stat> → **id** … ∈ *P*: **add** *First*(**id**) **to** *First*(**<stat>**)
<stat> → **write** … ∈ *P*: **add** *First*(**write**) **to** *First*(**<stat>**)
<stat> → **read** … ∈ *P*: **add** *First*(**read**) **to** *First*(**<stat>**)
Summary: *First*(**<stat>**) = {**id**, **write**, **read**}

<st-list> → **end** ∈ *P*: **add** *First*(**end**) **to** *First*(**<st-list>**)
<st-list> → **<stat>** … ∈ *P*: **add** *First*(**<stat>**) **to** *First*(**<st-list>**)
Summary: *First*(**<st-list>**) = {**id**, **write**, **read**, **end**}

<prog> → **begin** … ∈ *P*: **add** *First*(**begin**) **to** *First*(**<prog>**)
Summary: *First*(**<prog>**) = {**begin**}

Construction of LL Table

α	...	*a*	...
...			
A		α(*A*, *a*)	
...			

α(*A*, *a*) = $A \to X_1 X_2 \ldots X_n \in P$ if $a \in First(X_1)$; otherwise, α(*A*, *a*) is blank ⇒ **ERROR**

Task: LL table for SPL

	id	int	:=	...
<prog>				
<st-list>	2	id ∈ *First*(<stat>)		
<stat>	6	id ∈ *First*(id)		
<it-list>				
<item>	10	id ∈ *First*(id)		

Construct the rest analogically.

Rule *r*: $A \to X_1 X_2 \ldots X_n$		*First*(X_1)
1: <prog>	→ begin …	{begin}
2: <st-list>	→ <stat>…	{id, write, read}
3: <st-list>	→ end	{end}
4: <stat>	→ read …	{read}
5: <stat>	→ write …	{write}
6: <stat>	→ id …	{id}
7: <it-list>	→ , …	{,}
8: <it-list>	→)	{)}
9: <item>	→ int	{int}
10: <item>	→ id	{id}

Parsing Based on LL Table: Example

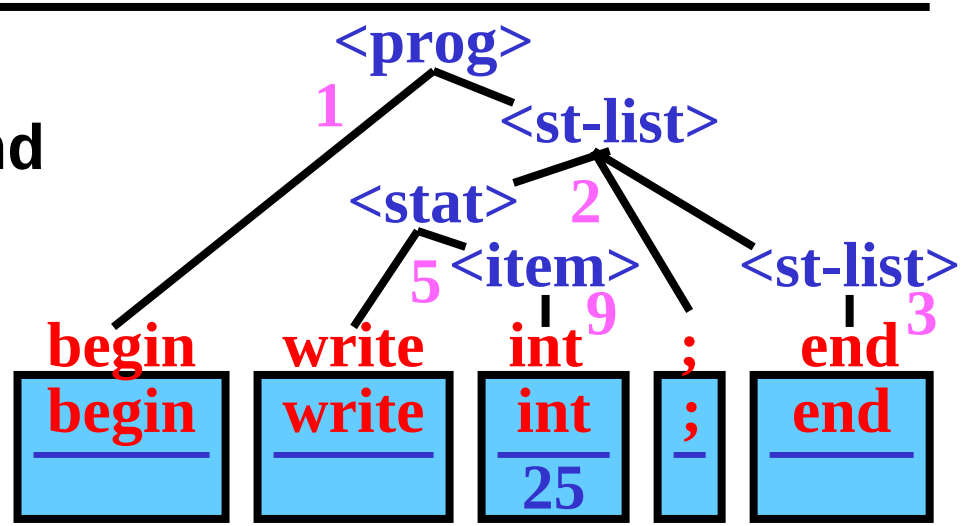
1: <prog> → begin <st-list>
2: <st-list> → <stat> ; <st-list>
3: <st-list> → end
4: <stat> → read id
5: <stat> → write <item>
6: <stat> → id := add (…
7: <it-list> → , <item> <it-list>
8: <it-list> →)
9: <item> → int
10: <item> → id

	beg	end	rd	wr	id	int	,	(	)	;	:=	add
<prog>	1											
<st-list>		3	2	2	2							
<stat>			4	5	6							
<it-list>							7		8			
<item>					10	9						

Source program:

```
begin write 25; end
```

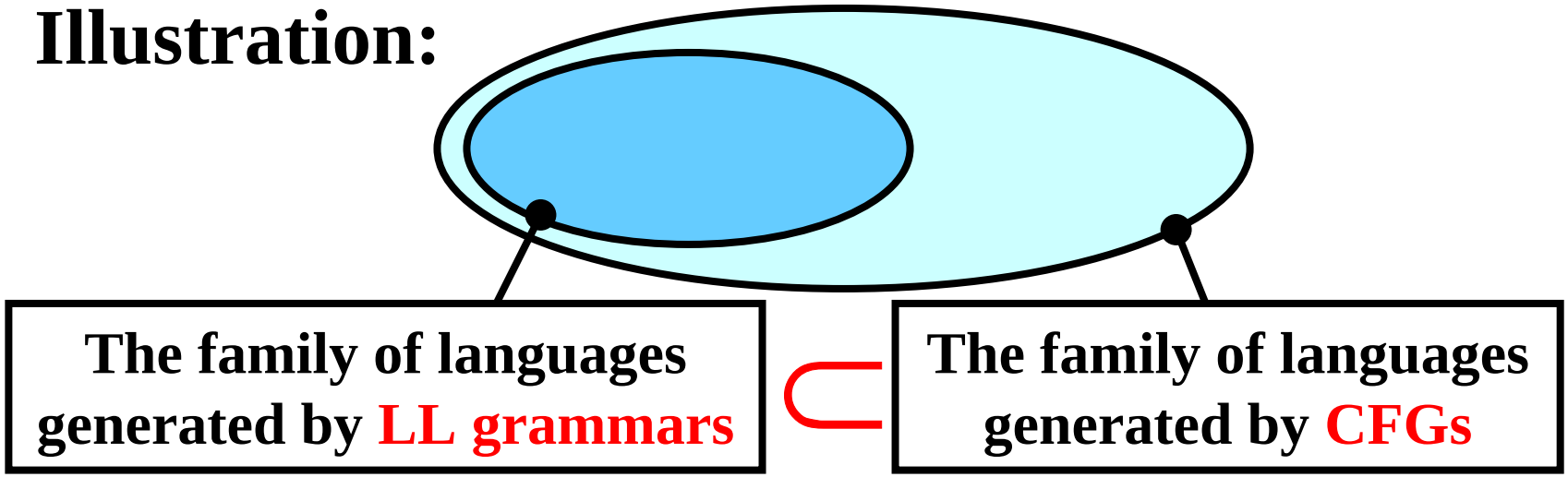
Lexical Analyzer



LL Grammars: Useful Transformations

Generaly: CFG are stronger that LL grammars

Illustration:



- **Some** CFGs can be converted to equivalent LL grammars

Basic conversions:

1) Factorization

2) Left recursion replacement

Note: A rule of the form $A \to Ax$, where $A \in N$, $x \in (N \cup T)^*$ is called a *left recursive rule*.

Factorization

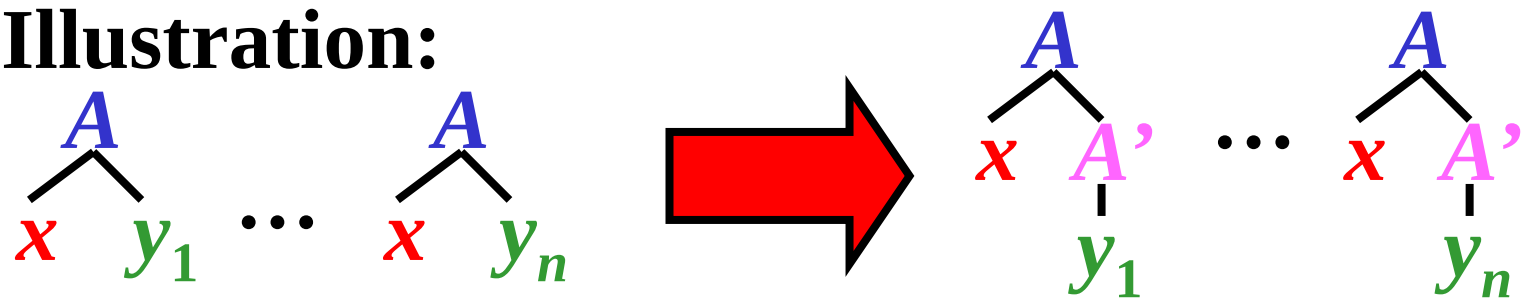
Idea: Replace rules of the form

$A \to xy_1,\ A \to xy_2,\ \dots,\ A \to xy_n$ with

$A \to xA',\ A' \to y_1,\ A' \to y_2,\ \dots,\ A' \to y_n,$

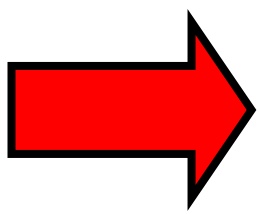
where A' is a new nonterminal

Illustration:



Example:

<stat> → write id
<stat> → write int

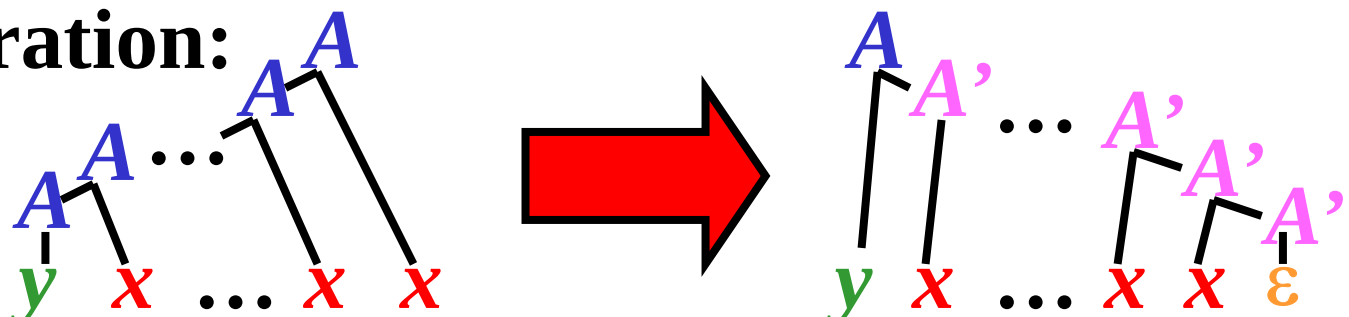


<stat> → write <item>
<item> → id
<item> → int

Left Recursion Replacement

Idea: Replace rules of the form $A \to Ax$, $A \to y$ with $A \to yA'$, $A' \to xA'$, $A' \to \varepsilon$, where A' is a new nonterminal.

Illustration:



Example:

$E \to E+T$
$E \to T$

$\Rightarrow$ $E \to TE'$, $E' \to +TE'$, $E' \to \varepsilon$

$T \to T*F$
$T \to F$

$\Rightarrow$ $T \to FT'$, $T' \to *FT'$, $T' \to \varepsilon$

$F \to (E)$
$F \to i$

$F \to (E)$
$F \to i$

LL Grammars with ε-rules: Introduction

Why ε-rules?

- elimination of the left recursion introduces ε-rule
- ε-rules often make the language specification clearer

Simplification of this part:

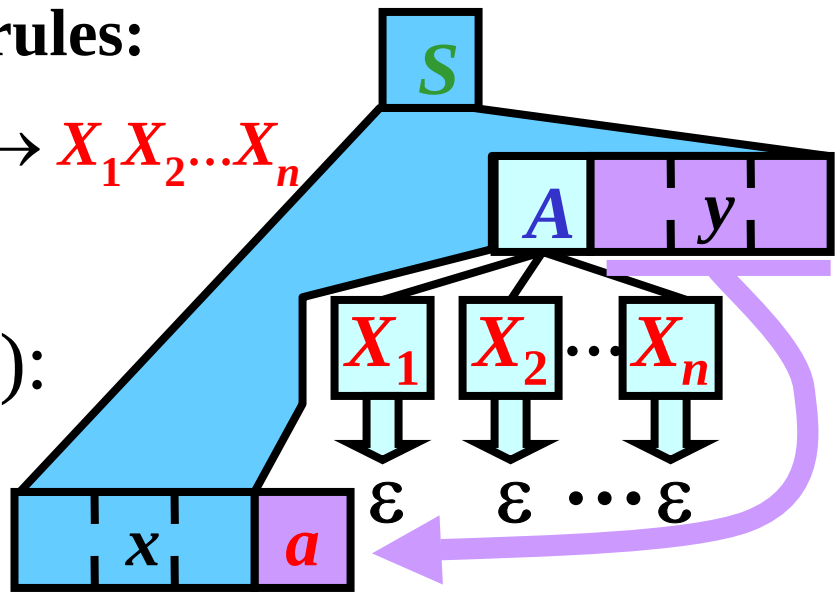
Assume that every input string of tokens ends with $.

Note: $ acts as an *end marker*.

Main problem with ε-rules:

Rule *r*: $A \rightarrow X_1 X_2 \ldots X_n$

Maybe: $a \notin First(A)$:



Note: We must define other sets: *Empty*, *Follow* and *Predict*.

Grammar for Arithmetical Expressions

- $G_{expr3} = (N, T, P, E)$, where
- $N = \{E, E', F, F', T\}$,
- $T = \{i, +, *, (,)\}$,
- $P = \{$ 1: $E \rightarrow TE'$, 2: $E' \rightarrow +TE'$,
  3: $E' \rightarrow \varepsilon$, 4: $T \rightarrow FT'$,
  5: $T' \rightarrow *FT'$, 6: $T' \rightarrow \varepsilon$,
  7: $F \rightarrow (E)$, 8: $F \rightarrow i$ $\}$

Example:

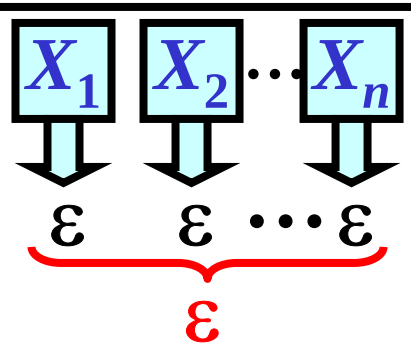
$$(i + i)*(i + i) \in L(G_{expr3})$$

Set *Empty*

Gist: ***Empty*(*x*) is the set that include ε if *x* derives the empty string; otherwise, *Empty*(*x*) is empty**

Definition: Let $G = (N, T, P, S)$ be a CFG.
$Empty(x) = \{\varepsilon\}$ if $x \Rightarrow^* \varepsilon$; otherwise,
$Empty(x) = \varnothing$, where $x \in (N \cup T)^*$.

Illustration:



$x = X_1 X_2 \ldots X_n \Rightarrow^* \varepsilon$

$Empty(x) = \{\varepsilon\}$

Algorithm: *Empty*(*X*)

- **Input:** $G = (N, T, P, S)$
- **Output:** *Empty*(*X*) for every $X \in N \cup T$

- **Method:**
- **for each** $a \in T$: *Empty*(a) := $\varnothing$
- **for each** $A \in N$:

  if $A \to \varepsilon \in P$ **then** *Empty*(A) := $\{\varepsilon\}$

  else *Empty*(A) := $\varnothing$
- **Apply the following rule until no *Empty* set can be changed:**
  - **if** $A \to X_1X_2\ldots X_n \in P$ **and** *Empty*(X_i) = $\{\varepsilon\}$ for all $i = 1,\ldots, n$ **then** *Empty*(A) = $\{\varepsilon\}$

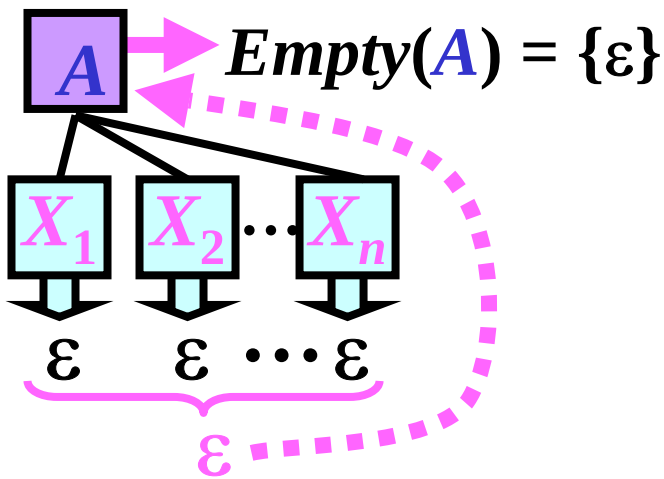
Previous Algorithm: Illustration

1) for each $a \in T$: $Empty(a) := \emptyset$ because $a \not\Rightarrow^* \varepsilon$

2) for each r**:** $A \to \varepsilon \in P$: $Empty(A) := \{\varepsilon\}$ because $A \Rightarrow^1 \varepsilon$ **[**r**]**

3) Apply the following rules until no *Empty* set can be changed:

- **if** $A \to X_1 X_2 \ldots X_n \in P$ **and** $Empty(X_i) = \{\varepsilon\}$ for all $i = 1,\ldots,n$ **then** $Empty(A) = \{\varepsilon\}$



Empty(*X*) for G_{expr3}: Example

G_{expr3} = (*N*, *T*, *P*, ***E***), where: *N* = {***E***, ***F***, ***T***}, *T* = {***i***, **+**, *****, **(**, **)**},

P = { 1: ***E*** → ***TE'***, 2: ***E'*** → **+*TE'***, 3: ***E'*** → ε, 4: ***T*** → ***FT'***

5: ***T'*** → ***** ***FT'***, 6: ***T'*** → ε, 7: ***F*** → **(*E*)**, 8: ***F*** → ***i*** }

Initialization:

Empty(***i***) := ∅		*Empty*(***E***)	:= ∅
Empty(**+**) := ∅		*Empty*(***E'***)	:= {ε}
Empty(*****) := ∅		*Empty*(***T***)	:= ∅
Empty(**(**) := ∅		*Empty*(***T'***)	:= {ε}
Empty(**)**) := ∅		*Empty*(***F***)	:= ∅

- **No *Empty* set can be changed.**

Algorithm: *First*(*X*)

- **Input:** $G = (N, T, P, S)$
- **Output:** *First*(*X*) for every $X \in N \cup T$

- **Method:**
- **for each** $a \in T$: *First*(a) := {a}
- **for each** $A \in N$: *First*(A) := $\varnothing$
- **Apply the following rule until no *First* set can be changed:**
- **if** $A \to X_1X_2\ldots X_{k-1}X_k\ldots X_n \in P$ **then**
  - add all symbols from *First*(X_1) to *First*(A)
  - **if** *Empty*(X_i) = {ε} for all $i = 1,\ldots, k\text{-}1$, where $k \le n$ **then** add all symbols from *First*(X_k) to *First*(A)

Previous Algorithm: Illustration

1) for each $a \in T$: $First(a) := \{a\}$ because $a \Rightarrow^0 a$

2) for each $A \in N$: $First(A) := \emptyset$ (inicialization)

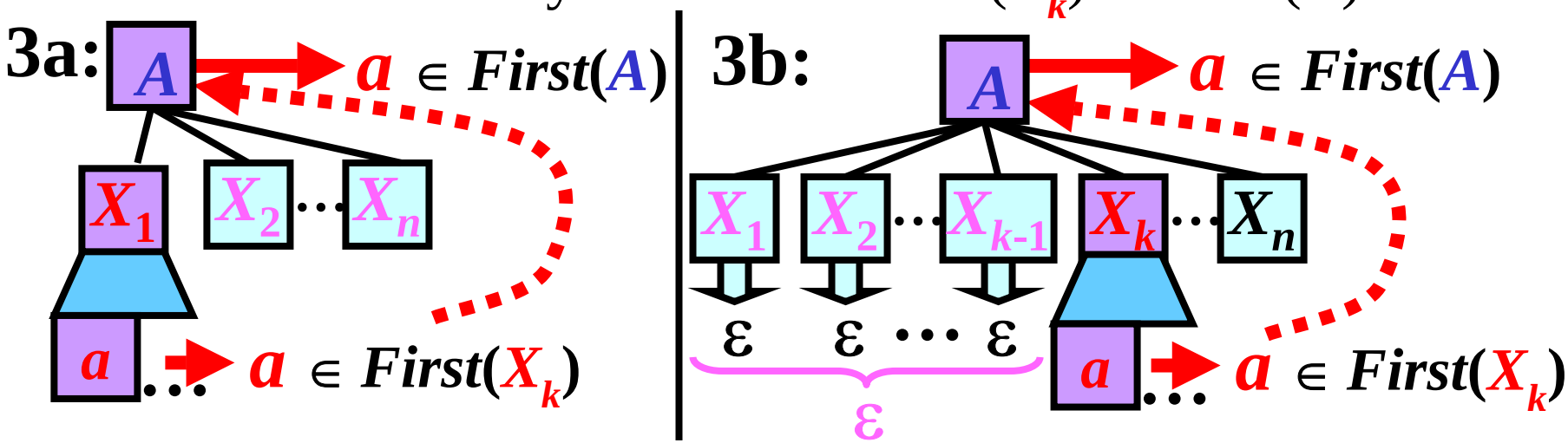
---

3) Apply the following rules until no *First* set or *Empty* set can be changed:

- **if** $A \rightarrow X_1 X_2 \ldots X_{k-1} X_k \ldots X_n \in P$ **then**

  3a) add all symbols from $First(X_1)$ to $First(A)$

  3b) if $Empty(X_i) = \{\varepsilon\}$ for all $i = 1, \ldots, k\text{-}1$, where $k < n$ **then** add all symbols from $First(X_k)$ to $First(A)$:



First(*X*) for G_{expr3}: Example

Initialization:

First(***i***)	:= {***i***}	*First*(***E***)	:= ∅
First(**+**)	:= {**+**}	*First*(***E'***)	:= ∅
First(*****)	:= {*****}	*First*(***T***)	:= ∅
First(**(**)	:= { **(** }	*First*(***T'***)	:= ∅
First(**)**)	:= { **)** }	*First*(***F***)	:= ∅

F → ***i*** ∈ *P*: **add** *First*(***i***) = {***i***} **to** *First*(***F***)
F → **(** ***E*** **)** ∈ *P*: **add** *First*(**(**) = {**(**} **to** *First*(***F***)
Summary: *First*(*F*) = {*i*, (}

T' → ***** ***F T'*** ∈ *P*: **add** *First* (*****) = {*****} **to** *First*(***T'***)
Summary: *First*(*T'*) = {*}

T → ***F T'*** ∈ *P*: **add** *First* (***F***) = {***i***, **(**} **to** *First*(***T***)
Summary: *First*(*T*) = {*i*, (}

E' → **+** ***T E'*** ∈ *P*: **add** *First* (**+**) = {**+**} **to** *First*(***E'***)
Summary: *First*(*E'*) = {+}

E → ***T E'*** ∈ *P*: **add** *First* (***T***) = {***i***, **(**} **to** *First*(***E***)
Summary: *First*(*E*) = {*i*, (}

- **No *First* set can be changed.**

First(*X*) & *Empty*(*X*) for G_{expr3}: Summary

G_{expr3} = (*N*, *T*, *P*, ***E***), where: *N* = {***E***, ***F***, ***T***}, *T* = {***i***, **+**, *****, **(**, **)**},
P = { **1**: ***E*** → ***TE'***, **2**: ***E'*** → **+*TE'***, **3**: ***E'*** → ε, **4**: ***T*** → ***FT'***
5: ***T'*** → ***** ***FT'***, **6**: ***T'*** → ε, **7**: ***F*** → **(** ***E*** **)**, **8**: ***F*** → ***i*** }

Set *Empty* for all $X \in N \cup T$:

Empty(***i***)	:= ∅	*Empty*(***E***)	:= ∅
Empty(**+**)	:= ∅	*Empty*(***E'***)	:= {ε}
Empty(*****)	:= ∅	*Empty*(***T***)	:= ∅
Empty(**(**)	:= ∅	*Empty*(***T'***)	:= {ε}
Empty(**)**)	:= ∅	*Empty*(***F***)	:= ∅

Set *First* for all $X \in N \cup T$:

First(***i***)	:= {***i***}	*First*(***E***)	:= {***i***, **(**}
First(**+**)	:= {**+**}	*First*(***E'***)	:= {**+**}
First(*****)	:= {*****}	*First*(***T***)	:= {***i***, **(**}
First(**(**)	:= { **(** }	*First*(***T'***)	:= {*****}
First(**)**)	:= { **)** }	*First*(***F***)	:= {***i***, **(**}

Note: for each ***a*** ∈ *T*: *Empty*(***a***) = ∅, *First*(***a***) = {***a***}

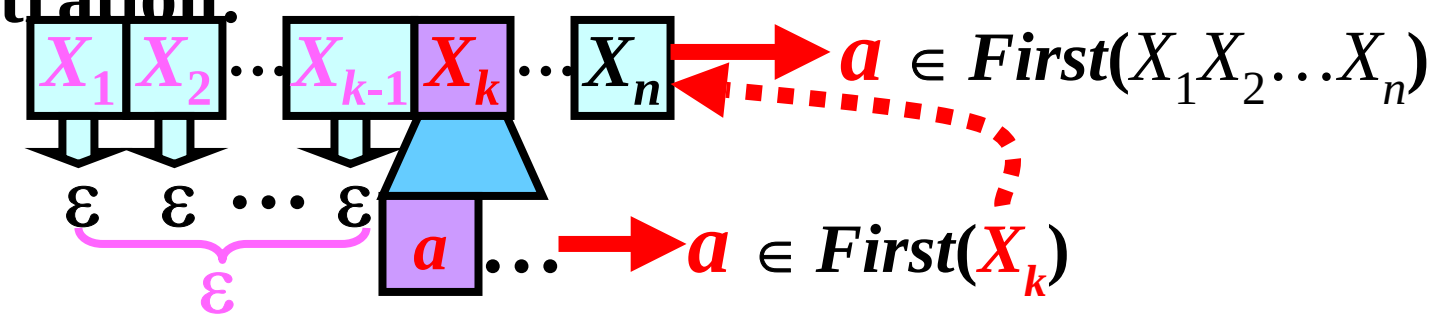
Algorithm: *First*($X_1X_2\ldots X_n$)

- **Input:** $G = (N, T, P, S)$; *First*(X) & *Empty*(X) for every $X \in N \cup T$; $x = X_1X_2\ldots X_n$, where $x \in (N \cup T)^+$
- **Output:** *First*($X_1X_2\ldots X_n$)

- **Method:**
- *First*($X_1X_2\ldots X_n$) := *First*(X_1)
- **Apply the following rule until nothing can be added to *First*($X_1X_2\ldots X_{k-1}X_k\ldots X_n$):**
  - **if** *Empty*(X_i) = {ε} for all $i = 1,\ldots,k\text{-}1$, where $k \le n$ **then** add all symbols from *First*(X_k) to *First*($X_1X_2\ldots X_n$)

! Note: *First*(ε) = ∅

Illustration:



$First(X_1X_2\ldots X_n)$: Example

$G_{expr3} = (N, T, P, \mathbf{E})$, where: $N = \{\mathbf{E}, \mathbf{F}, \mathbf{T}\}$, $T = \{\mathbf{i}, +, *, (,)\}$,

$P = \{$ 1: $\mathbf{E} \rightarrow \mathbf{TE'}$, 2: $\mathbf{E'} \rightarrow +\mathbf{TE'}$, 3: $\mathbf{E'} \rightarrow \varepsilon$, 4: $\mathbf{T} \rightarrow \mathbf{FT'}$

5: $\mathbf{T'} \rightarrow *\mathbf{FT'}$, 6: $\mathbf{T'} \rightarrow \varepsilon$, 7: $\mathbf{F} \rightarrow (\mathbf{E})$, 8: $\mathbf{F} \rightarrow \mathbf{i}$ $\}$

Set *Empty* & *First* for all $X \in N$:

$Empty(\mathbf{E})$	$:= \varnothing$	$First(\mathbf{E})$	$:= \{\mathbf{i}, (\}$
$Empty(\mathbf{E'})$	$:= \{\varepsilon\}$	$First(\mathbf{E'})$	$:= \{+\}$
$Empty(\mathbf{T})$	$:= \varnothing$	$First(\mathbf{T})$	$:= \{\mathbf{i}, (\}$
$Empty(\mathbf{T'})$	$:= \{\varepsilon\}$	$First(\mathbf{T'})$	$:= \{*\}$
$Empty(\mathbf{F})$	$:= \varnothing$	$First(\mathbf{F})$	$:= \{\mathbf{i}, (\}$

Task: $First(\mathbf{E'T'FET})$

1) $First(\mathbf{E'T'FET}) := First(\mathbf{E'}) = \{+\}$

2) $First(\mathbf{E'T'FET})$: add $First(\mathbf{T'}) = \{*\}$ to $First(\mathbf{E'T'FET})$

$Empty(\mathbf{E'}) = \{\varepsilon\}$

3) $First(\mathbf{E'T'FET})$: add $First(\mathbf{F}) = \{\mathbf{i}, (\}$ to $First(\mathbf{E'T'FET})$

$Empty(\mathbf{E'}) = Empty(\mathbf{T'}) = \{\varepsilon\}$

Summary: $First(\mathbf{E'T'FET}) = \{+, *, \mathbf{i}, (\}$

Algorithm: *Empty*($X_1X_2\ldots X_n$)

- **Input:** $G = (N, T, P, S)$; *Empty*(X) for every $X \in N \cup T$;
  $x = X_1X_2\ldots X_n$, where $x \in (N \cup T)^+$
- **Output:** *Empty*($X_1X_2\ldots X_n$)

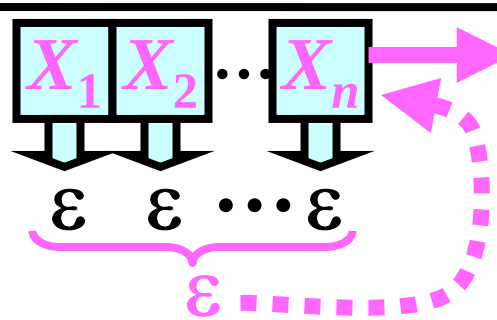
- **Method:**
- **if** *Empty*(X_i) = {ε} for all $i = 1,\ldots,n$ **then**
  Empty($X_1X_2\ldots X_n$) := {ε}

  else

  Empty($X_1X_2\ldots X_n$) := ∅

! Note: *Empty*(ε) = {ε}

Illustration:



Empty($X_1X_2\ldots X_n$) **= {ε}**

Empty($X_1X_2\ldots X_n$): Example

G_{expr3} = (N, T, P, ***E***), where: N = {***E***, ***F***, ***T***}, T = {***i***, **+**, *****, **(**, **)**},

P = { **1**: ***E*** → ***TE'***, **2**: ***E'*** → **+*TE'***, **3**: ***E'*** → ε, **4**: ***T*** → ***FT'***

5: ***T'*** → ***** ***FT'***, **6**: ***T'*** → ε, **7**: ***F*** → **(** ***E*** **)**, **8**: ***F*** → ***i*** }

Set *Empty* for all $X \in N$**:**

Empty(***E***)	:= ∅
Empty(***E'***)	:= {ε}
Empty(***T***)	:= ∅
Empty(***T'***)	:= {ε}
Empty(***F***)	:= ∅

Task: *Empty*(***E'T'***)

Empty(***E'***) = *Empty*(***T'***) = **{ε}, so** *Empty*(***E'T'***) = **{ε}**

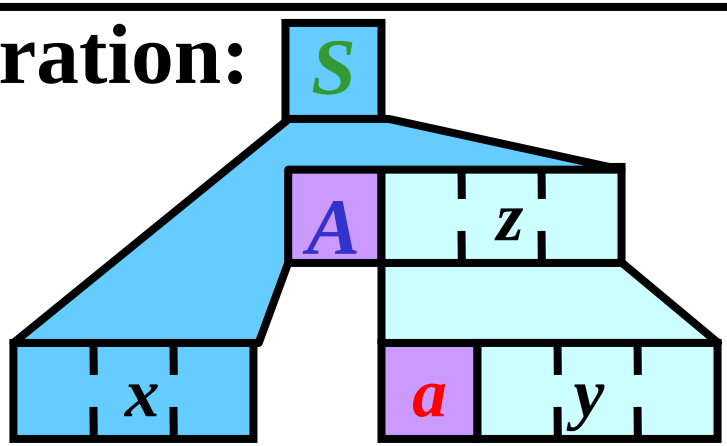
Set *Follow*

Gist: ***Follow*(*A*) is the set of all terminals that can come right after *A* in a sentential form of *G***

Definition: Let $G = (N, T, P, S)$ be a CFG. For every $A \in N$, we define the set *Follow*(A) as

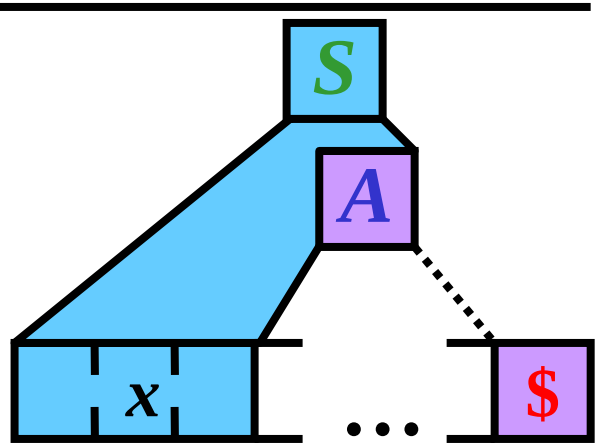
$$Follow(A) = \{a: a \in T, S \Rightarrow^* xAay, x, y \in (N \cup T)^*\} \cup \{\$: S \Rightarrow^* xA, x \in (N \cup T)^*\}$$

Illustration:



$S \Rightarrow^* xAz \Rightarrow^* xAay$

$a \in Follow(A)$



$S \Rightarrow^* xA$

$\$ \in Follow(A)$

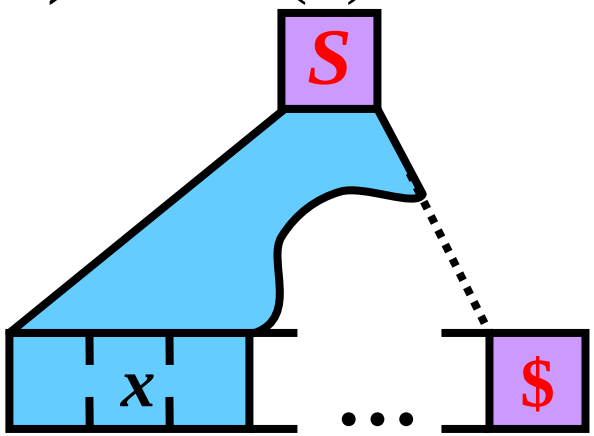
Algorithm: *Follow*(*A*)

- **Input:** $G = (N, T, P, S)$;
- **Output:** *Follow*(*A*) for every $A \in N$

- **Method:**
- *Follow*(***S***) := {**$**};
- **Apply the following rules until no *Follow* set can be changed:**
- **if** $A \rightarrow xBy \in P$ **then**
  - **if** $y \neq \varepsilon$ **then**
    add all symbols from *First*(***y***) to *Follow*(***B***);
  - **if** *Empty*(***y***) = {ε} **then**
    add all symbols from *Follow*(***A***) to *Follow*(***B***);

Previous Algorithm: Illustration

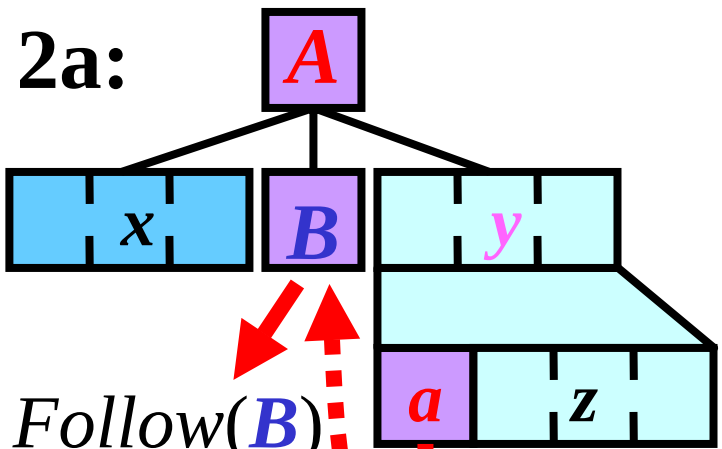
1) *Follow*(***S***) := {**$**}



2) Apply the following rules until no *Follow* set can be changed:

- **if** $A \rightarrow xBy \in P$ **then**
  - **2a) if** $y \neq \varepsilon$ **then** add all symbols from *First*(***y***) to *Follow*(***B***)
  - **2b) if** *Empty*(***y***) = {ε} **then** add all symbols from *Follow*(***A***) to *Follow*(***B***)

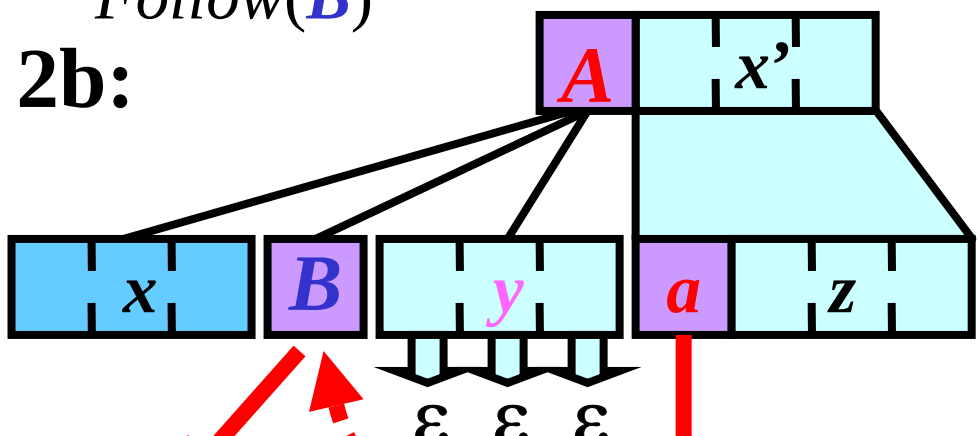
2a:



a ∈ *Follow*(***B***)

a ∈ *First*(***y***)

2b:



a ∈ *Follow*(***B***)

a ∈ *Follow*(***A***)

Follow(*X*) for G_{expr3}: Example 1/3

First(**E**)	:= {**i**, **(**}	*Empty*(**E**)	:= ∅	*Follow*(**E**)	:= ∅
First(**E'**)	:= {**+**}	*Empty*(**E'**)	:= {ε}	*Follow*(**E'**)	:= ∅
First(**T**)	:= {**i**, **(**}	*Empty*(**T**)	:= ∅	*Follow*(**T**)	:= ∅
First(**T'**)	:= {*****}	*Empty*(**T'**)	:= {ε}	*Follow*(**T'**)	:= ∅
First(**F**)	:= {**i**, **(**}	*Empty*(**F**)	:= ∅	*Follow*(**F**)	:= ∅

0) *Follow*(**E**) := {**$**}

1) **F** → **(** **E** **)** ∈ *P*: **add** *First*(**)**) = {**)**} **to** *Follow*(**E**)

≠ ε

Summary: *Follow*(**E**) = {**$**, **)**}

2) **E** → **T** **E'** ∈ *P*: **add** *Follow*(**E**) = {**$**, **)**} **to** *Follow*(**E'**)

ε: *Empty*(ε) = {ε}

E → **T** **E'** ∈ *P*: **add** *First*(**E'**) = {**+**} **to** *Follow*(**T**)

≠ ε

E → **T** **E'** ∈ *P*: **add** *Follow*(**E**) = {**$**, **)**} **to** *Follow*(**T**)

Empty(**E'**) = {ε}

Summary: *Follow*(**E'**) = {**$**, **)**}, *Follow*(**T**) = {**+**, **$**, **)**}

Follow(*X*) for G_{expr3}: Example 2/3

First(**E**)	:= {**i**, **(**}	*Empty*(**E**)	:= ∅	*Follow*(**E**)	:= {**$**, **)**}
First(**E'**)	:= {**+**}	*Empty*(**E'**)	:= {ε}	*Follow*(**E'**)	:= {**$**, **)**}
First(**T**)	:= {**i**, **(**}	*Empty*(**T**)	:= ∅	*Follow*(**T**)	:= {**+**, **$**, **)**}
First(**T'**)	:= {*****}	*Empty*(**T'**)	:= {ε}	*Follow*(**T'**)	:= ∅
First(**F**)	:= {**i**, **(**}	*Empty*(**F**)	:= ∅	*Follow*(**F**)	:= ∅

3) **E'** → +**T**<u>**E'**</u> ∈ *P*: **add** *Follow*(**E'**) = {**$**, **)**} **to** *Follow*(**E'**)

ε: *Empty*(ε) = {ε}

E' → +**T**<u>**E'**</u> ∈ *P*: **add** *First*(**E'**) = {**+**} **to** *Follow*(**T**)

≠ ε

E' → +**T**<u>**E'**</u> ∈ *P*: **add** *Follow*(**E'**) = {**$**, **)**} **to** *Follow*(**T**)

Empty(**E'**) = {ε}

Summary: Nothing is changed

4) **T** → **F**<u>**T'**</u> ∈ *P*: **add** *Follow*(**T**) = {**+**, **$**, **)**} **to** *Follow*(**T'**)

ε: *Empty*(ε) = {ε}

T → **F**<u>**T'**</u> ∈ *P*: **add** *First*(**T'**) = {*****} **to** *Follow*(**F**)

≠ ε

T → **F**<u>**T'**</u> ∈ *P*: **add** *Follow*(**T**) = {**+**, **$**, **)**} **to** *Follow*(**F**)

Empty(**T'**) = {ε}

Summary: *Follow*(**T'**) = {**+**, **$**, **)**}, *Follow*(**F**) = {*****, **+**, **$**, **)**}

Follow(*X*) for G_{expr3}: Example 3/3

First(**E**)	:= {**i**, **(**}	*Empty*(**E**)	:= ∅	*Follow*(**E**)	:= {**$**, **)**}
First(**E'**)	:= {**+**}	*Empty*(**E'**)	:= {ε}	*Follow*(**E'**)	:= {**$**, **)**}
First(**T**)	:= {**i**, **(**}	*Empty*(**T**)	:= ∅	*Follow*(**T**)	:= {**+**, **$**, **)**}
First(**T'**)	:= {*****}	*Empty*(**T'**)	:= {ε}	*Follow*(**T'**)	:= {**+**, **$**, **)**}
First(**F**)	:= {**i**, **(**}	*Empty*(**F**)	:= ∅	*Follow*(**F**)	:= {*****, **+**, **$**, **)**}

5) **T'** → ***F**$\underbrace{\mathbf{T'}}_{\varepsilon}$ ∈ *P*: **add** *Follow*(**T'**) = {**+**, **$**, **)**} **to** *Follow*(**T'**)

ε: *Empty*(ε) = {ε}

T' → ***F**$\underbrace{\mathbf{T'}}_{\neq \varepsilon}$ ∈ *P*: **add** *First*(**T'**) = {*****} **to** *Follow*(**F**)

T' → ***F**$\underbrace{\mathbf{T'}}_{\neq \varepsilon}$ ∈ *P*: **add** *Follow*(**T'**) = {**+**, **$**, **)**} **to** *Follow*(**F**)

Empty(**T'**) = {ε}

End: Nothing is changed.

Summary:

Follow(**E**) := {**$**, **)**}
Follow(**E'**) := {**$**, **)**}
Follow(**T**) := {**+**, **$**, **)**}
Follow(**T'**) := {**+**, **$**, **)**}
Follow(**F**) := {*****, **+**, **$**, **)**}

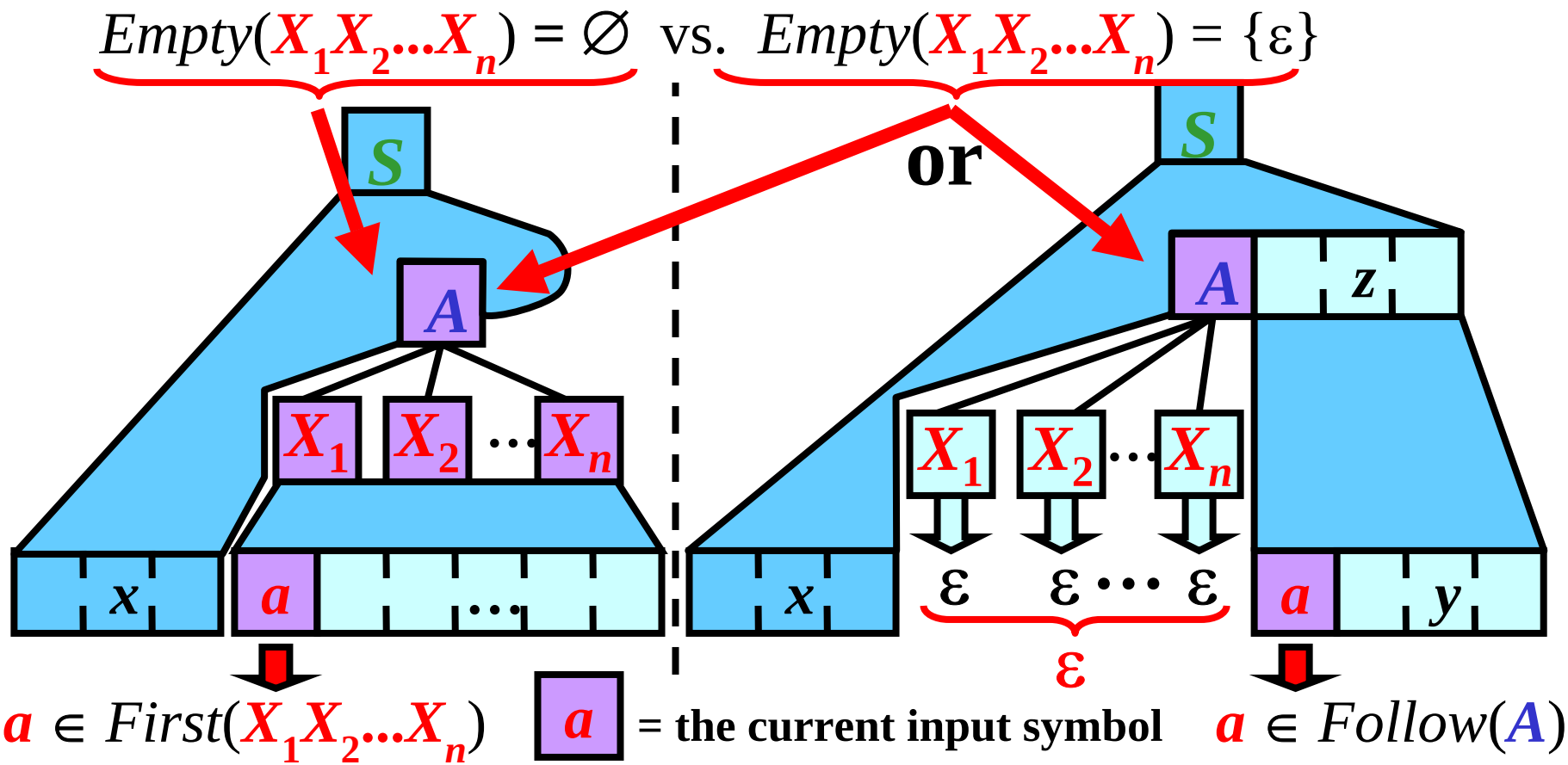
Set *Predict*

Gist: ***Predict*(*A* → *x*) is the set of all terminals that can begin a string obtained by a derivation started by using *A* → *x*.**

Definition: Let $G = (N, T, P, S)$ be a CFG. For every $A \to x \in P$, we define *Predict*$(A \to x)$ so that

- if *Empty*$(x) = \{\varepsilon\}$ then
  Predict$(A \to x)$ = *First*$(x) \cup$ *Follow*(A)
- if *Empty*$(x) = \varnothing$ then
  Predict$(A \to x)$ = *First*(x)

Set *Predict*($A \to X_1X_2...X_n$): Illustration



Summary: if *Empty*($X_1X_2...X_n$) = {ε} then *Predict*($A \to X_1X_2...X_n$) = *First*($X_1X_2...X_n$) ∪ *Follow*(A); otherwise, *Predict*($A \to X_1X_2...X_n$) = *First*($X_1X_2...X_n$)

Predict($A \rightarrow x$) for G_{expr3}: Example 1/2

First(***E***)	:= {***i***, **(**}	*Empty*(***E***)	:= ∅	*Follow*(***E***)	:= {**$**, **)**}
First(***E'***)	:= {**+**}	*Empty*(***E'***)	:= {ε}	*Follow*(***E'***)	:= {**$**, **)**}
First(***T***)	:= {***i***, **(**}	*Empty*(***T***)	:= ∅	*Follow*(***T***)	:= {**+**, **$**, **)**}
First(***T'***)	:= {***** }	*Empty*(***T'***)	:= {ε}	*Follow*(***T'***)	:= {**+**, **$**, **)**}
First(***F***)	:= {***i***, **(**}	*Empty*(***F***)	:= ∅	*Follow*(***F***)	:= {*****, **+**, **$**, **)**}

1: ***E*** → ***TE'***

Empty(***TE'***) = ∅ because *Empty*(***T***) = ∅

Predict(**1**) := *First*(***TE'***) = *First*(***T***) = {***i***, **(**}

2: ***E'*** → **+*TE'***

Empty(**+*TE'***) = ∅ because *Empty*(***T***) = ∅

Predict(**2**) := *First*(**+*TE'***) = *First*(**+**) = {**+**}

3: ***E'*** → ε

Empty(ε) = {ε}

Predict(**3**) := *First*(ε) ∪ *Follow*(***E'***) = ∅ ∪ {**$**, **)**} = {**$**, **)**}

4: ***T*** → ***FT'***

Empty(***FT'***) = ∅ because *Empty*(***F***) = ∅

Predict(**4**) := *First*(***FT'***) = *First*(***F***) = {***i***, **(**}

Predict($A \rightarrow x$) for G_{expr3}: Example 2/2

First(**E**)	:= {**i**, **(**}	*Empty*(**E**)	:= ∅	*Follow*(**E**)	:= {**$**, **)**}
First(**E'**)	:= {**+**}	*Empty*(**E'**)	:= {ε}	*Follow*(**E'**)	:= {**$**, **)**}
First(**T**)	:= {**i**, **(**}	*Empty*(**T**)	:= ∅	*Follow*(**T**)	:= {**+**, **$**, **)**}
First(**T'**)	:= {*****}	*Empty*(**T'**)	:= {ε}	*Follow*(**T'**)	:= {**+**, **$**, **)**}
First(**F**)	:= {**i**, **(**}	*Empty*(**F**)	:= ∅	*Follow*(**F**)	:= {*****, **+**, **$**, **)**}

5: **T'** → ***FT'**

Empty(***FT'**) = ∅ because *Empty*(**F**) = ∅

Predict(**5**) := *First*(***FT'**) = *First*(*****) = {*****}

6: **T'** → ε

Empty(ε) = {ε}

Predict(**6**) := *First*(ε) ∪ *Follow*(**T'**) = ∅ ∪ {**+**, **$**, **)**} = {**+**, **$**, **)**}

7: **F** → **(E)**

Empty(**(E)**) = ∅ because *Empty*(**E**) = ∅

Predict(**7**) := *First*(**(E)**) = *First*(**(**) = {**(**}

8: **F** → **i**

Empty(**i**) = ∅

Predict(**8**) := *First*(**i**) = {**i**}

Construction of LL Table

α	...	*a*	...
...			
A		α(*A*, *a*)	
...			

α(*A*, *a*) = $A \to X_1 X_2 \ldots X_n \in P$ if $a \in Predict(A \to X_1 X_2 \ldots X_n)$; otherwise, α(*A*, *a*) is blank.

Task: LL table for G_{expr1}

	i	+	*	(	)	$
E	1 ← *i* ∈ *Predict*(1)					
E'						
T	4 ← *i* ∈ *Predict*(4)					
T'						
F	8 ← *i* ∈ *Predict*(8)					

Construct the rest analogically.

Rule *r*	***Predict*(*r*)**
1: $E \to TE'$	{*i*, (}
2: $E' \to +TE'$	{+}
3: $E' \to \varepsilon$	{$,)}
4: $T \to FT'$	{*i*, (}
5: $T' \to *FT'$	{*}
6: $T' \to \varepsilon$	{+, $,)}
7: $F \to (E)$	{(}
8: $F \to i$	{*i*}

Parsing Based on LL Table: Example

	i	+	*	(	)	$
E	1			1		
E'		2			3	3
T	4			4		
T'		6	5		6	6
F	8			7		

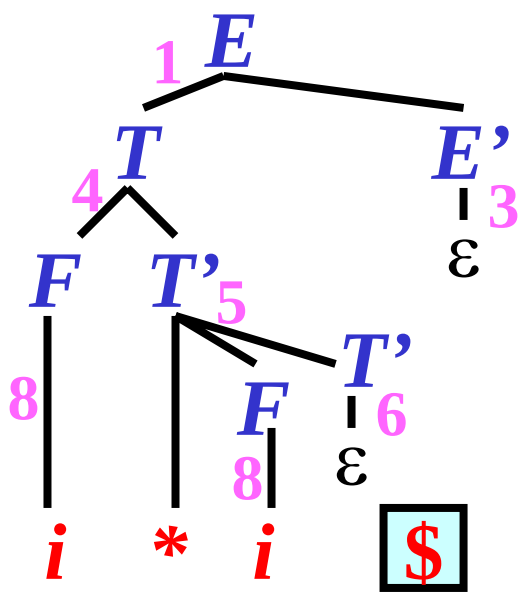
1: $E \rightarrow TE'$ 5: $T' \rightarrow *FT'$

2: $E' \rightarrow +TE'$ 6: $T' \rightarrow \varepsilon$

3: $E' \rightarrow \varepsilon$ 7: $F \rightarrow (E)$

4: $T \rightarrow FT'$ 8: $F \rightarrow i$

Question: $i * i \in \mathrm{L}(G_{expr3})$**?**



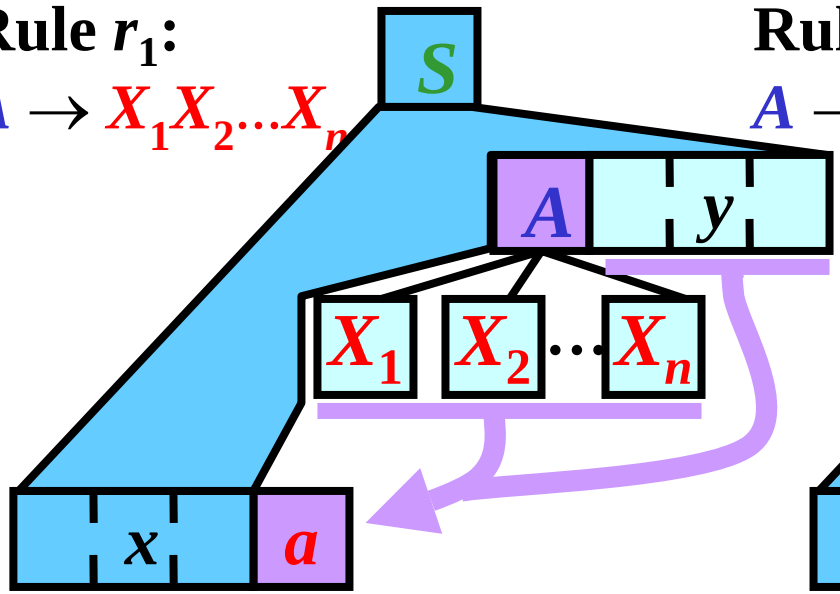
LL Grammars with ε-rules: Definition

Definition: Let $G = (N, T, P, S)$ be a CFG. G is an *LL grammar* if for every $a \in T$ and every $A \in N$ there is **no more than one** A-rule $A \to X_1 X_2 \ldots X_n \in P$ such that $a \in Predict(A \to X_1 X_2 \ldots X_n)$

Illustration:

Rule r_1:

$A \to X_1 X_2 \ldots X_n$

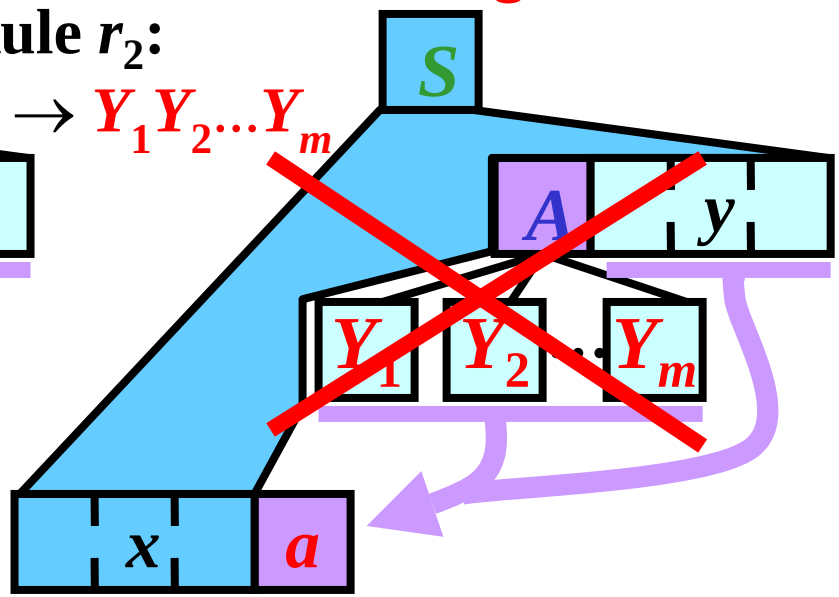


$a \in Predict(A \to X_1 X_2 \ldots X_n)$

Ruled out in an LL grammar

Rule r_2:

$A \to Y_1 Y_2 \ldots Y_m$



$a \in Predict(A \to Y_1 Y_2 \ldots Y_m)$

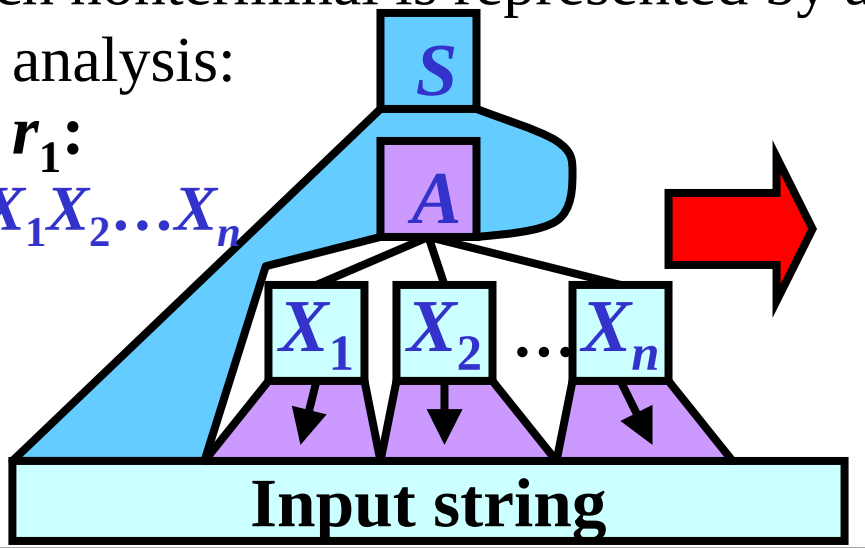
LL Analyzer Implementation

1) Recursive-Descent Parsing

• Each nonterminal is represented by a procedure, which perform its analysis:

Rule r_1:

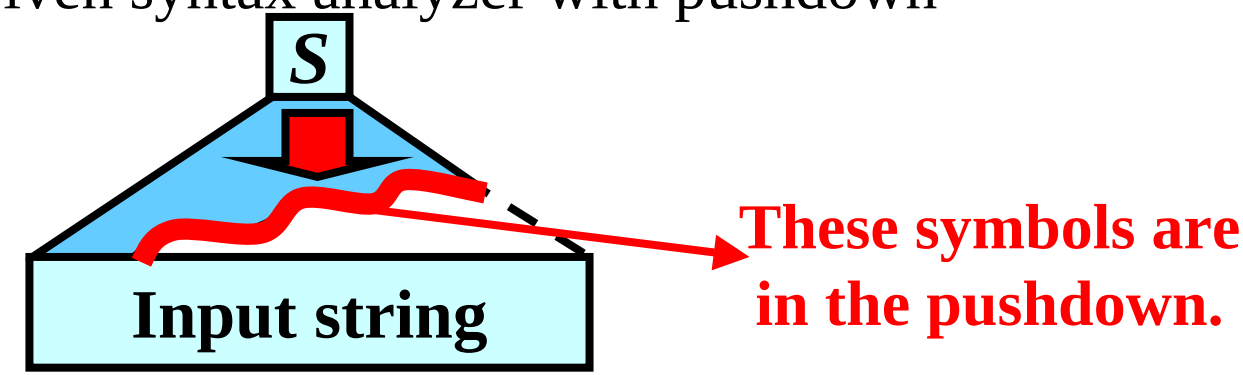
$A \rightarrow X_1 X_2 \ldots X_n$




```
function A: boolean;
begin
  {X1 analysis}
  {X2 analysis}
      …
  {Xn analysis}
end
```

2) Predictive Parsing

• Table-driven syntax analyzer with pushdown

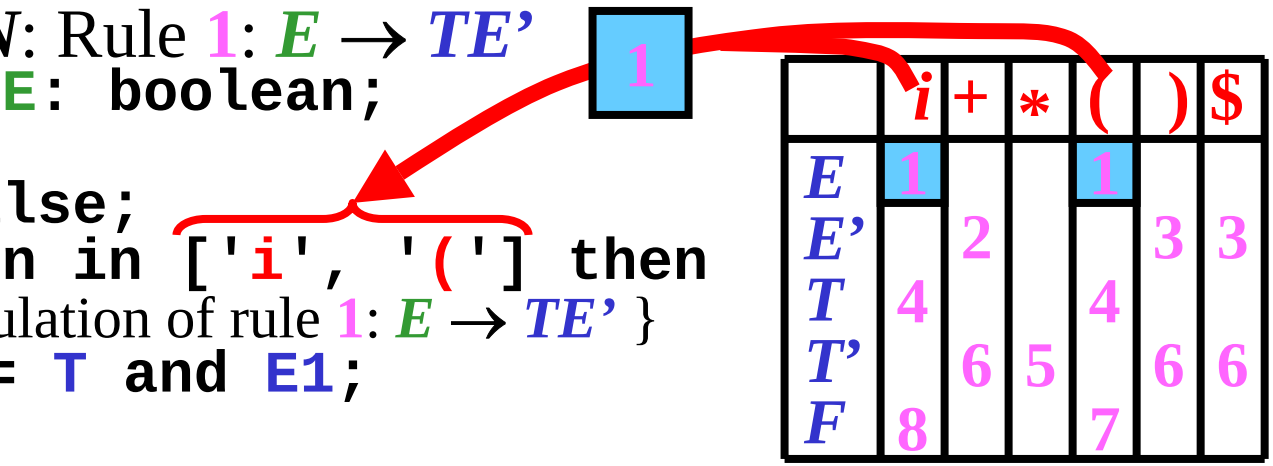


Recursive Descent: Example 1/4

```
Procedure GetNextToken;
begin
{ this procedure get the next token to global variable "token"}
end
```

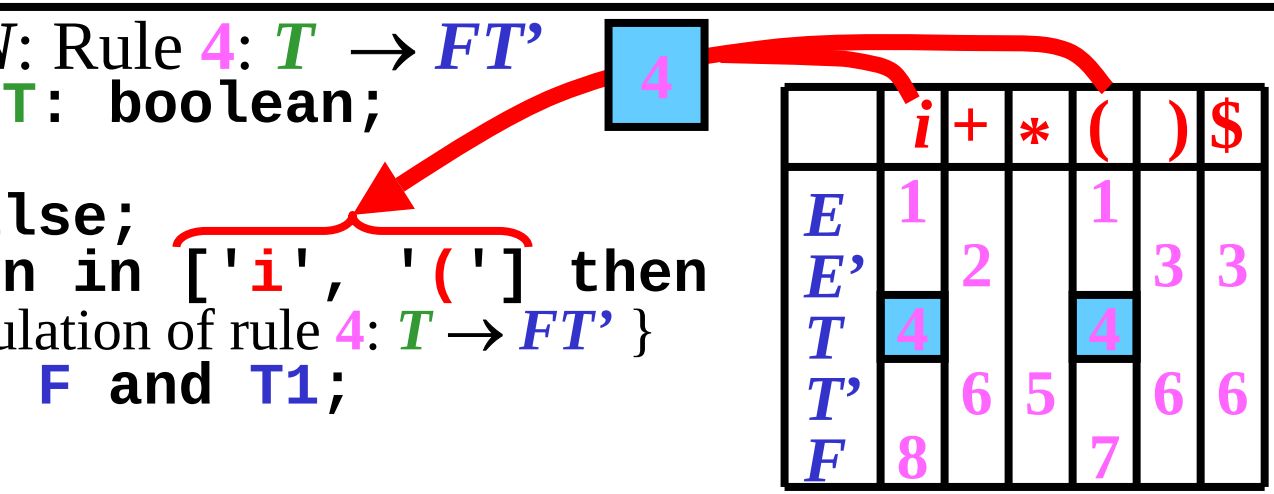
• For $E \in N$: Rule 1: $E \rightarrow TE'$

```
function E: boolean;
begin
   E := false;
   if token in ['i', '('] then
        { simulation of rule 1: E → TE' }
        E := T and E1;
end;
```



• For $T \in N$: Rule 4: $T \rightarrow FT'$

```
function T: boolean;
begin
   T := false;
   if token in ['i', '('] then
        { simulation of rule 4: T → FT' }
        T := F and T1;
end;
```

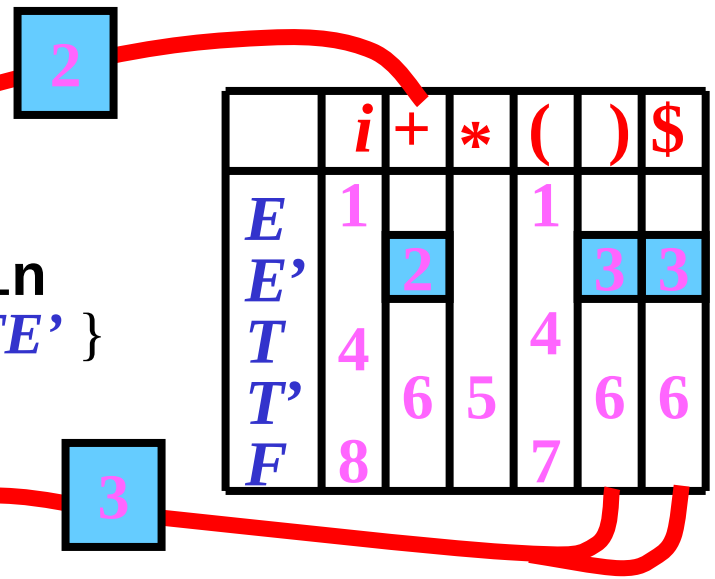


Recursive Descent: Example 2/4

- For *E'* ∈ *N*: Rules 2: *E'* → +*TE'*, 3: *E'* → ε

```
function E1: boolean;
begin
   E1 := false;
   if token = '+' then begin
      { simulation of rule 2: E' → +TE' }
      GetNextToken;
      E1 := T and E1;
   end
   else
   if token in [')', '$'] then
      { simulation of rule 3: E' → ε}
      E1 := true;
end;
```

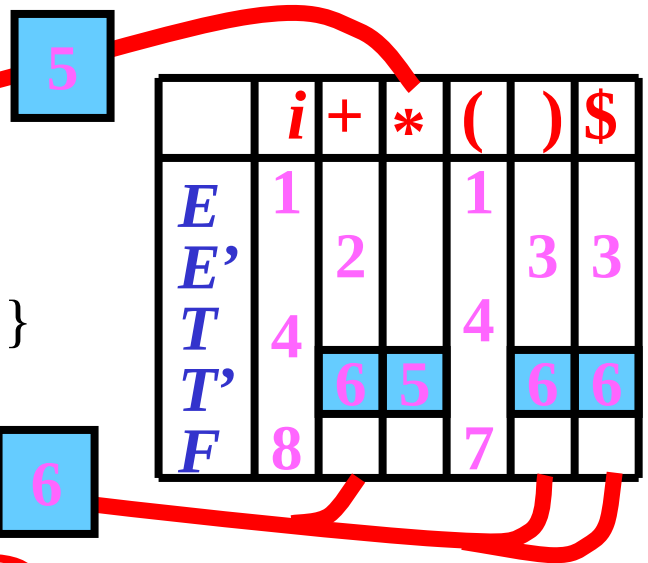
	i	+	*	(	)	$
E	1			1		
E'		2			3	3
T	4			4		
T'		6	5		6	6
F	8			7		



Recursive Descent: Example 3/4

- For $T' \in N$: Rules 5: $T' \rightarrow *FT'$, 6: $T' \rightarrow \varepsilon$

```
function T1: boolean;
begin
   T1 := false;
   if token = '*' then begin
      { simulation of rule 5: T' → *FT' }
      GetNextToken;
      T1 := F and T1;
   end
   else
   if token in ['+', ')', '$'] then
      { simulation of rule 6: T' → ε}
      T1 := true;
end;
```



Recursive Descent: Example 4/4

- For $F \in N$: Rules 7: $F \rightarrow (E)$, 8: $F \rightarrow i$

```
function F: boolean;
begin
   F := false;
   if token = '(' then begin
       { simulation of rule 7: F → (E) }
       GetNextToken;
       if E then begin
          F := (token = ')');
          GetNextToken;
       end;
   end
   else
   if token = 'i' then begin
       { simulation of rule 8: F → i }
       F := true;
       GetNextToken;
   end;
end;
```

	i	+	*	(	)	$
E	1			1		
E'		2			3	3
T	4			4		
T'		6	5		6	6
F	8			7		

Main body:

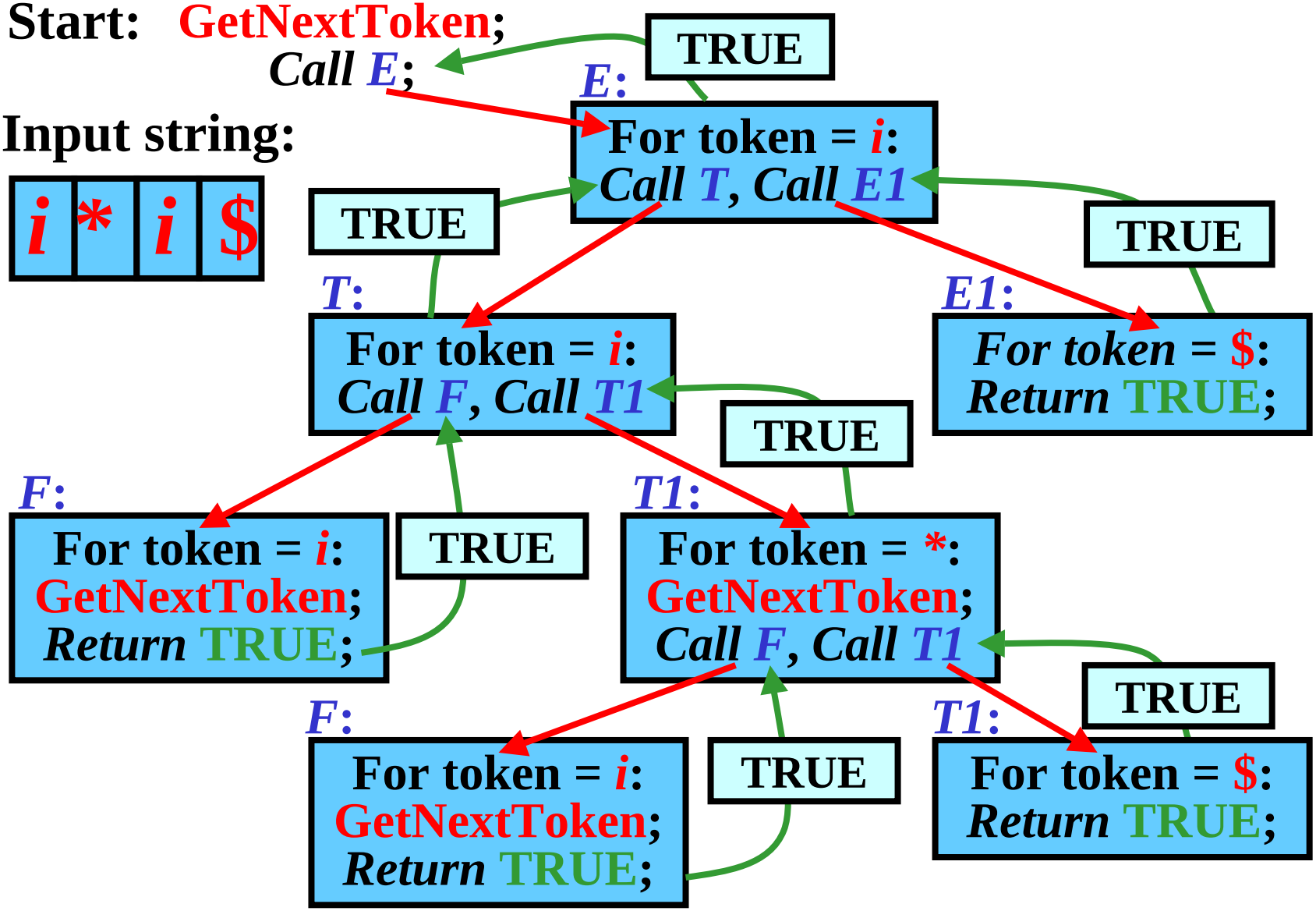
```
begin
   GetNextToken;
   if E then
      write('OK')
   else
      write('ERROR')
end.
```

Recursive Descent: Illustration for *i***i*$

Start: **GetNextToken;**
Call E;

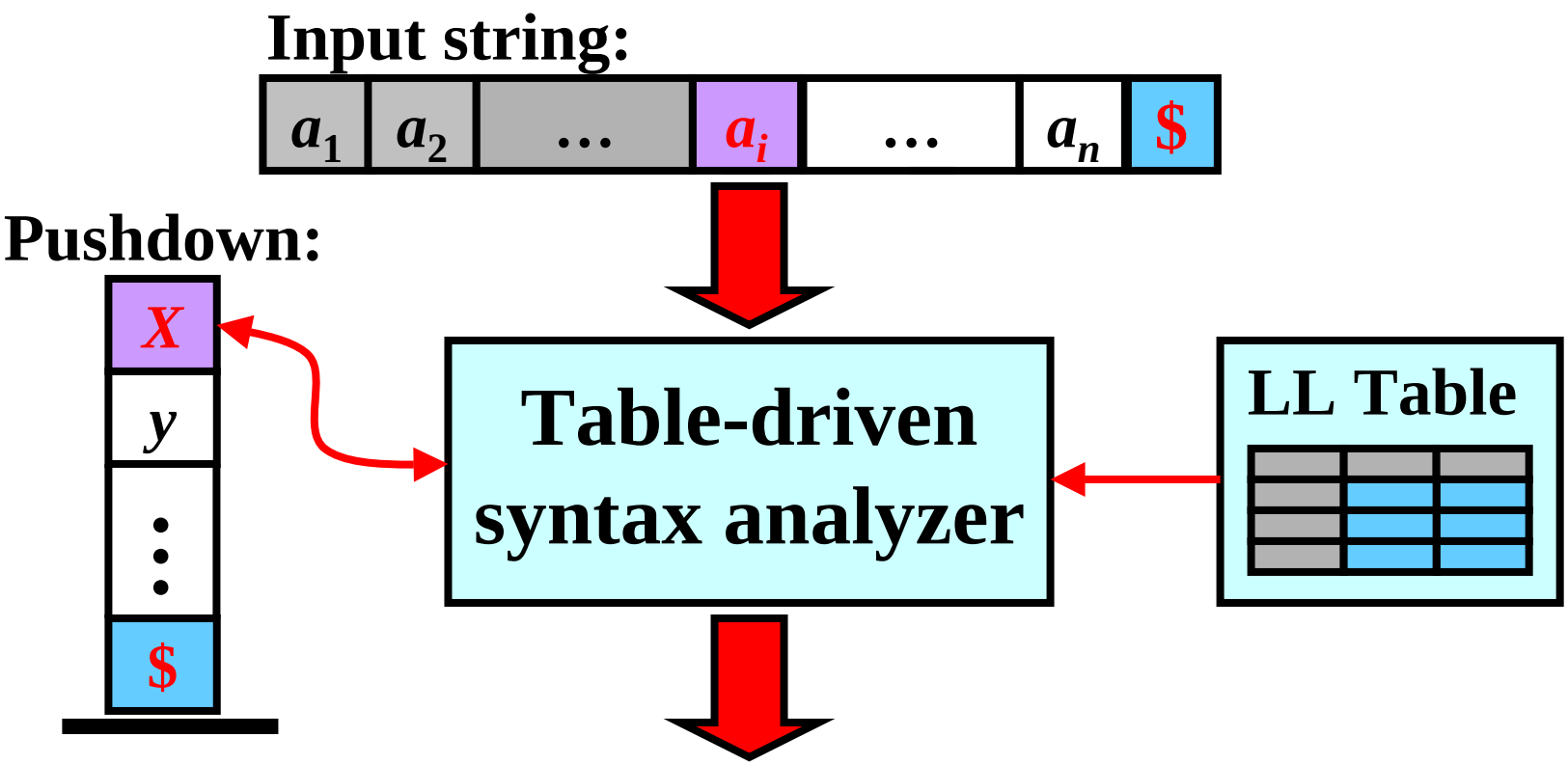
Input string:

i	*	*i*	$



Predictive Parsing

- **Model of table-driven syntax analyzer:**



Left parse = sequence of rules used in the leftmost derivation of the input string.

Table-Driven Parsing: Algorithm

- **Input:** LL-table for $G=(N, T, P, S)$; $x \in T^*$
- **Output:** Left parse of x if $x \in L(G)$; otherwise, error

---

- **Method:**
- push(**$**) & push(***S***) onto the pushdown;
- **while** the pushdown is not empty **do**
  - let ***X*** = the pushdown top and ***a*** = the current token
  - **case *X* of:**
    - ***X*** = **$:** **if *a* = $ then success**
      else error;
    - ***X*** $\in T$**:** **if *X* = *a* then** pop(***X***) & read next ***a*** from input string
      else error;
    - ***X*** $\in N$**:** **if *r*: *X*** $\rightarrow$ ***x*** $\in$ LL-table[***X***, ***a***] **then** replace ***X*** with reversal(***x***) on the pushdown & write ***r*** to output
      else error;

end

Table-Driven Parsing: Example

	i	+	*	(	)	$
E	1			1		
E'		2			3	3
T	4			4		
T'		6	5		6	6
F	8			7		

Rules:

1: $E \rightarrow TE'$
2: $E' \rightarrow +TE'$
3: $E' \rightarrow \varepsilon$
4: $T \rightarrow FT'$
5: $T' \rightarrow *FT'$
6: $T' \rightarrow \varepsilon$
7: $F \rightarrow (E)$
8: $F \rightarrow i$

Input string: *i * i $*

Pushdown	Input	Rule	Derivation
$*E*	*i*i$*		*E*
$*E'T*	*i*i$*	1: $E \rightarrow TE'$	$\Rightarrow TE'$
$*E'T'F*	*i*i$*	4: $T \rightarrow FT'$	$\Rightarrow FT'E'$
$*E'T'i*	*i*i$*	8: $F \rightarrow i$	$\Rightarrow iT'E'$
$*E'T'*	**i$*	5: $T' \rightarrow *FT'$	$\Rightarrow i*FT'E'$
$*E'T'F**	**i$*		
$*E'T'F*	*i$*	8: $F \rightarrow i$	$\Rightarrow i*iT'E'$
$*E'T'i*	*i$*		
$*E'T'*	*$*	6: $T' \rightarrow \varepsilon$	$\Rightarrow i*iE'$
$*E'*	*$*	3: $E' \rightarrow \varepsilon$	$\Rightarrow i*i$
$	*$*		

Success

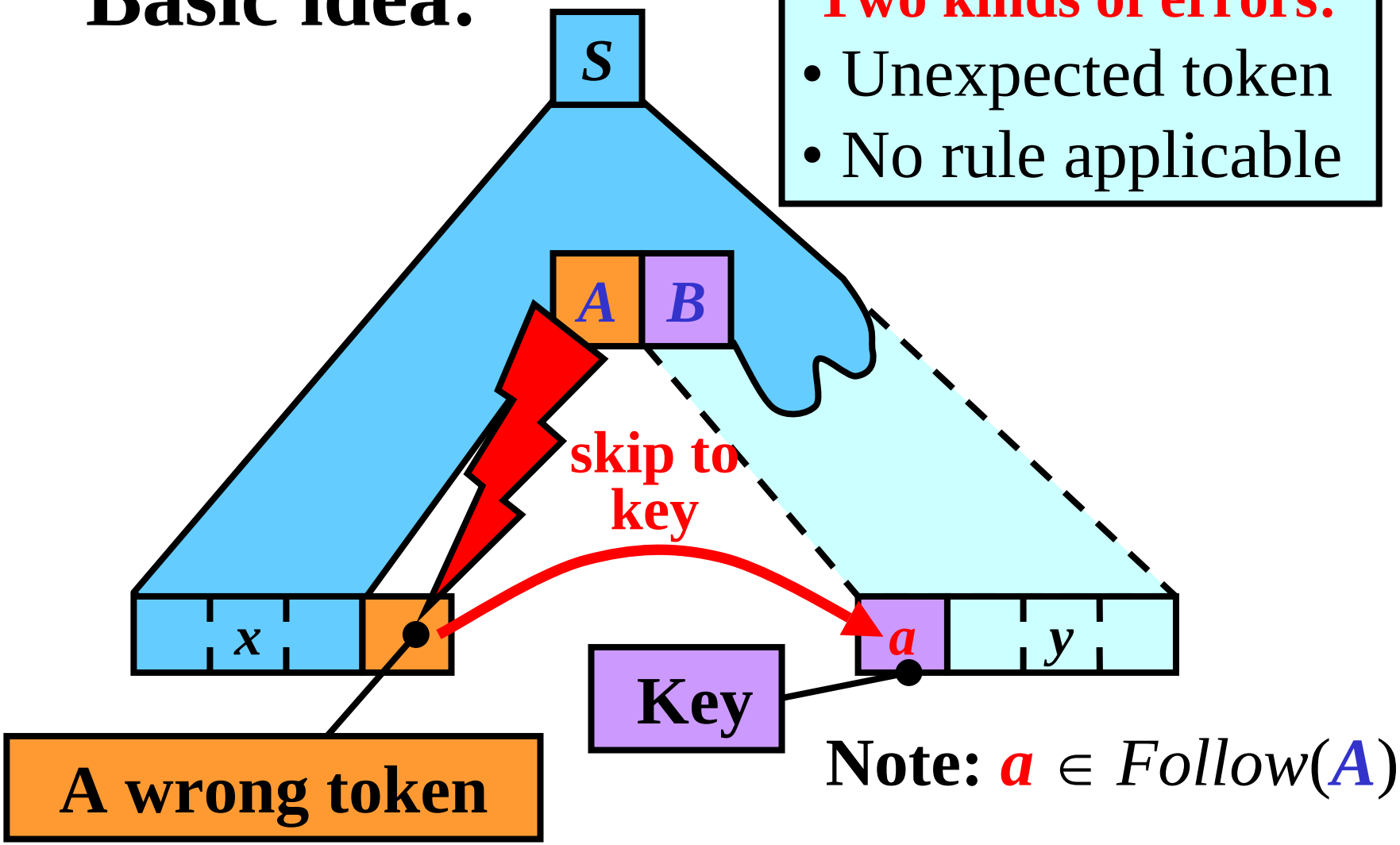
Left parse: 1485863

Handling Errors: Introduction

Basic idea:

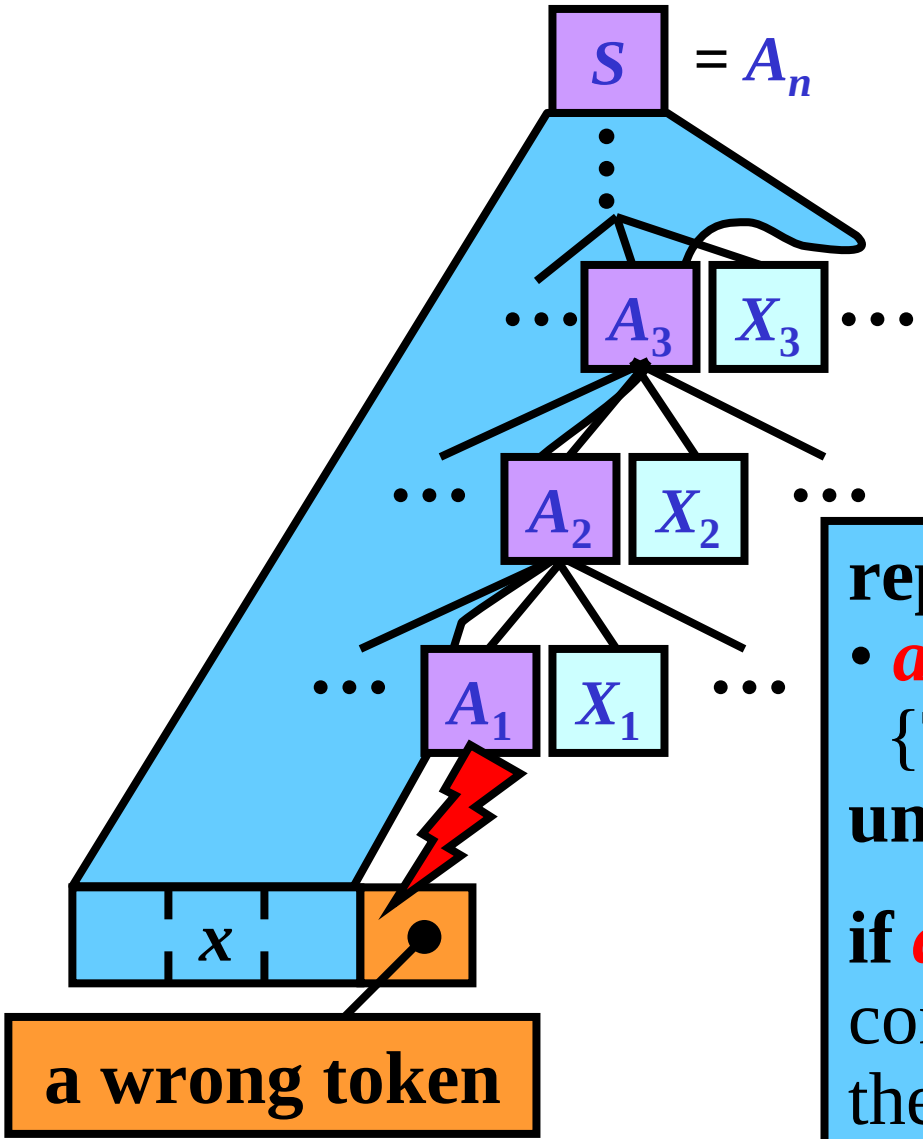
Two kinds of errors:

- Unexpected token
- No rule applicable



Note: $a \in Follow(A)$

Panic-Mode (Hartmann) Error Recovery



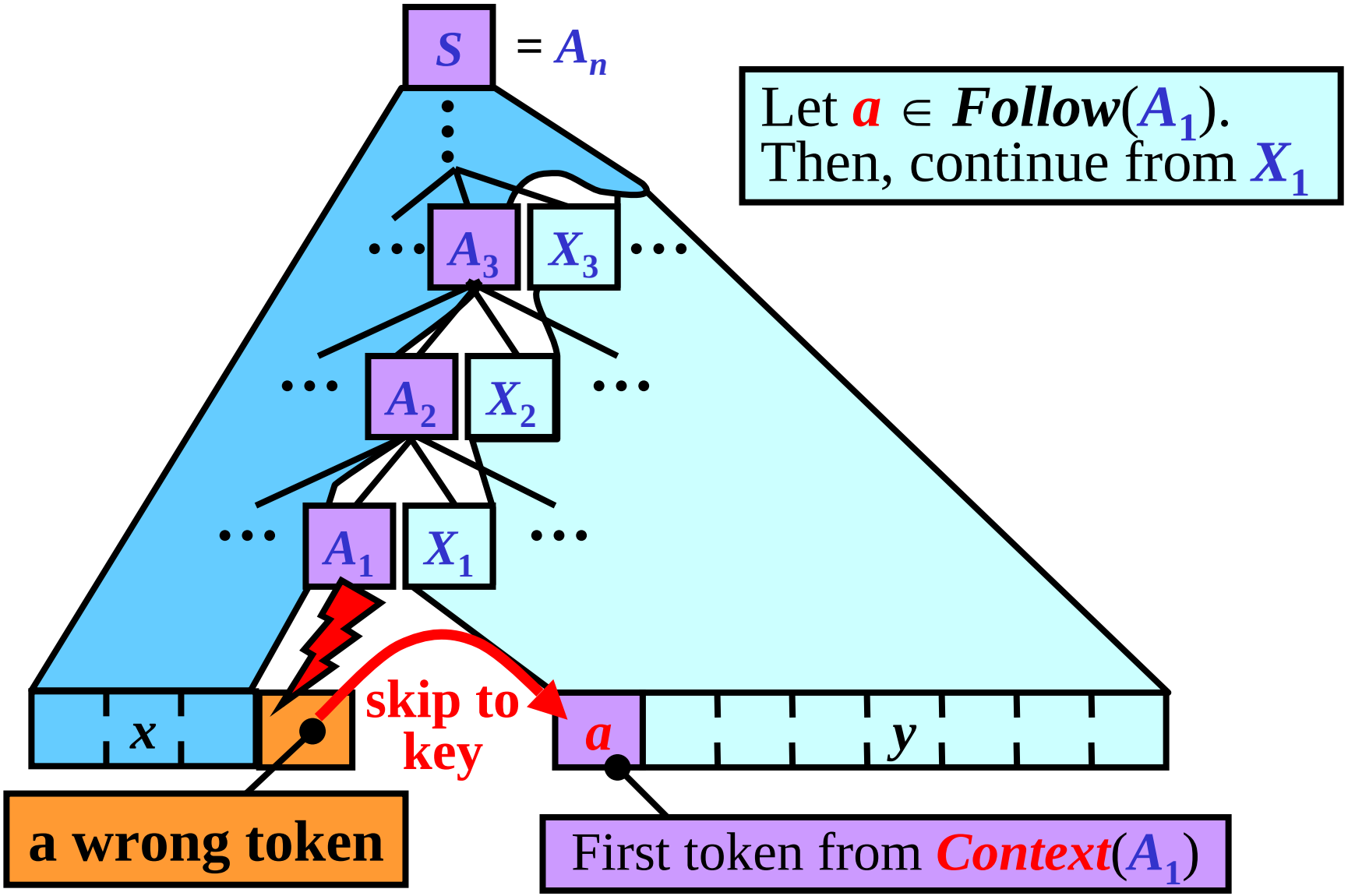
- Let ***Context***(A_1) = ***Follow***(A_1) ∪ ***Follow***(A_2) ∪ … ***Follow***(A_n)

repeat
- ***a*** := GetNextToken;
  {These tokens are skipped}

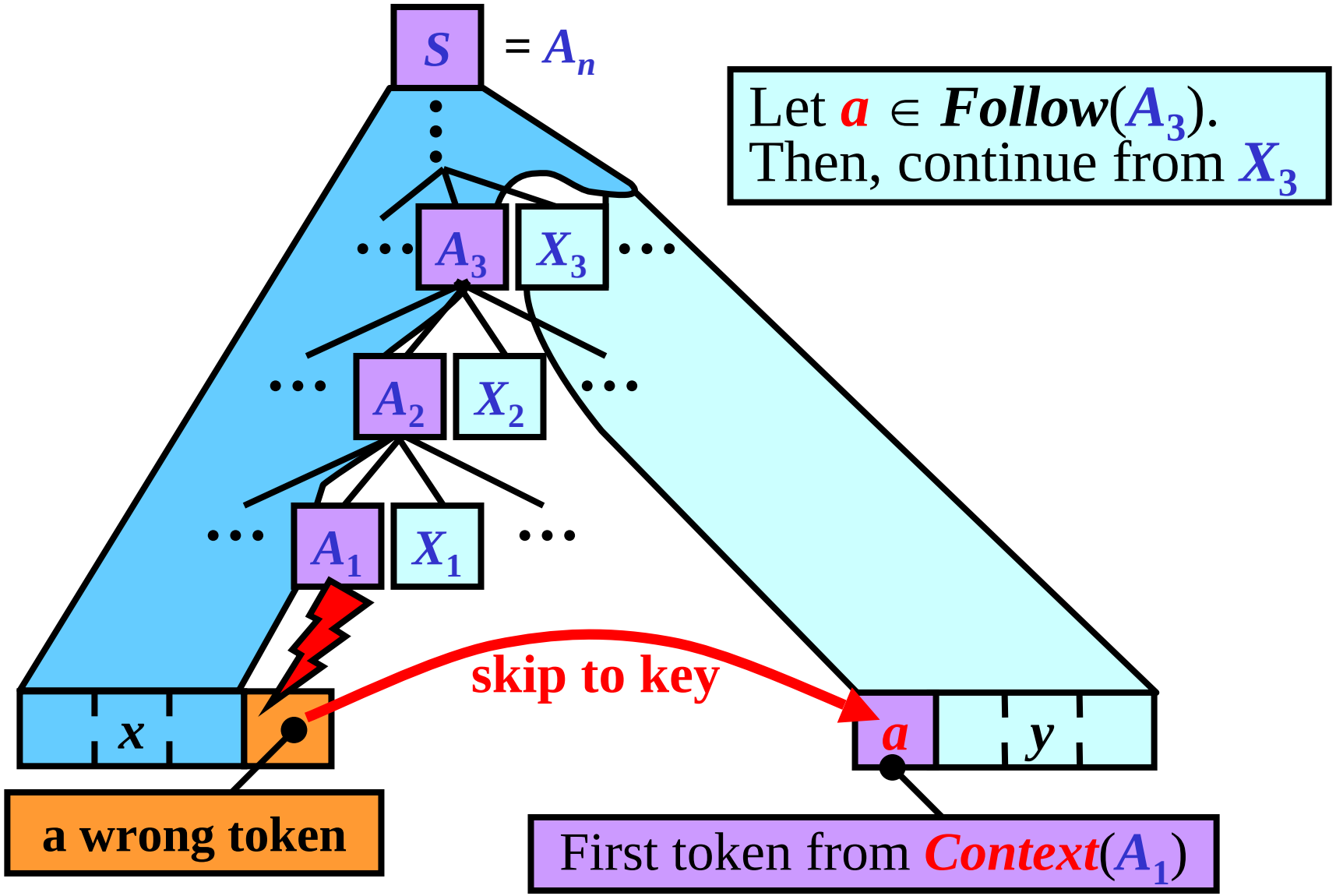
until ***a*** in ***Context***(A_1)

if ***a*** in ***Follow***(A_i) **then** continue with parsing from the symbol X_i.

Panic-Mode Recovery: Illustration 1/2



Panic-Mode Recovery: Illustration 2/2



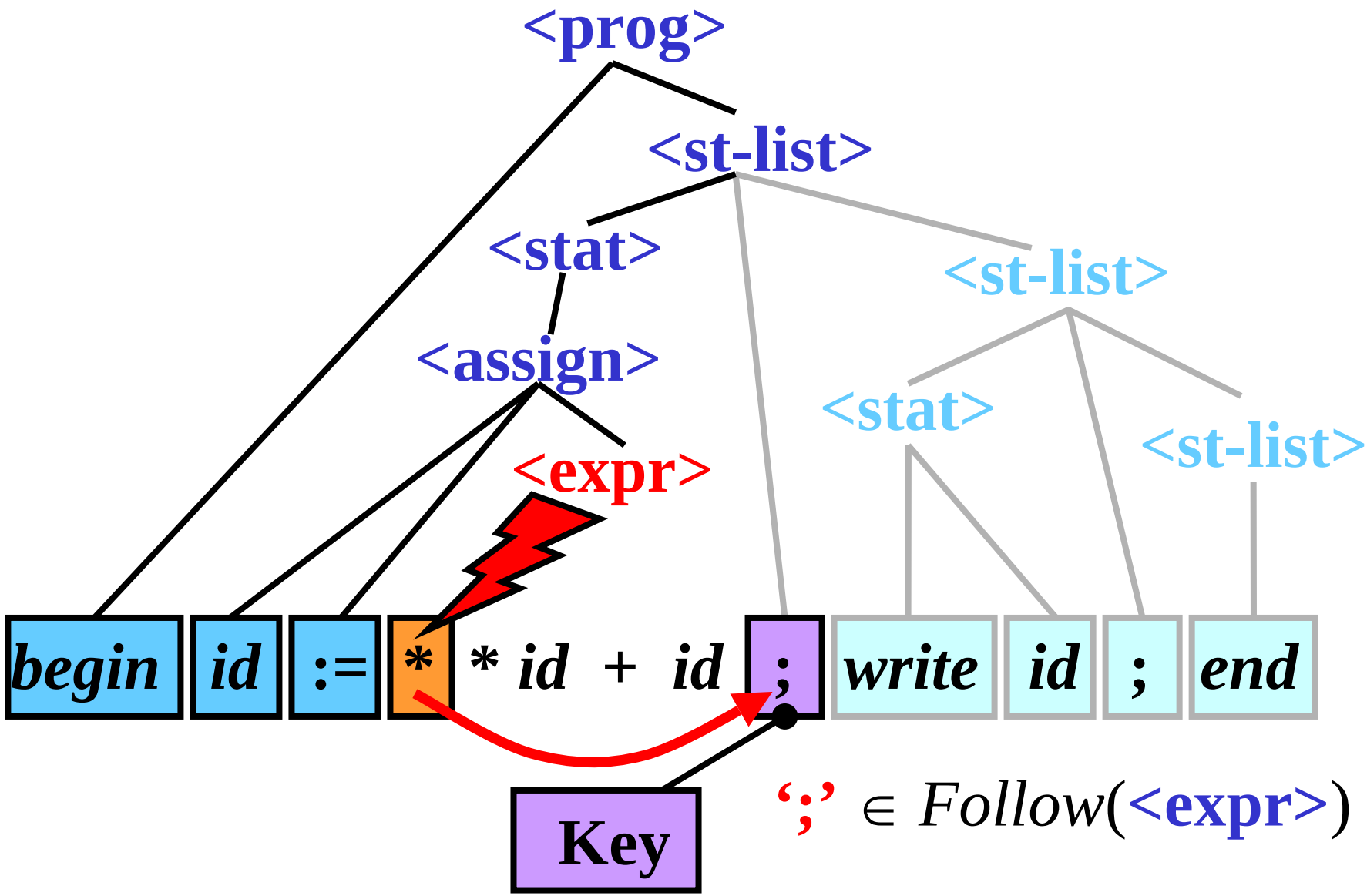
Context(*X*) for Predictive Parser: Variant I

For $G = (N, T, P, S)$,

***Context*(*A*) = *Follow*(*A*) for every $A \in N$**

- **Method:**
- Let ***A*** be pushdown top & no rule is applicable:
- **repeat**
  a := GetNextToken;
  {These tokens are skipped}
  until ***a*** in ***Context***(***A***)
- pop ***A*** from the pushdown**;**

Variant I: Example



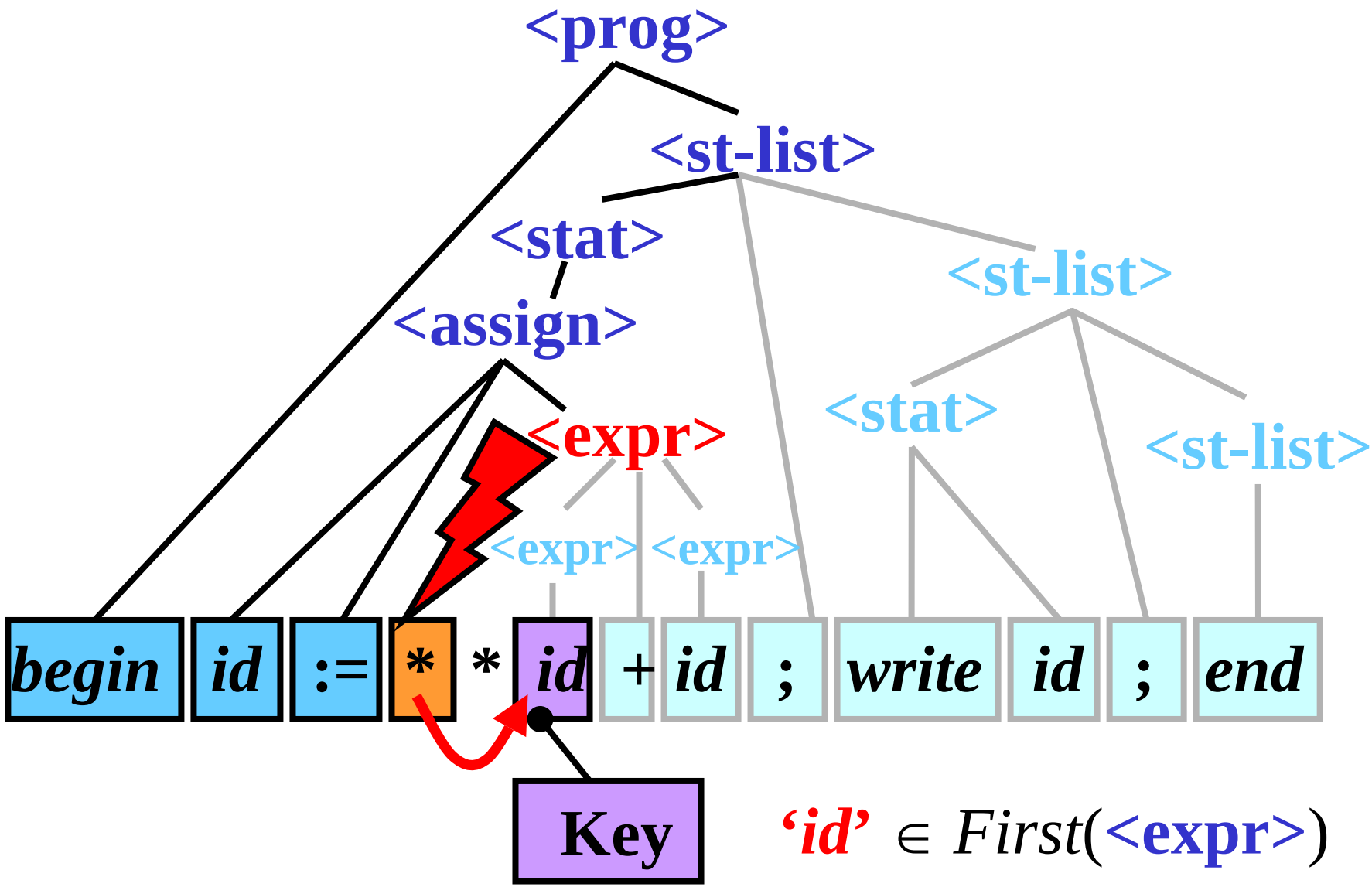
Context(*X*) for Predictive Parser: Variant II

For *G* = (*N*, *T*, *P*, *S*),

Context*(*A*) = *First*(*A*) ∪ *Follow*(*A*) for every *A* ∈ *N

- **Method:**
- Let ***A*** be pushdown top & no rule is applicable:
- **repeat**
  a := GetNextToken;
  {These tokens are skipped}
  until ***a*** in ***Context***(***A***)
- **if** ***a*** ∈ *First*(***A***) **then** resume according to ***A***
  else pop ***A*** from the pushdown // ***a*** ∈ *Follow*(***A***)

Variant II: Example



'id' ∈ *First*(<expr>)