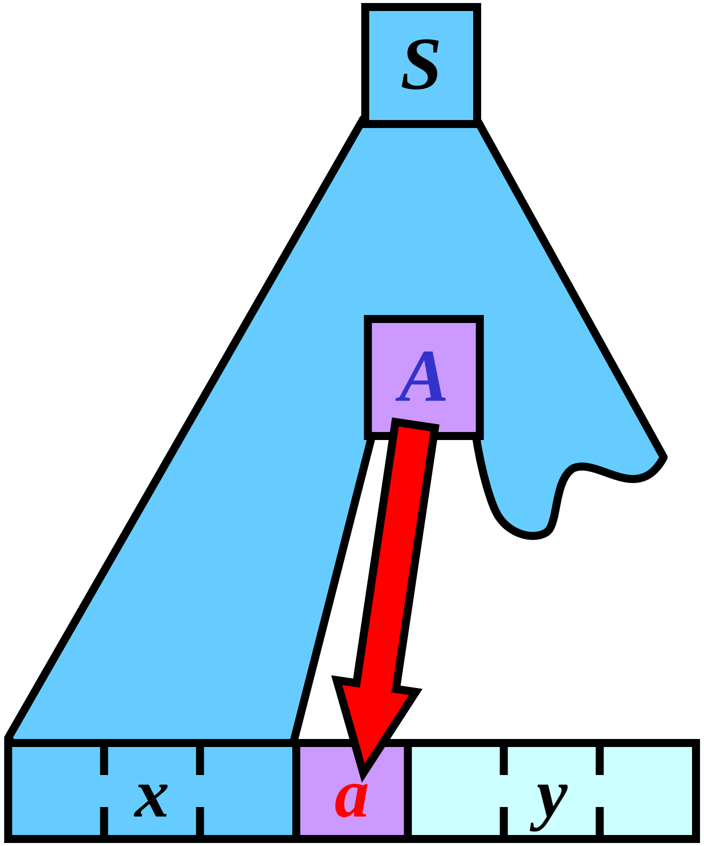
Top-Down Parsing

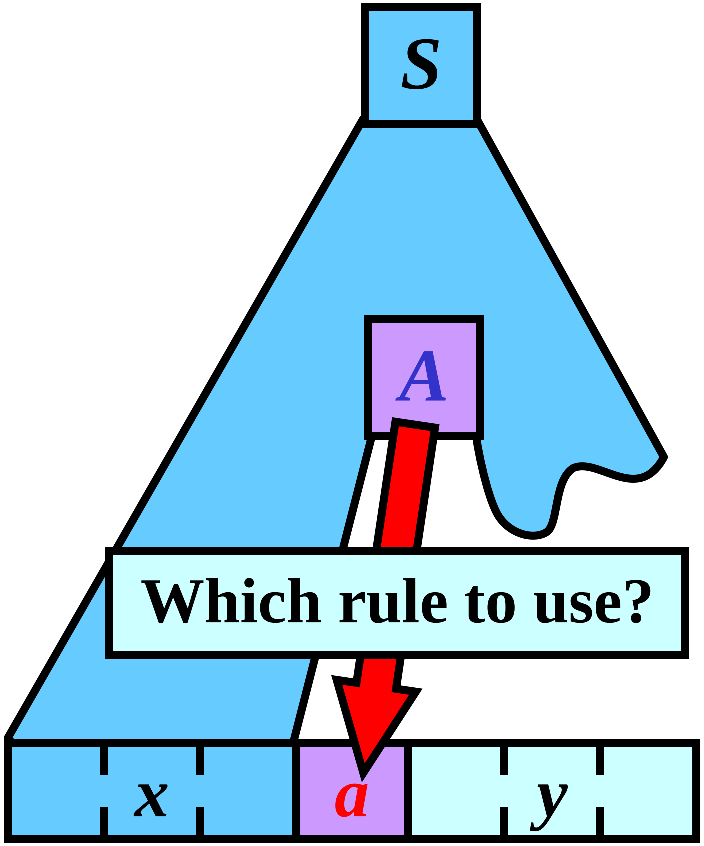
Top-Down Parsing: Introduction

Problem:



Top-Down Parsing: Introduction

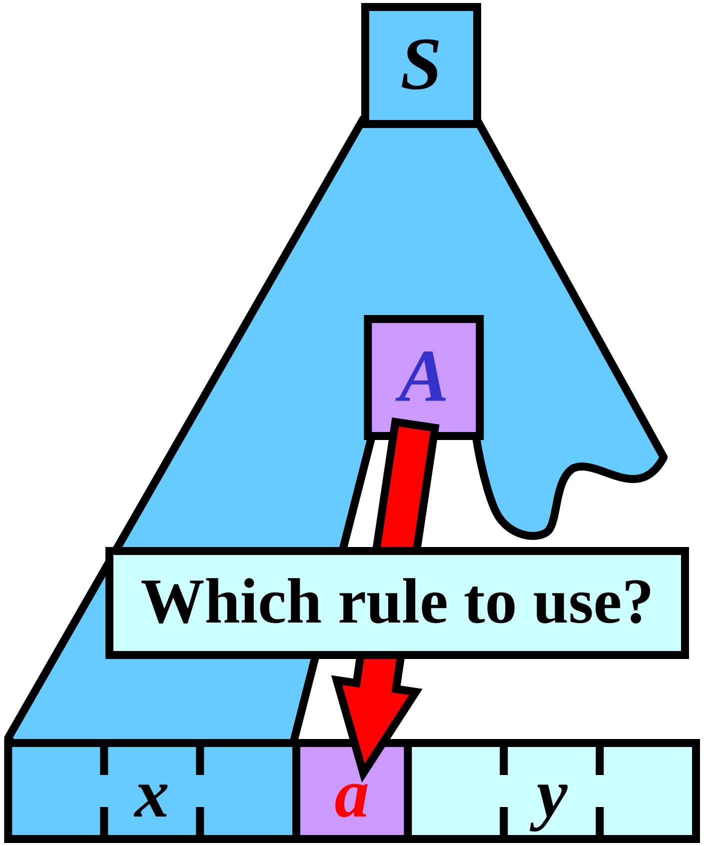
Problem:



Which rule to use?

Top-Down Parsing: Introduction

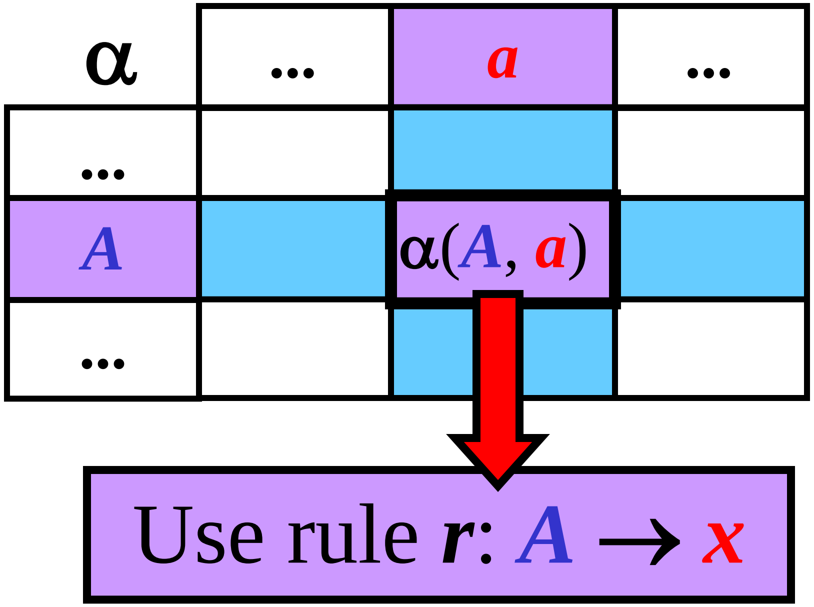
Problem:



Which rule to use?

Basic idea:

Table:

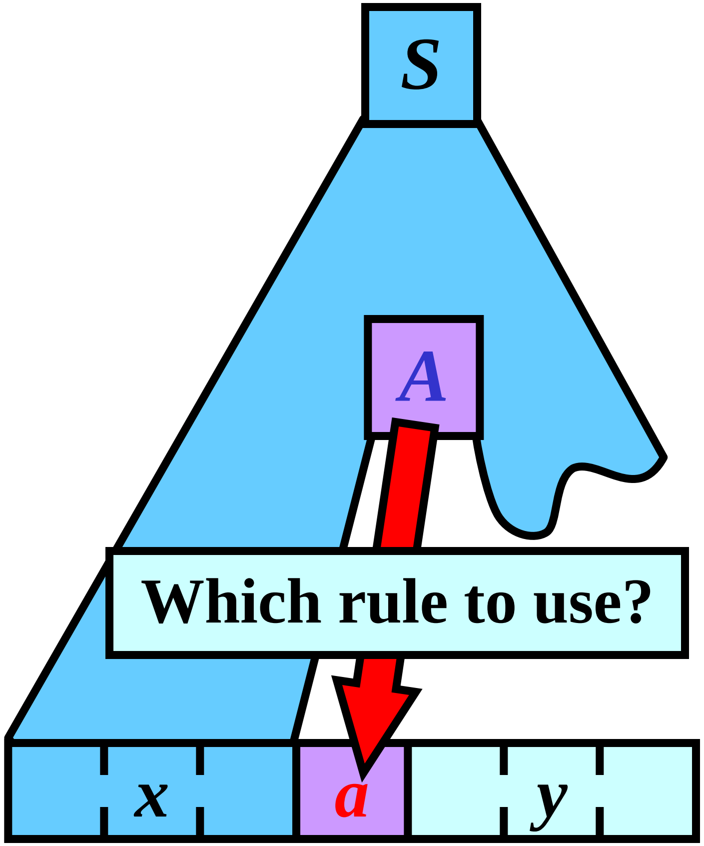


α	...	a	...
...			
A		$\alpha(A, a)$	
...			

Use rule r: $A \rightarrow x$

Top-Down Parsing: Introduction

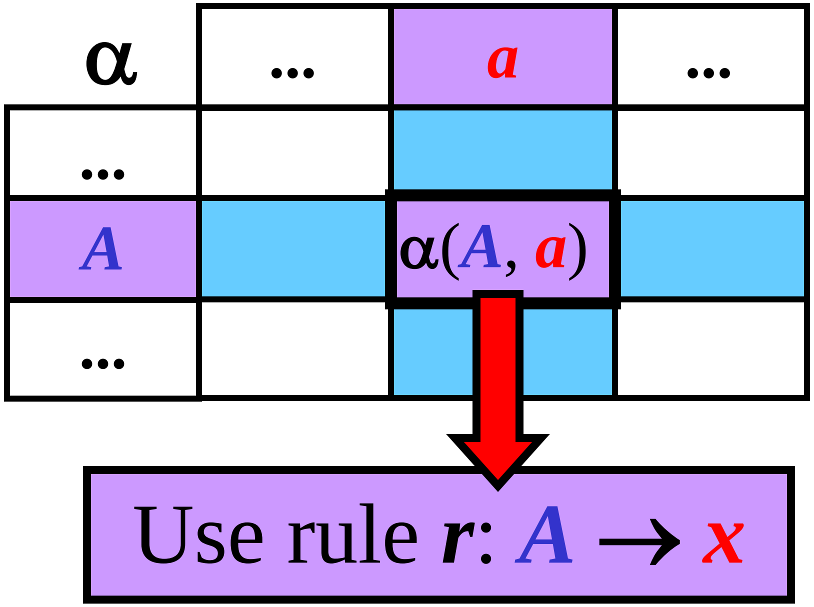
Problem:



Which rule to use?

Basic idea:

Table:



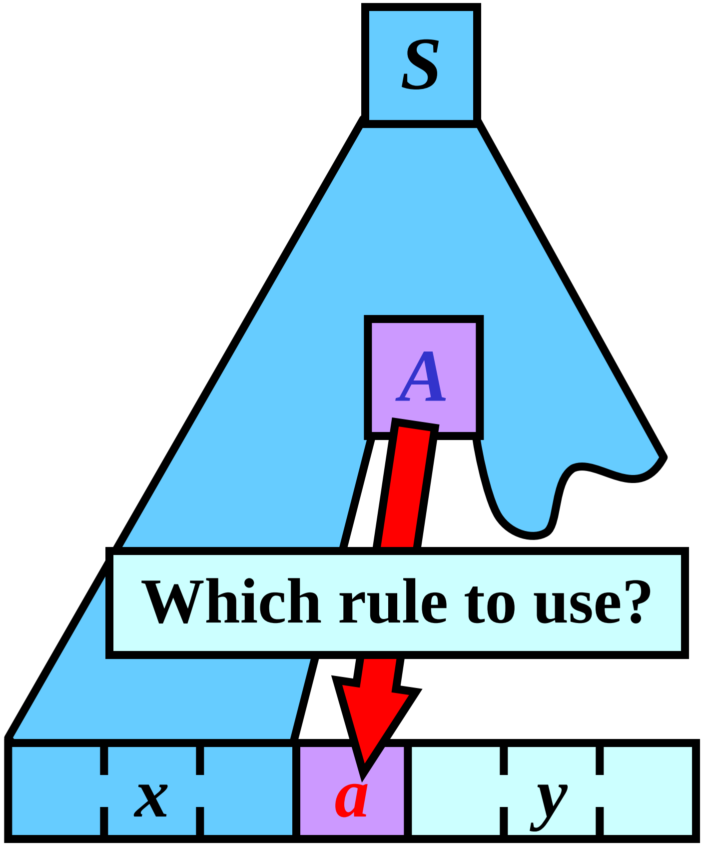
α	...	***a***	...
...			
A		α(*A*, *a*)	
...			

Use rule ***r***: ***A*** → ***x***

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

Top-Down Parsing: Introduction

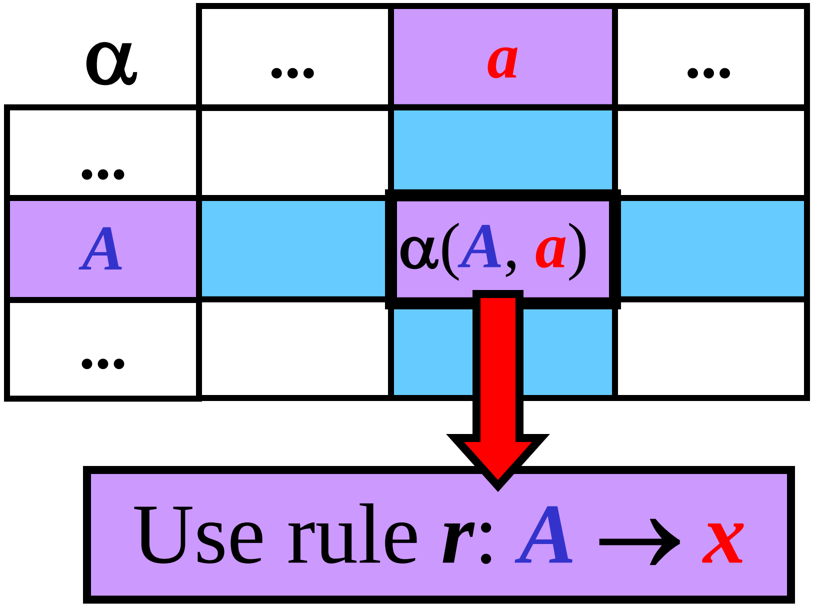
Problem:



Which rule to use?

Basic idea:

Table:



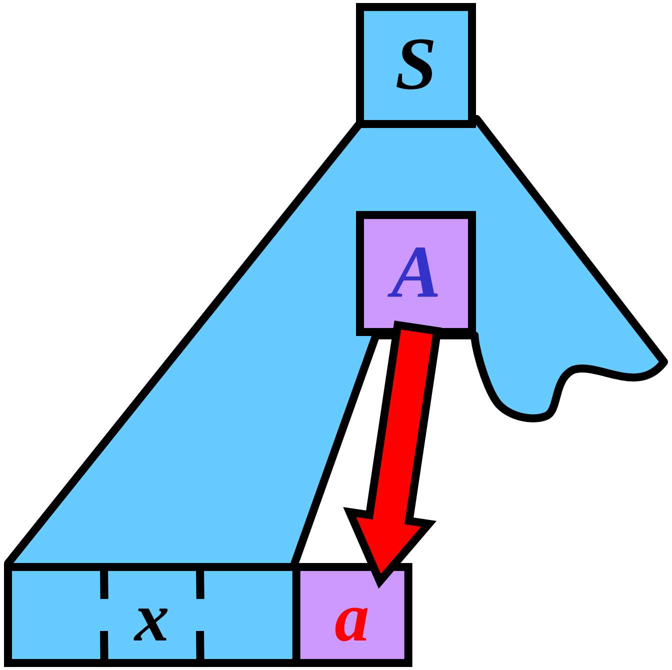
α	...	a	...
...			
A		α(A, a)	
...			

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

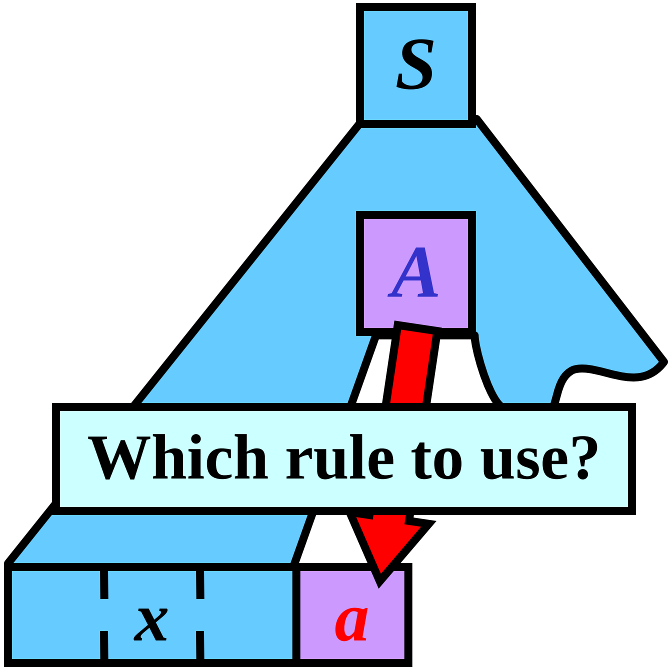
Question: Could you construct this table for **any** CFG?

Answer: **NO**

A Table-Based Selection of a Rule

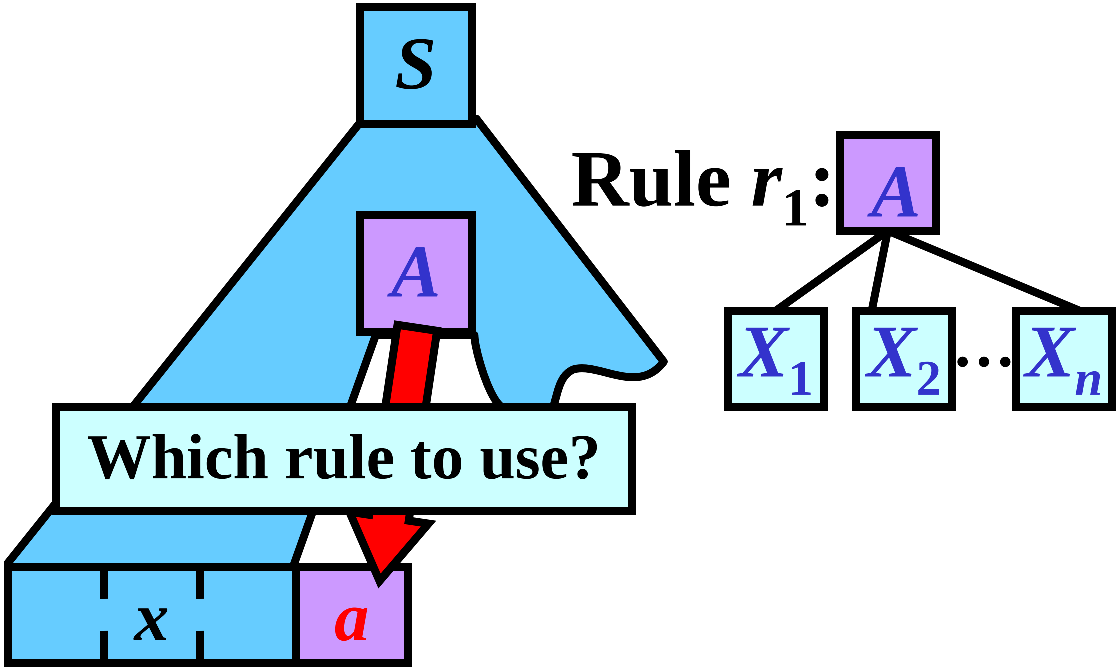


A Table-Based Selection of a Rule



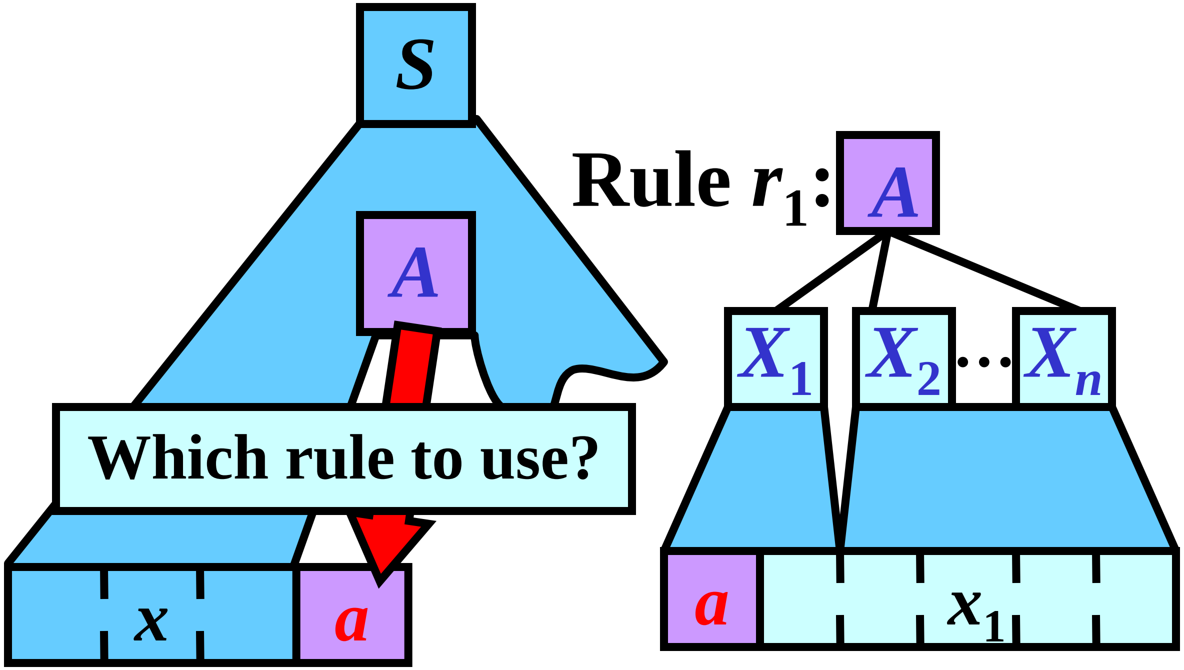
Which rule to use?

A Table-Based Selection of a Rule

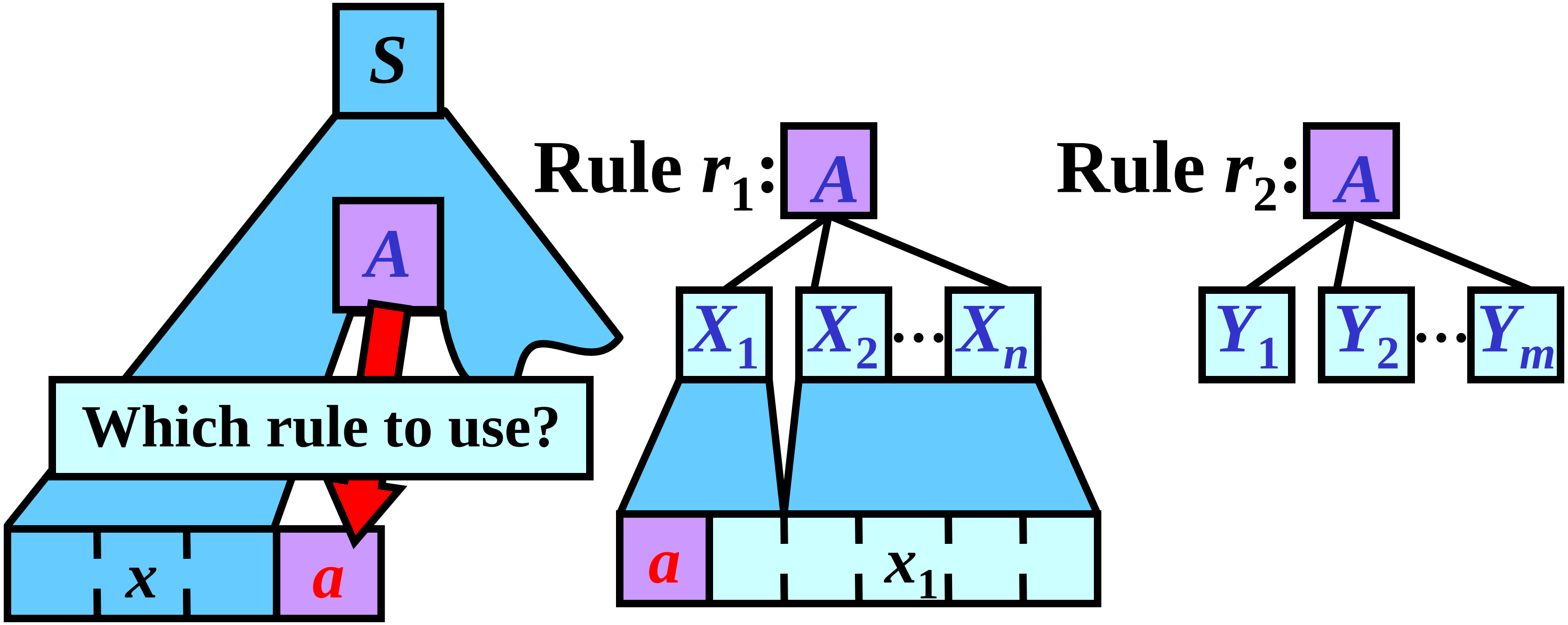


Which rule to use?

A Table-Based Selection of a Rule



A Table-Based Selection of a Rule

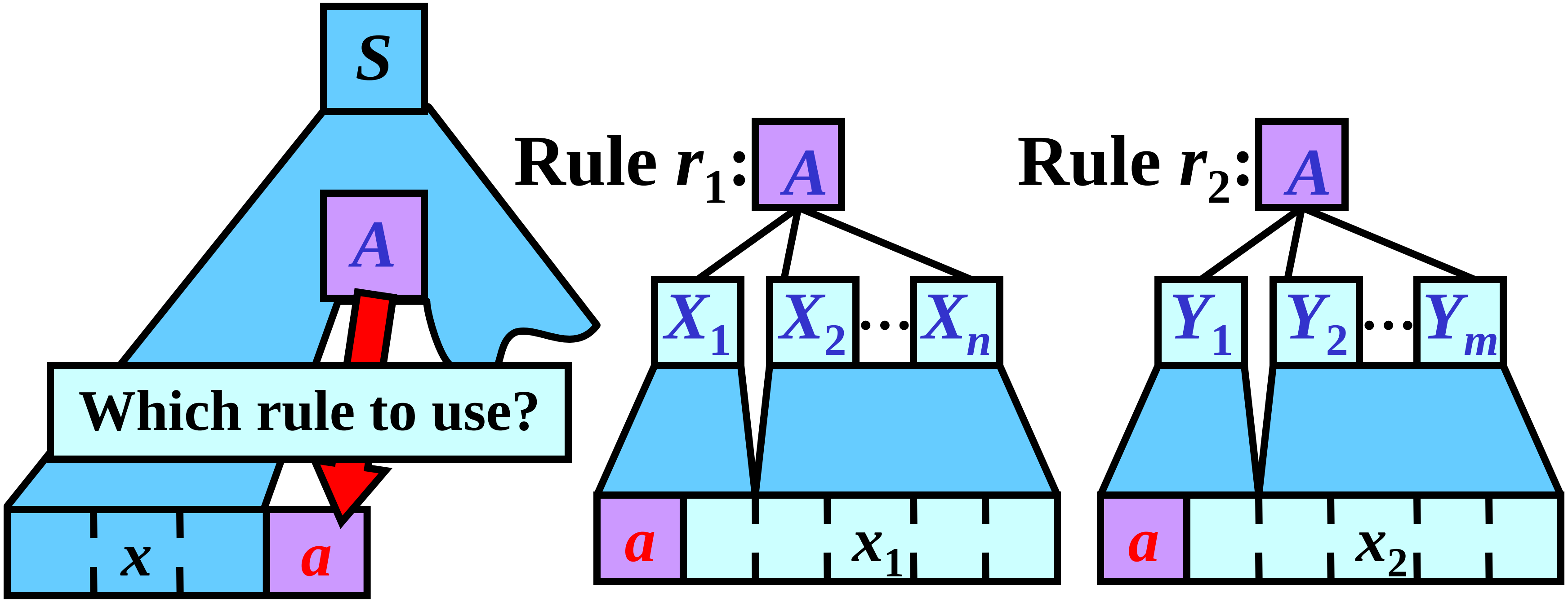


Which rule to use?

Rule r_1: $A \to X_1 X_2 \dots X_n$

Rule r_2: $A \to Y_1 Y_2 \dots Y_m$

A Table-Based Selection of a Rule



A Table-Based Selection of a Rule

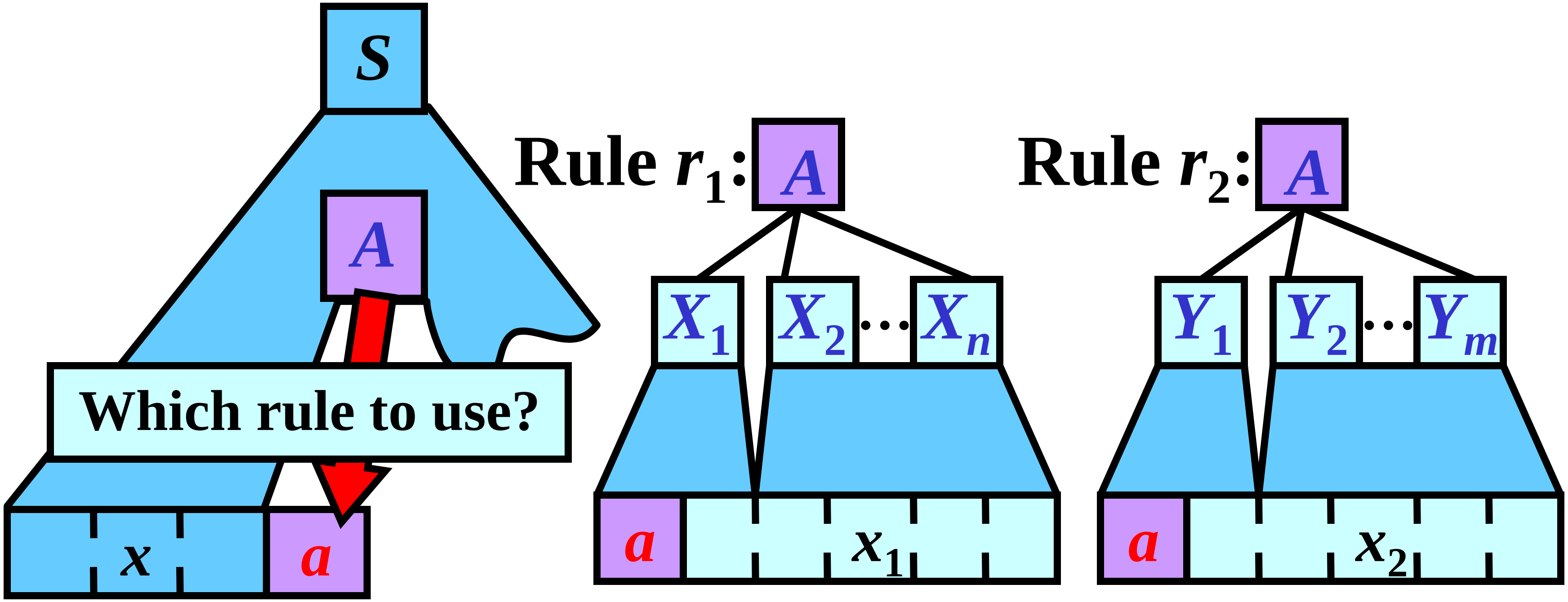


Table:

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

A Table-Based Selection of a Rule

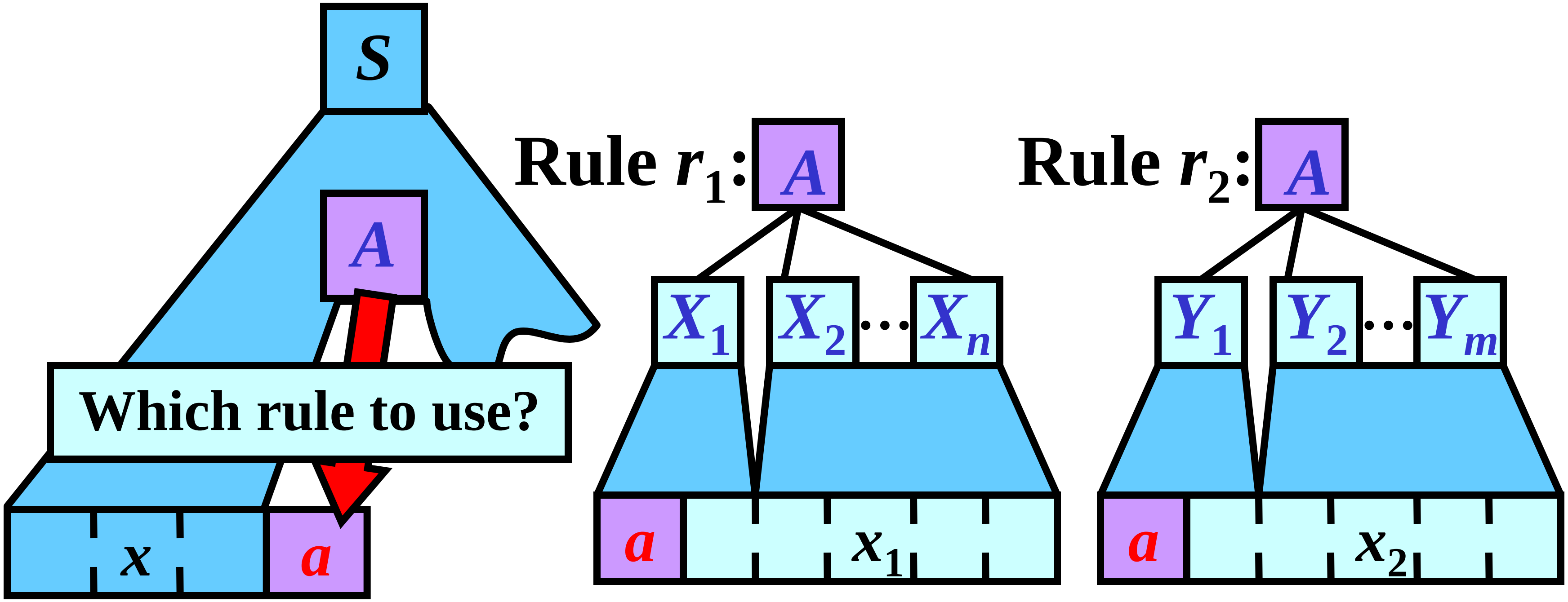
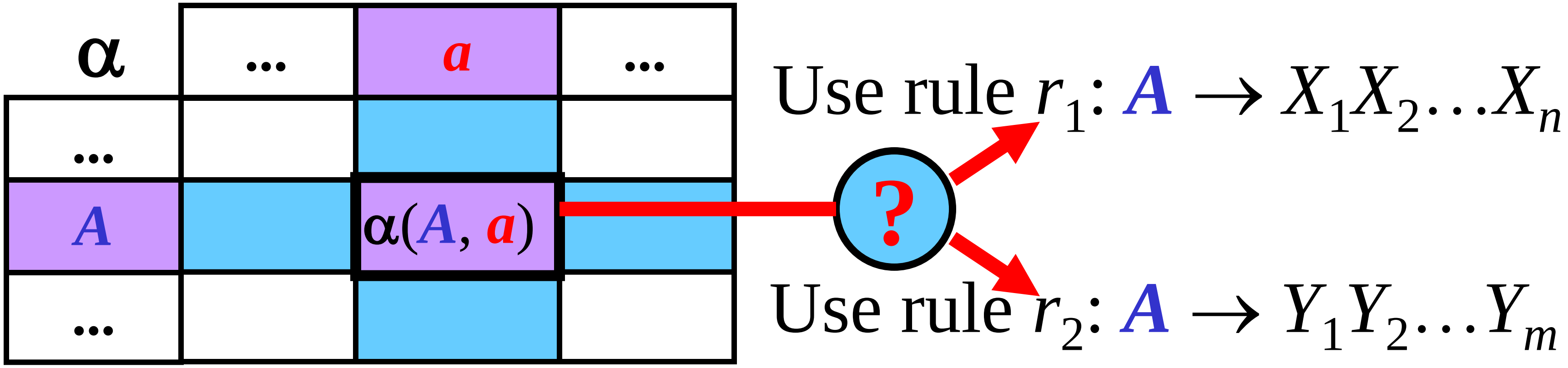


Table:



α	...	a	...
...			
A		α(A, a)	
...			

Use rule r_1: $A \to X_1X_2\ldots X_n$

Use rule r_2: $A \to Y_1Y_2\ldots Y_m$

Set *First*

Gist: *First*(x) is the set of all terminals that can begin a string 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 $First(x) = \{a: a \in T, x \Rightarrow^* ay; y \in (N \cup T)^*\}$.

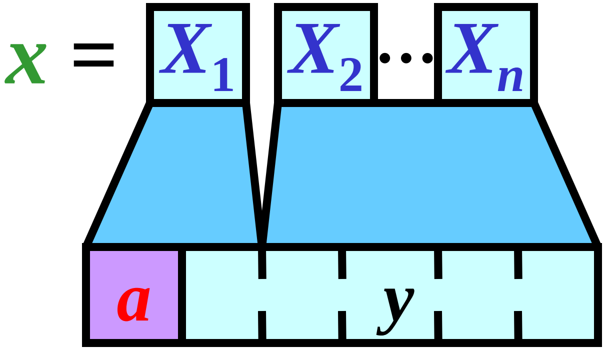
Illustration: $x = \boxed{X_1}\,\boxed{X_2}\cdots\boxed{X_n}$

Set *First*

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

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

Illustration:

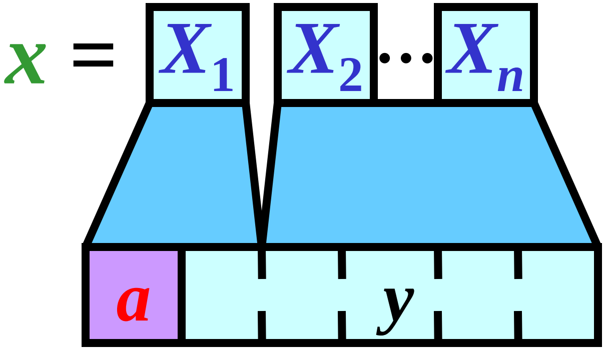


Set *First*

Gist: *First*(x) is the set of all terminals that can begin a string 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:



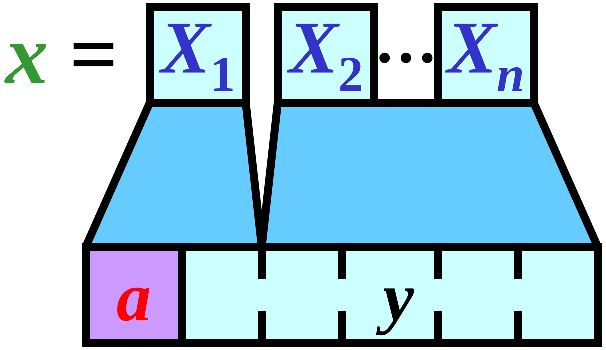
$$x = X_1X_2 \ldots X_n \Rightarrow^* ay$$

Set *First*

Gist: *First*(x) is the set of all terminals that can begin a string 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

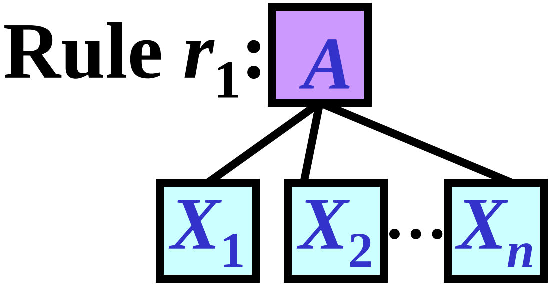
Definition: Let $G = (N, T, P, S)$ be a CFG without ε-rules. G is an *LL grammar* if for every $a \in T$ and every $A \in N$ there is **no more than one** rule $A \to X_1X_2...X_n \in P$ such that $a \in First(X_1X_2...X_n)$

Illustration:

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 $A \rightarrow X_1X_2 \ldots X_n \in P$ such that $a \in First(X_1X_2 \ldots X_n)$

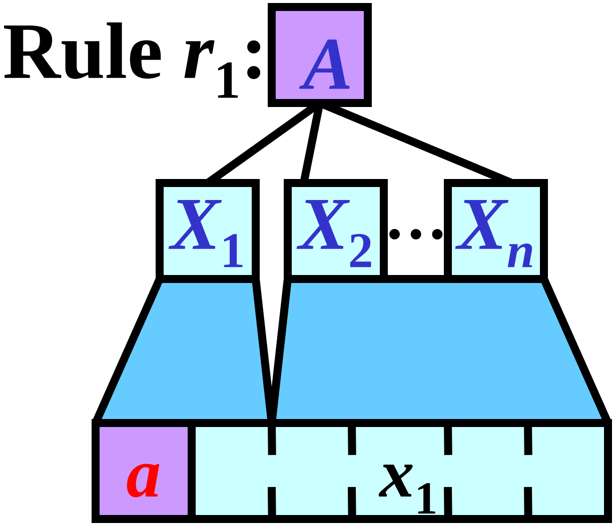
Illustration:



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 $A \to X_1X_2\ldots X_n \in P$ such that $a \in First(X_1X_2\ldots X_n)$

Illustration:

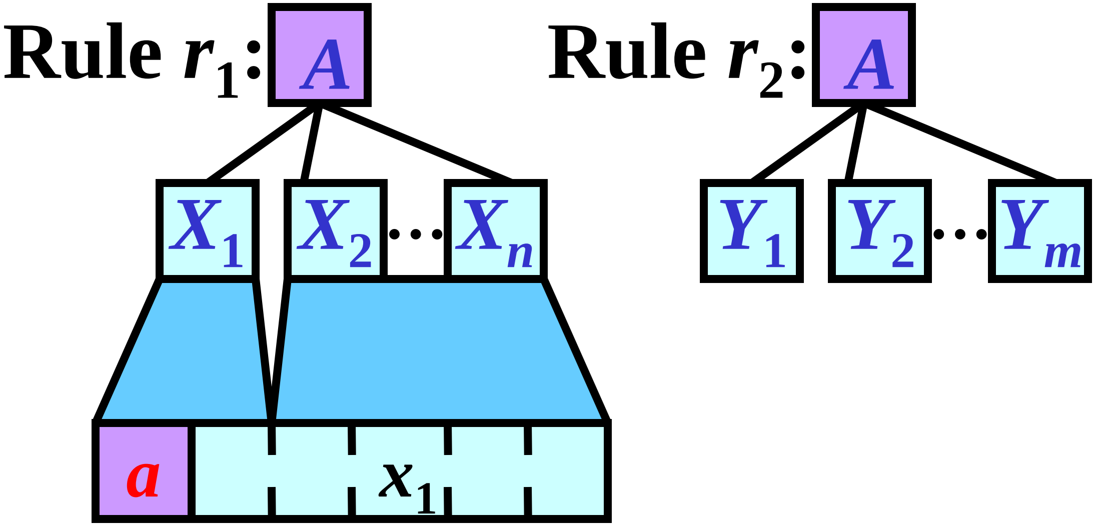


$a \in First(X_1X_2\ldots X_n)$

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 $A \to X_1X_2\ldots X_n \in P$ such that $a \in First(X_1X_2\ldots X_n)$

Illustration:

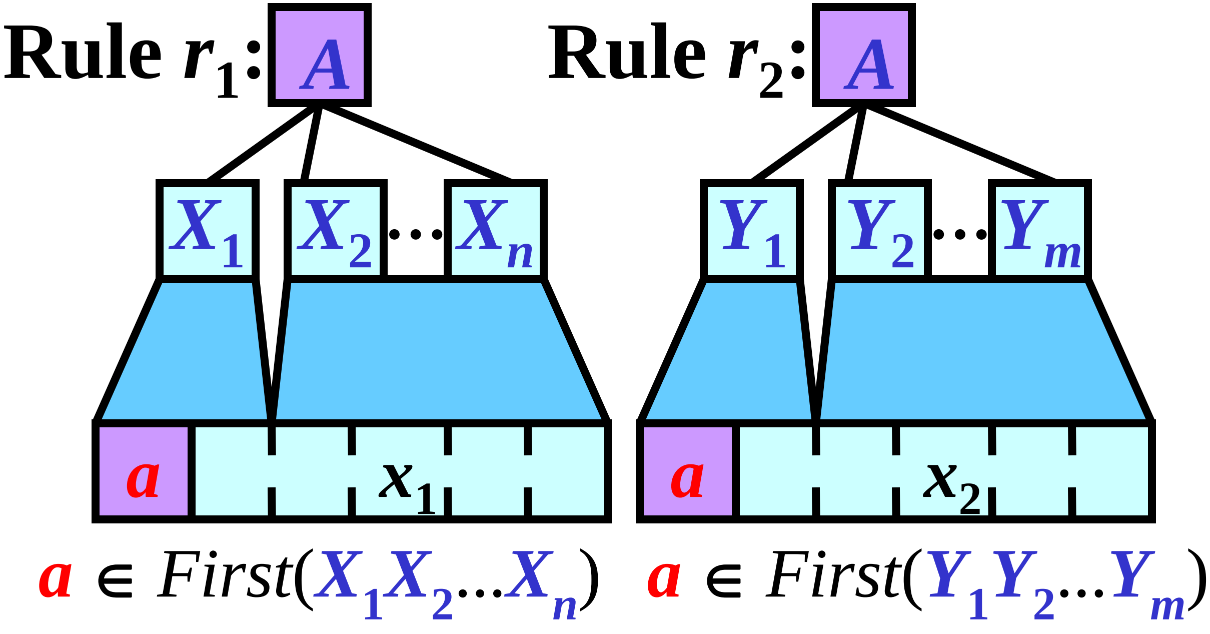


$a \in First(X_1X_2\ldots X_n)$

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:

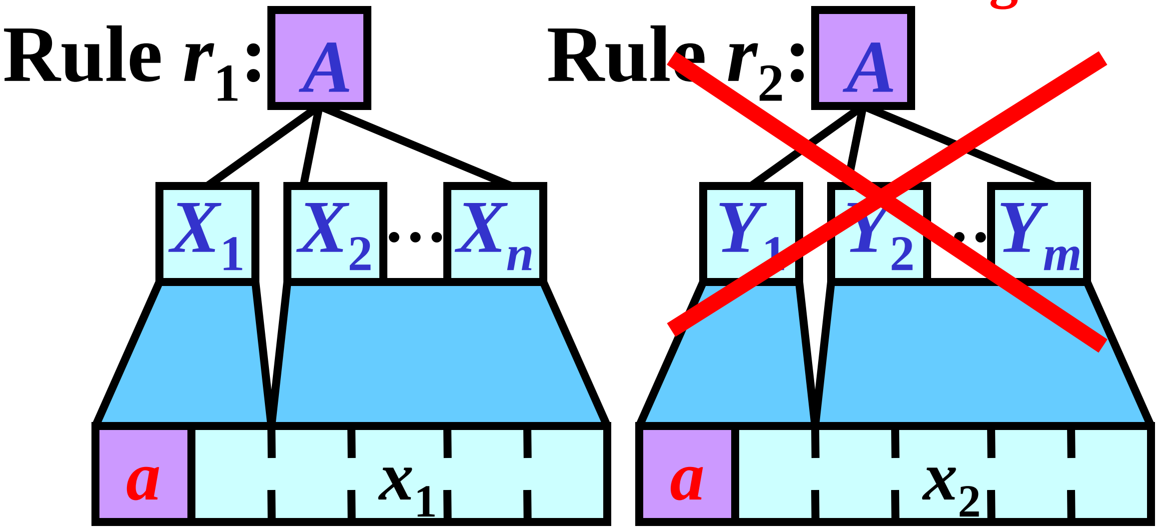


$a \in First(X_1X_2…X_n)$ $a \in First(Y_1Y_2…Y_m)$

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:



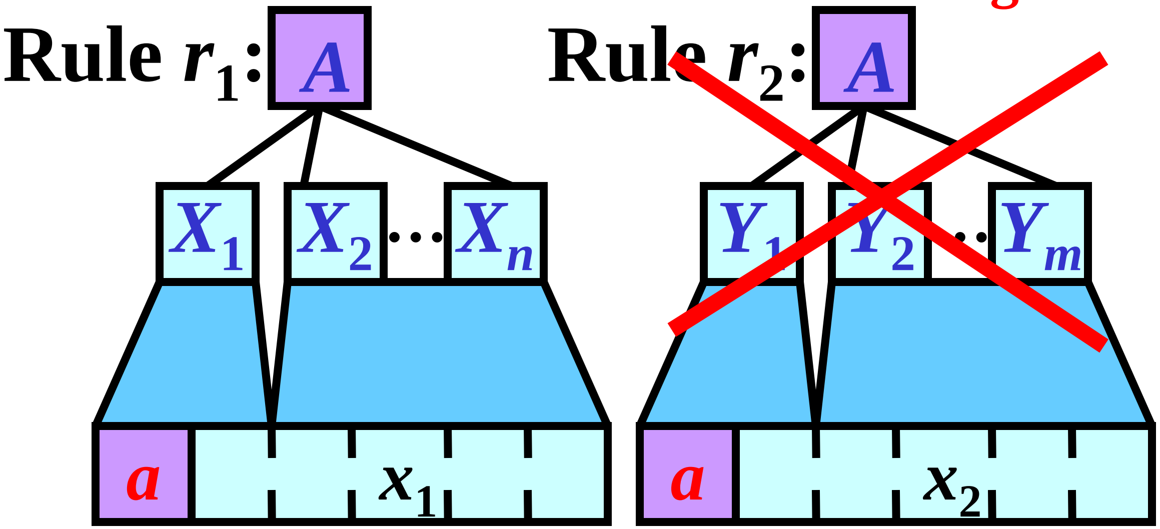
$a \in First(X_1X_2...X_n)$ $a \in First(Y_1Y_2...Y_m)$

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 $A \rightarrow X_1X_2\dots X_n \in P$ such that $a \in First(X_1X_2\dots X_n)$

Illustration:

Ruled out in an LL grammar



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

Table:

α	...	a	...
...			
A		$\alpha(A, a)$	
...			

Only rule r_1:
$A \rightarrow X_1X_2\dots X_n$

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

- **Method:**
- for each $\boldsymbol{a} \in T$: *First*($\boldsymbol{a}$) := {$\boldsymbol{a}$}
- **Apply the following rule until no *First* set can be changed:**
- if $\boldsymbol{A} \rightarrow \boldsymbol{X_1 X_2 \dots X_n} \in P$, then add *First*($\boldsymbol{X_1}$) to *First*($\boldsymbol{A}$)

Illustration:

Algorithm: *First*(*X*)

- **Input:** $G = (N, T, P, S)$ **without ε-rules**
- **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 $A \rightarrow X_1X_2\ldots X_n \in P$, then add $First(X_1)$ to $First(A)$

Illustration:

1) for each $a \in T$:

$First(a) := \{a\}$

because $a \Rightarrow^0 a$

Algorithm: *First*(X)

- **Input:** $G = (N, T, P, S)$ **without ε-rules**
- **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 $A \rightarrow X_1X_2\ldots X_n \in P$, then add *First*($X_1$) to *First*($A$)

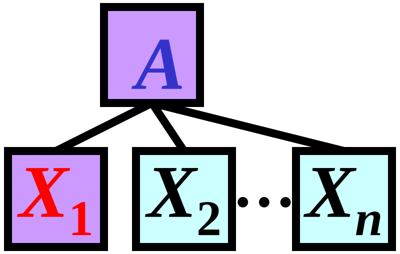
Illustration:

1) for each $a \in T$:

***First*(a) := {a}**

because $a \Rightarrow^0 a$

2)



Algorithm: *First*(X)

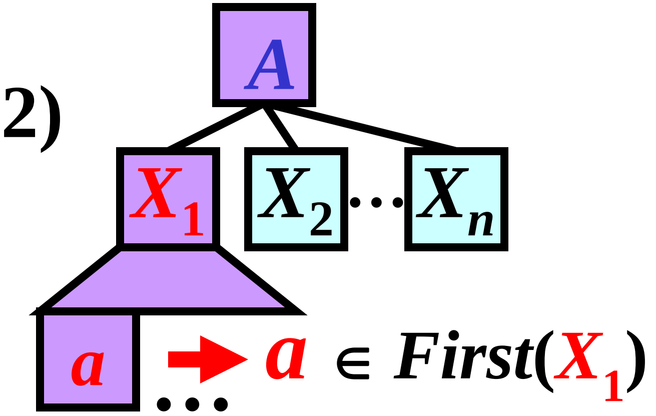
- **Input:** $G = (N, T, P, S)$ **without ε-rules**
- **Output:** $First(X)$ for every $X \in N \cup T$

- **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)



Algorithm: *First*(*X*)

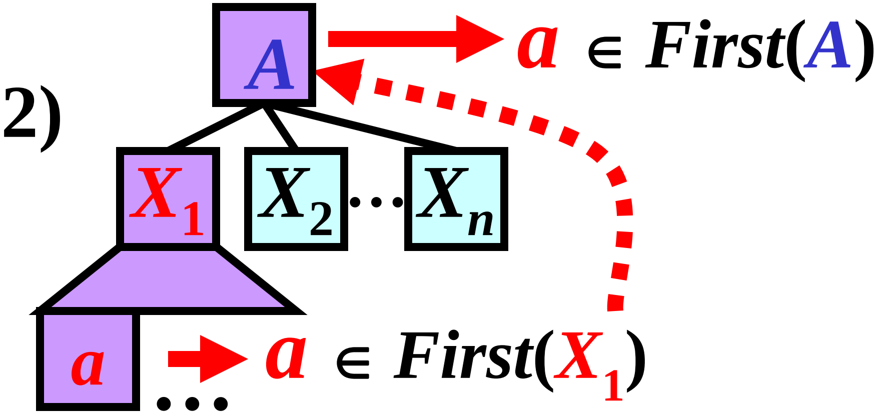
- **Input:** $G = (N, T, P, S)$ **without ε-rules**
- **Output:** $First(X)$ for every $X \in N \cup T$

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

First(**id**) := {**id**}
First(**int**) := {**int**}
First(**:=**) := {**:=**}
First(**add**) := {**add**}

First(**,**) := {**,**}
First(**(**) := {**(**}
First(**)**) := {**)**}
First(**;**) := {**;**}

First(*X*) for SPL: Example

First(**begin**) := {**begin**}
First(**end**) := {**end**}
First(**read**) := {**read**}
First(**write**) := {**write**}

First(**id**) := {**id**}
First(**int**) := {**int**}
First(**:=**) := {**:=**}
First(**add**) := {**add**}

First(**,**) := {**,**}
First(**(**) := {**(**}
First(**)**) := {**)**}
First(**;**) := {**;**}

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

First(*X*) for SPL: Example

First(**begin**) := {**begin**}
First(**end**) := {**end**}
First(**read**) := {**read**}
First(**write**) := {**write**}

First(**id**) := {**id**}
First(**int**) := {**int**}
First(**:=**) := {**:=**}
First(**add**) := {**add**}

First(**,**) := {**,**}
First(**(**) := {**(**}
First(**)**) := {**)**}
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>**) = {**)**, **,**}

First(*X*) for SPL: Example

First(**begin**) := {**begin**}
First(**end**) := {**end**}
First(**read**) := {**read**}
First(**write**) := {**write**}

First(**id**) := {**id**}
First(**int**) := {**int**}
First(**:=**) := {**:=**}
First(**add**) := {**add**}

First(**,**) := {**,**}
First(**(**) := {**(**}
First(**)**) := {**)**}
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**}

First(*X*) for SPL: Example

First(**begin**) := {**begin**}
First(**end**) := {**end**}
First(**read**) := {**read**}
First(**write**) := {**write**}

First(**id**) := {**id**}
First(**int**) := {**int**}
First(**:=**) := {**:=**}
First(**add**) := {**add**}

First(**,**) := {**,**}
First(**(**) := {**(**}
First(**)**) := {**)**}
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**}

First(*X*) for SPL: Example

First(**begin**) := {**begin**}
First(**end**) := {**end**}
First(**read**) := {**read**}
First(**write**) := {**write**}

First(**id**) := {**id**}
First(**int**) := {**int**}
First(**:=**) := {**:=**}
First(**add**) := {**add**}

First(**,**) := {**,**}
First(**(**) := {**(**}
First(**)**) := {**)**}
First(**;**) := {**;**}

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

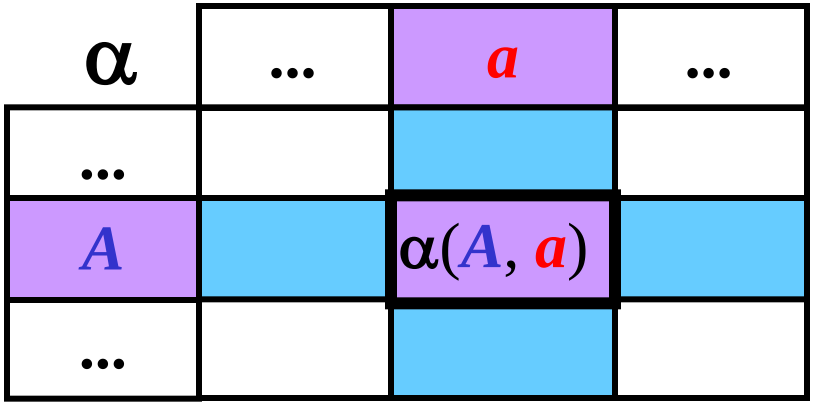
<it-list> → **)** ∈ *P*: **add** *First*(**)**) **to** *First*(**<it-list>**)
<it-list> → **,** … ∈ *P*: **add** *First*(**,**) **to** *First*(**<it-list>**)
Summary: *First*(**<it-list>**) = {**)**, **,**}

<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**}

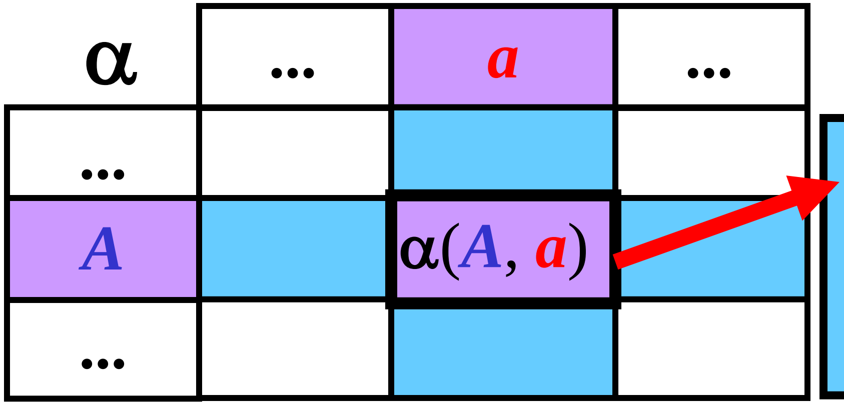
<prog> → **begin** … ∈ *P*: **add** *First*(**begin**) **to** *First*(**<prog>**)
Summary: *First*(**<prog>**) = {**begin**}

Construction of LL Table



α	…	*a*	…
…			
A		α(*A*, *a*)	
…			

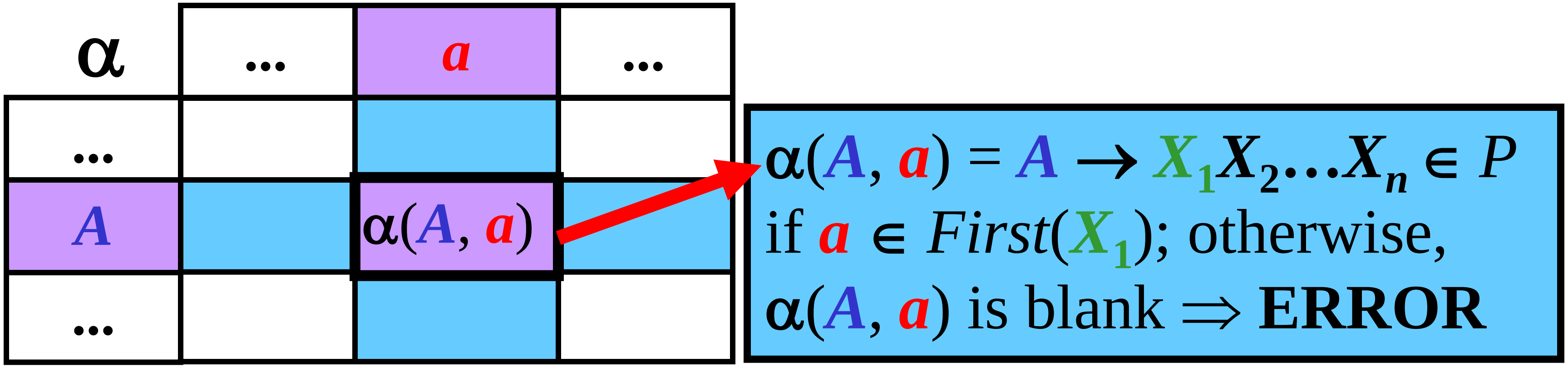
Construction of LL Table



α	…	a	…
…			
A		$\alpha(A, a)$	
…			

$\alpha(A, a) = A \to X_1X_2\ldots X_n \in P$
if $a \in First(X_1)$; otherwise,
$\alpha(A, a)$ is blank $\Rightarrow$ **ERROR**

Construction of LL Table



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

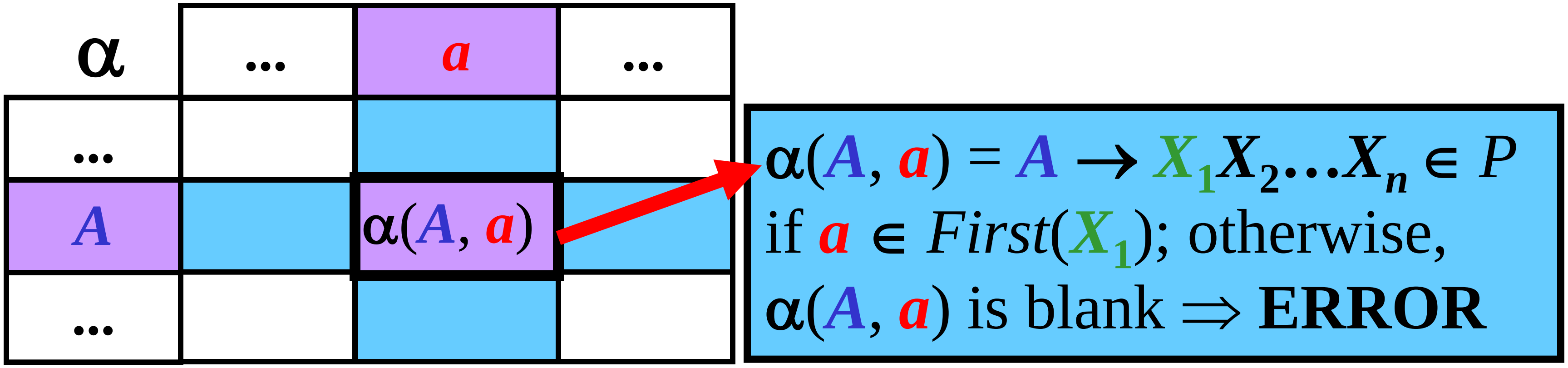
$\alpha(A, a) = A \to X_1X_2\ldots X_n \in P$
if $a \in First(X_1)$; otherwise,
$\alpha(A, a)$ is blank ⇒ **ERROR**

Task: LL table for SPL

	id	**int**	**:=**	**•••**
<prog>				
<st-list>				
<stat>				
<it-list>				
<item>				

Rule *r*: $A \to X_1X_2\ldots X_n$	$First(X_1)$
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**}

Construction of LL Table



$\alpha(A, a) = A \rightarrow X_1X_2\ldots X_n \in P$
if $a \in First(X_1)$; otherwise,
$\alpha(A, a)$ is blank ⇒ **ERROR**

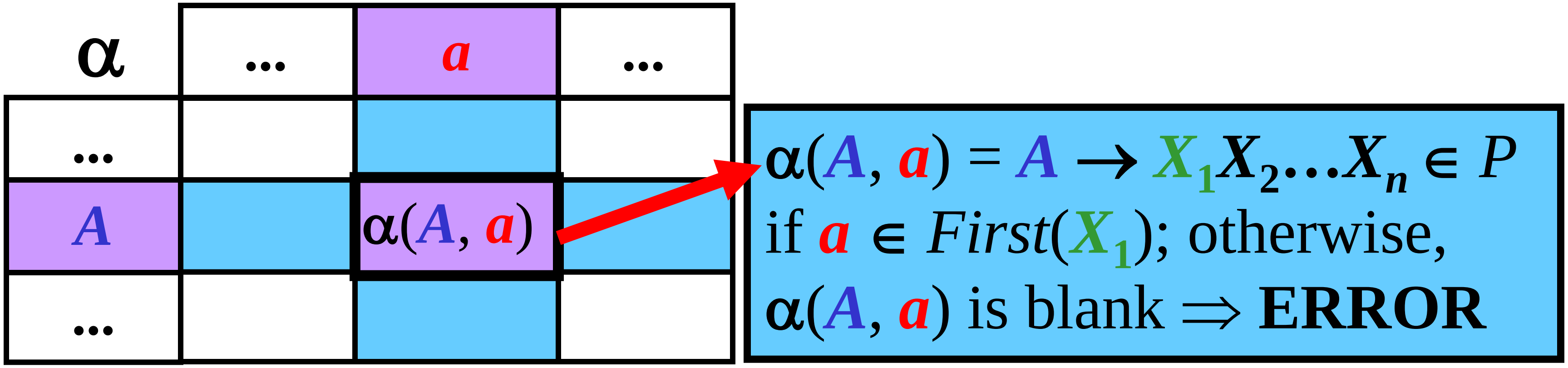
Task: LL table for SPL

	id	int	:=	…
<prog>				
<st-list>	2			
<stat>				
<it-list>				
<item>				

id ∈ *First*(<stat>)

Rule ***r*****:** $A \rightarrow X_1X_2\ldots X_n$	$First(X_1)$
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}

Construction of LL Table



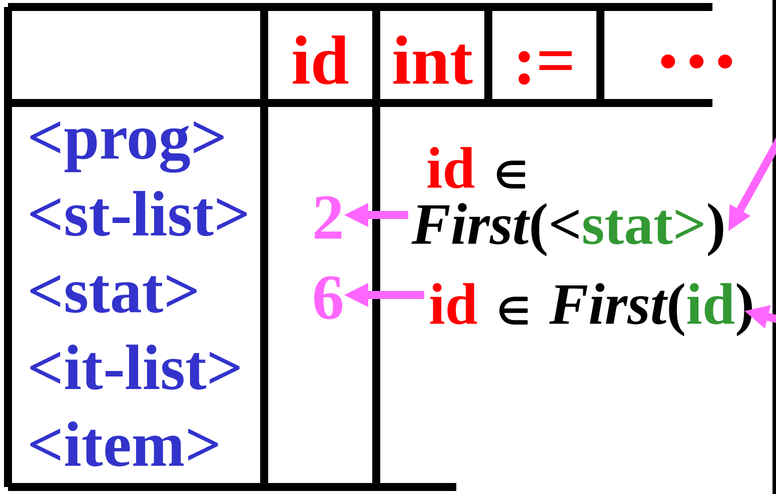
α	…	*a*	…
…			
A		α(*A*, *a*)	
…			

$\alpha(A, a) = A \rightarrow X_1X_2\ldots X_n \in P$

if $a \in First(X_1)$; otherwise,

$\alpha(A, a)$ is blank ⇒ **ERROR**

Task: LL table for SPL



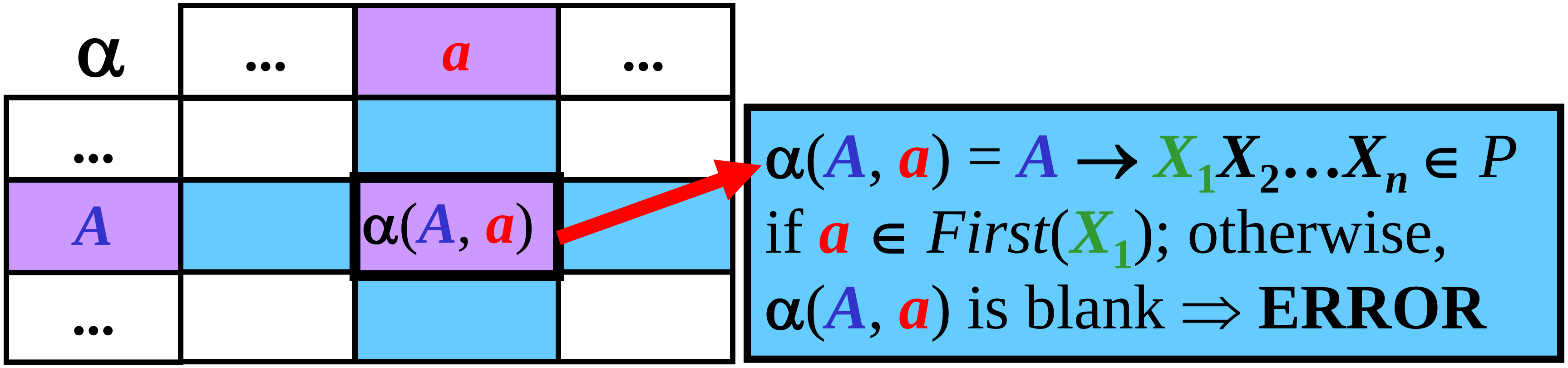
	id	int	:=	…
<prog>				
<st-list>	2			
<stat>	6			
<it-list>				
<item>				

id ∈ *First*(<stat>)

id ∈ *First*(id)

Rule *r*: $A \rightarrow X_1X_2\ldots X_n$	*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}

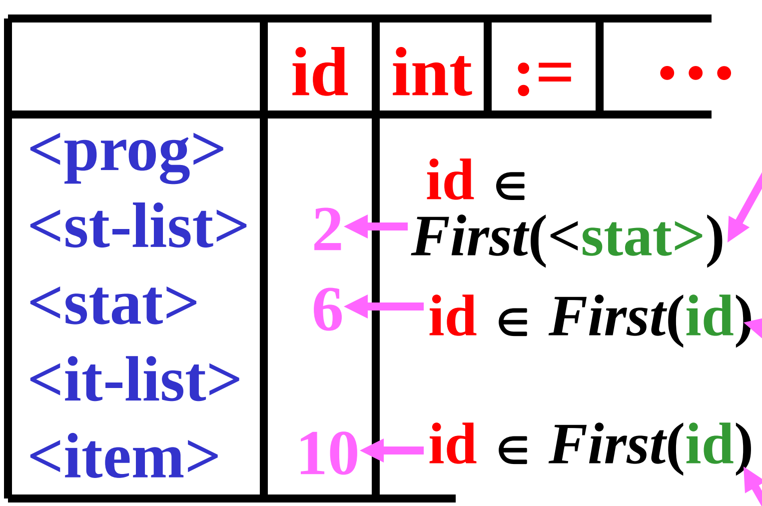
Construction of LL Table



α	…	*a*	…
…			
A		α(*A*, *a*)	
…			

α(*A*, *a*) = $A \to X_1X_2\ldots X_n \in P$ if *a* ∈ *First*(X_1); otherwise, α(*A*, *a*) is blank ⇒ **ERROR**

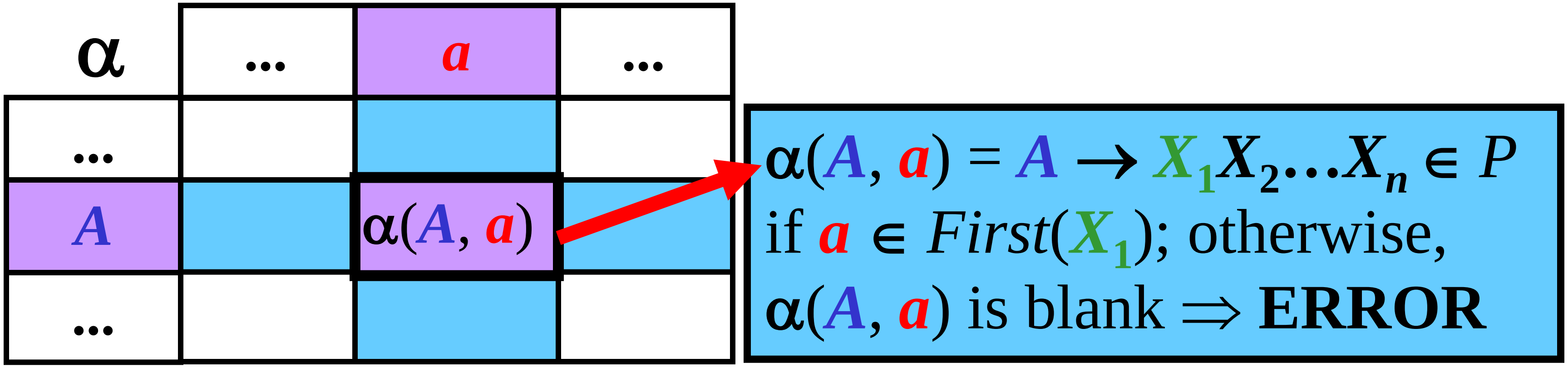
Task: LL table for SPL



	id	int	:=	…
<prog>				
<st-list>	2			
<stat>	6			
<it-list>				
<item>	10			

Rule *r*: $A \to X_1X_2\ldots X_n$	*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}

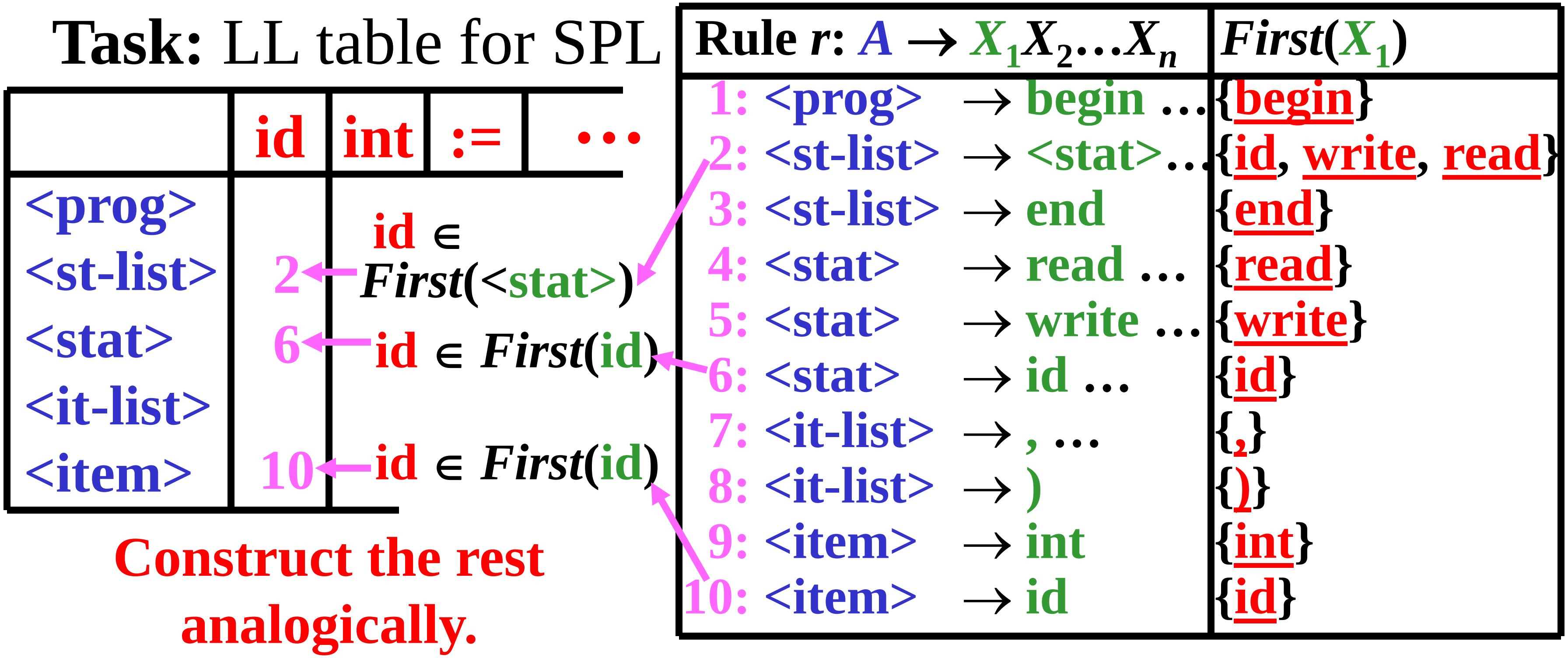
Construction of LL Table



α	…	*a*	…
…			
A		α(*A*, *a*)	
…			

$\alpha(A, a) = A \to X_1X_2\ldots X_n \in P$
if $a \in First(X_1)$; otherwise,
$\alpha(A, a)$ is blank ⇒ **ERROR**

Task: LL table for SPL



	id	int	:=	…
<prog>				
<st-list>	2			
<stat>	6			
<it-list>				
<item>	10			

id ∈ *First*(<stat>)

id ∈ *First*(id)

id ∈ *First*(id)

Rule *r*: $A \to X_1X_2\ldots X_n$	*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}

Construct the rest analogically.

Parsing Based on LL Table: Example

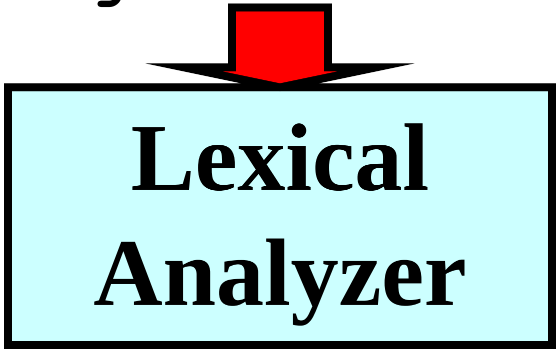
1: <prog> → begin <st-list>
2: <st-list> → <stat> ; <st-list>
3: <st-list> → end
4: <stat> → read id
5: <stat> → write <item>
6: <stat> → id := add (…
7: <it-list> → , <item> <it-list>
8: <it-list> →)
9: <item> → int
10: <item> → id

	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						

<prog>

Source program:

```
begin write 25; end
```



Lexical Analyzer

Parsing Based on LL Table: Example

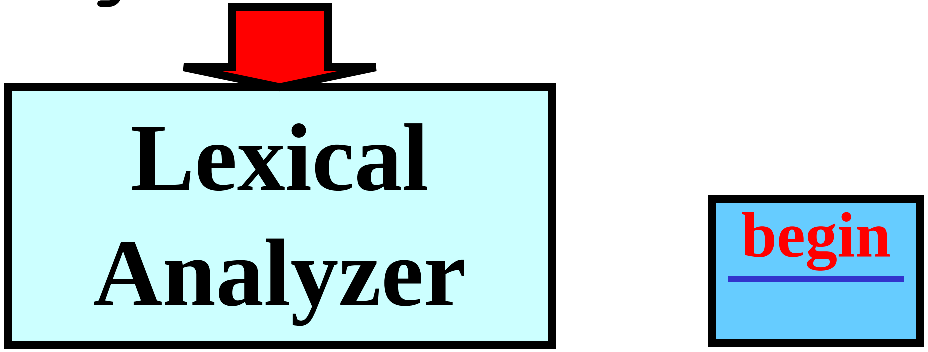
1: <prog> → begin <st-list>
2: <st-list> → <stat> ; <st-list>
3: <st-list> → end
4: <stat> → read id
5: <stat> → write <item>
6: <stat> → id := add (…
7: <it-list> → , <item> <it-list>
8: <it-list> →)
9: <item> → int
10: <item> → id

	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						

<prog>

Source program:

```
begin write 25; end
```



Parsing Based on LL Table: Example

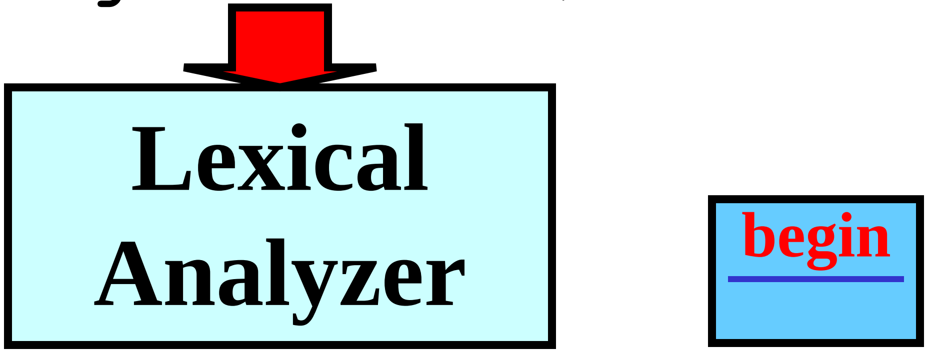
1: <prog> → begin <st-list>
2: <st-list> → <stat> ; <st-list>
3: <st-list> → end
4: <stat> → read id
5: <stat> → write <item>
6: <stat> → id := add (…
7: <it-list> → , <item> <it-list>
8: <it-list> →)
9: <item> → int
10: <item> → id

	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						

<prog>

Source program:

```
begin write 25; end
```



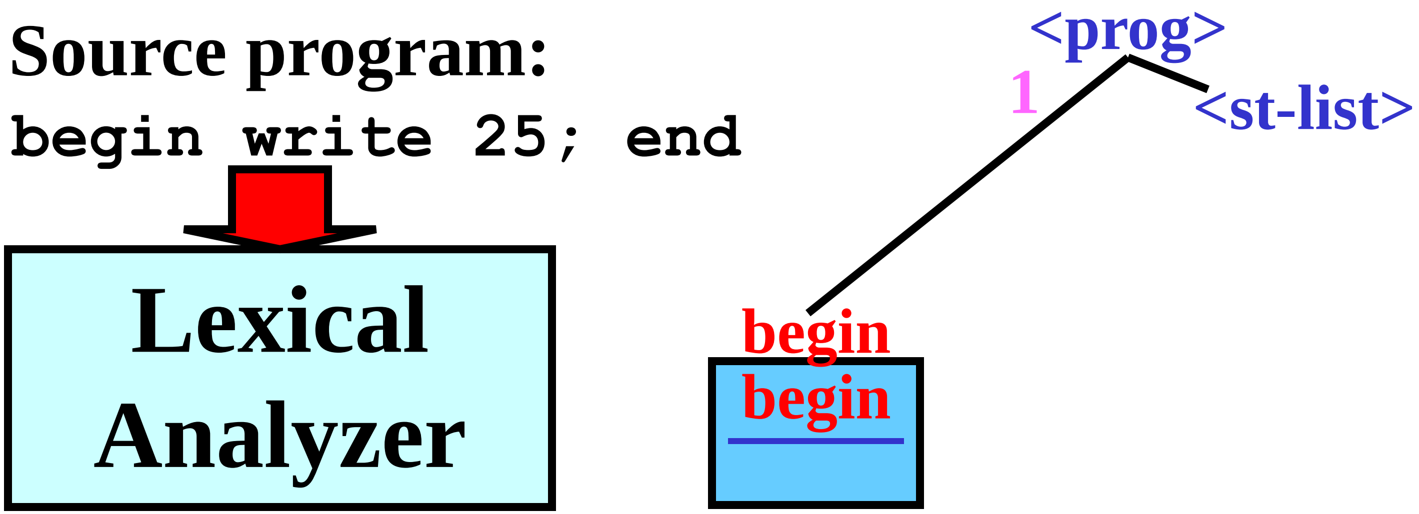
Lexical Analyzer

begin

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`

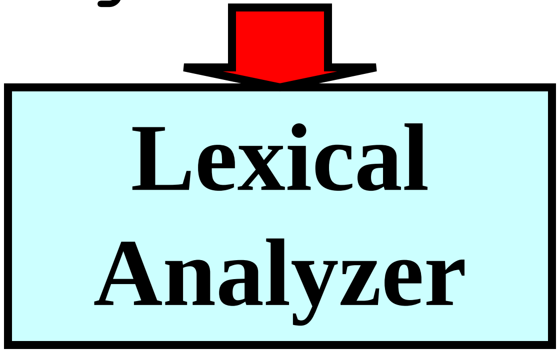
Parsing Based on LL Table: Example

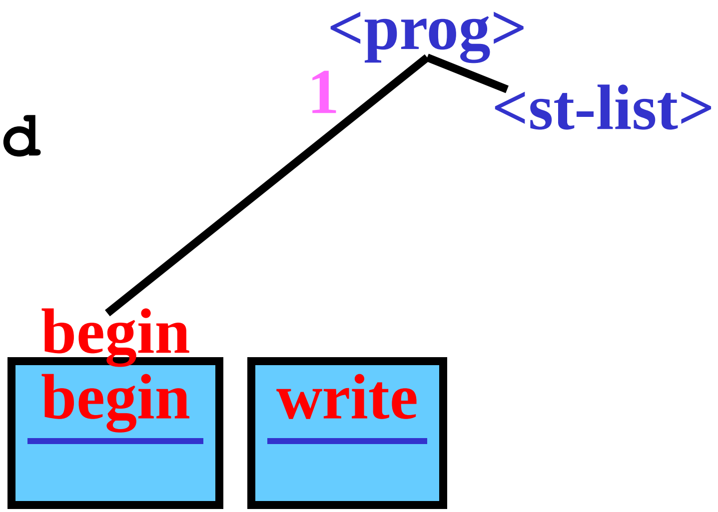
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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`





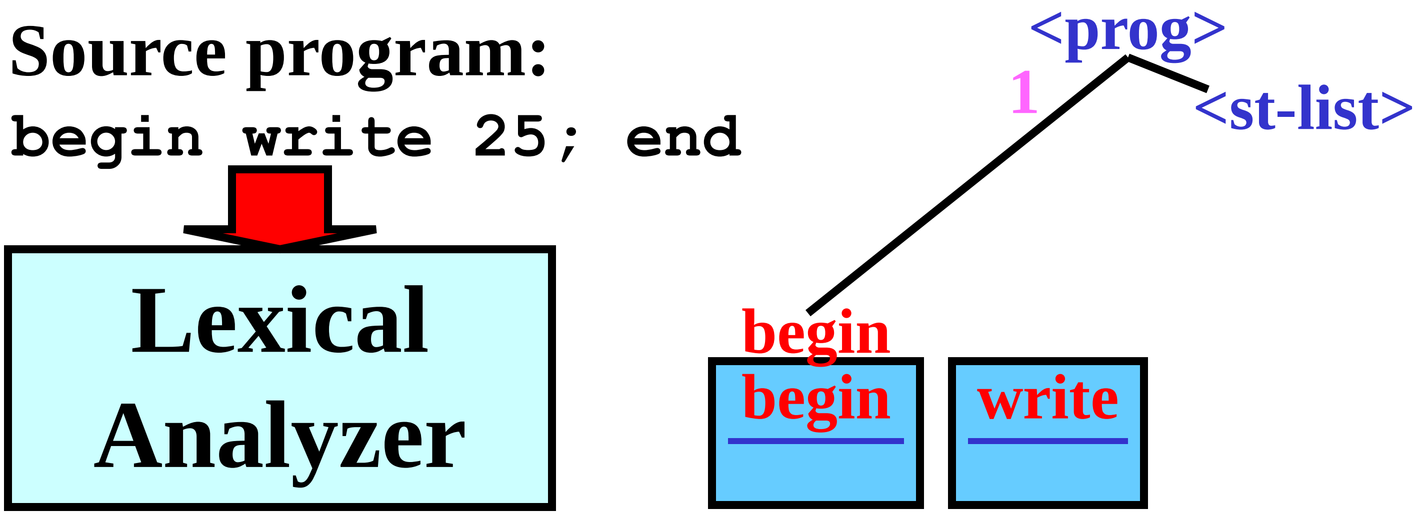
Parsing Based on LL Table: Example

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`



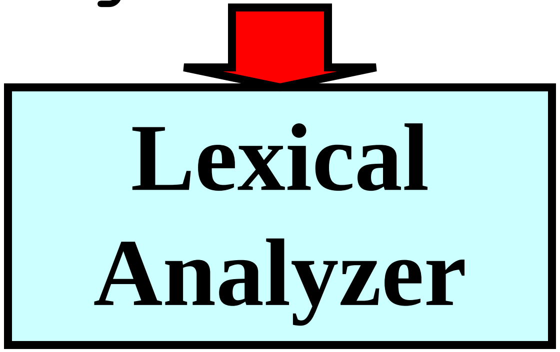
Parsing Based on LL Table: Example

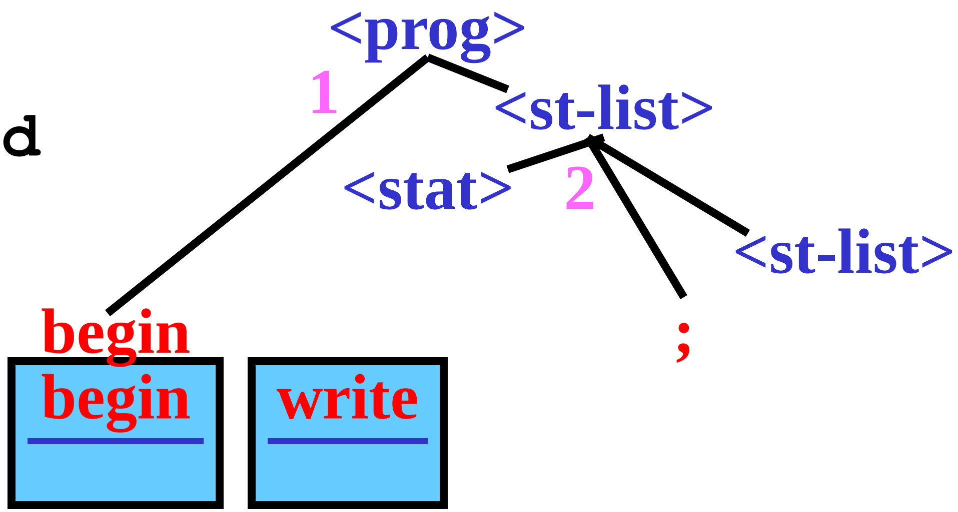
1: <prog> → begin <st-list>
2: <st-list> → <stat> ; <st-list>
3: <st-list> → end
4: <stat> → read id
5: <stat> → write <item>
6: <stat> → id := add (…
7: <it-list> → , <item> <it-list>
8: <it-list> →)
9: <item> → int
10: <item> → id

	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`





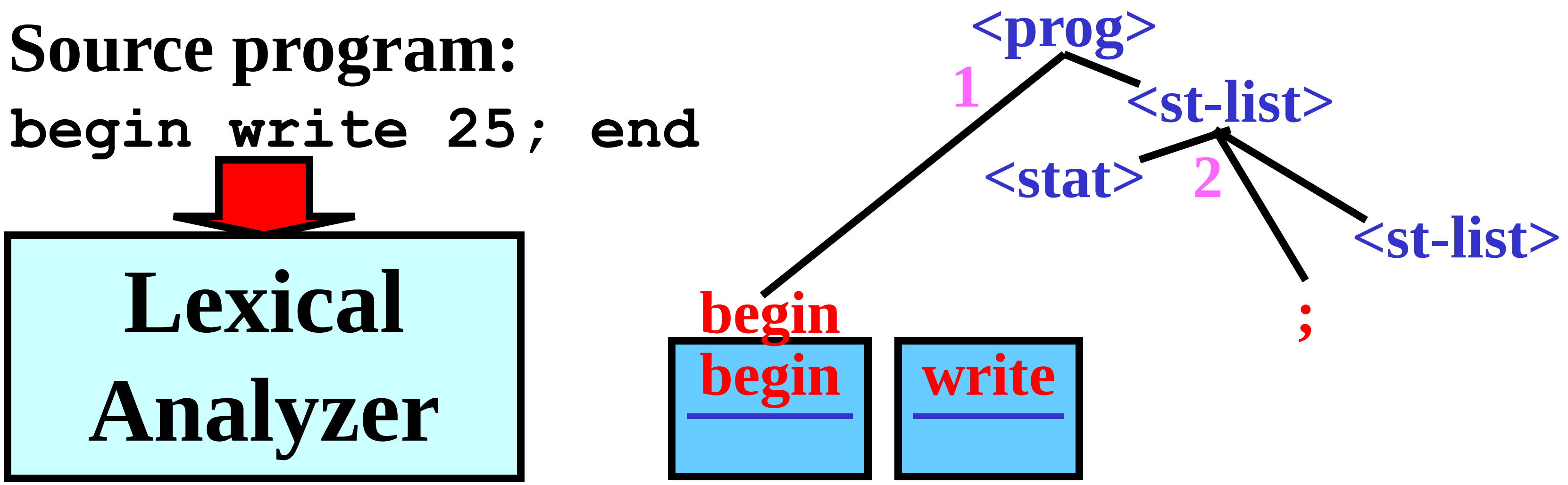
Parsing Based on LL Table: Example

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
```



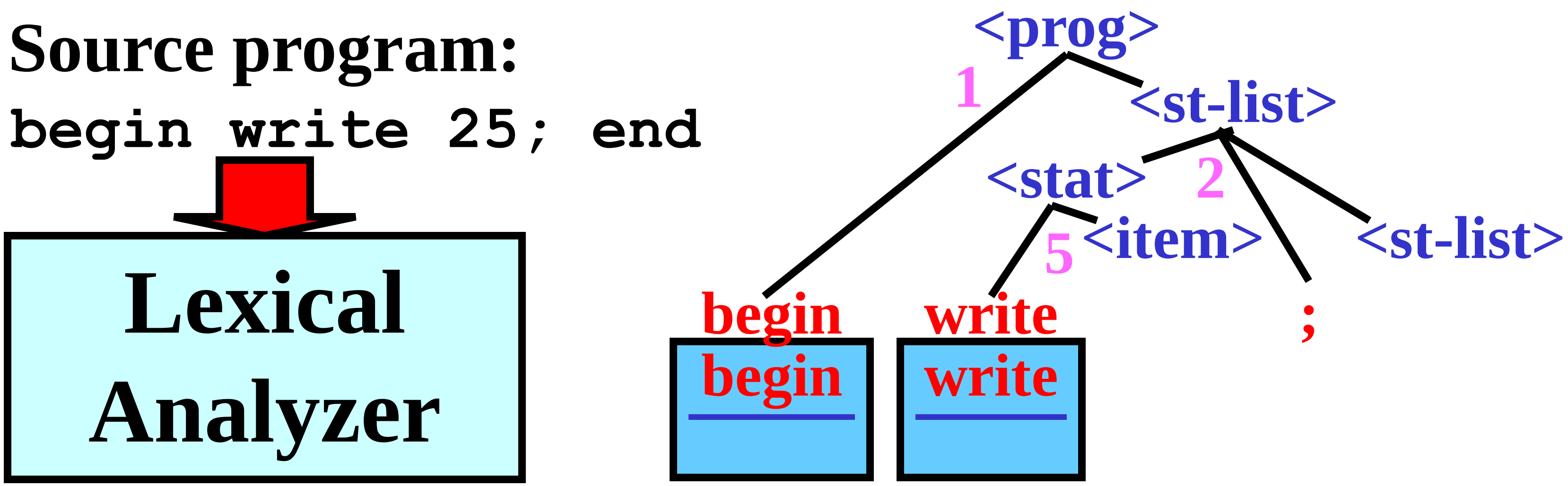
Parsing Based on LL Table: Example

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
```



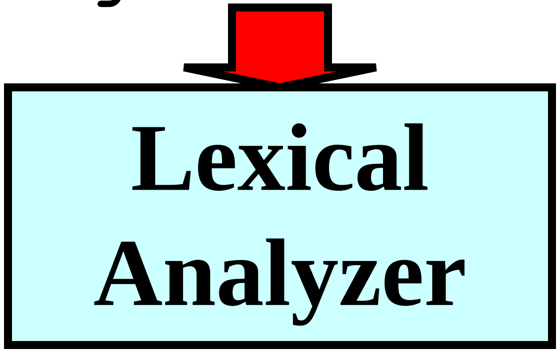
Parsing Based on LL Table: Example

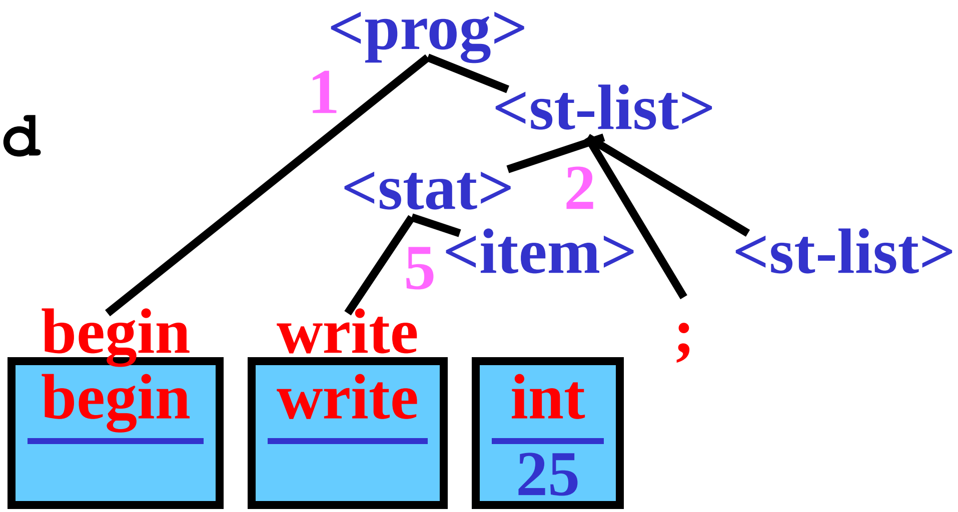
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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`





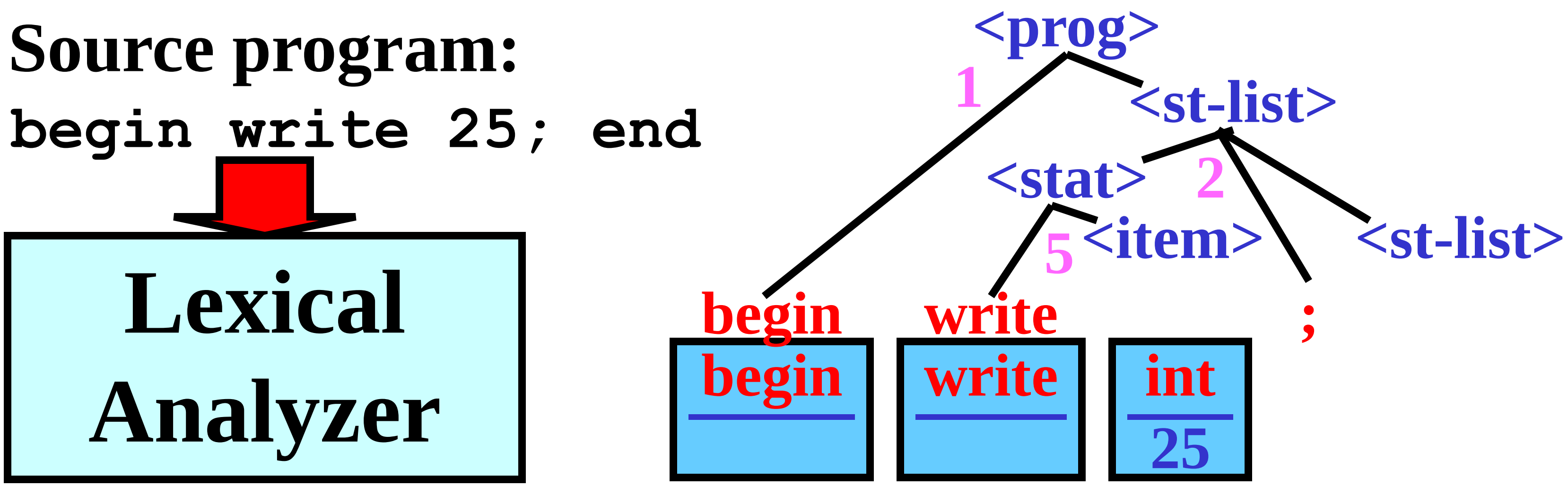
Parsing Based on LL Table: Example

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
```



Parsing Based on LL Table: Example

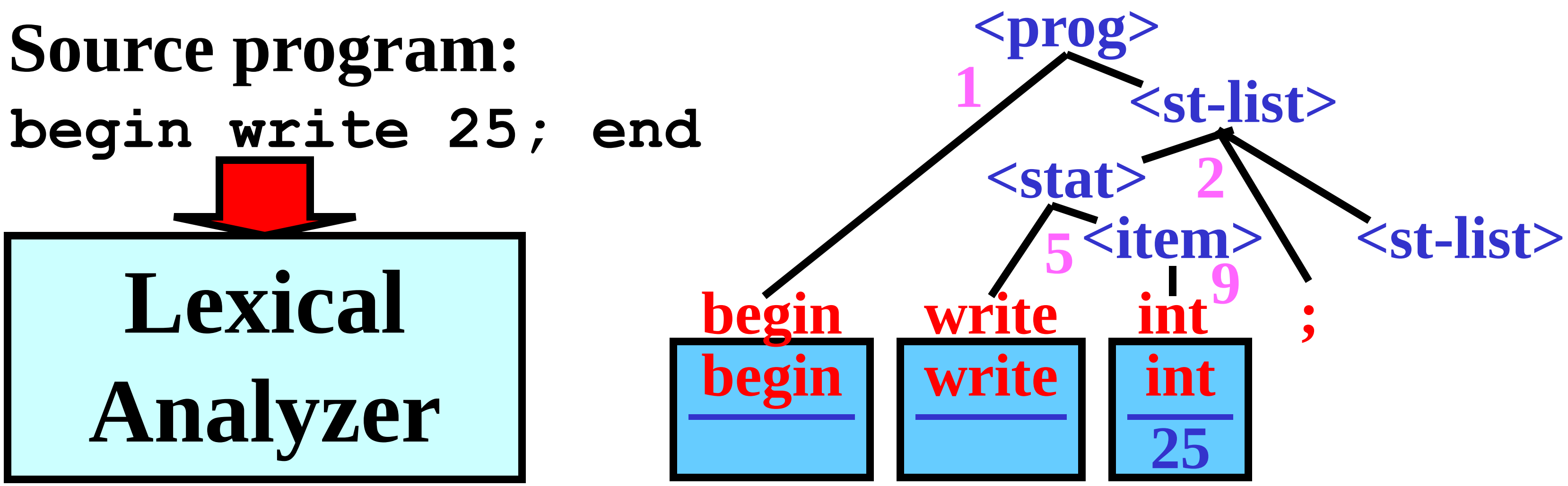
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>

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
```



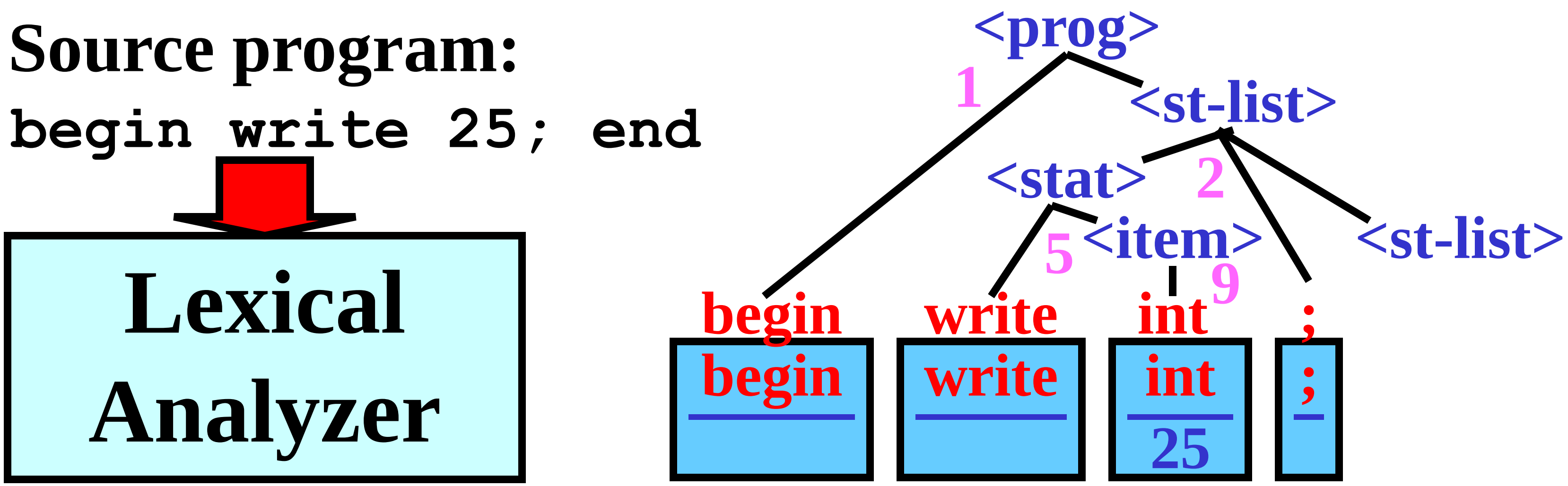
Parsing Based on LL Table: Example

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`



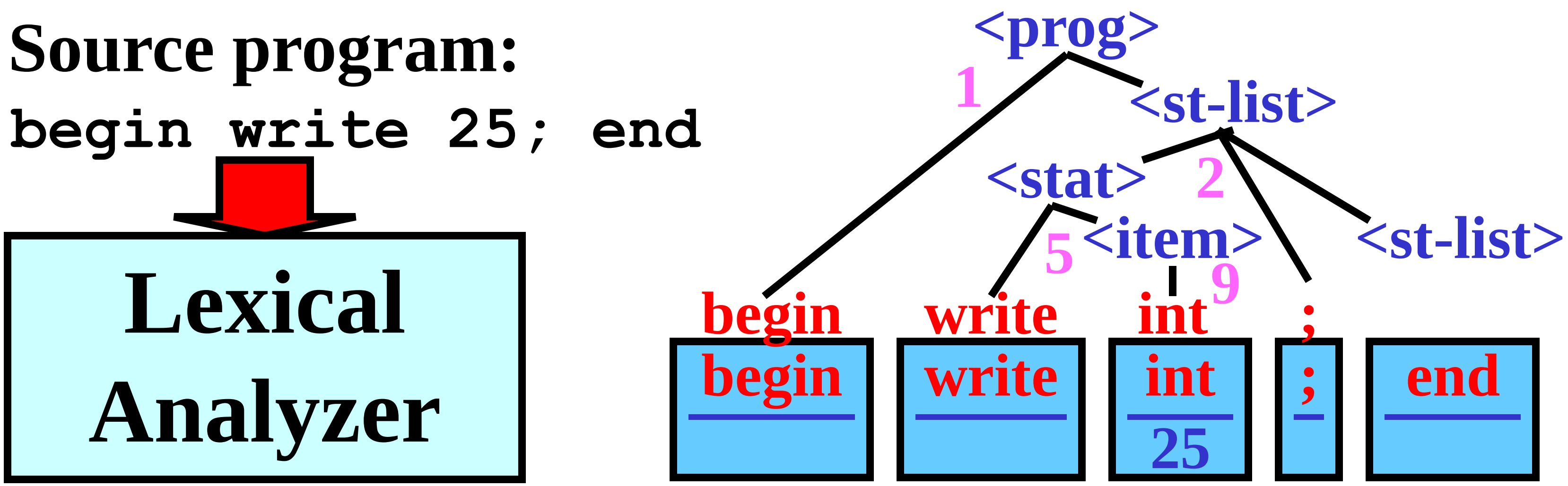
Parsing Based on LL Table: Example

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
```



Parsing Based on LL Table: Example

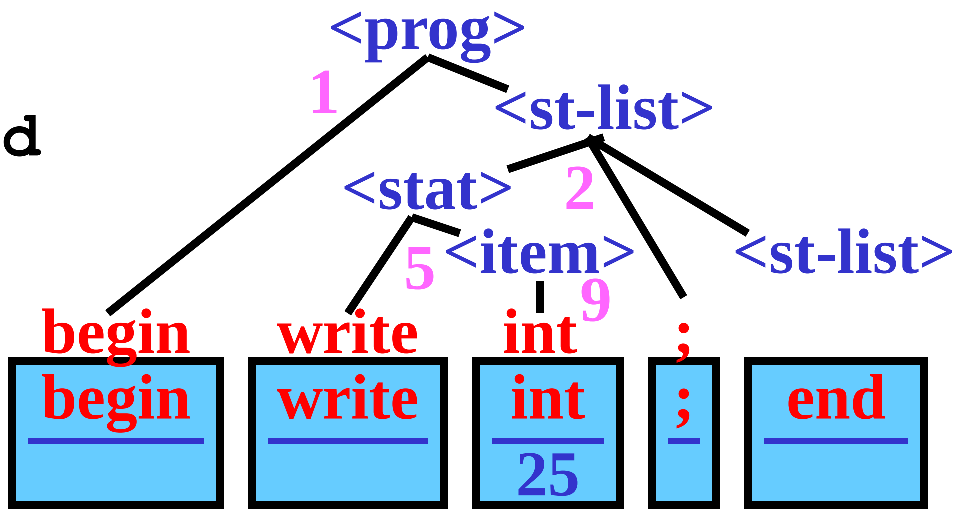
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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



Parsing Based on LL Table: Example

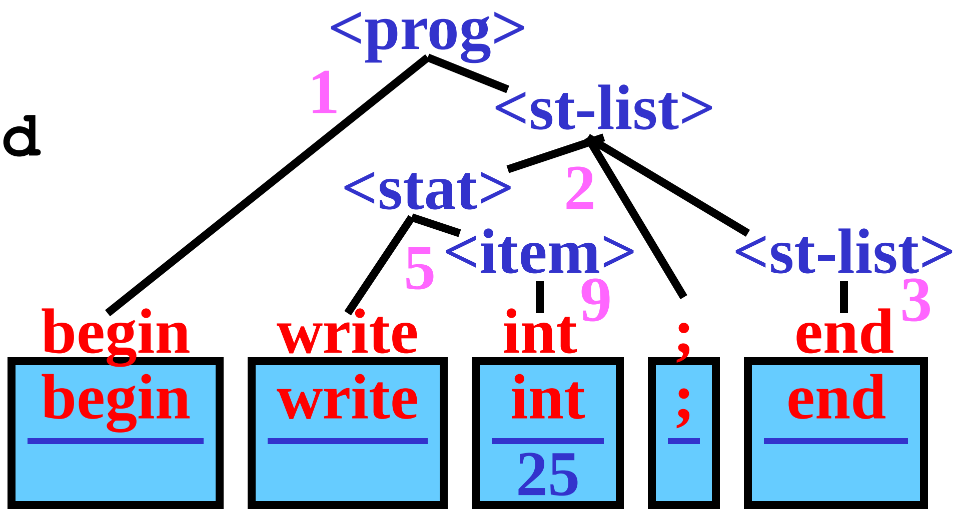
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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`

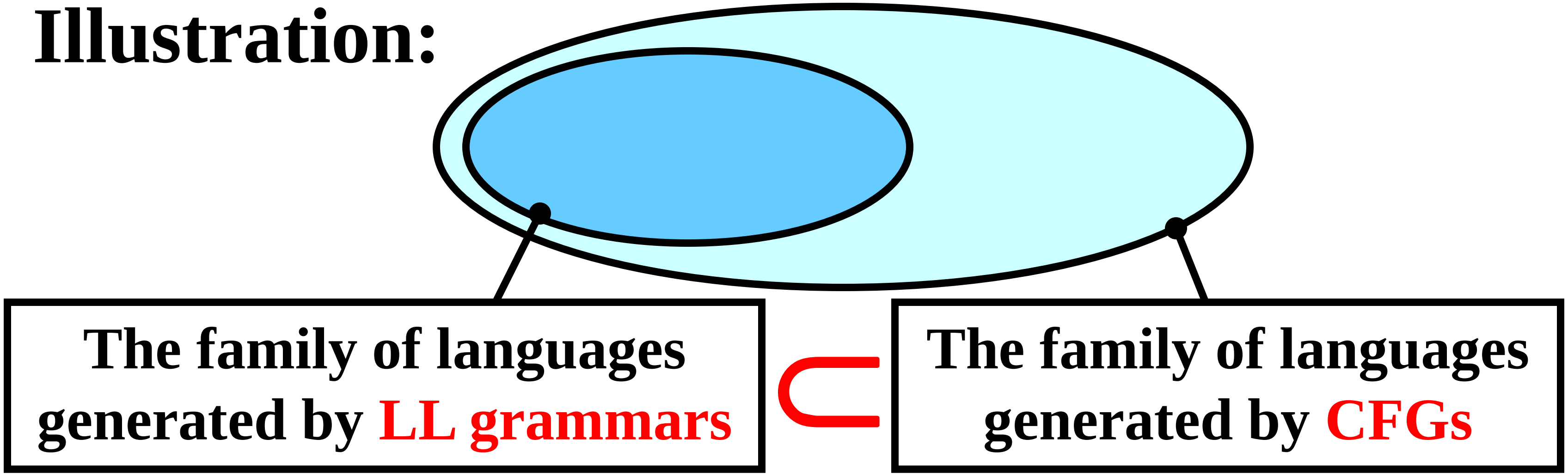
Lexical Analyzer



LL Grammars: Useful Transformations

Generaly: CFG are stronger that LL grammars

Illustration:



The family of languages generated by LL grammars $\subset$ **The family of languages generated by CFGs**

- **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

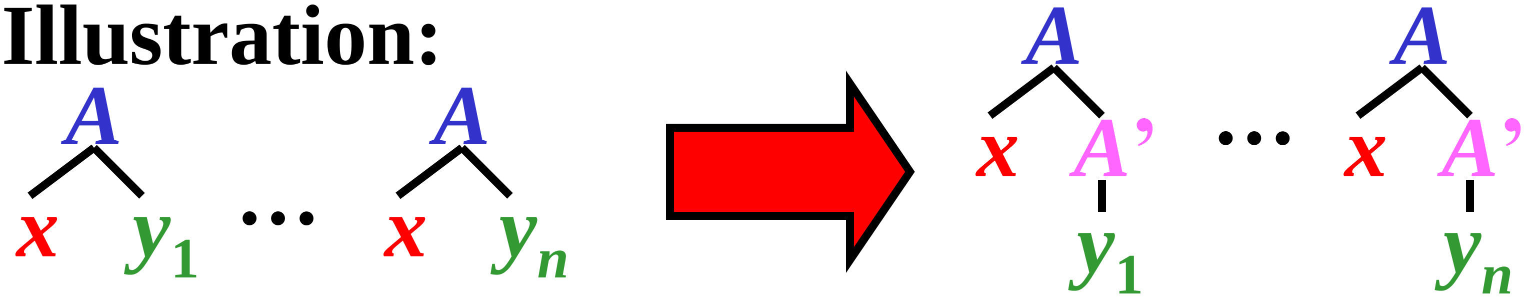
Idea: Replace rules of the form

$$A \rightarrow xy_1,\ A \rightarrow xy_2, \ldots, A \rightarrow xy_n$$ with

$$A \rightarrow xA',\ A' \rightarrow y_1,\ A' \rightarrow y_2, \ldots, A' \rightarrow y_n,$$

where A' is a new nonterminal

Illustration:



Factorization

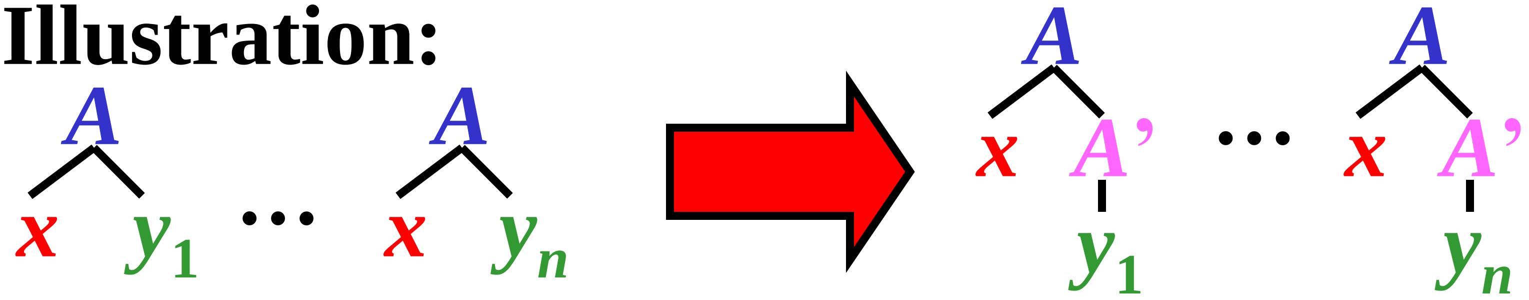
Idea: Replace rules of the form

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

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

where A' is a new nonterminal

Illustration:



Example:

<stat> → **write id**

<stat> → **write int**

Factorization

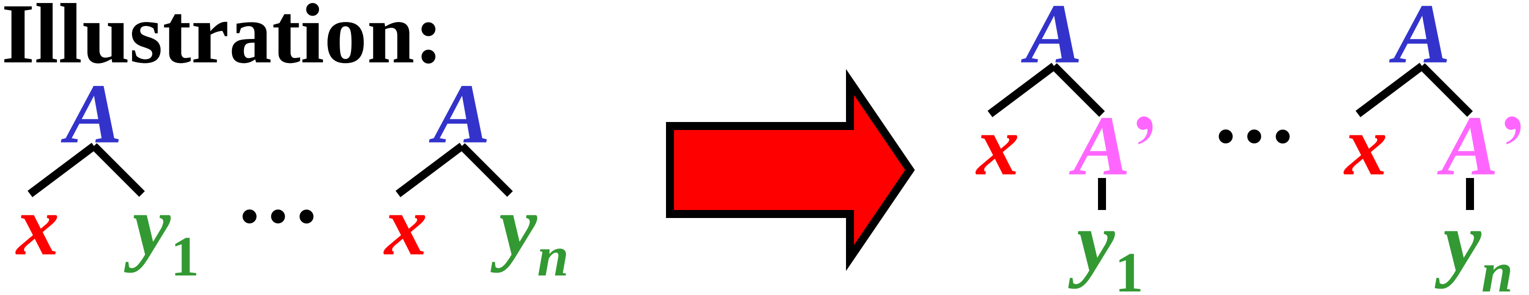
Idea: Replace rules of the form

$$A \to xy_1,\ A \to xy_2,\ \dots,\ A \to xy_n \text{ with}$$

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

where A' is a new nonterminal

Illustration:



Example:

<stat> → write id

<stat> → write int

<stat> → write <item>

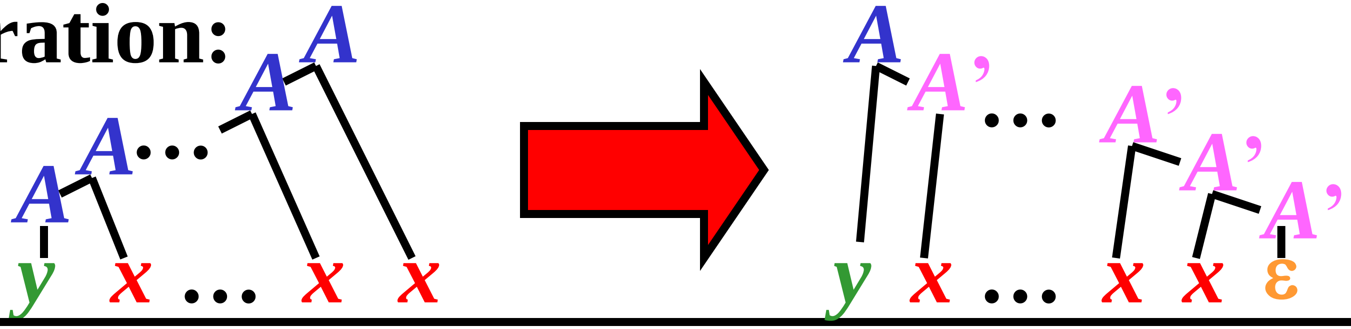
<item> → id

<item> → int

Left Recursion Replacement

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

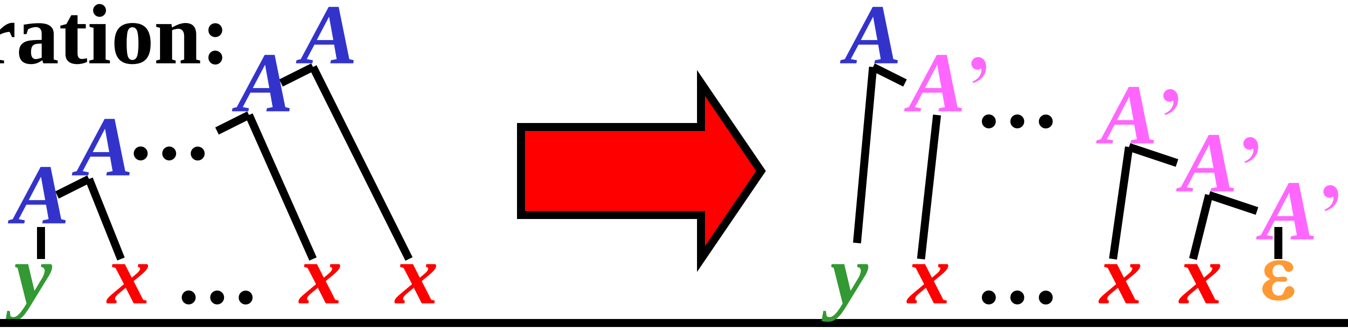
Illustration:



Left Recursion Replacement

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

Illustration:



Example:

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F$

$T \rightarrow F$

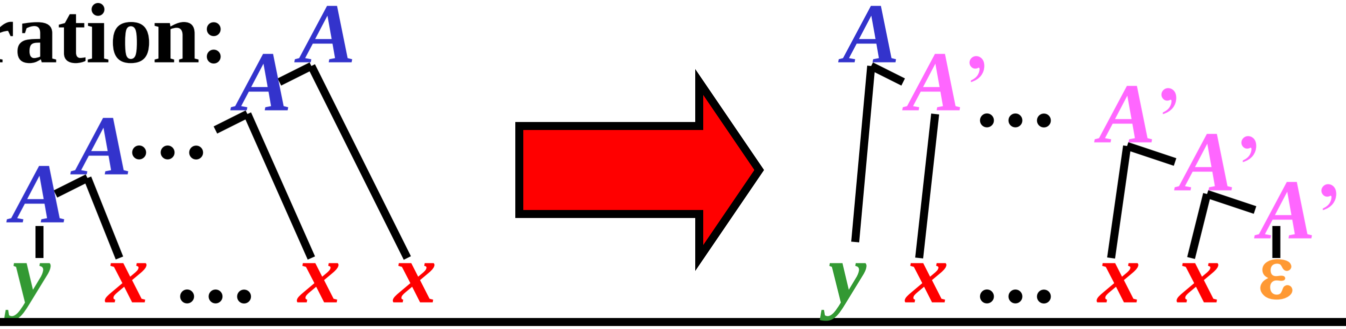
$F \rightarrow (E)$

$F \rightarrow i$

Left Recursion Replacement

Idea: Replace rules of the form *A* → *Ax*, *A* → *y* with *A* → *yA'*, *A'* → *xA'*, *A'* → ε, where *A'* is a new nonterminal.

Illustration:



Example:

E → *E+T*
E → *T*
T → *T*F*
T → *F*
F → (*E*)
F → *i*

E → *TE'*, *E'* → *+TE'*, *E'* → ε

T → *FT'*, *T'* → **FT'*, *T'* → ε

F → (*E*)
F → *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:

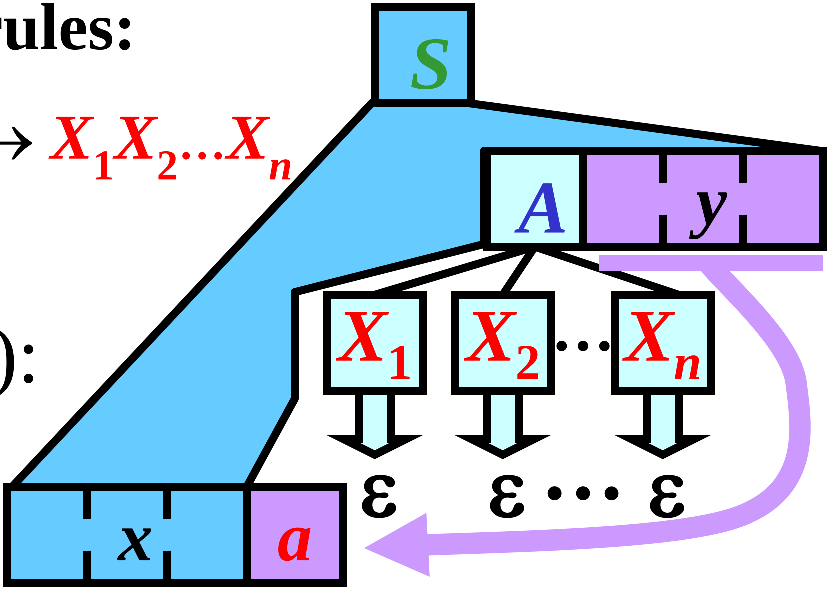
Assume that every input string of tokens ends with $.

Note: $ acts as an *end marker*.

Main problem with ε-rules:

Rule *r*: $A \rightarrow X_1X_2…X_n$

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



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

Grammar for Arithmetical Expressions

- $G_{expr3} = (N, T, P, \mathbf{E})$, where
- $N = \{\mathbf{E}, \mathbf{E'}, \mathbf{T}, \mathbf{T'}, \mathbf{F}\}$,
- $T = \{\mathbf{i}, \mathbf{+}, \mathbf{*}, \mathbf{(}, \mathbf{)}\}$,
- $P = \{$ 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: $\mathbf{F} \rightarrow \mathbf{i}$ $\}$

Example:

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

Set *Empty*

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

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

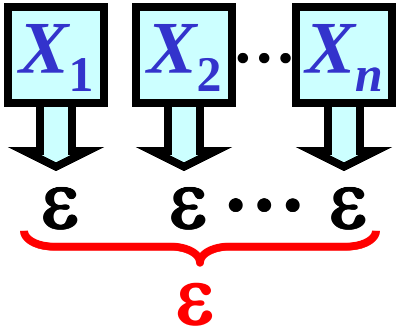
Illustration: $\mathbf{x} = \boxed{X_1}\,\boxed{X_2}\cdots\boxed{X_n}$

Set *Empty*

Gist: ***Empty*(*x*) is the set that includes 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,
$Empty(x) = \emptyset$, where $x \in (N \cup T)^*$.

Illustration: x =

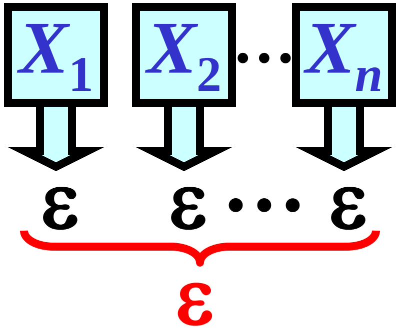


Set *Empty*

Gist: ***Empty*(*x*) is the set that includes 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,
$Empty(x) = \emptyset$, where $x \in (N \cup T)^*$.

Illustration: $x =$



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

⇓

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

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 $A \rightarrow \varepsilon \in P$ then $Empty(A) := \{\varepsilon\}$
                                 else $Empty(A) := \varnothing$
```

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

Previous Algorithm: Illustration

1) for each $\boldsymbol{a} \in T$: *Empty*($\boldsymbol{a}$) := $\varnothing$ because $\boldsymbol{a} \not\Rightarrow^* \boldsymbol{\varepsilon}$

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

---

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

Previous Algorithm: Illustration

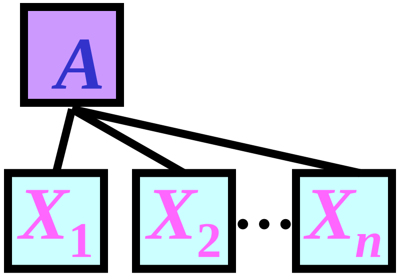
1) for each $\boldsymbol{a} \in T$: $Empty(\boldsymbol{a}) := \varnothing$ because $\boldsymbol{a} \not\Rightarrow^* \varepsilon$

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

---

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

- **if** $A \rightarrow X_1X_2 \ldots X_n \in P$ **and** $Empty(X_i) = \{\varepsilon\}$ for all $i = 1,\ldots,n$ **then** $Empty(A) = \{\varepsilon\}$



Previous Algorithm: Illustration

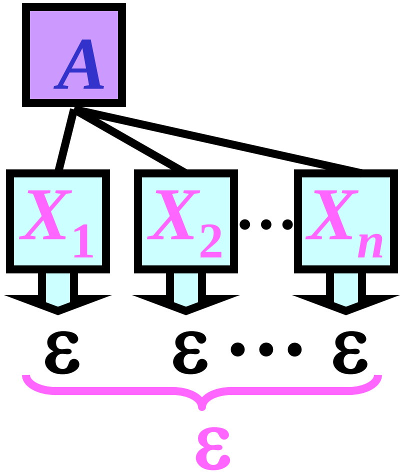
1) for each ***a*** $\in$ *T*: *Empty*(***a***) := $\varnothing$ because ***a*** $\not\Rightarrow^*$ **ε**

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

---

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

- **if** ***A*** $\rightarrow$ $X_1X_2 \dots X_n$ $\in P$ **and** *Empty*(X_i) = {ε} for all *i* = 1,…,*n* **then** *Empty*(***A***) = {ε}



Previous Algorithm: Illustration

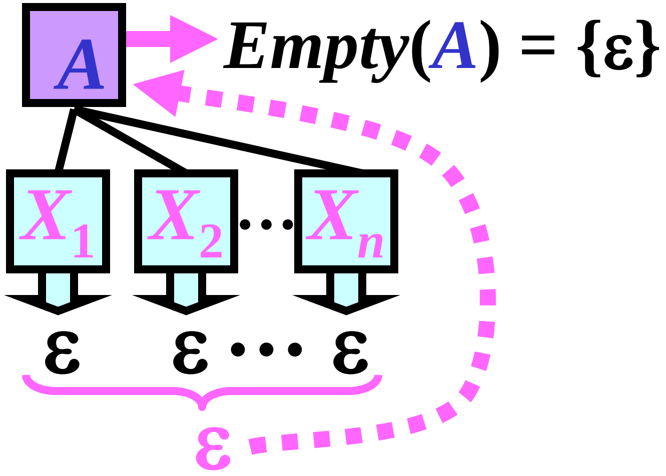
1) for each ***a*** $\in T$: *Empty*(***a***) := $\varnothing$ because ***a*** $\not\Rightarrow^*$ **ε**

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

---

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

- **if** $A \rightarrow X_1X_2 \dots X_n \in P$ **and** *Empty*(X_i) = {ε} for all $i = 1,\dots,n$ **then** *Empty*(***A***) = {ε}



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

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

$P = \{$ 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: $F \rightarrow i$ $\}$

Initialization:

Empty(i) := $\emptyset$
Empty($+$) := $\emptyset$
Empty($*$) := $\emptyset$
Empty($($) := $\emptyset$
Empty($)$) := $\emptyset$

Empty(E) := $\emptyset$
Empty(E') := $\{\varepsilon\}$
Empty(T) := $\emptyset$
Empty(T') := $\{\varepsilon\}$
Empty(F) := $\emptyset$

- **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** $A \rightarrow X_1X_2\ldots X_{k-1}X_k\ldots X_n \in P$ **then**
    - add all symbols from $First(X_1)$ to $First(A)$
    - **if** $Empty(X_i) = \{\varepsilon\}$ for all $i = 1,\ldots, k$-1, where $k \leq n$ **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*** $\Rightarrow^{\mathbf{0}}$ ***a***

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

---

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

- **if** $A \rightarrow X_1X_2 \ldots X_{k-1}X_k \ldots X_n \in P$ **then**

Previous Algorithm: Illustration

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

2) for each $\boldsymbol{A} \in N$: *First*($\boldsymbol{A}$) := ∅ (inicialization)

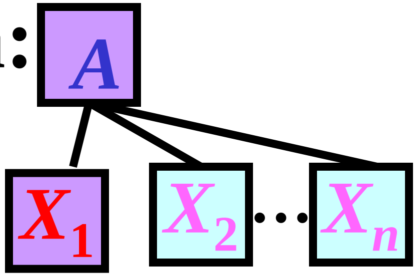
---

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

- **if** $A \rightarrow X_1X_2 \ldots X_{k-1}X_k \ldots X_n \in P$ **then**

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

3a:



Previous Algorithm: Illustration

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

2) for each ***A*** ∈ *N*: *First*(***A***) := ∅ (inicialization)

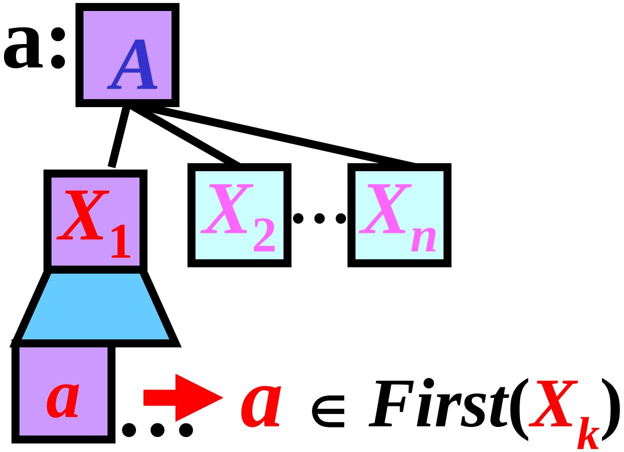
---

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

- **if** $A \rightarrow X_1X_2 \dots X_{k-1}X_k \dots X_n \in P$ **then**

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

3a:



Previous Algorithm: Illustration

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

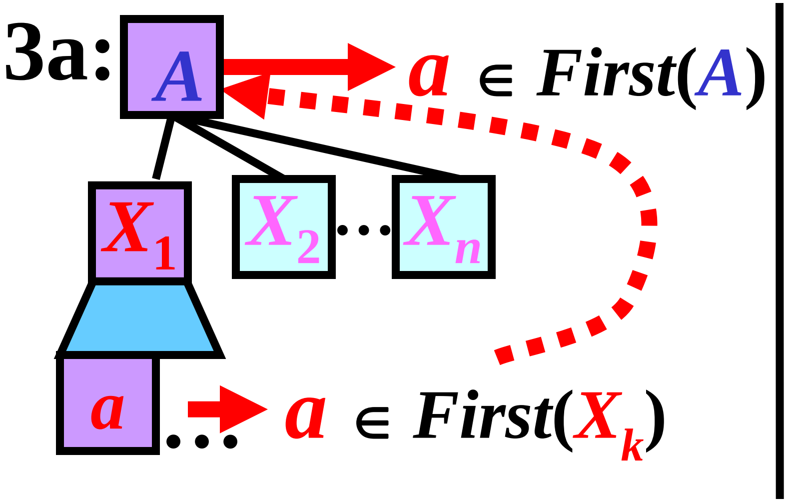
2) for each $A \in N$: *First*(A) := ∅ (inicialization)

---

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

- **if** $A \rightarrow X_1X_2 \dots X_{k-1}X_k \dots X_n \in P$ **then**

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



Previous Algorithm: Illustration

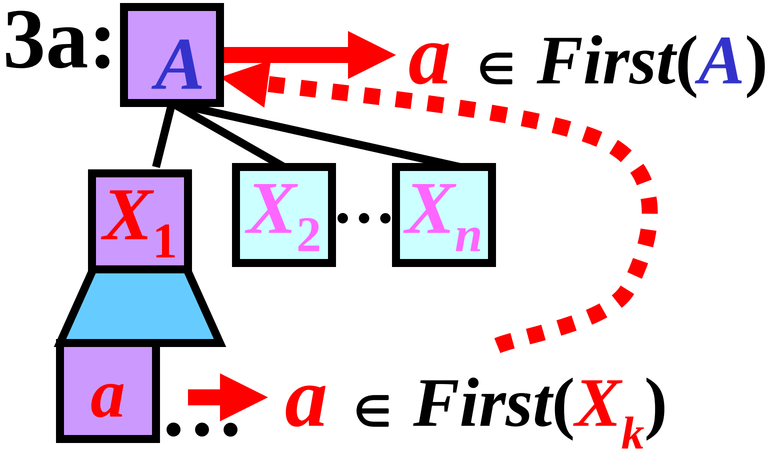
1) for each $a \in T$: $First(a) := \{a\}$ because $a \Rightarrow^0 a$

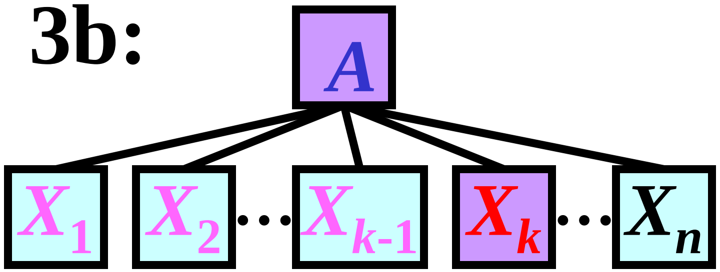
2) for each $A \in N$: $First(A) := \emptyset$ (inicialization)

---

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

- **if** $A \rightarrow X_1X_2 \dots X_{k-1}X_k \dots X_n \in P$ **then**
  - **3a)** add all symbols from $First(X_1)$ to $First(A)$
  - **3b) if** $Empty(X_i) = \{\varepsilon\}$ for all $i = 1, \dots, k\text{-}1$, where $k < n$ **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)

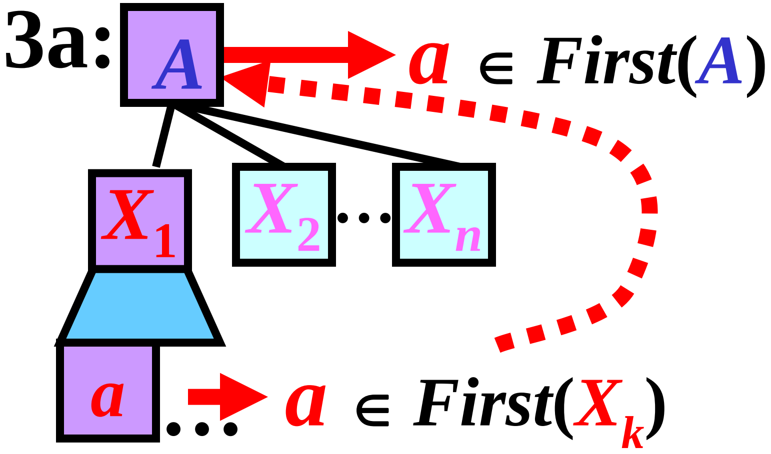
---

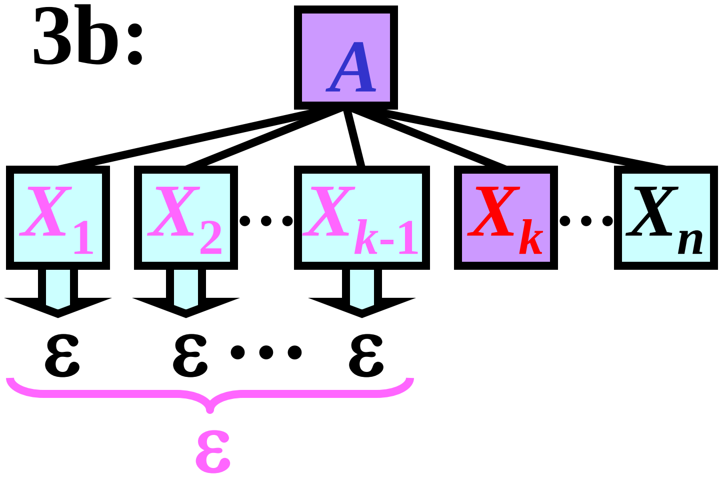
3) Apply the following rules until no *First* set or *Empty* set can be changed:

- **if** $A \rightarrow X_1X_2 \dots X_{k-1}X_k \dots X_n \in P$ **then**

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

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





Previous Algorithm: Illustration

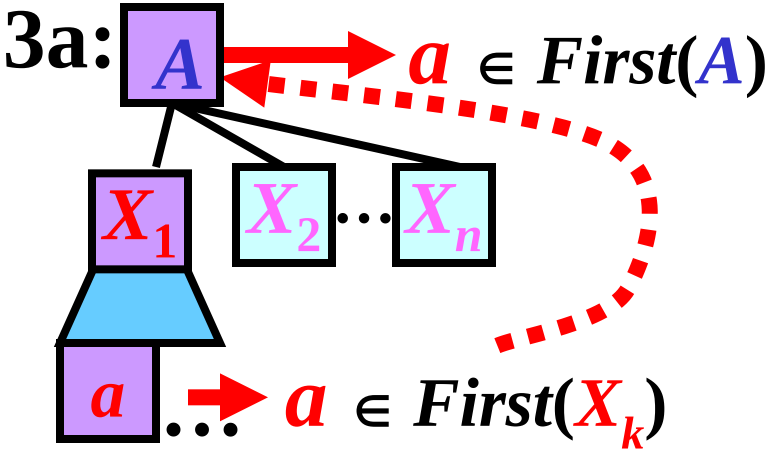
1) for each ***a*** $\in T$: *First*(***a***) := {***a***} because $\boldsymbol{a} \Rightarrow^{\mathbf{0}} \boldsymbol{a}$

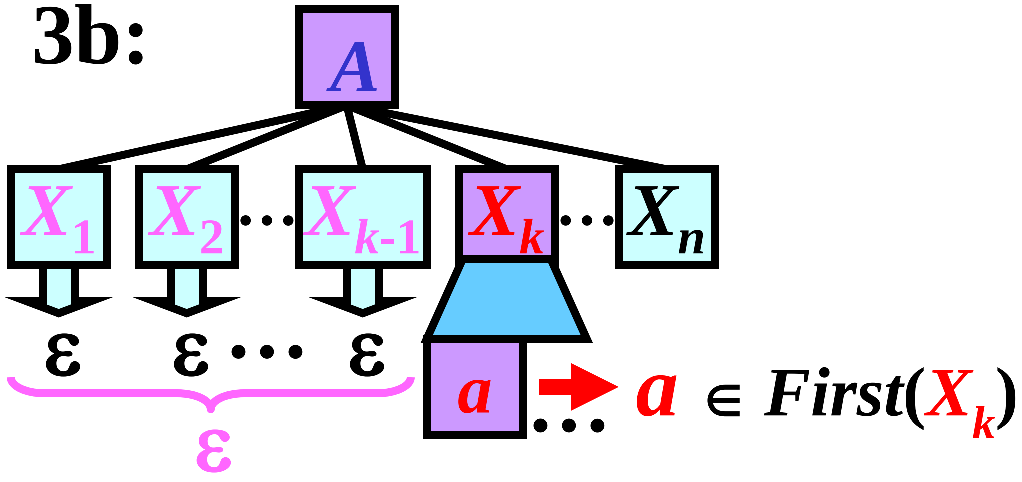
2) for each ***A*** $\in N$: *First*(***A***) := ∅ (inicialization)

---

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

- **if** $A \rightarrow X_1X_2 \dots X_{k-1}X_k \dots X_n \in P$ **then**
  - **3a)** add all symbols from *First*(X_1) to *First*(***A***)
  - **3b) if** *Empty*(X_i) = {ε} for all $i = 1, \dots, k\text{-}1$, where $k < n$ **then** add all symbols from *First*(X_k) to *First*(***A***):





Previous Algorithm: Illustration

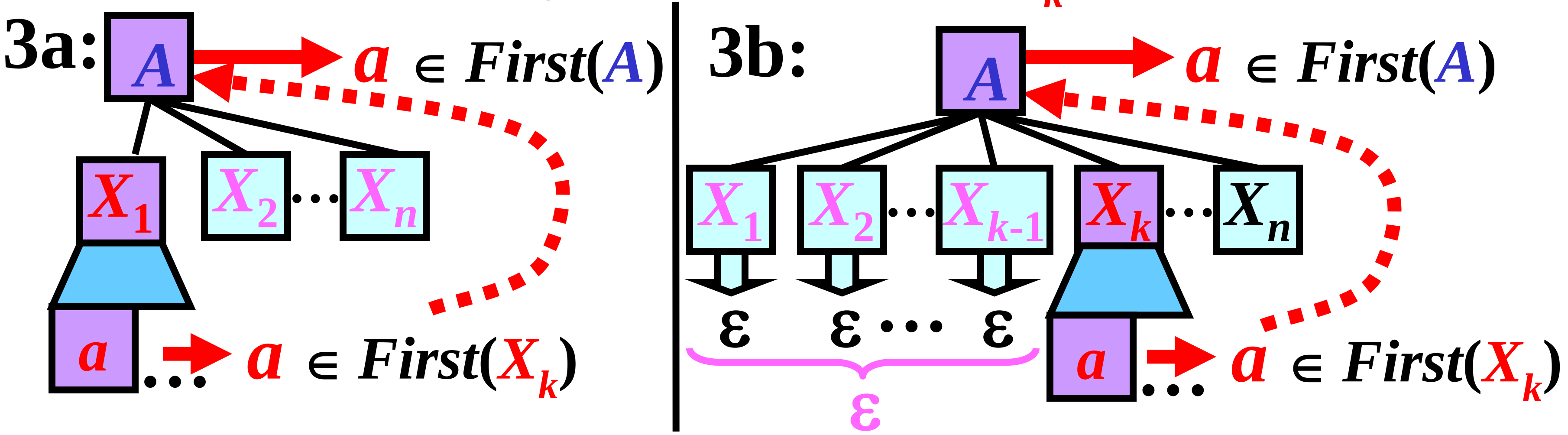
1) for each $a \in T$: *First*(a) := {a} because $a \Rightarrow^0 a$

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

---

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

- **if** $A \rightarrow X_1X_2 \dots X_{k-1}X_k \dots X_n \in P$ **then**
  - **3a)** add all symbols from *First*(X_1) to *First*(A)
  - **3b) if** *Empty*(X_i) = {ε} for all $i = 1, \dots, k$-1, where $k < n$ **then** add all symbols from *First*(X_k) to *First*(A):



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

Initialization:

$First(\mathbf{i}) := \{\mathbf{i}\}$
$First(+) := \{+\}$
$First(*) := \{*\}$
$First(\,(\,) := \{\,(\,\}$
$First(\,)\,) := \{\,)\,\}$

$First(E) := \varnothing$
$First(E') := \varnothing$
$First(T) := \varnothing$
$First(T') := \varnothing$
$First(F) := \varnothing$

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*, (}

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

Initialization:

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

$F \to i \in P$: **add** *First*(**i**) = {**i**} **to** *First*(**F**)

$F \to (E) \in P$: **add** *First*(**(**) = {**(**} **to** *First*(**F**)

Summary: *First*(*F*) = {*i*, (}

$T' \to *FT' \in P$: **add** *First* (*****) = {*****} **to** *First*(**T’**)

Summary: *First*(*T’*) = {*}

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

Initialization:

First(**i**) := {**i**}
First(+) := {+}
First(*) := {*}
First(( ) := { (}
First( )) := {) }

First(***E***) := ∅
First(***E’***) := ∅
First(***T***) := ∅
First(***T’***) := ∅
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’ → ***FT’** ∈ *P*: **add** *First* (*****) = {*****} **to** *First*(***T’***)
Summary: *First*(*T’*) = {*}

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

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

Initialization:

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

$F \rightarrow i \in P$: **add** *First*(*i*) = {*i*} **to** *First*(*F*)

$F \rightarrow (E) \in P$: **add** *First*(() = {(} **to** *First*(*F*)

Summary: *First*(*F*) = {*i*, (}

$T' \rightarrow *FT' \in P$: **add** *First* (*) = {*} **to** *First*(*T'*)

Summary: *First*(*T'*) = {*}

$T \rightarrow FT' \in P$: **add** *First* (*F*) = {*i*, (} **to** *First*(*T*)

Summary: *First*(*T*) = {*i*, (}

$E' \rightarrow +TE' \in P$: **add** *First* (+) = {+} **to** *First*(*E'*)

Summary: *First*(*E'*) = {+}

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

Initialization:

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

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

E' → +*TE'* ∈ *P*: **add** *First* (+) = {+} **to** *First*(*E'*)
Summary: *First*(*E'*) = {+}

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

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

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

E' → **+*TE'*** ∈ *P*: **add** *First* (+) = {+} **to** *First*(***E'***)
Summary: *First*(*E'*) = {+}

E → ***TE'*** ∈ *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: $E \rightarrow TE'$, 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: $F \rightarrow i$ $\}$

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

$Empty(i) := \emptyset$
$Empty(+) := \emptyset$
$Empty(*) := \emptyset$
$Empty(() := \emptyset$
$Empty()) := \emptyset$

$Empty(E) := \emptyset$
$Empty(E') := \{\varepsilon\}$
$Empty(T) := \emptyset$
$Empty(T') := \{\varepsilon\}$
$Empty(F) := \emptyset$

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

$First(i) := \{i\}$
$First(+) := \{+\}$
$First(*) := \{*\}$
$First(() := \{(\}$
$First()) := \{)\}$

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

Note: for each $a \in T$: $Empty(a) = \emptyset$, $First(a) = \{a\}$

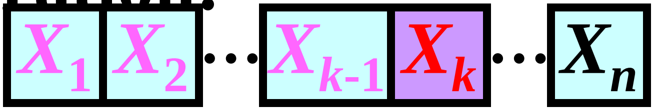
Algorithm: *First*($X_1X_2\dots X_n$)

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

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

! Note: *First*(ε) = ∅

Illustration:



$X_1\ X_2 \dots X_{k-1}\ X_k \dots X_n$

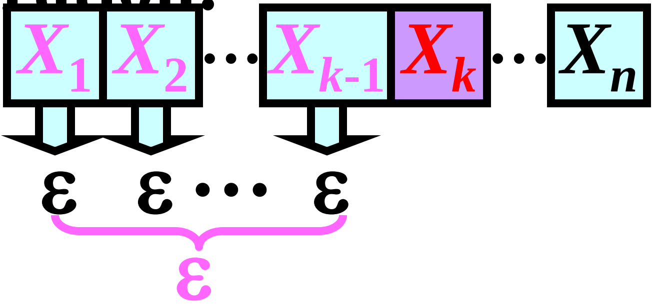
Algorithm: $First(X_1X_2…X_n)$

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

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

! Note: $First(\varepsilon) = \varnothing$

Illustration:



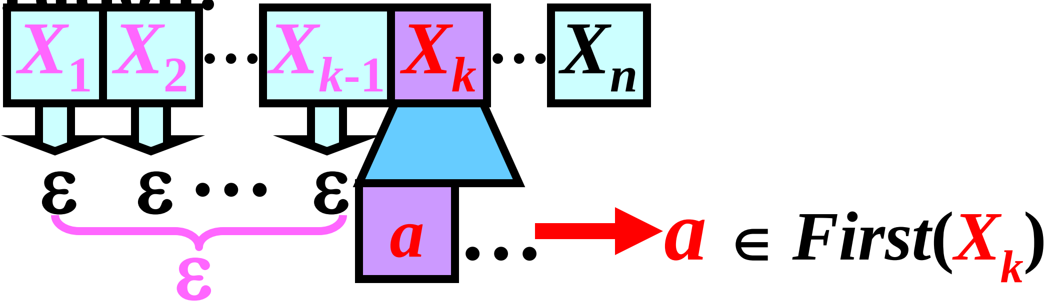
Algorithm: $First(X_1X_2\ldots X_n)$

- **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** $\boldsymbol{First(X_1X_2 \ldots X_{k-1} X_k \ldots X_n)}$**:**
  - **if** $Empty(\boldsymbol{X_i}) = \{\varepsilon\}$ for all $i = 1,\ldots,k\text{-}1$, where $k \le n$ **then** add all symbols from $First(\boldsymbol{X_k})$ to $First(X_1X_2\ldots X_n)$

! Note: $First(\varepsilon) = \emptyset$

Illustration:



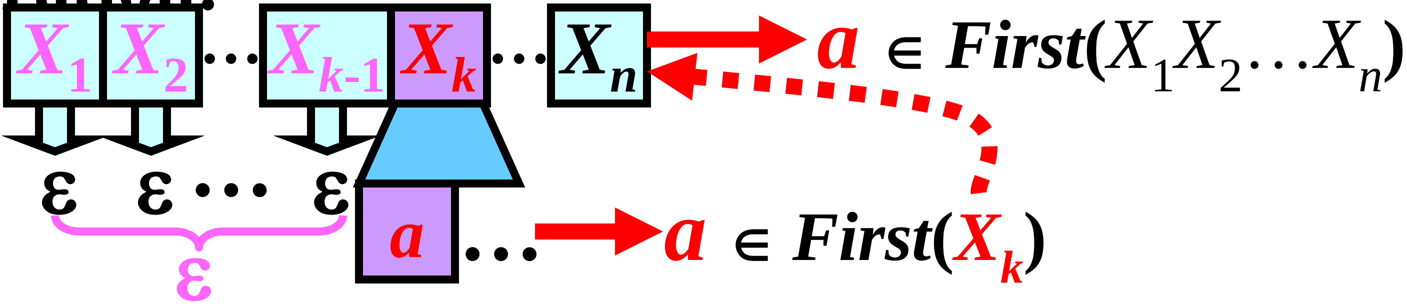
Algorithm: *First*($X_1X_2\ldots X_n$)

- **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 $i = 1,\ldots,k$-1, where $k \leq n$ **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

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

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

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

Set *Empty* & *First* for all $X \in N$:	*Empty*(E)	$:= \emptyset$	*First*(E)	$:= \{i, (\}$
	Empty(E')	$:= \{\varepsilon\}$	*First*(E')	$:= \{+\}$
	Empty(T)	$:= \emptyset$	*First*(T)	$:= \{i, (\}$
	Empty(T')	$:= \{\varepsilon\}$	*First*(T')	$:= \{*\}$
	Empty(F)	$:= \emptyset$	*First*(F)	$:= \{i, (\}$

Task: *First*$(E'T'FET)$

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

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

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

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

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

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

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** $Empty(X_i) = \{\varepsilon\}$ for all $i = 1,\ldots,n$ **then**

```
    $Empty(X_1X_2\ldots X_n) := \{\varepsilon\}$
else
    $Empty(X_1X_2\ldots X_n) := \emptyset$
```

! Note: $Empty(\varepsilon) = \{\varepsilon\}$

Illustration: $\boxed{X_1}\boxed{X_2}\ldots\boxed{X_n}$

Algorithm: $Empty(X_1X_2\ldots X_n)$

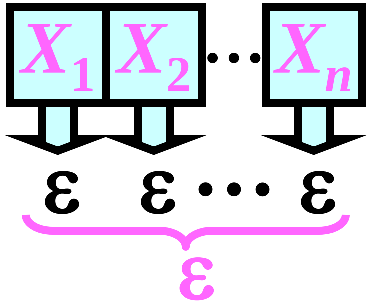
- **Input:** $G = (N, T, P, S)$; $Empty(X)$ for every $X \in N \cup T$;
  $x = X_1X_2\ldots X_n$, where $x \in (N \cup T)^+$
- **Output:** $Empty(X_1X_2\ldots X_n)$

- **Method:**

```
• if $Empty(X_i) = \{\varepsilon\}$ for all $i = 1,\ldots,n$ then
        $Empty(X_1X_2\ldots X_n) := \{\varepsilon\}$
  else
        $Empty(X_1X_2\ldots X_n) := \varnothing$
```

! Note: $Empty(\varepsilon) = \{\varepsilon\}$

Illustration:



Algorithm: *Empty*($X_1X_2…X_n$)

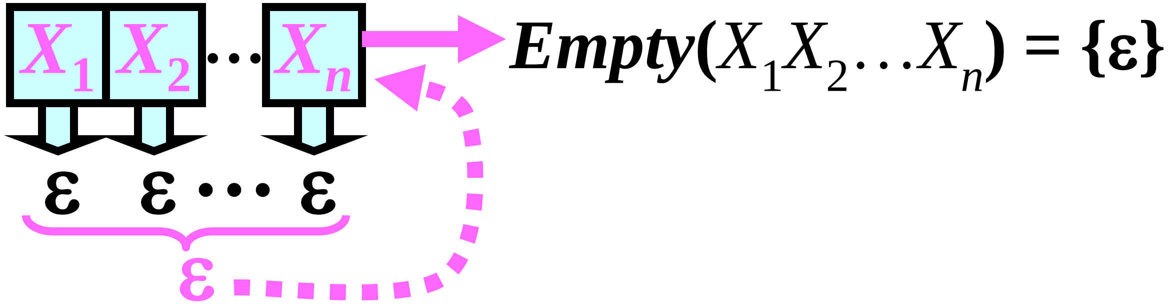
- **Input:** $G = (N, T, P, S)$; *Empty*(X) for every $X \in N \cup T$;
  $x = X_1X_2…X_n$, where $x \in (N \cup T)^+$
- **Output:** *Empty*($X_1X_2…X_n$)

- **Method:**

```
• if $Empty(X_i) = \{\varepsilon\}$ for all $i = 1,…,n$ then
      $Empty(X_1X_2…X_n) := \{\varepsilon\}$
  else
      $Empty(X_1X_2…X_n) := \emptyset$
```

! Note: *Empty*(ε) = $\{\varepsilon\}$

Illustration:



$Empty(X_1X_2\ldots X_n)$: Example

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

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

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

$Empty(E) := \varnothing$
$Empty(E') := \{\varepsilon\}$
$Empty(T) := \varnothing$
$Empty(T') := \{\varepsilon\}$
$Empty(F) := \varnothing$

Task: $Empty(E'T')$

$Empty(E') = Empty(T') = \{\varepsilon\}$, **so** $Empty(E'T') = \{\varepsilon\}$

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

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

Illustration:

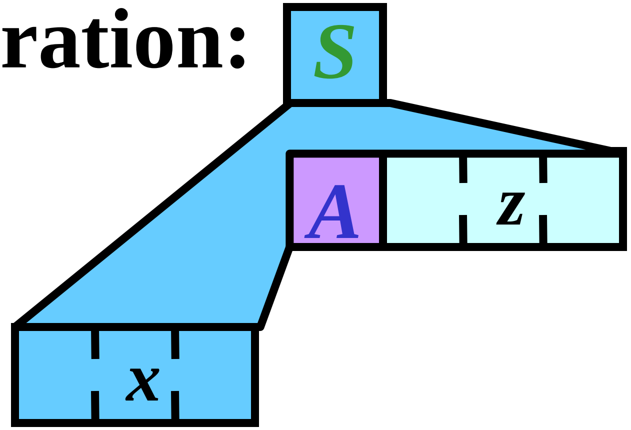
Set *Follow*

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:



$S \Rightarrow^* xAz$

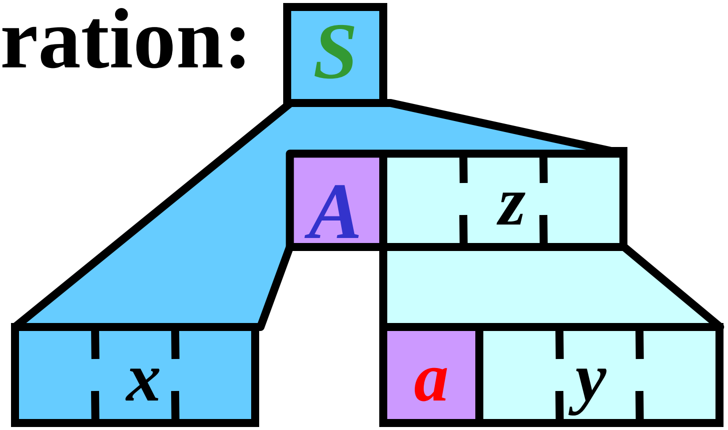
Set *Follow*

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:



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

$$\Downarrow$$

$$a \in Follow(A)$$

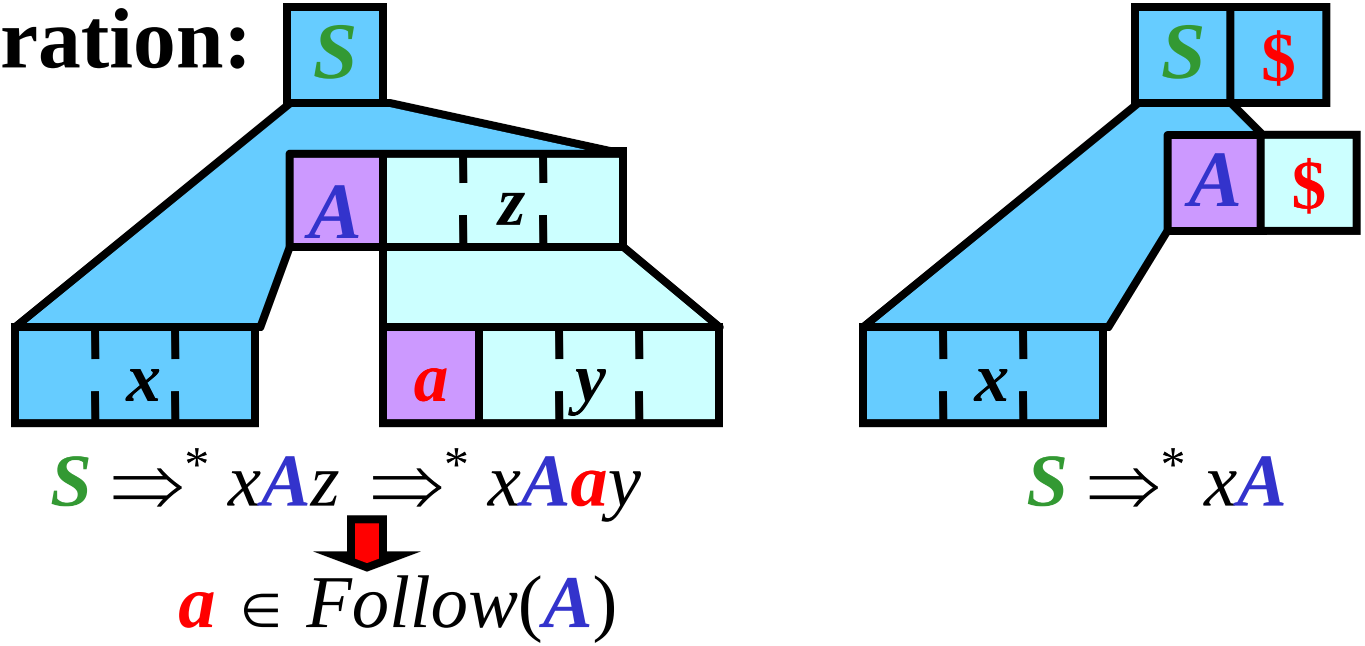
Set *Follow*

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:



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

$a \in Follow(A)$

$S \Rightarrow^* xA$

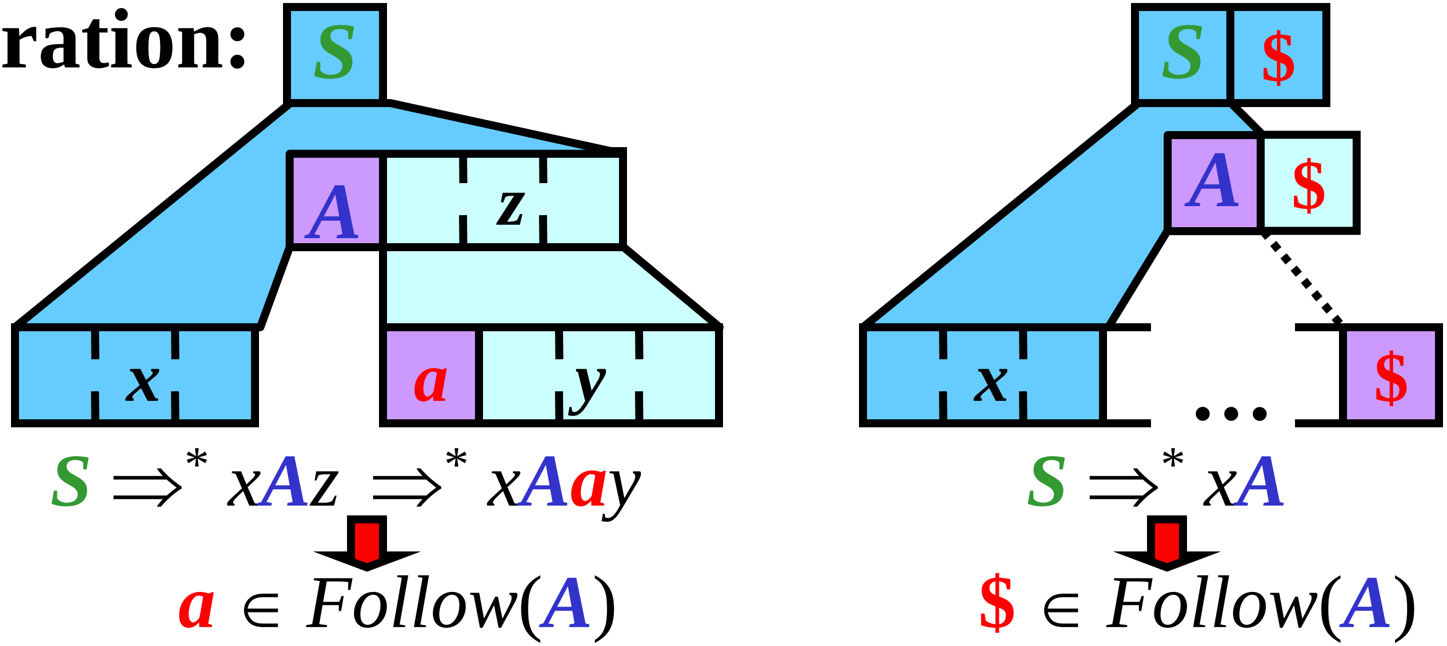
Set *Follow*

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:



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

$a \in Follow(A)$

$S \Rightarrow^* xA$

$\$ \in Follow(A)$

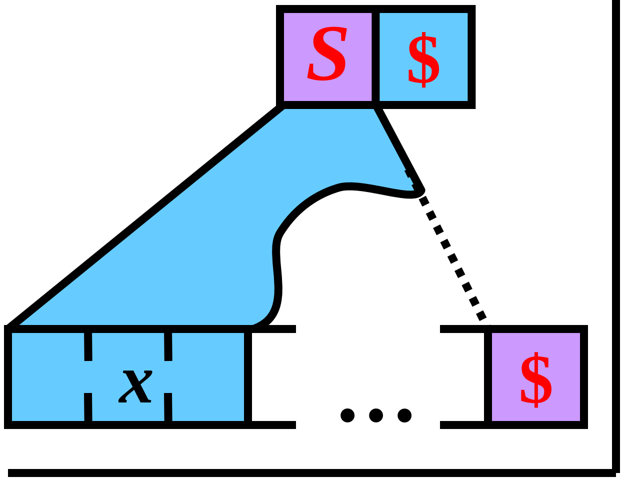
Algorithm: *Follow*(*A*)

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

- **Method:**
- *Follow*(**S**) := {**$**};
- **Apply the following rules until no *Follow* set can be changed:**
- **if** $\boldsymbol{A} \rightarrow x\boldsymbol{B}y \in P$ **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***) := {**$**}

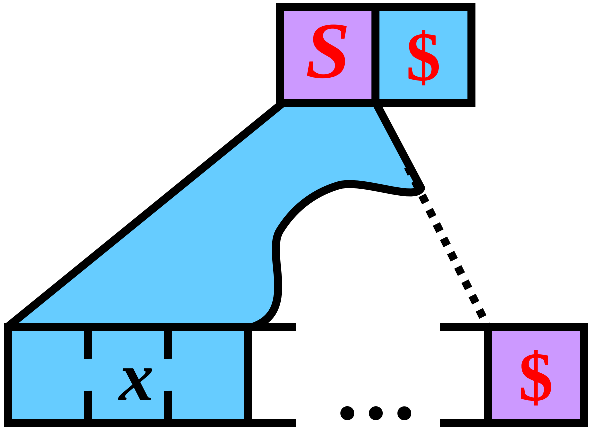


2) Apply the following rules until no *Follow* set can be changed:

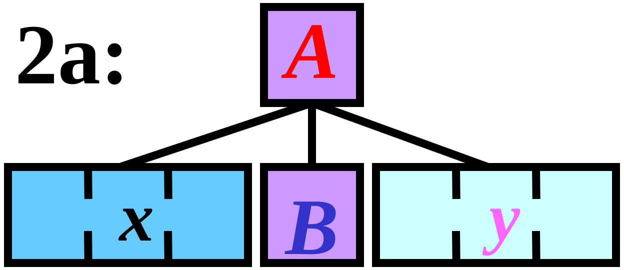
- **if $A \rightarrow xBy \in P$ then**

Previous Algorithm: Illustration

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



2a:

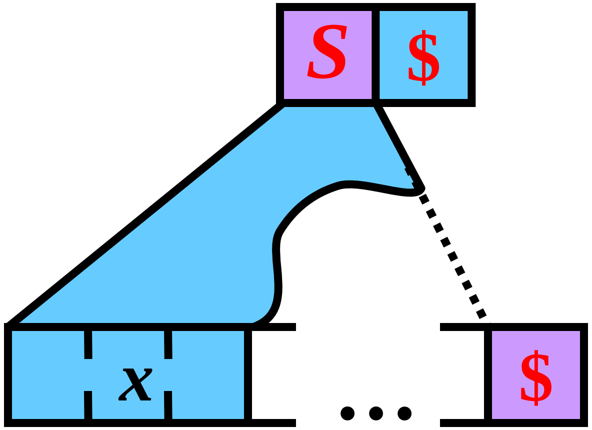


2) Apply the following rules until no *Follow* set can be changed:

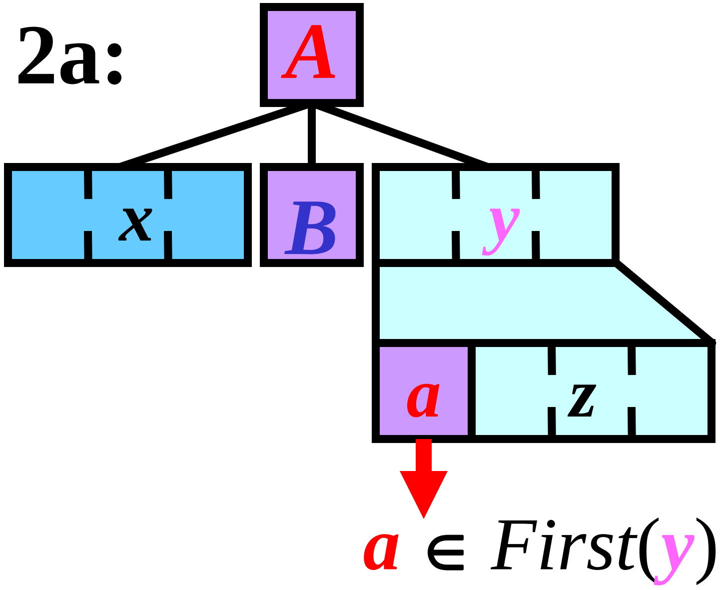
- **if** $A \rightarrow xBy \in P$ **then**
  - **2a) if** $y \neq \varepsilon$ **then** add all symbols from *First*(y) to *Follow*(B)

Previous Algorithm: Illustration

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



2a:

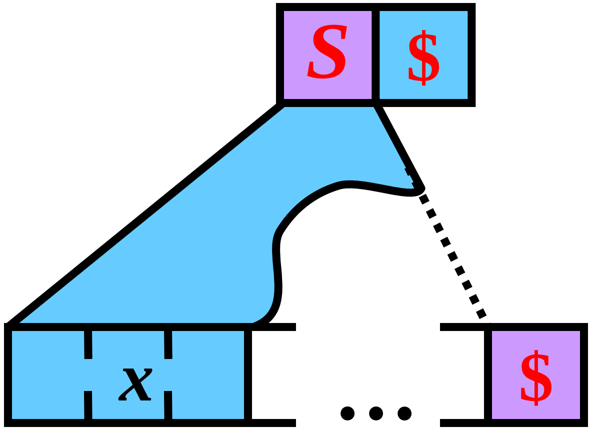


2) Apply the following rules until no *Follow* set can be changed:

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

Previous Algorithm: Illustration

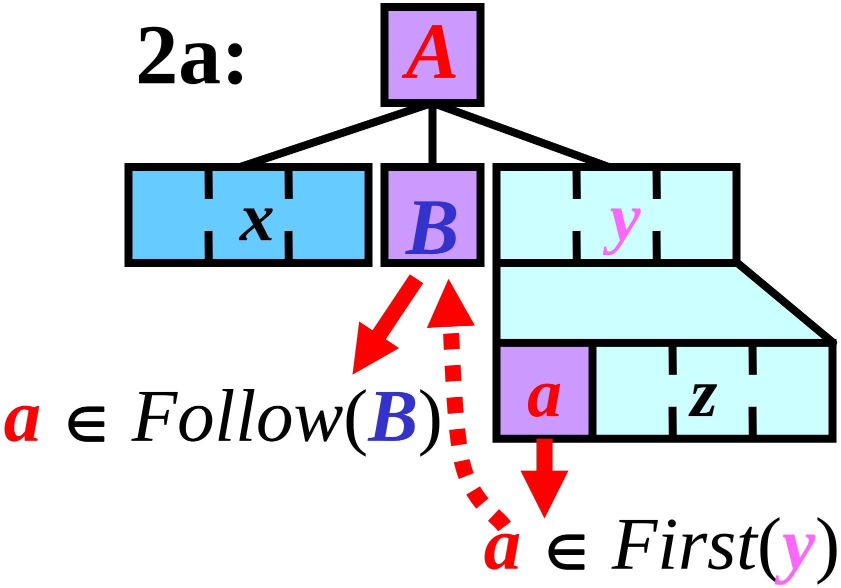
1) *Follow*(***S***) := {**$**}



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

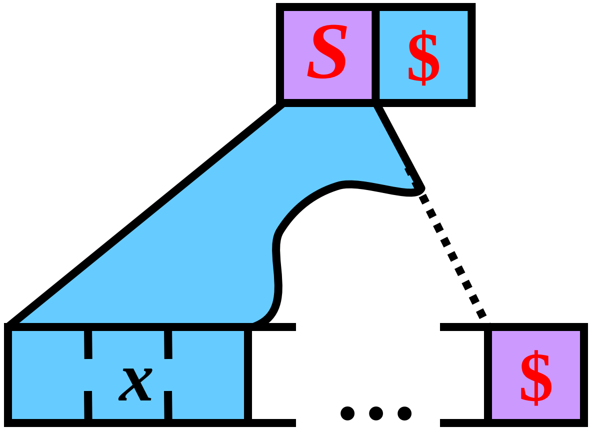
- **if** $A \rightarrow xBy \in P$ **then**
  2a) if $y \neq \varepsilon$ **then** add all symbols from *First*(y) to *Follow*(B)

2a:

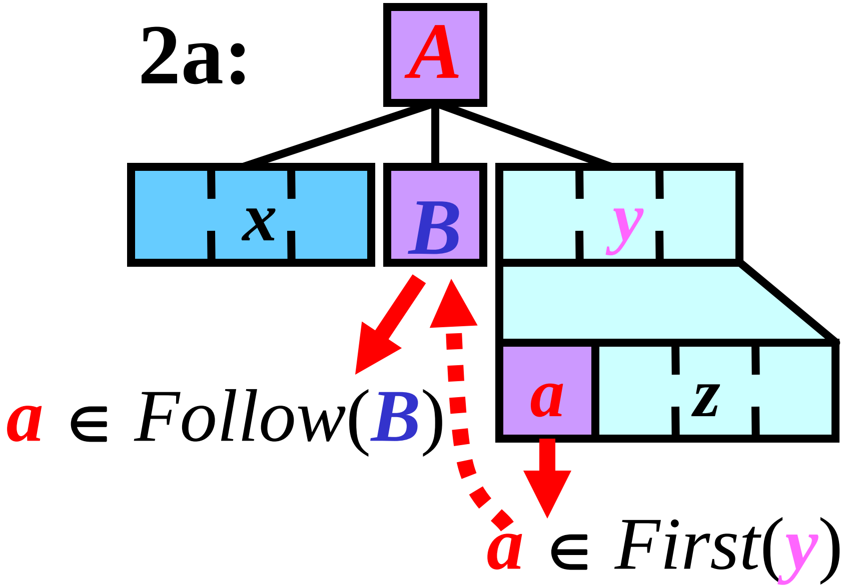


Previous Algorithm: Illustration

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



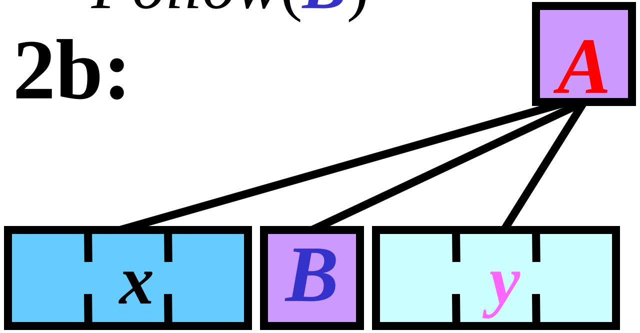
2a:



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

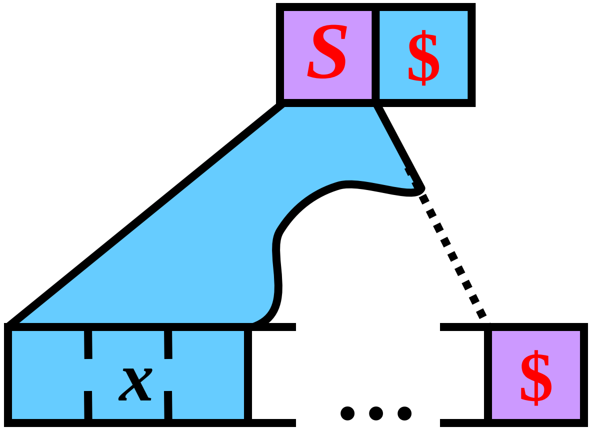
- **if** $A \rightarrow xBy \in P$ **then**
  - **2a) if** $y \neq \varepsilon$ **then** add all symbols from *First*(y) to *Follow*(B)
  - **2b) if** *Empty*(y) = {ε} **then** add all symbols from *Follow*(A) to *Follow*(B)

2b:



Previous Algorithm: Illustration

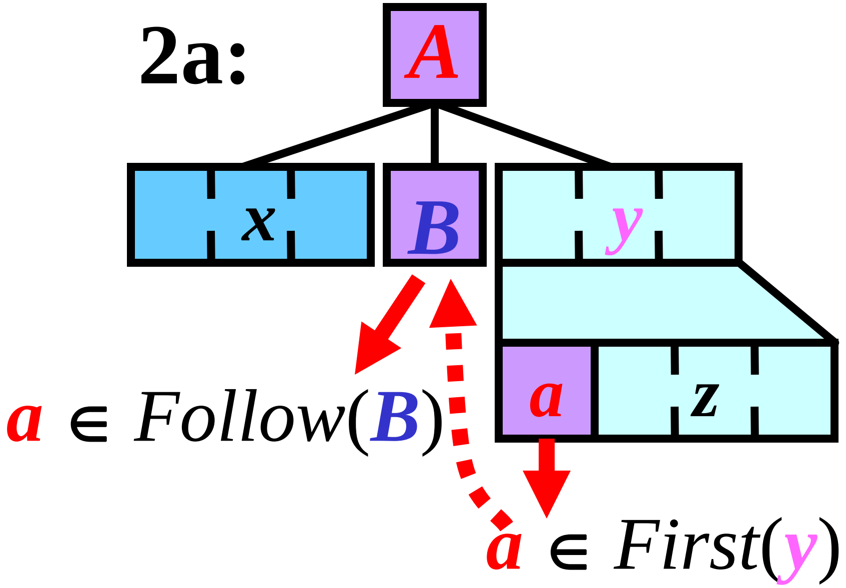
1) *Follow*(***S***) := {$}



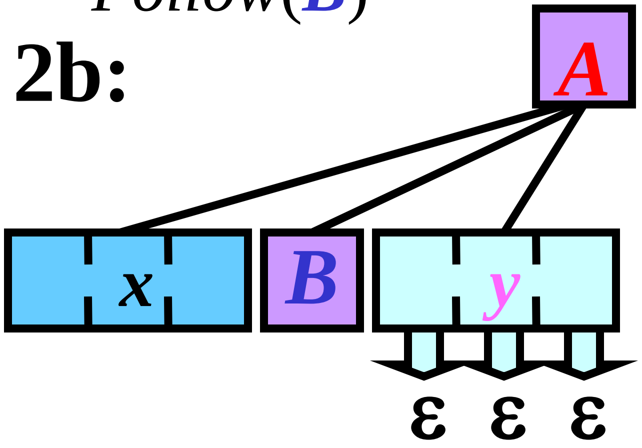
2) Apply the following rules until no *Follow* set can be changed:

- **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:

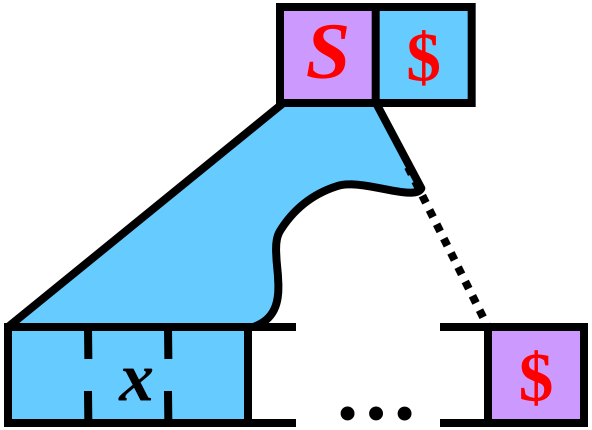


2b:



Previous Algorithm: Illustration

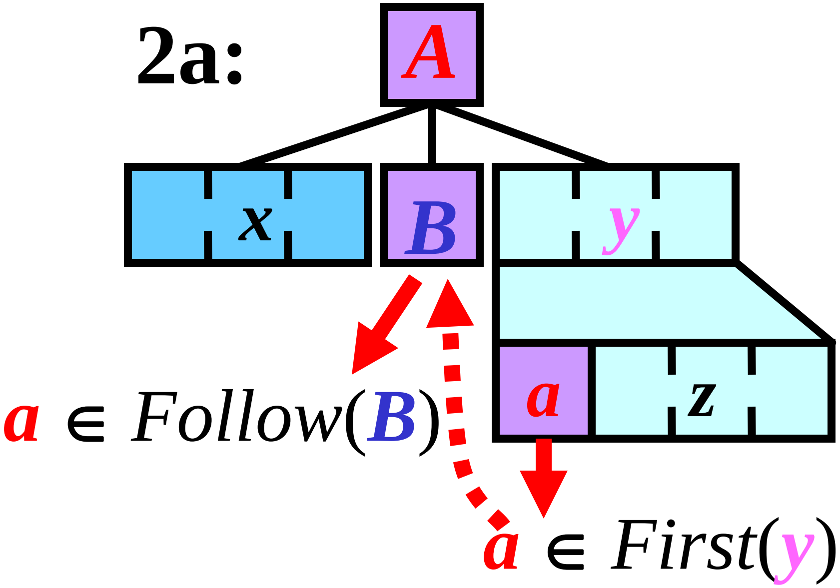
1) $\mathit{Follow}(S) := \{\$\}$



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

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

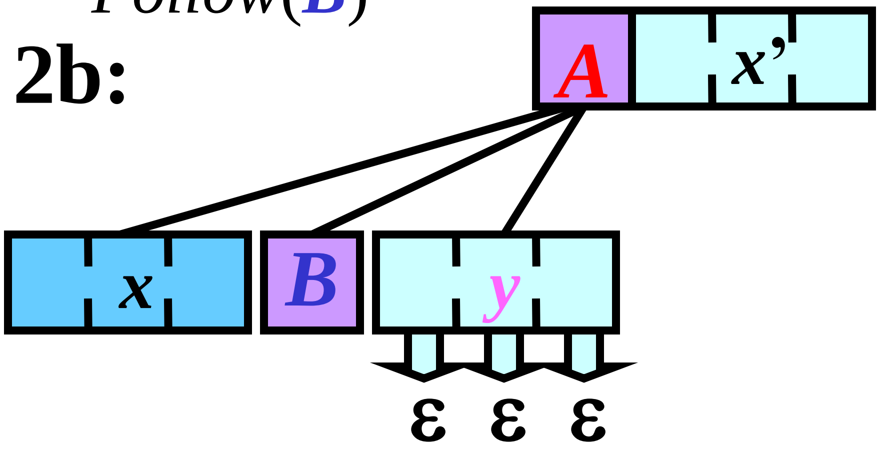
2a:



$a \in \mathit{Follow}(B)$

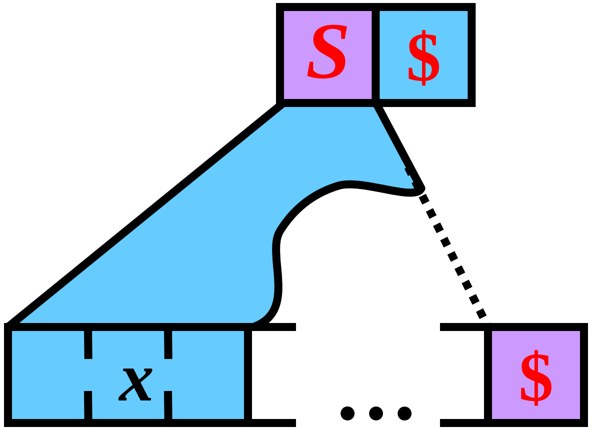
$a \in \mathit{First}(y)$

2b:



Previous Algorithm: Illustration

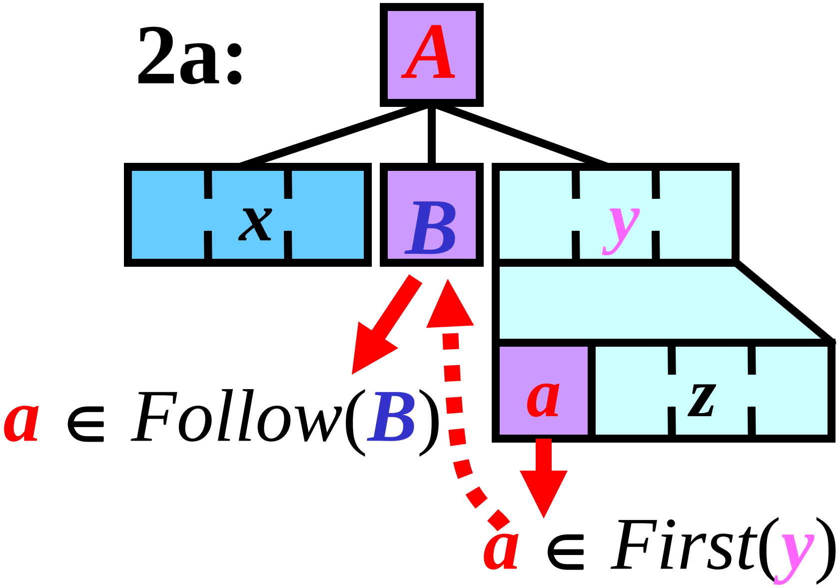
1) *Follow*(***S***) := {**$**}



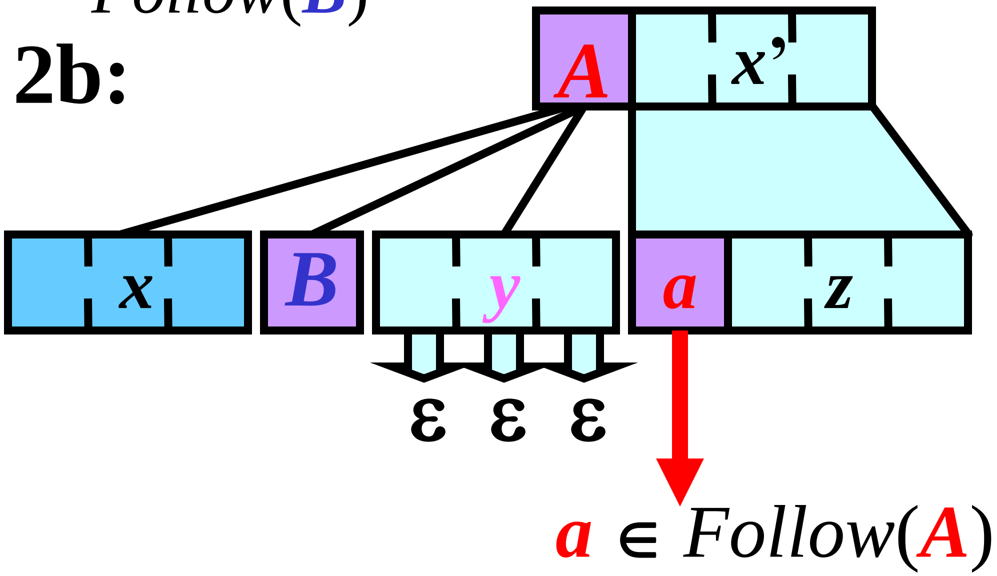
2) Apply the following rules until no *Follow* set can be changed:

- **if** *A* → *xBy* ∈ *P* **then**
  - **2a) if** *y* ≠ ε **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:

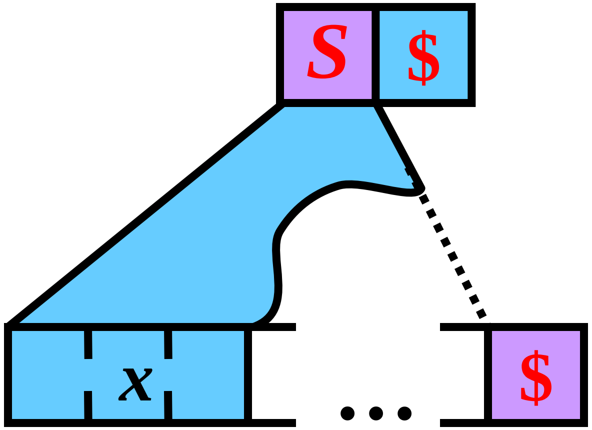


2b:



Previous Algorithm: Illustration

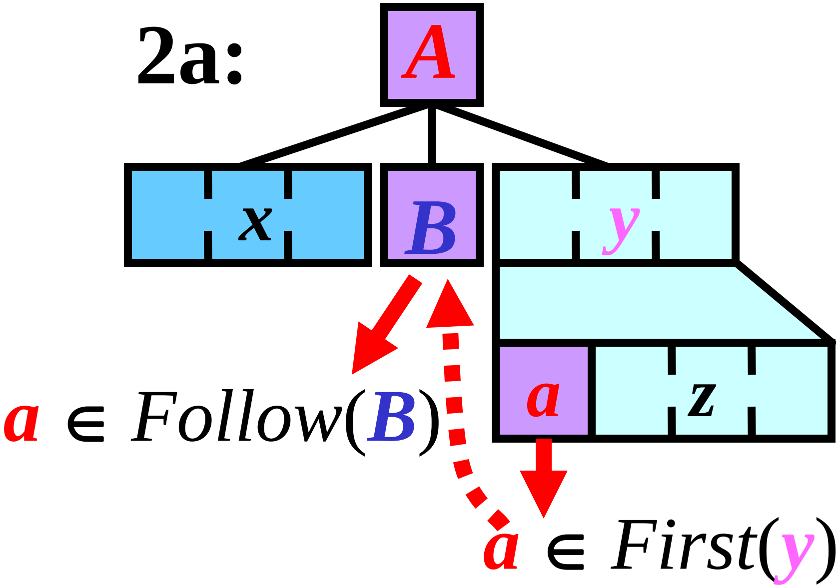
1) *Follow*(**S**) := {**$**}



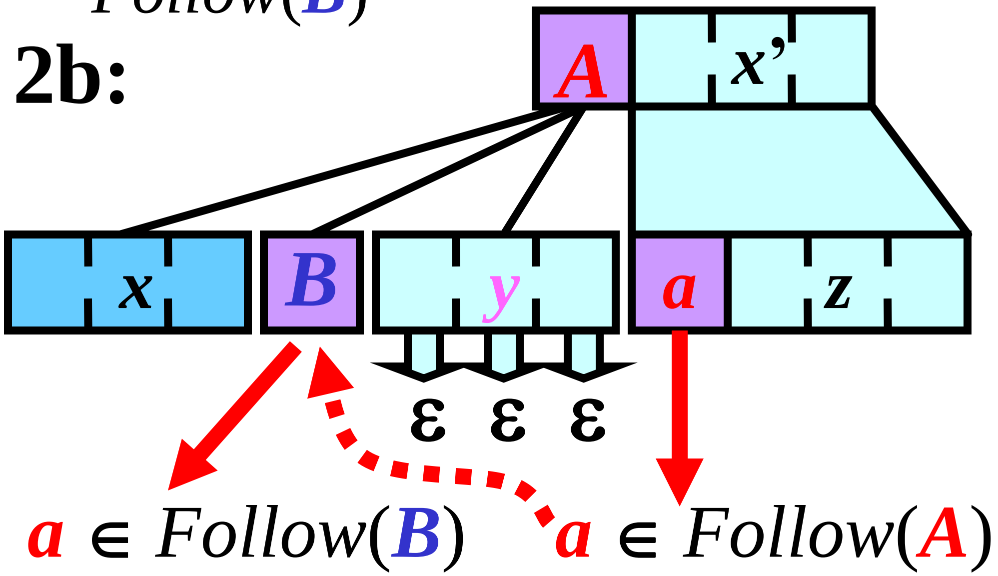
2) Apply the following rules until no *Follow* set can be changed:

- **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:



2b:



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**) := ∅

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***) := {**$**}

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

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

---

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

---

1) $F \to (\ E\ \underbrace{)}_{\neq \varepsilon} \in P$**:**

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

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

---

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

---

1) $F \rightarrow (E \underbrace{)}_{\neq \varepsilon} \in P$**:** **add** $First()) = \{)\}$ **to** $Follow(E)$

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) $F \rightarrow (E \underbrace{)}_{\neq \varepsilon} \in P$**:** **add** *First*()) = {)} **to** *Follow*(***E***)

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

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) $F \to (\ E\ \underbrace{)}_{\neq \varepsilon} \in P$: **add** *First*()) = {)} **to** *Follow*(***E***)

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

2) $E \to TE'\underbrace{\;}_{\varepsilon} \in P$:

ε: *Empty*(ε) = {ε}

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) $\boldsymbol{F} \rightarrow \boldsymbol{(}\ \boldsymbol{E}\ \underbrace{\boldsymbol{)}}_{\neq\ \varepsilon} \in P$**:** **add** *First*()) = {)} **to** *Follow*(***E***)

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

2) $\boldsymbol{E} \rightarrow \boldsymbol{TE'}\underbrace{\;\;}_{\varepsilon:\ Empty(\varepsilon)\ =\ \{\varepsilon\}} \in P$**:** **add** *Follow*(***E***) = {**$**, **)**} **to** *Follow*(***E’***)

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) $F \rightarrow (E \underbrace{)}_{\neq \varepsilon} \in P$: **add** *First*()) = {)} **to** *Follow*(***E***)

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

2) $E \rightarrow TE' \underbrace{\quad}_{\varepsilon} \in P$: **add** *Follow*(***E***) = {**$**, **)**} **to** *Follow*(***E'***)

ε: *Empty*(ε) = {ε}

$E \rightarrow T\underbrace{E'}_{\neq \varepsilon} \in P$:

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

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

---

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

---

1) $F \to (E \underbrace{)}_{\neq \varepsilon} \in P$: **add** $First()) = \{)\}$ **to** $Follow(E)$

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

---

2) $E \to TE'\underbrace{\quad}_{\varepsilon:\ Empty(\varepsilon) = \{\varepsilon\}} \in P$: **add** $Follow(E) = \{\$,)\}$ **to** $Follow(E')$

$E \to T\underbrace{E'}_{\neq \varepsilon} \in P$: **add** $First(E') = \{+\}$ **to** $Follow(T)$

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

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

---

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

---

1) $F \to (\, E \underbrace{)}_{\neq \varepsilon} \in P$: **add** $First()) = \{)\}$ **to** $Follow(E)$

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

---

2) $E \to TE' \underbrace{\ }_{\varepsilon:\ Empty(\varepsilon) = \{\varepsilon\}} \in P$: **add** $Follow(E) = \{\$,)\}$ **to** $Follow(E')$

$E \to T\underbrace{E'}_{\neq \varepsilon} \in P$: **add** $First(E') = \{+\}$ **to** $Follow(T)$

$E \to T\underbrace{E'} \in P$:

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

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) $F \rightarrow (E \underbrace{)}_{\neq \varepsilon} \in P$: **add** *First*()) = {)} **to** *Follow*(***E***)

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

---

2) $E \rightarrow TE'\underbrace{\;}_{\varepsilon:\ Empty(\varepsilon) = \{\varepsilon\}} \in P$: **add** *Follow*(***E***) = {$,)} **to** *Follow*(***E’***)

$E \rightarrow T\underbrace{E'}_{\neq \varepsilon} \in P$: **add** *First*(***E’***) = {+} **to** *Follow*(***T***)

$E \rightarrow T\underbrace{E'}_{Empty(E') = \{\varepsilon\}} \in P$: **add** *Follow*(***E***) = {$,)} **to** *Follow*(***T***)

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) $F \to (E \underbrace{)}_{\neq \varepsilon} \in P$: **add** *First*()) = {)} **to** *Follow*(*E*)

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

---

2) $E \to TE'\underbrace{\ }_{\varepsilon:\ Empty(\varepsilon) = \{\varepsilon\}} \in P$: **add** *Follow*(*E*) = {$,)} **to** *Follow*(*E'*)

$E \to T\underbrace{E'}_{\neq \varepsilon} \in P$: **add** *First*(*E'*) = {+} **to** *Follow*(*T*)

$E \to T\underbrace{E'}_{Empty(E') = \{\varepsilon\}} \in P$: **add** *Follow*(*E*) = {$,)} **to** *Follow*(*T*)

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

$\mathit{Follow}(X)$ for G_{expr3}: Example 2/3

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

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

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

3) $E' \rightarrow +TE'\underbrace{\quad}_{\varepsilon} \in P$: **add** *Follow*($E'$) = $\{\$,)\}$ **to** *Follow*(E')

ε: *Empty*(ε) = $\{\varepsilon\}$

$E' \rightarrow +T\underbrace{E'}_{\neq \varepsilon} \in P$: **add** *First*($E'$) = $\{+\}$ **to** *Follow*(T)

$E' \rightarrow +T\underbrace{E'} \in P$: **add** *Follow*($E'$) = $\{\$,)\}$ **to** *Follow*(T)

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

Summary: Nothing is changed

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) $E' \to +TE'\underbrace{\ }_{\varepsilon} \in P$: **add** *Follow*(*E'*) = {$,)} **to** *Follow*(*E'*)

ε: *Empty*(ε) = {ε}

$E' \to +T\underbrace{E'}_{\neq\varepsilon} \in P$: **add** *First*(*E'*) = {+} **to** *Follow*(*T*)

$E' \to +T\underbrace{E'} \in P$: **add** *Follow*(*E'*) = {$,)} **to** *Follow*(*T*)

Empty(*E'*) = {ε}

Summary: Nothing is changed

---

4) $T \to FT'\underbrace{\ }_{\varepsilon} \in P$: **add** *Follow*(*T*) = {+, $,)} **to** *Follow*(*T'*)

ε: *Empty*(ε) = {ε}

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

$T \to F\underbrace{T'} \in P$: **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***, **(**}
First(***E’***) := {**+**}
First(***T***) := {***i***, **(**}
First(***T’***) := {*****}
First(***F***) := {***i***, **(**}

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

Follow(***E***) := {**$**, **)**}
Follow(***E’***) := {**$**, **)**}
Follow(***T***) := {**+**, **$**, **)**}
Follow(***T’***) := {**+**, **$**, **)**}
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) $T' \rightarrow *FT'\underbrace{\;}_{\varepsilon:\ Empty(\varepsilon) = \{\varepsilon\}} \in P$: **add** *Follow*($T'$) = {+, $,)} **to** *Follow*($T'$)

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

$T' \rightarrow *F\underbrace{T'}_{Empty(T') = \{\varepsilon\}} \in P$: **add** *Follow*($T'$) = {+, $,)} **to** *Follow*($F$)

End: Nothing is changed.

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

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

5) $T' \to *FT'\underbrace{\;\;}_{\varepsilon:\ Empty(\varepsilon) = \{\varepsilon\}} \in P$: **add** $Follow(T') = \{+, \$,)\}$ **to** $Follow(T')$

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

$T' \to *F\underbrace{T'}_{Empty(T') = \{\varepsilon\}} \in P$: **add** $Follow(T') = \{+, \$,)\}$ **to** $Follow(F)$

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 $\mathbf{A} \rightarrow \mathbf{x} \in P$, we define $Predict(\mathbf{A} \rightarrow \mathbf{x})$ so that

- if $Empty(\mathbf{x}) = \{\varepsilon\}$ then

  $Predict(\mathbf{A} \rightarrow \mathbf{x}) = First(\mathbf{x}) \cup Follow(\mathbf{A})$
- if $Empty(\mathbf{x}) = \emptyset$ then

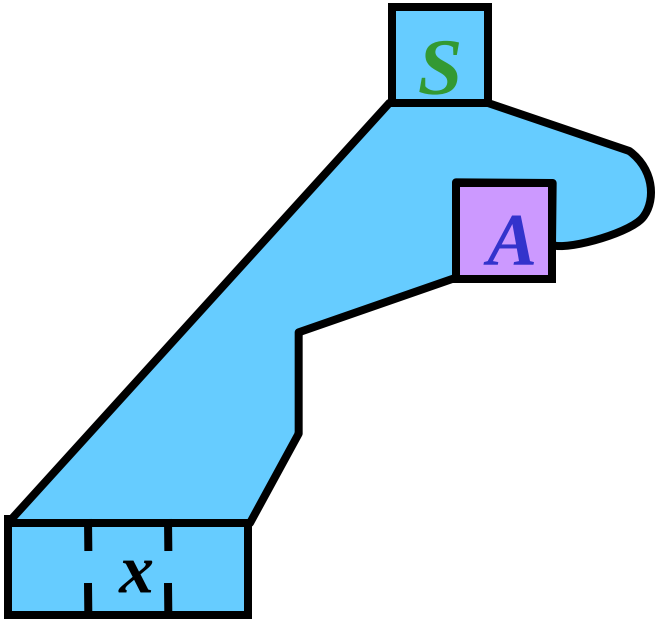
  $Predict(\mathbf{A} \rightarrow \mathbf{x}) = First(\mathbf{x})$

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

$Empty(\mathbf{X_1X_2...X_n}) = \varnothing$ vs. $Empty(\mathbf{X_1X_2...X_n}) = \{\varepsilon\}$

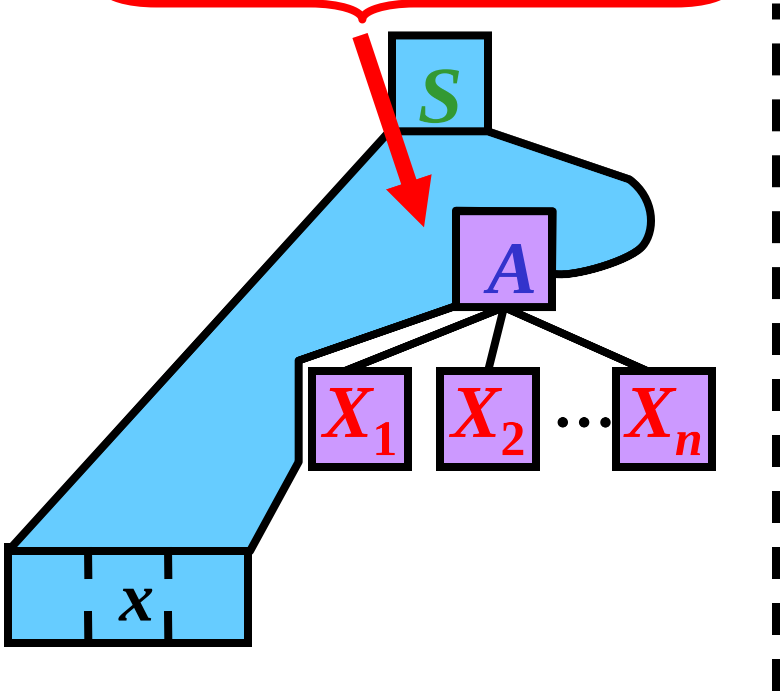
Set $Predict(A \rightarrow X_1X_2 \ldots X_n)$: Illustration

$Empty(\mathbf{X_1X_2 \ldots X_n}) = \emptyset$ vs. $Empty(\mathbf{X_1X_2 \ldots X_n}) = \{\varepsilon\}$



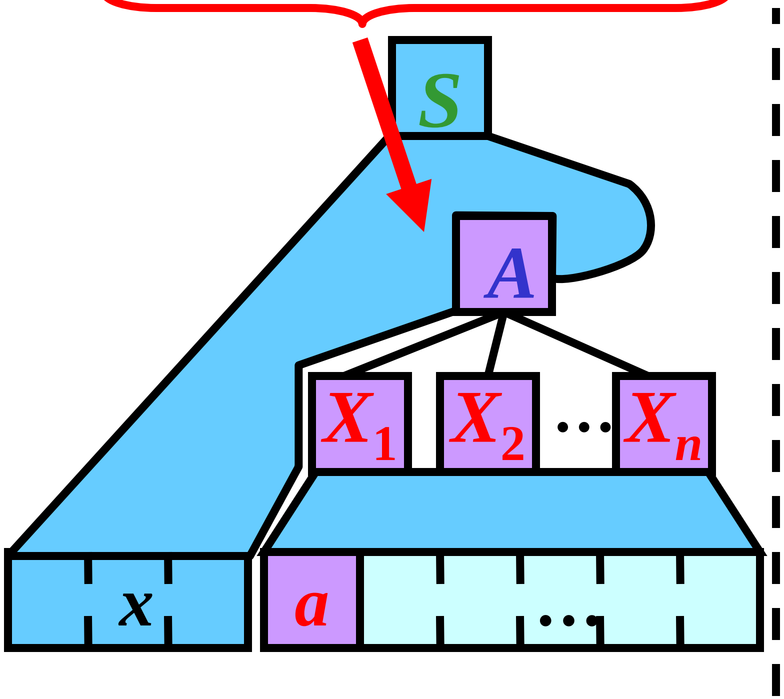
Set $Predict(A \rightarrow X_1X_2...X_n)$: Illustration

$\underbrace{Empty(X_1X_2...X_n) = \varnothing}$ vs. $Empty(X_1X_2...X_n) = \{\varepsilon\}$



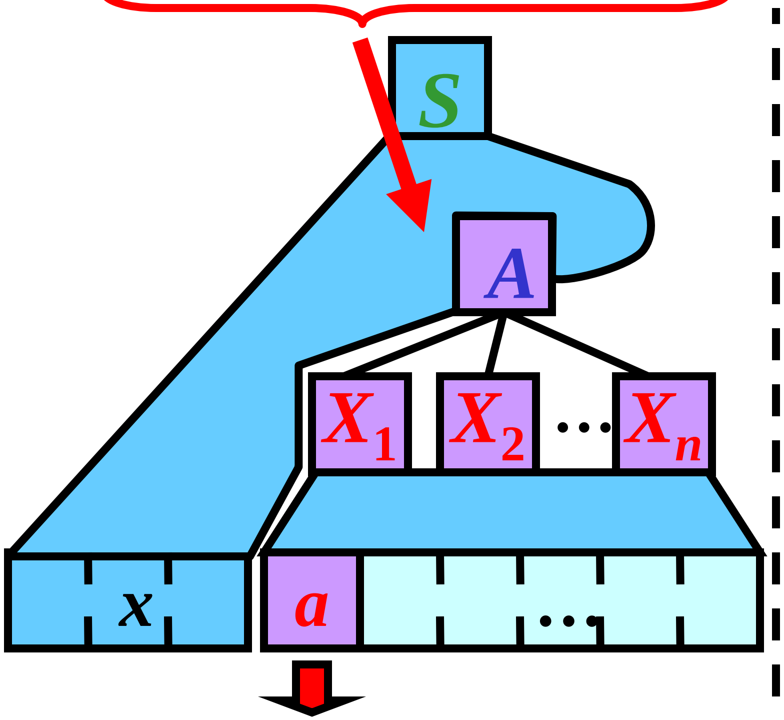
Set *Predict*($A \to X_1X_2...X_n$): Illustration

$\underbrace{Empty(X_1X_2...X_n) = \varnothing}$ vs. $Empty(X_1X_2...X_n) = \{\varepsilon\}$



Set *Predict*($A \rightarrow X_1X_2 \ldots X_n$): Illustration

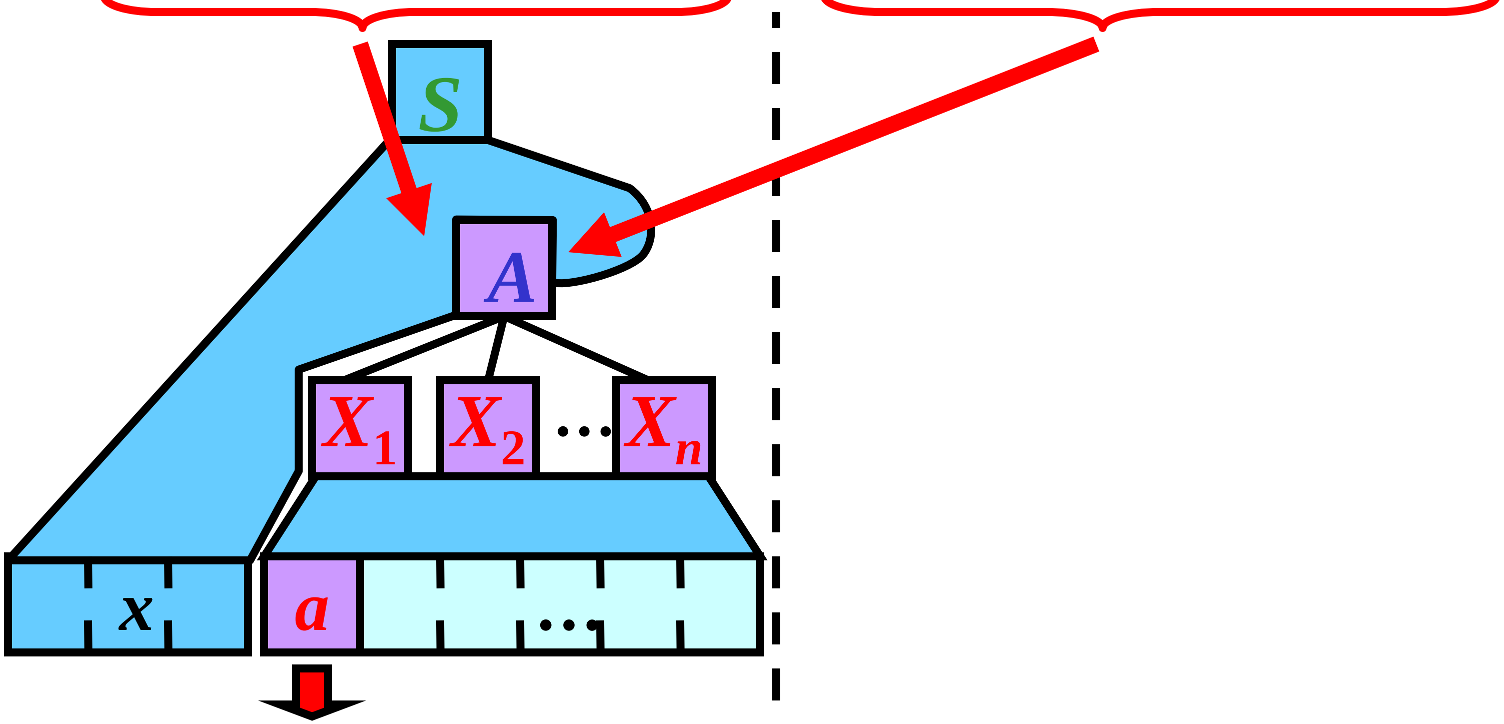
$\underbrace{\mathit{Empty}(X_1X_2\ldots X_n) = \emptyset}$ vs. $\mathit{Empty}(X_1X_2\ldots X_n) = \{\varepsilon\}$



$a \in \mathit{First}(X_1X_2\ldots X_n)$

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

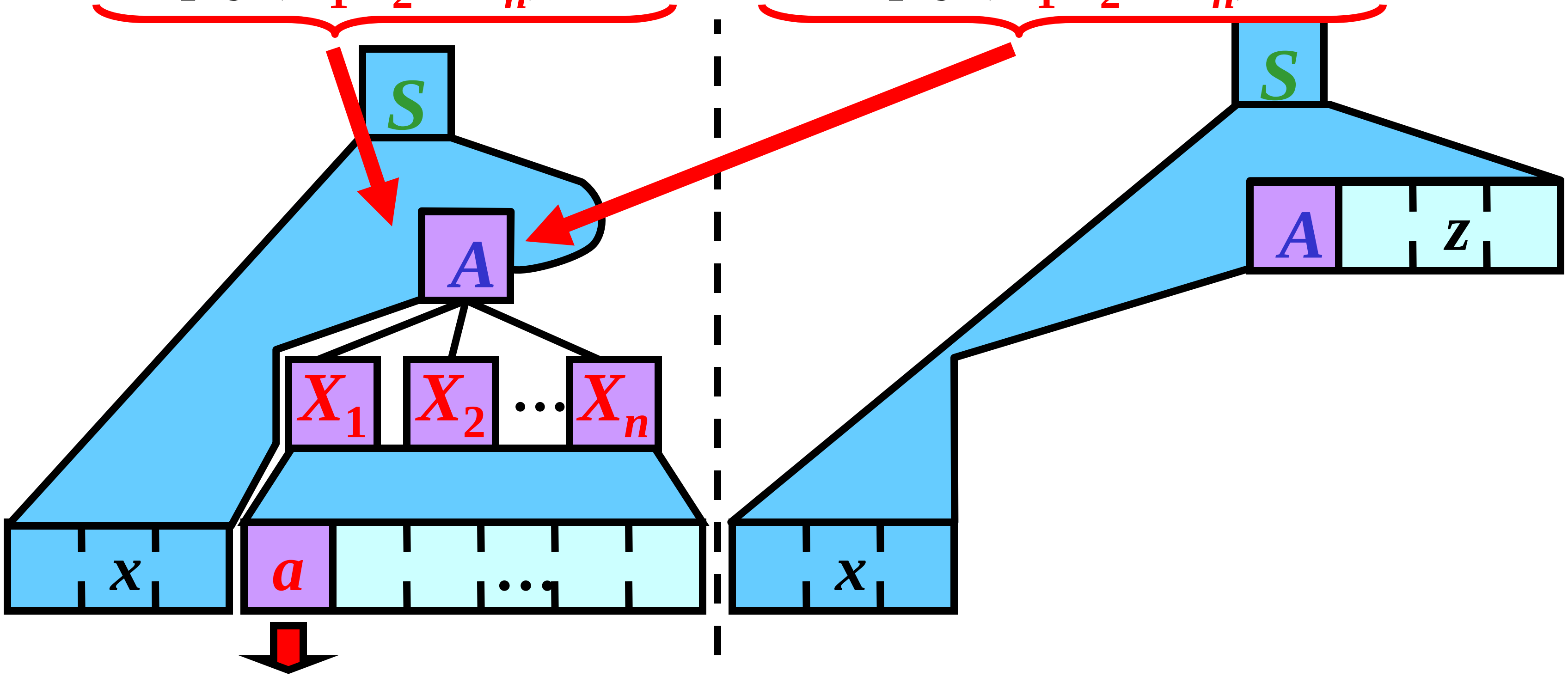
$Empty(X_1X_2...X_n) = \emptyset$ vs. $Empty(X_1X_2...X_n) = \{\varepsilon\}$



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

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

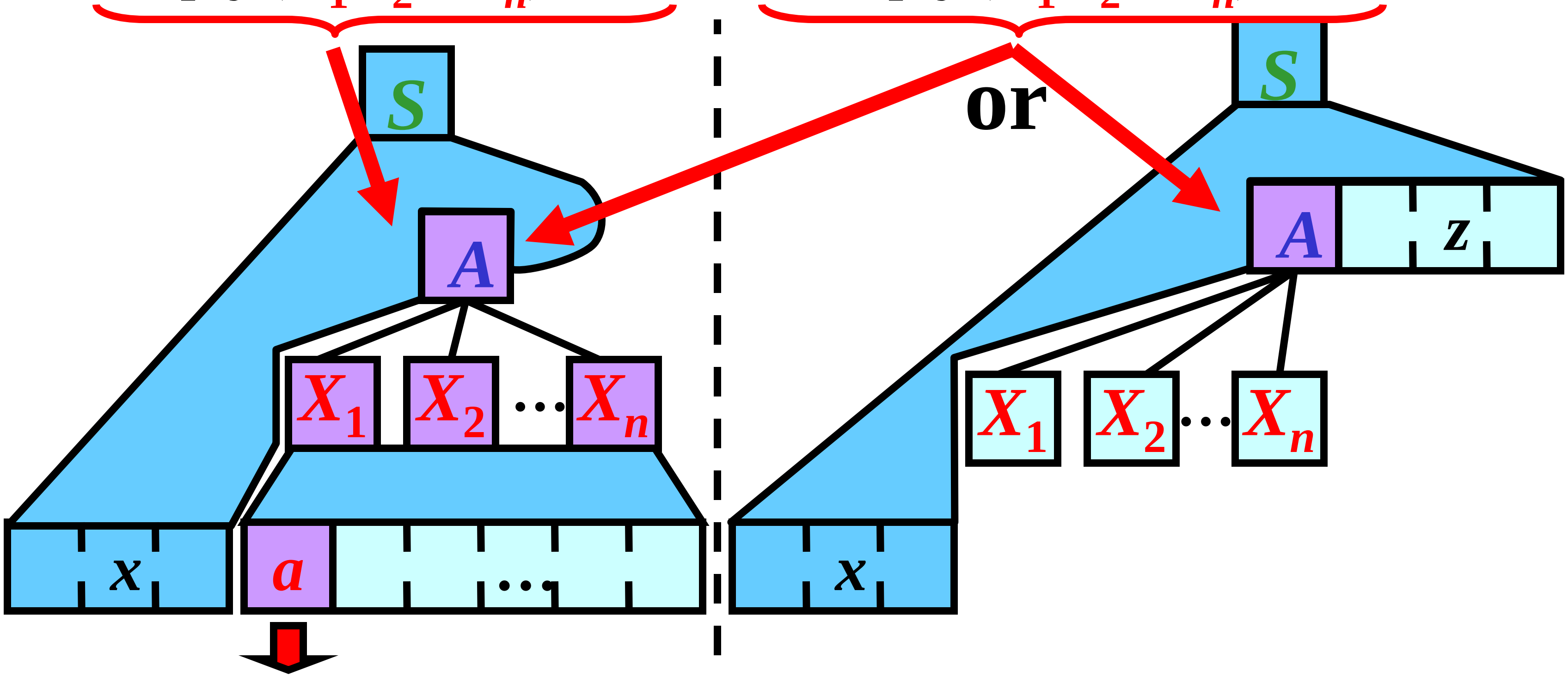
$Empty(X_1X_2...X_n) = \varnothing$ vs. $Empty(X_1X_2...X_n) = \{\varepsilon\}$



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

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

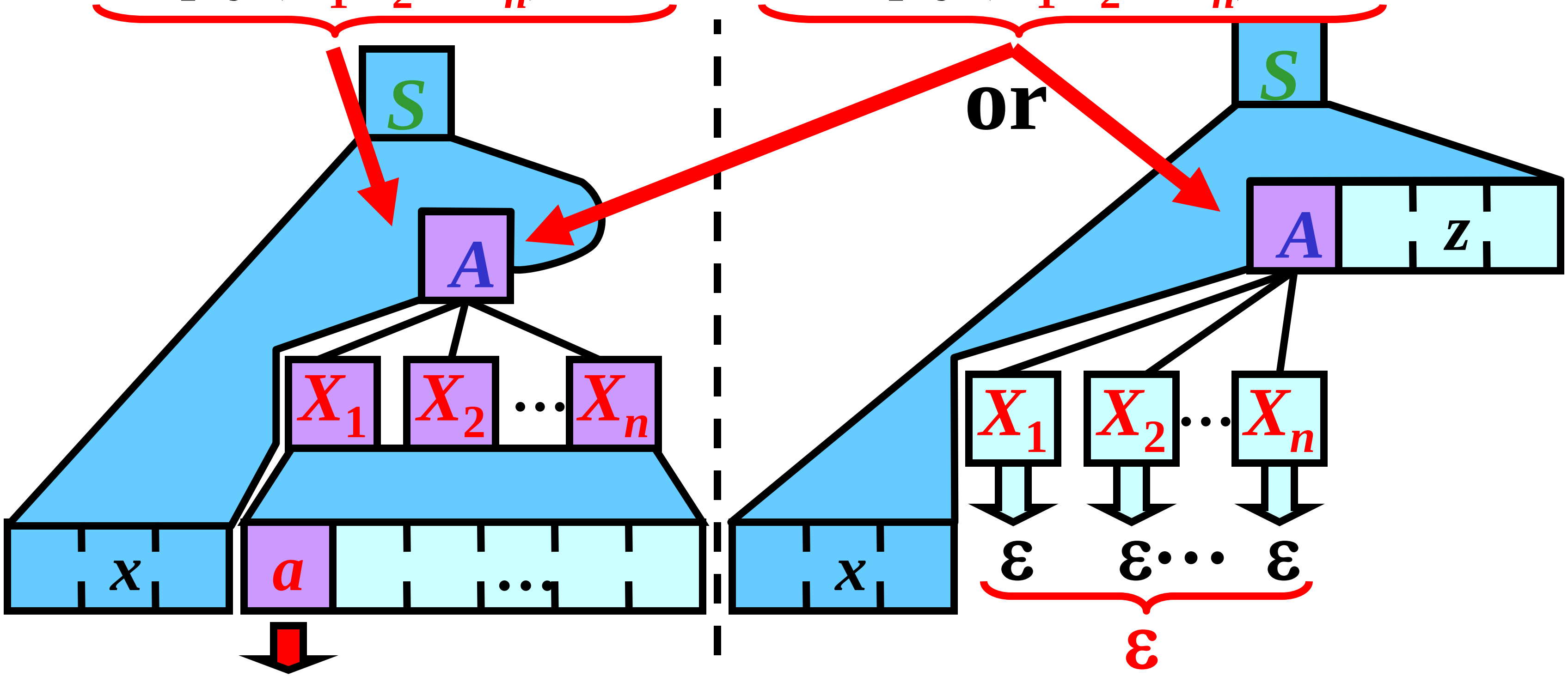
$Empty(X_1X_2...X_n) = \emptyset$ vs. $Empty(X_1X_2...X_n) = \{\varepsilon\}$



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

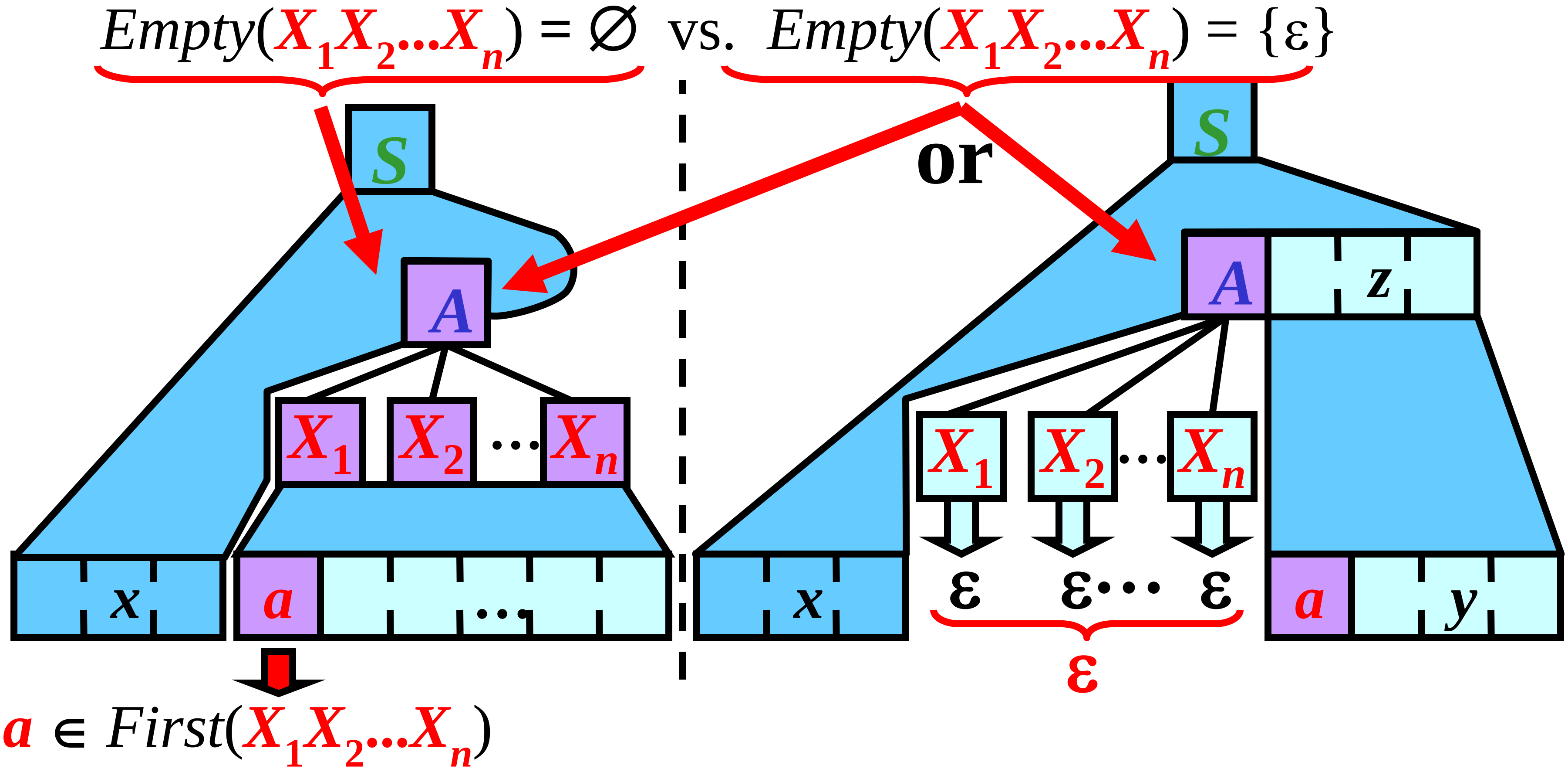
Set *Predict*($A \rightarrow X_1X_2...X_n$): Illustration

Empty($X_1X_2...X_n$) = ∅ vs. *Empty*($X_1X_2...X_n$) = {ε}



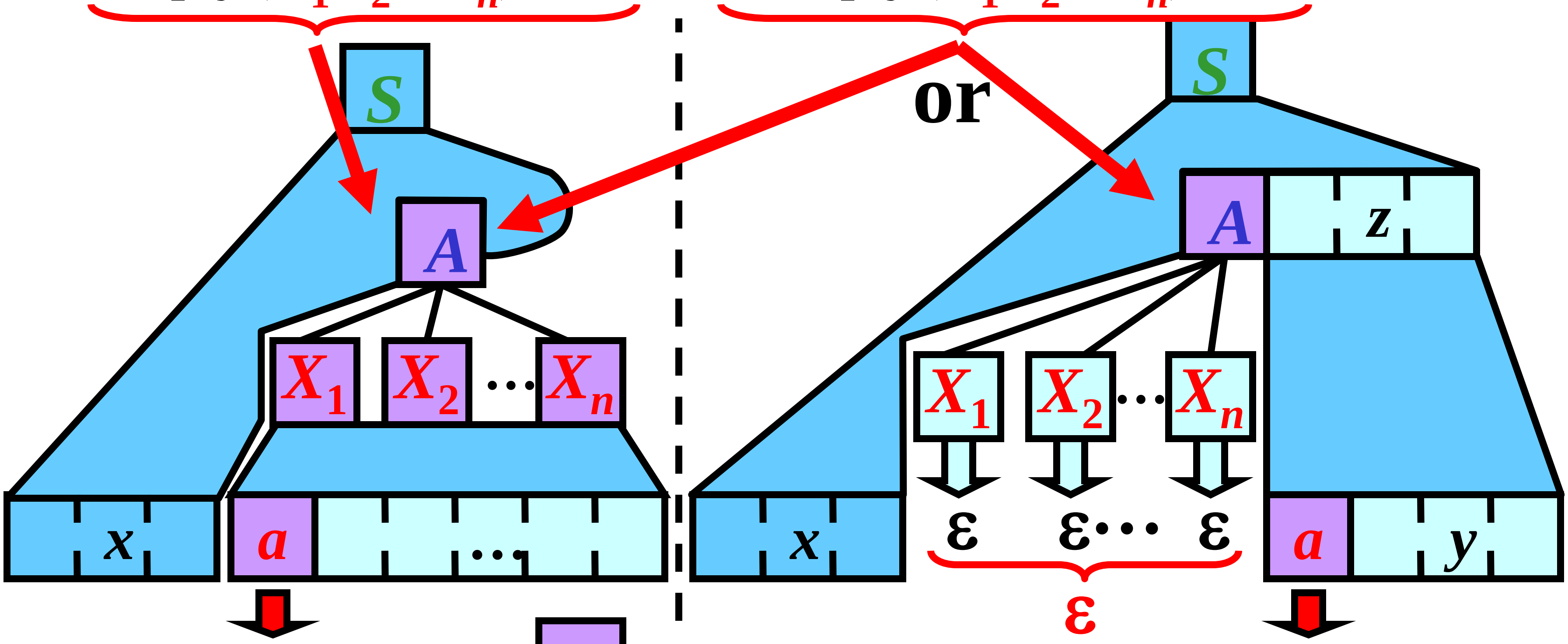
a ∈ *First*($X_1X_2...X_n$)

Set $Predict(A \rightarrow X_1X_2 \ldots X_n)$: Illustration



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

Empty($X_1X_2...X_n$) = ∅ vs. *Empty*($X_1X_2...X_n$) = {ε}

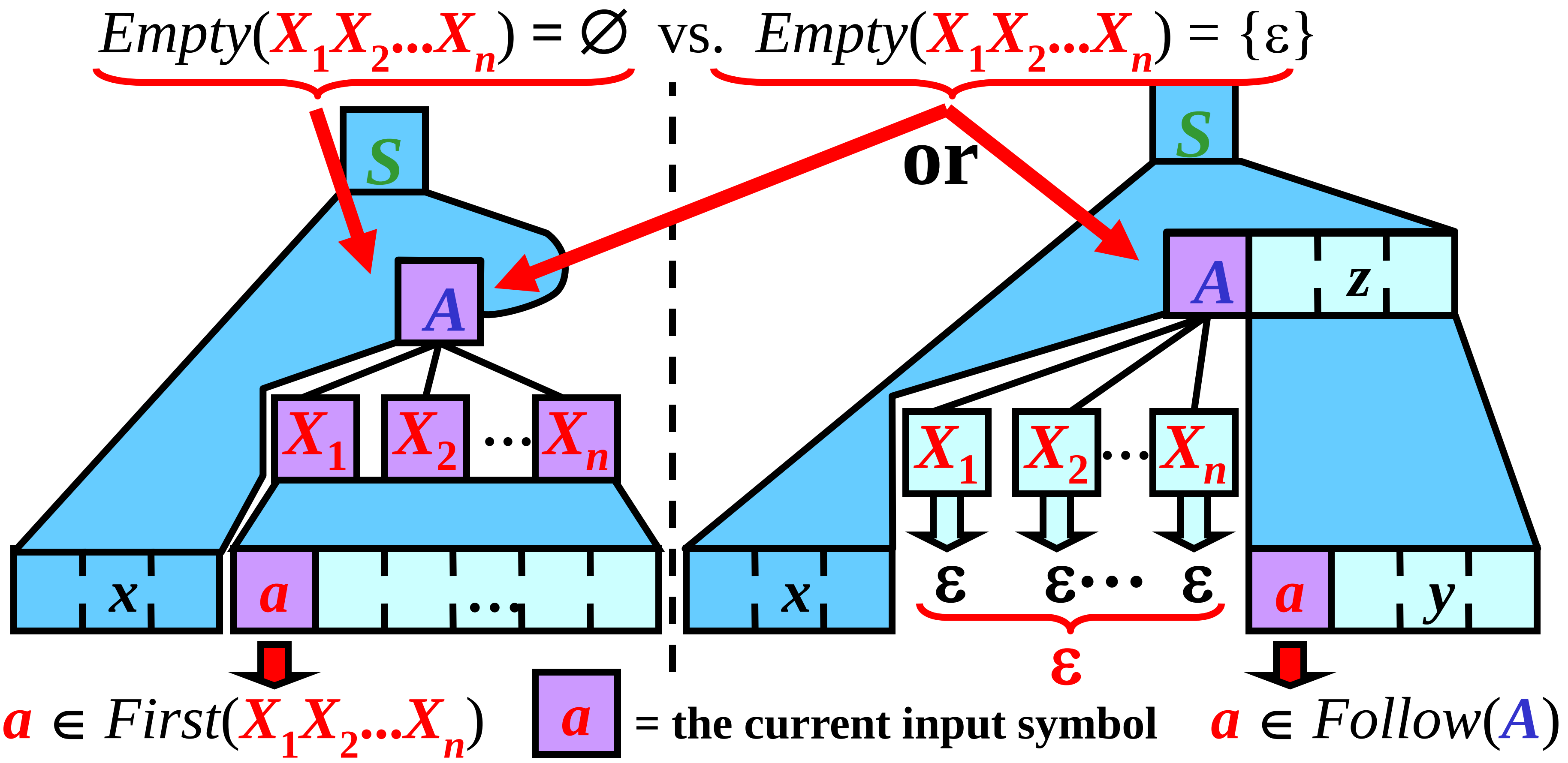


a ∈ *First*($X_1X_2...X_n$)

a = **the current input symbol**

a ∈ *Follow*(**A**)

Set *Predict*($A \rightarrow X_1X_2\ldots X_n$): Illustration



$Empty(X_1X_2\ldots X_n) = \varnothing$ vs. $Empty(X_1X_2\ldots X_n) = \{\varepsilon\}$

$a \in First(X_1X_2\ldots X_n)$ $\boxed{a}$ **= the current input symbol** $a \in Follow(A)$

Summary: if $Empty(X_1X_2\ldots X_n) = \{\varepsilon\}$ then

$Predict(A \rightarrow X_1X_2\ldots X_n) = First(X_1X_2\ldots X_n) \cup Follow(A)$;

otherwise, $Predict(A \rightarrow X_1X_2\ldots X_n) = First(X_1X_2\ldots X_n)$

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

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

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

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

1: $E \rightarrow TE'$

$Empty(TE') = \emptyset$ because $Empty(T) = \emptyset$

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

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

Predict(*A* → *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*(**+**) = ∅
Predict(2) := *First*(**+*TE’***) = *First*(**+**) = **{+}**

3: ***E’*** → ε

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

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

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

$\mathit{Empty}(E) := \emptyset$
$\mathit{Empty}(E') := \{\varepsilon\}$
$\mathit{Empty}(T) := \emptyset$
$\mathit{Empty}(T') := \{\varepsilon\}$
$\mathit{Empty}(F) := \emptyset$

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

$1\colon E \rightarrow TE'$

$\mathit{Empty}(TE') = \emptyset$ because $\mathit{Empty}(T) = \emptyset$
$\mathit{Predict}(1) := \mathit{First}(TE') = \mathit{First}(T) = \{i, (\}$

$2\colon E' \rightarrow +TE'$

$\mathit{Empty}(+TE') = \emptyset$ because $\mathit{Empty}(+) = \emptyset$
$\mathit{Predict}(2) := \mathit{First}(+TE') = \mathit{First}(+) = \{+\}$

$3\colon E' \rightarrow \varepsilon$

$\mathit{Empty}(\varepsilon) = \{\varepsilon\}$
$\mathit{Predict}(3) := \mathit{First}(\varepsilon) \cup \mathit{Follow}(E') = \emptyset \cup \{\$,)\} = \{\$,)\}$

$4\colon T \rightarrow FT'$

$\mathit{Empty}(FT') = \emptyset$ because $\mathit{Empty}(F) = \emptyset$
$\mathit{Predict}(4) := \mathit{First}(FT') = \mathit{First}(F) = \{i, (\}$

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

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

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

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

$\mathit{Empty}(E) := \varnothing$
$\mathit{Empty}(E') := \{\varepsilon\}$
$\mathit{Empty}(T) := \varnothing$
$\mathit{Empty}(T') := \{\varepsilon\}$
$\mathit{Empty}(F) := \varnothing$

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

5: $T' \to *FT'$

$\mathit{Empty}(*FT') = \varnothing$ because $\mathit{Empty}(*) = \varnothing$

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

Predict(A → 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(*) = ∅
Predict(5) := First(*FT’) = First(*) = {*}

6: T’ → ε

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

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

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

5: $T' \rightarrow *FT'$

$Empty(*FT') = \varnothing$ because $Empty(*) = \varnothing$

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

6: $T' \rightarrow \varepsilon$

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

$Predict(6) := First(\varepsilon) \cup Follow(T') = \varnothing \cup \{+, \$,)\} = \{+, \$,)\}$

7: $F \rightarrow (E)$

$Empty((E)) = \varnothing$ because $Empty(() = \varnothing$

$Predict(7) := First((E)) = First(() = \{(\}$

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

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

$Empty(E) := \emptyset$
$Empty(E') := \{\varepsilon\}$
$Empty(T) := \emptyset$
$Empty(T') := \{\varepsilon\}$
$Empty(F) := \emptyset$

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

5: $T' \to *FT'$

$Empty(*FT') = \emptyset$ because $Empty(*) = \emptyset$
$Predict(5) := First(*FT') = First(*) = \{*\}$

6: $T' \to \varepsilon$

$Empty(\varepsilon) = \{\varepsilon\}$
$Predict(6) := First(\varepsilon) \cup Follow(T') = \emptyset \cup \{+, \$,)\} = \{+, \$,)\}$

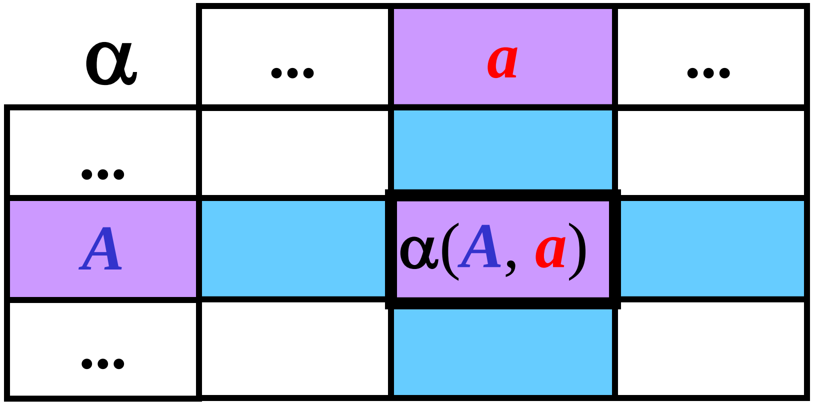
7: $F \to (E)$

$Empty((E)) = \emptyset$ because $Empty(() = \emptyset$
$Predict(7) := First((E)) = First(() = \{(\}$

8: $F \to i$

$Empty(i) = \emptyset$
$Predict(8) := First(i) = \{i\}$

Construction of LL Table



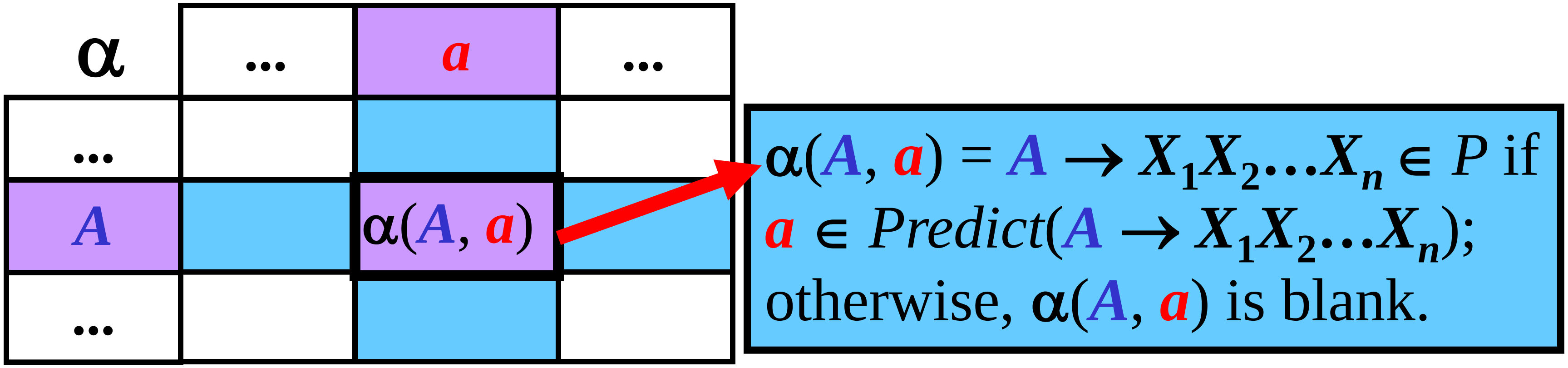
α	…	*a*	…
…			
A		α(*A*, *a*)	
…			

Construction of LL Table

α	...	a	...
...			
A		$\alpha(A, a)$	
...			

$\alpha(A, a) = A \to X_1X_2\ldots X_n \in P$ if $a \in \mathit{Predict}(A \to X_1X_2\ldots X_n)$; otherwise, $\alpha(A, a)$ is blank.

Construction of LL Table



α	…	a	…
…			
A		$\alpha(A, a)$	
…			

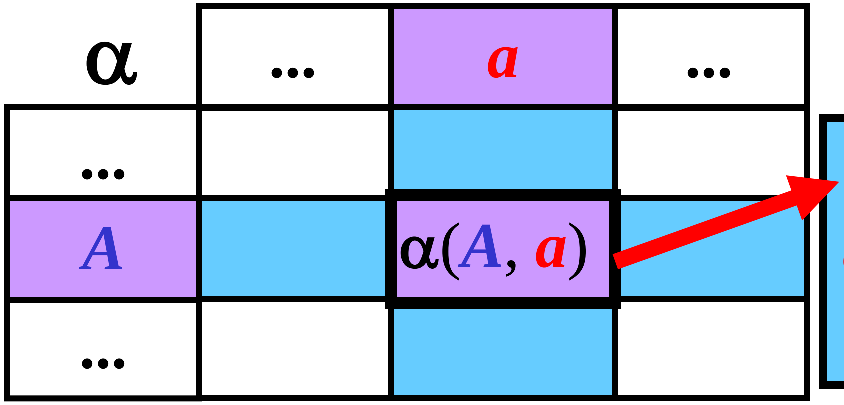
$\alpha(A, a) = A \to X_1X_2\ldots X_n \in P$ if $a \in Predict(A \to X_1X_2\ldots X_n)$; otherwise, $\alpha(A, a)$ is blank.

Task: LL table for G_{expr1}

	i	+	*	(	)	$
E						
E'						
T						
T'						
F						

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}

Construction of LL Table



α	…	*a*	…
…			
A		α(*A*, *a*)	
…			

$\alpha(A, a) = A \to X_1X_2\ldots X_n \in P$ if $a \in Predict(A \to X_1X_2\ldots X_n)$; otherwise, $\alpha(A, a)$ is blank.

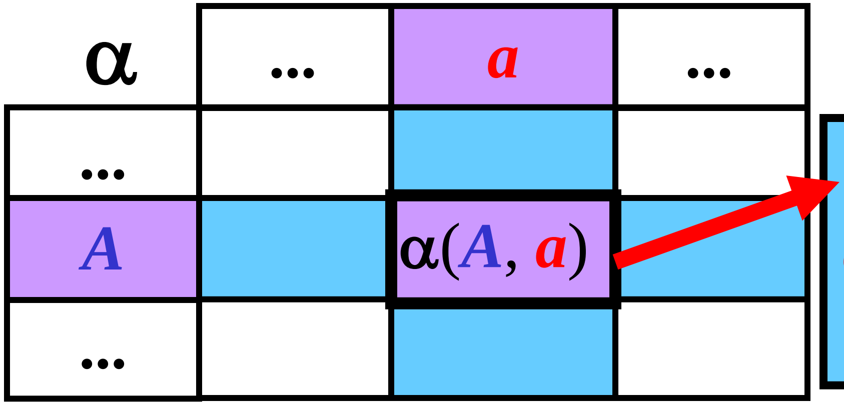
Task: LL table for G_{expr1}

	i	+	*	(	)	$
E	1					
E'						
T						
T'						
F						

$i \in Predict(1)$

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*}

Construction of LL Table



α	…	*a*	…
…			
A		α(*A*, *a*)	
…			

$\alpha(A, a) = A \to X_1X_2\ldots X_n \in P$ if $a \in Predict(A \to X_1X_2\ldots X_n)$; otherwise, $\alpha(A, a)$ is blank.

Task: LL table for G_{expr1}

	i	+	*	(	)	$
E	1					
E'						
T	4					
T'						
F						

$i \in Predict(1)$

$i \in Predict(4)$

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*}

Construction of LL Table

α	…	a	…
…			
A		α(A, a)	
…			

α(A, a) = $A \rightarrow X_1X_2\ldots X_n \in P$ if $a \in Predict(A \rightarrow X_1X_2\ldots X_n)$; otherwise, α($A$, a) is blank.

Task: LL table for G_{expr1}

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

$i \in Predict(1)$

$i \in Predict(4)$

$i \in Predict(8)$

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

Construction of LL Table

α	…	*a*	…
…			
A		α(*A*, *a*)	
…			

$\alpha(A, a) = A \to X_1X_2\dots X_n \in P$ if $a \in Predict(A \to X_1X_2\dots X_n)$; otherwise, $\alpha(A, a)$ is blank.

Task: LL table for G_{expr1}

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

i ∈ ***Predict***(1)

i ∈ ***Predict***(4)

i ∈ ***Predict***(8)

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*}

Construct the rest analogically.

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: $E \rightarrow TE'$
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: $F \rightarrow i$

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

E

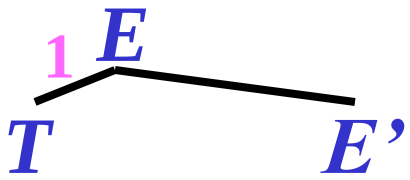
i $*$ i $\boxed{\$}$

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: $E \rightarrow TE'$
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: $F \rightarrow i$

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



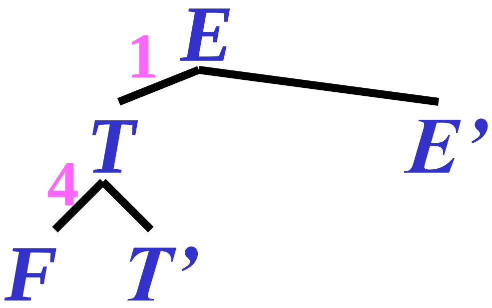
i * i $\boxed{\$}$

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: $E \rightarrow TE'$
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: $F \rightarrow i$

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



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: $E \to TE'$

2: $E' \to +TE'$

3: $E' \to \varepsilon$

4: $T \to FT'$

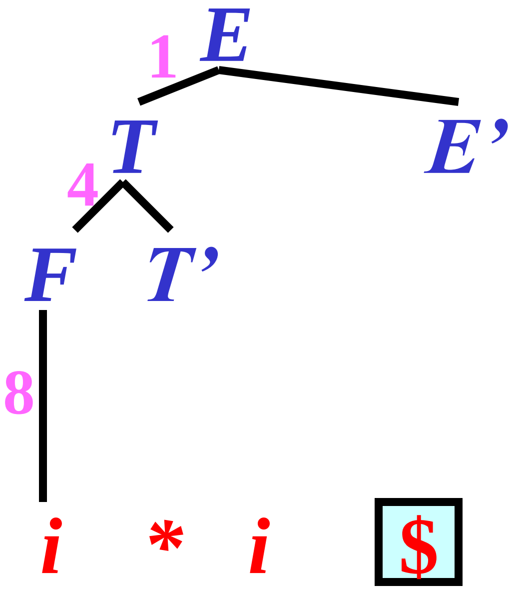
5: $T' \to *FT'$

6: $T' \to \varepsilon$

7: $F \to (E)$

8: $F \to i$

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

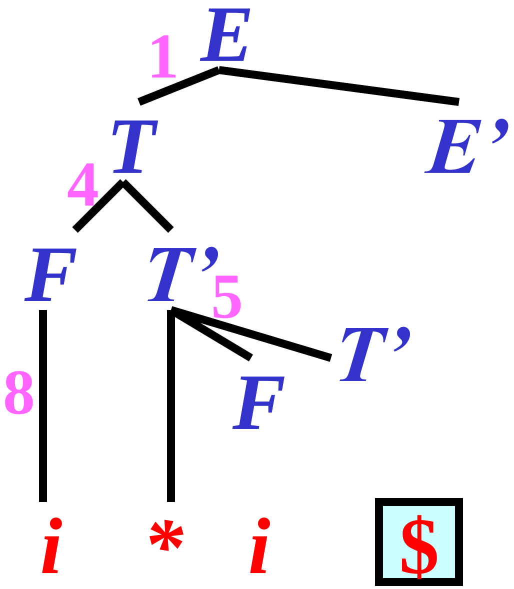


Parsing Based on LL Table: Example

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

1: *E* → *TE’*
2: *E’* → +*TE’*
3: *E’* → ε
4: *T* → *FT’*
5: *T’* → **FT’*
6: *T’* → ε
7: *F* → (*E*)
8: *F* → *i*

Question: ***i*** ***** ***i*** ∈ $L(G_{expr3})$**?**

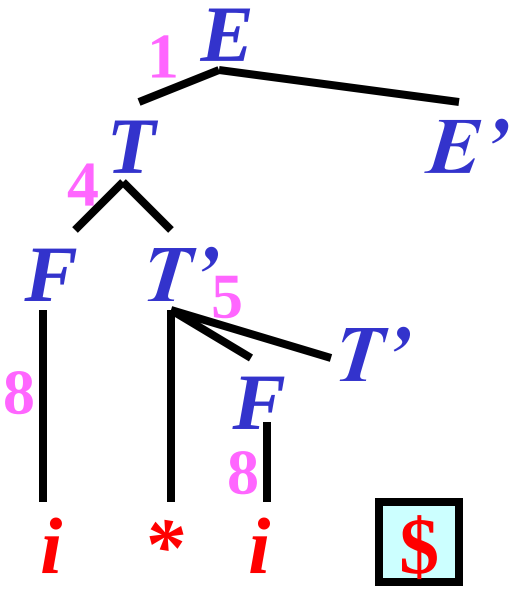


Parsing Based on LL Table: Example

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

1: $E \rightarrow TE'$
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: $F \rightarrow i$

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

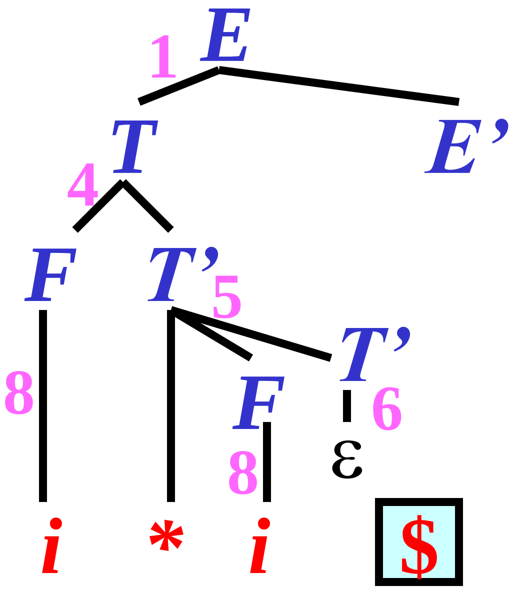


Parsing Based on LL Table: Example

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

1: *E* → *TE’*
2: *E’* → +*TE’*
3: *E’* → ε
4: *T* → *FT’*
5: *T’* → **FT’*
6: *T’* → ε
7: *F* → (*E*)
8: *F* → *i*

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



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* → *TE'*

2: *E'* → +*TE'*

3: *E'* → ε

4: *T* → *FT'*

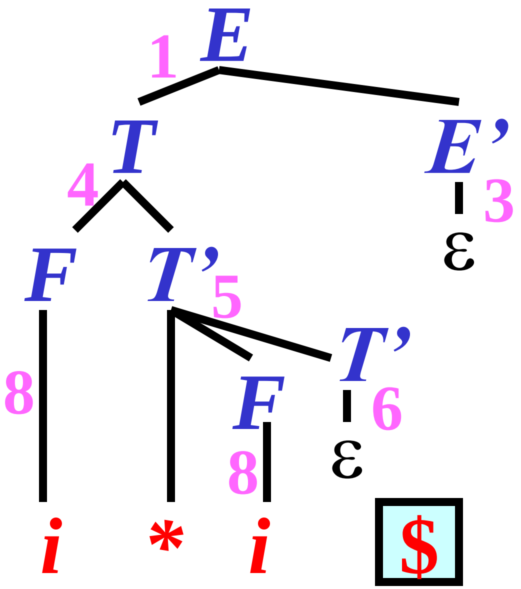
5: *T'* → **FT'*

6: *T'* → ε

7: *F* → (*E*)

8: *F* → *i*

Question: ***i*** *** ***i*** ∈ L(G_{expr3})**?**



LL Grammars with ε-rules: Definition

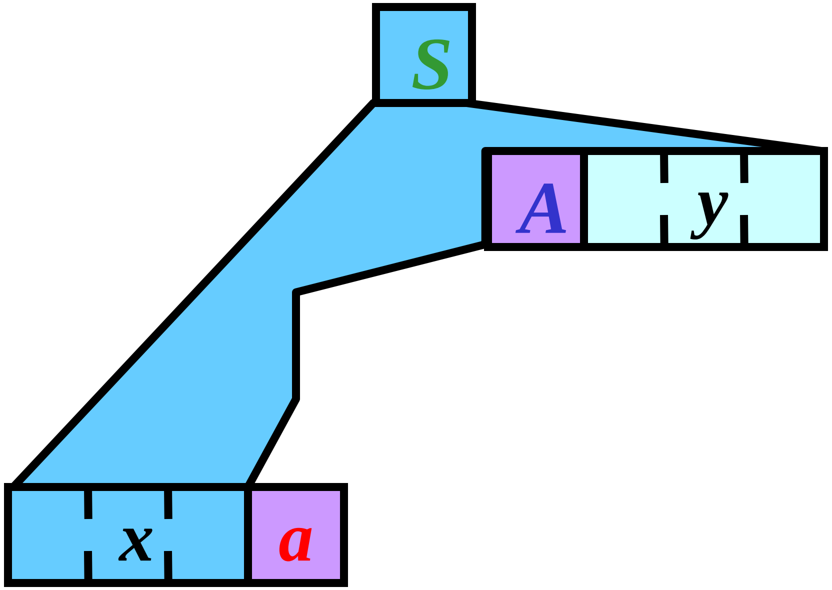
Definition: Let $G = (N, T, P, S)$ be a CFG. G is an *LL grammar* if for every $a \in T$ and every $A \in N$ there is **no more than one** A-rule $A \to X_1X_2\ldots X_n \in P$ such that $a \in Predict(A \to X_1X_2\ldots X_n)$

Illustration:

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 $A \to X_1X_2\ldots X_n \in P$ such that $a \in Predict(A \to X_1X_2\ldots X_n)$

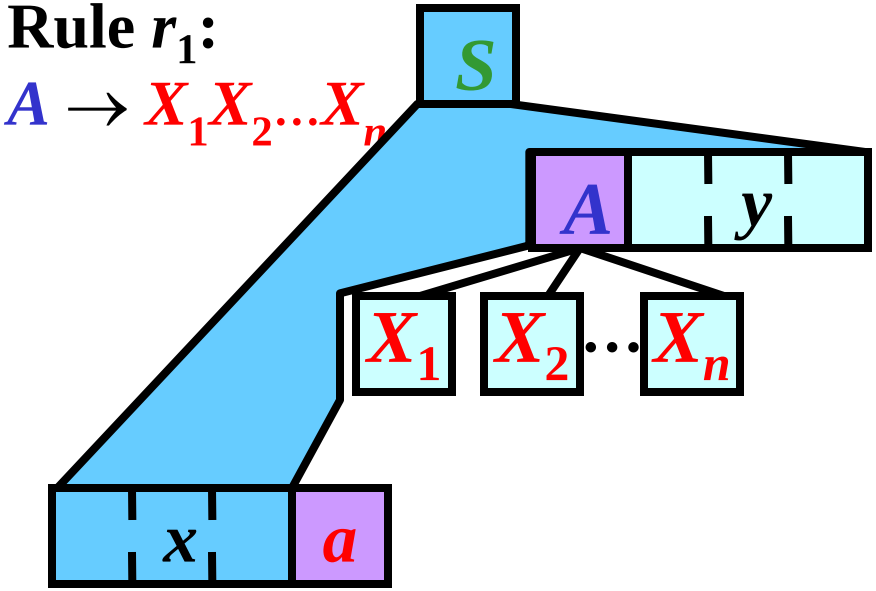
Illustration:



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 $A \rightarrow X_1X_2…X_n \in P$ such that $a \in Predict(A \rightarrow X_1X_2…X_n)$

Illustration:



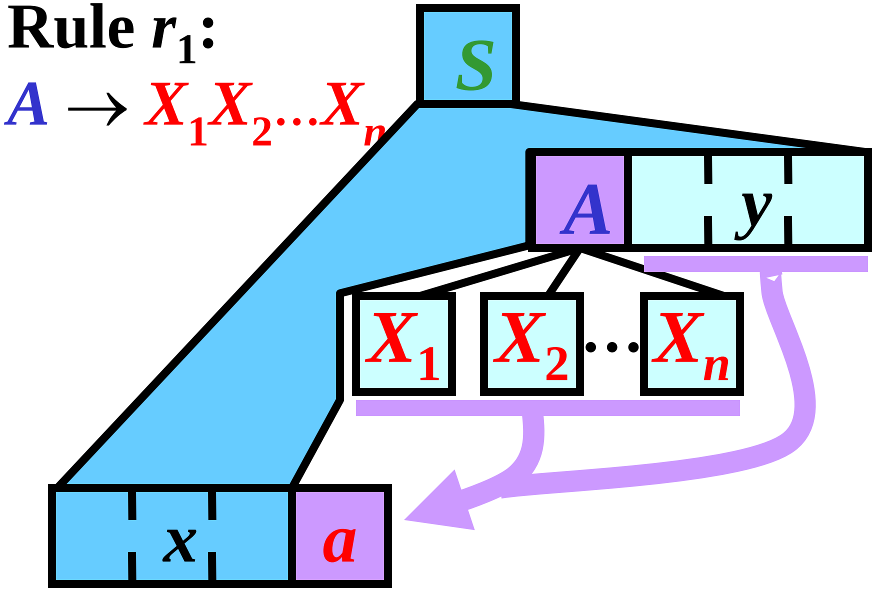
LL Grammars with ε-rules: Definition

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 $A \rightarrow X_1X_2...X_n \in P$ such that $a \in Predict(A \rightarrow X_1X_2...X_n)$

Illustration:

Rule r_1:

$A \rightarrow X_1X_2...X_n$



$a \in Predict(A \rightarrow X_1X_2...X_n)$

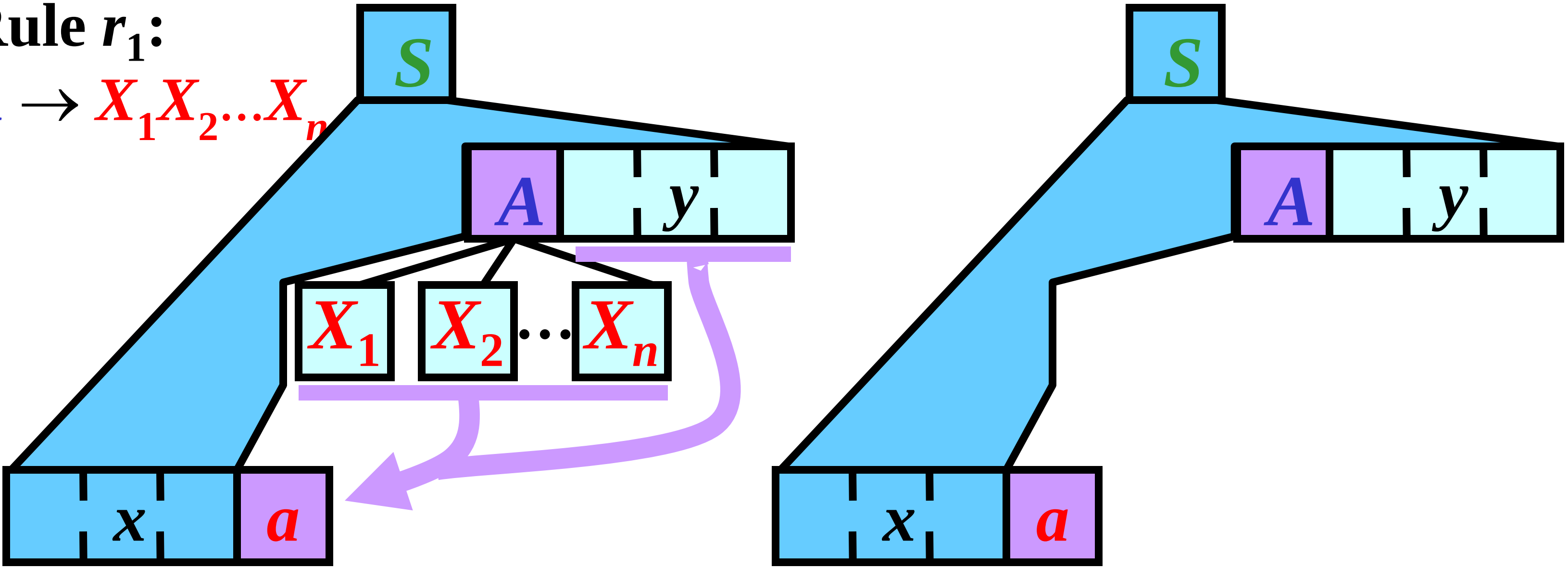
LL Grammars with ε-rules: Definition

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 $A \rightarrow X_1X_2\ldots X_n \in P$ such that $a \in Predict(A \rightarrow X_1X_2\ldots X_n)$

Illustration:

Rule r_1:

$A \rightarrow X_1X_2\ldots X_n$

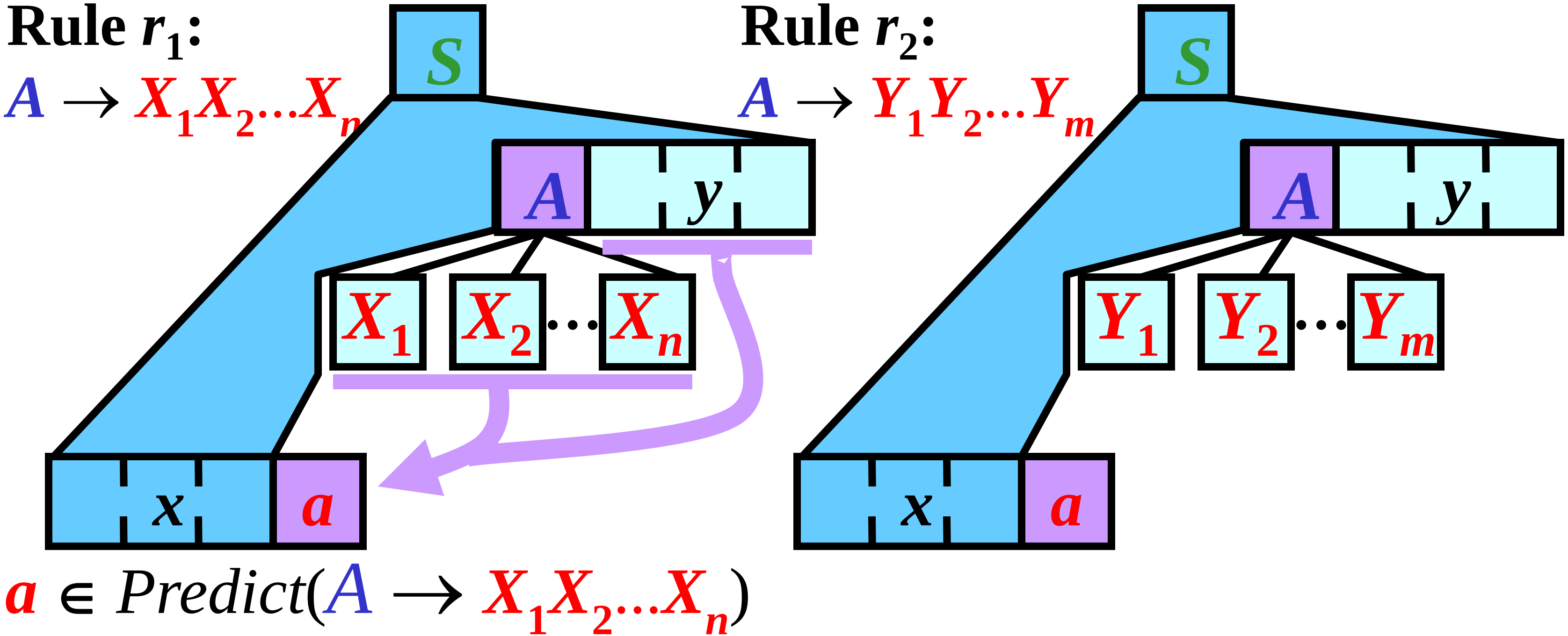


$a \in Predict(A \rightarrow X_1X_2\ldots X_n)$

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 $A \rightarrow X_1X_2…X_n \in P$ such that $a \in Predict(A \rightarrow X_1X_2…X_n)$

Illustration:



$a \in Predict(A \rightarrow X_1X_2…X_n)$

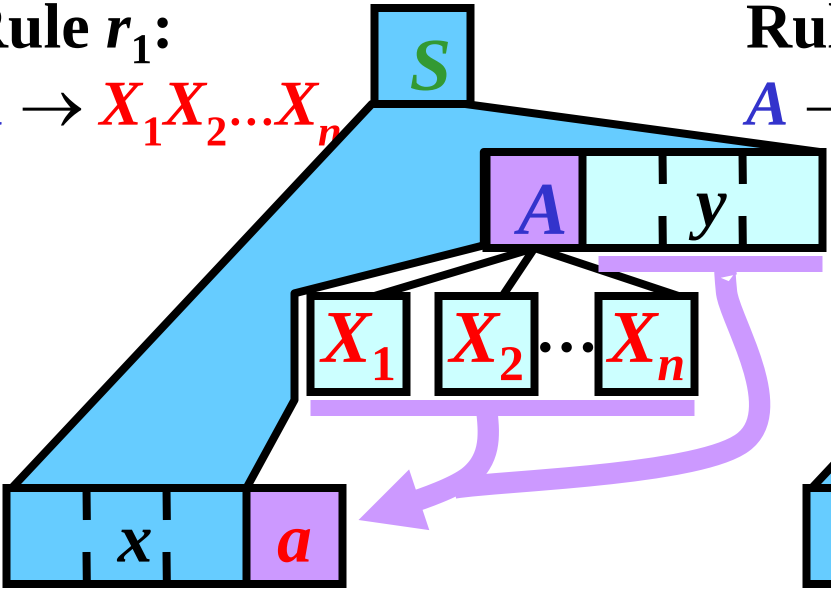
LL Grammars with ε-rules: Definition

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 $A \to X_1X_2...X_n \in P$ such that $a \in Predict(A \to X_1X_2...X_n)$

Illustration:

Rule r_1:

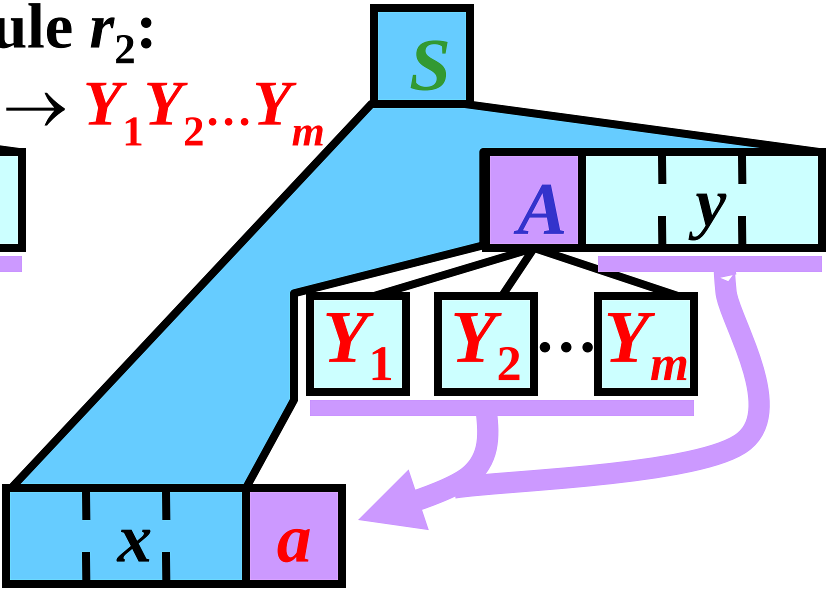
$A \to X_1X_2...X_n$



$a \in Predict(A \to X_1X_2...X_n)$

Rule r_2:

$A \to Y_1Y_2...Y_m$



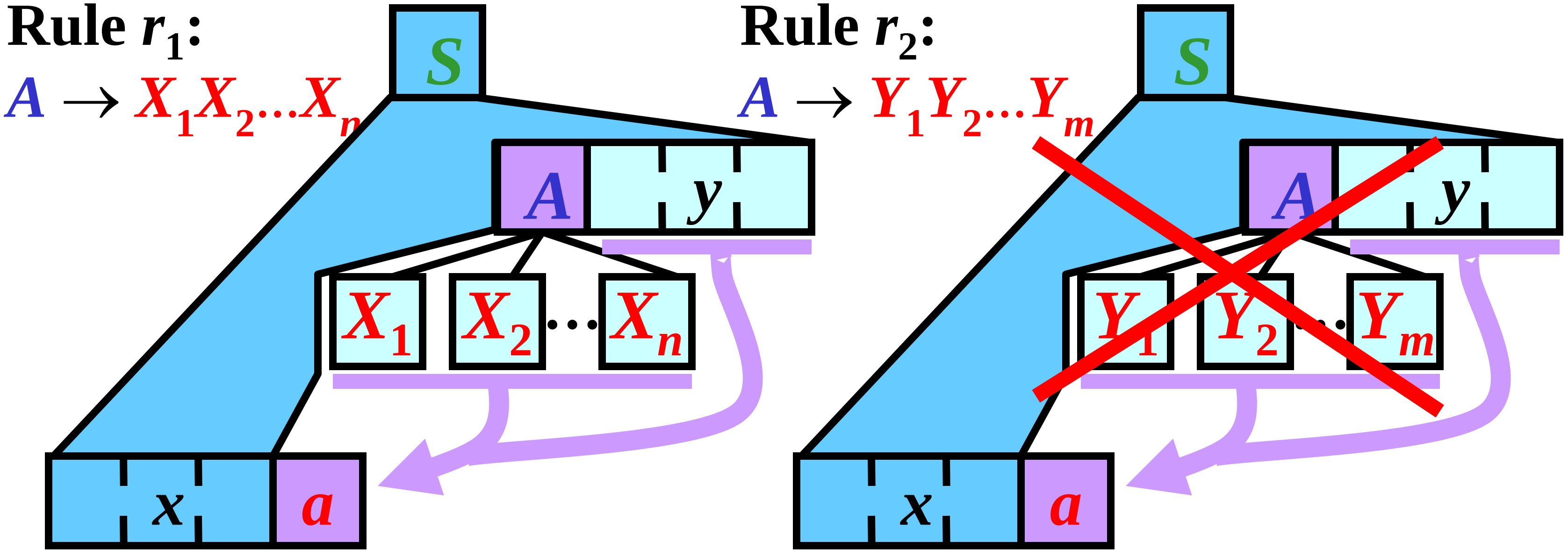
$a \in Predict(A \to Y_1Y_2...Y_m)$

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 $A \to X_1X_2\ldots X_n \in P$ such that $a \in Predict(A \to X_1X_2\ldots X_n)$

Illustration:

Ruled out in an LL grammar



Rule r_1:

$A \to X_1X_2\ldots X_n$

Rule r_2:

$A \to Y_1Y_2\ldots Y_m$

$a \in Predict(A \to X_1X_2\ldots X_n)$

$a \in Predict(A \to Y_1Y_2\ldots Y_m)$

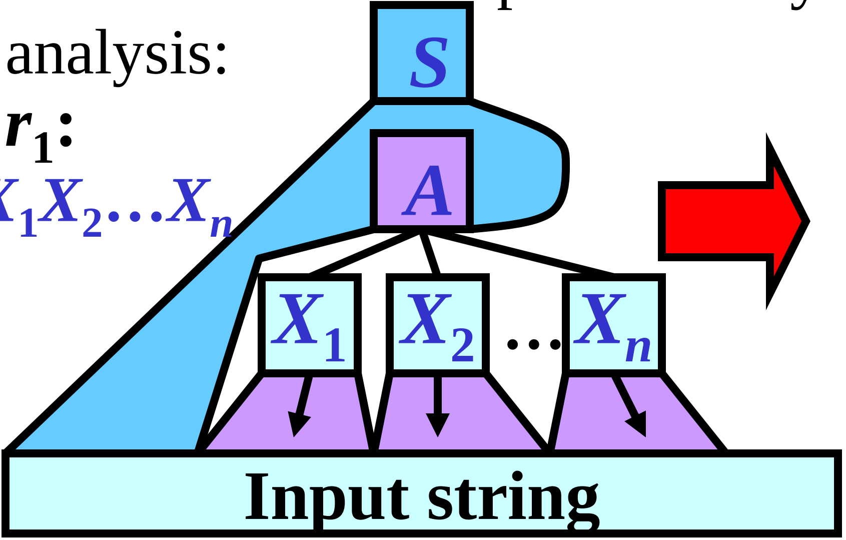
LL Analyzer Implementation

1) Recursive-Descent Parsing

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

Rule r_1:

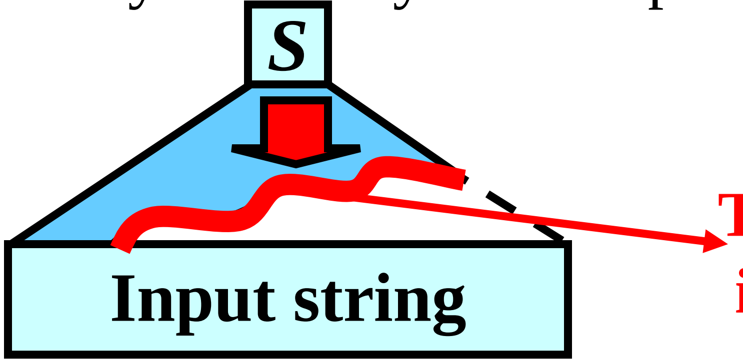
$A \rightarrow X_1X_2\ldots X_n$



```
function A: boolean;
begin
  {$X_1$ analysis}
  {$X_2$ analysis}
        …
  {$X_n$ analysis}
end
```

2) Predictive Parsing

- Table-driven syntax analyzer with pushdown



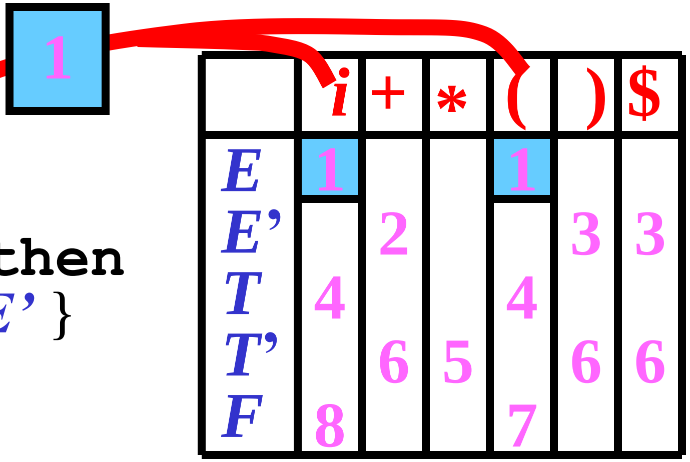
These symbols are in the pushdown.

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: $E \to TE'$

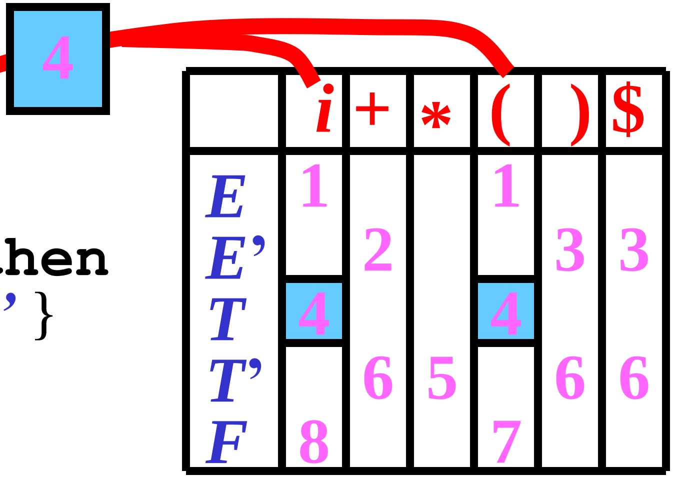
```
function E: boolean;
begin
   E := false;
   if token in ['i', '('] then
         { simulation of rule 1: E → TE' }
         E := T and E1;
end;
```



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

- For $T \in N$: Rule 4: $T \to FT'$

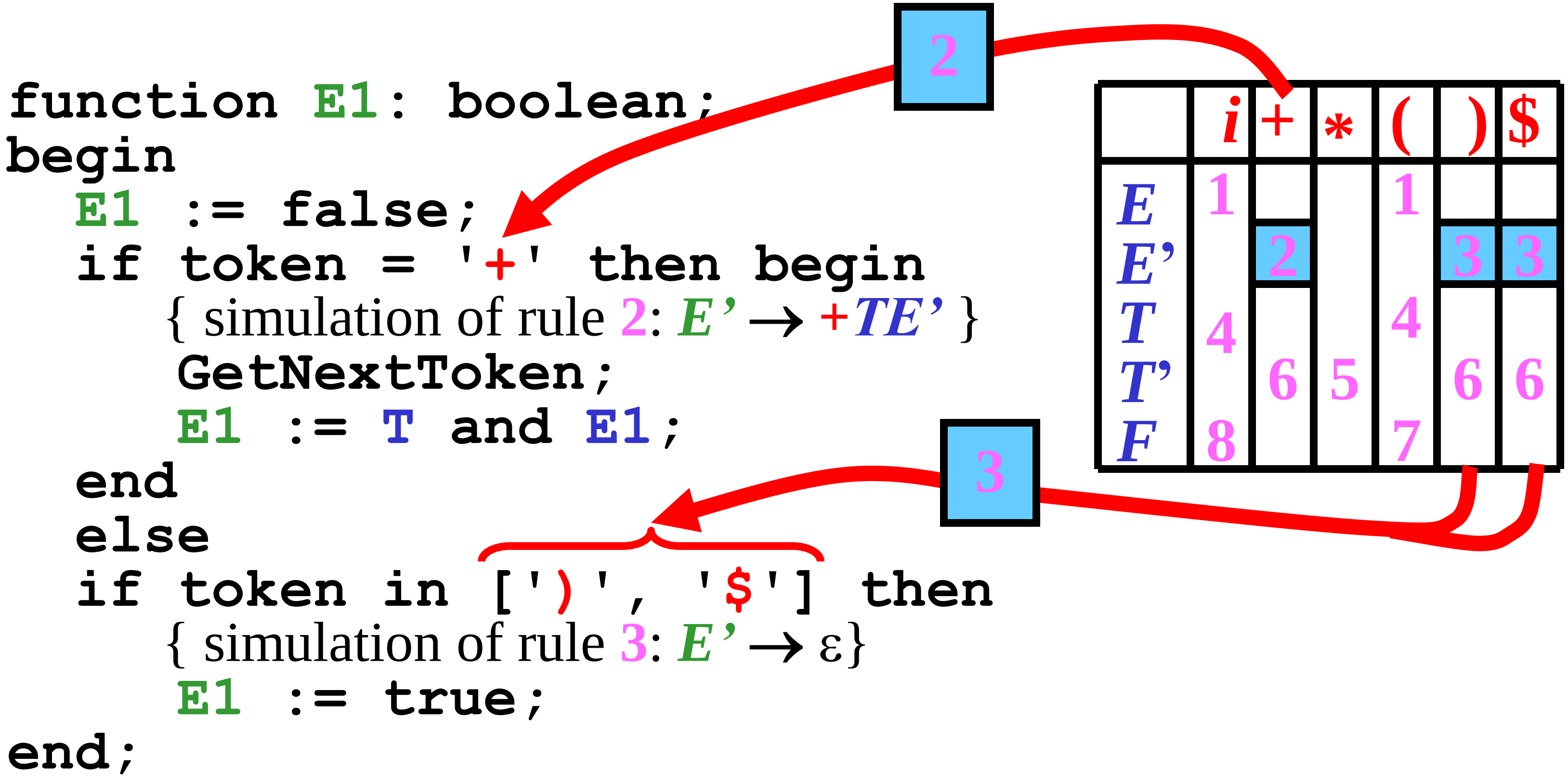
```
function T: boolean;
begin
   T := false;
   if token in ['i', '('] then
         { simulation of rule 4: T → FT' }
         T := F and T1;
end;
```



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

Recursive Descent: Example 2/4

- For *E'* ∈ *N*: Rules 2: $E' \to +TE'$, 3: $E' \to \varepsilon$



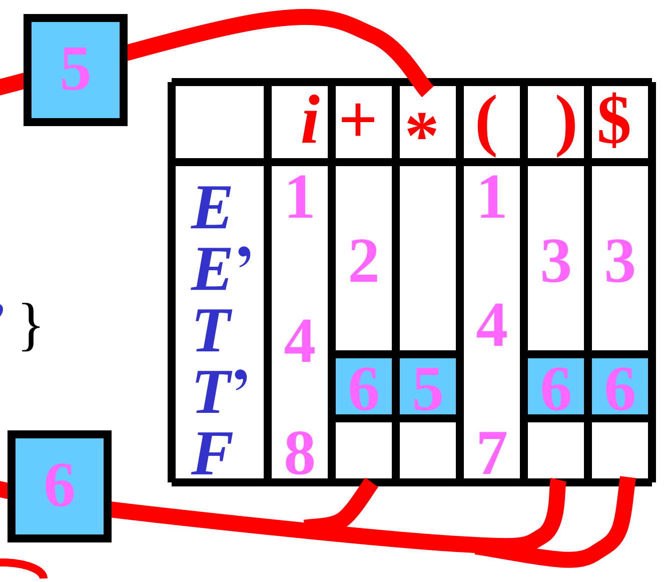
```
function E1: boolean;
begin
   E1 := false;
   if token = '+' then begin
      { simulation of rule 2: E' → +TE' }
      GetNextToken;
      E1 := T and E1;
   end
   else
   if token in [')', '$'] then
      { simulation of rule 3: E' → ε}
      E1 := true;
end;
```

	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'* ∈ *N*: Rules 5: *T'* → **FT'*, 6: *T'* → ε

```
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;
```



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

Recursive Descent: Example 4/4

- For *F* ∈ *N*: Rules 7: *F* → (*E*) , 8: *F* → *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:

Input string:

i * i $

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

Start: **GetNextToken;**
***Call E*;**

Input string:

_i_ * _i_ $

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

Start: **GetNextToken**;
Call E;

***E*:**

For token = *i*:
Call T*, *Call E1

Input string:

***i* * *i* $**

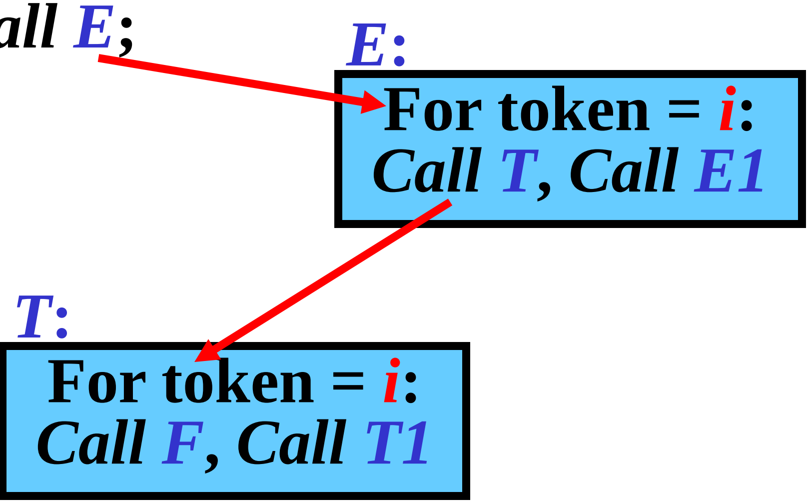
Recursive Descent: Illustration for *i***i*$

Start: **GetNextToken**;

***Call E*;**

Input string:

***i* * *i* $**



***E*:**

For token = *i*:

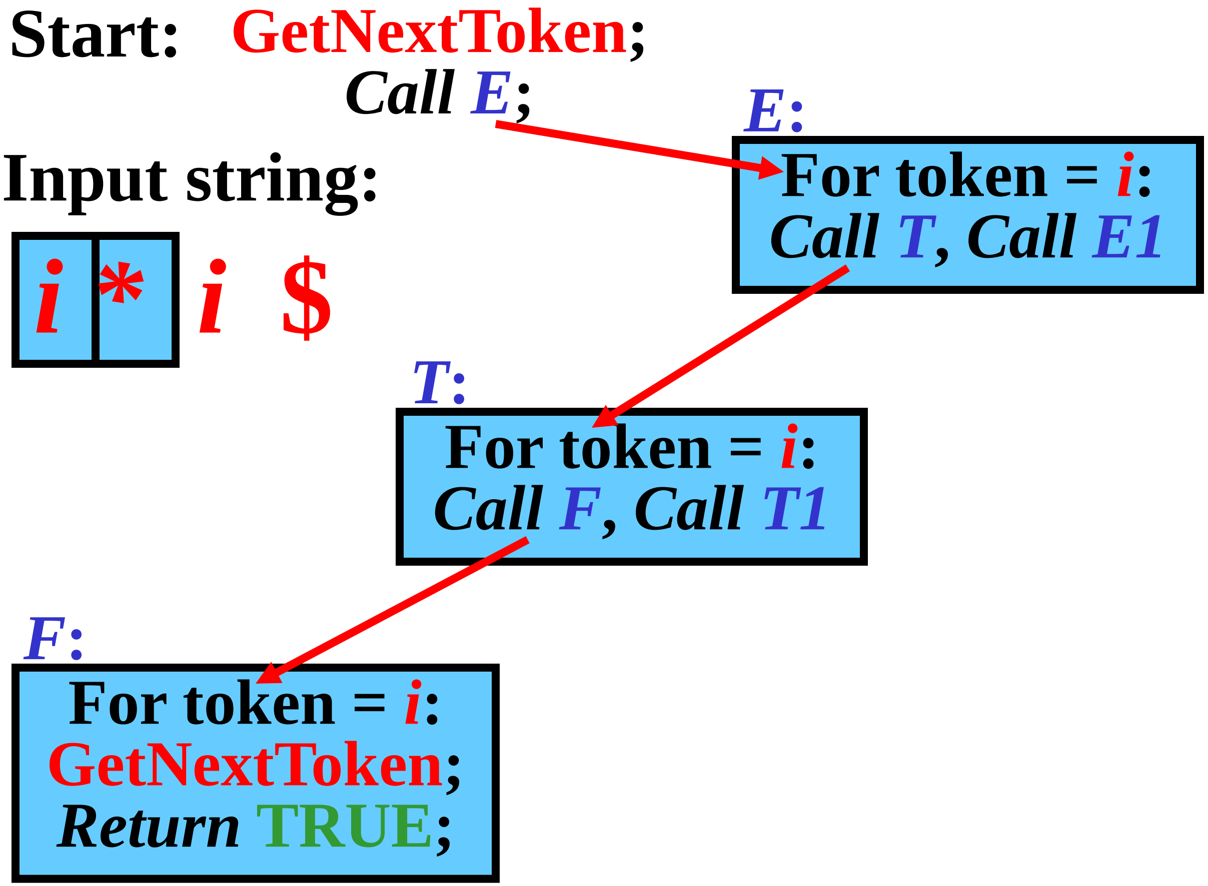
Call T*, *Call E1

***T*:**

For token = *i*:

Call F*, *Call T1

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



Start: GetNextToken;

***Call E*;**

Input string:

***i* * *i* $**

***E*:**

For token = *i*:
Call T*, *Call E1

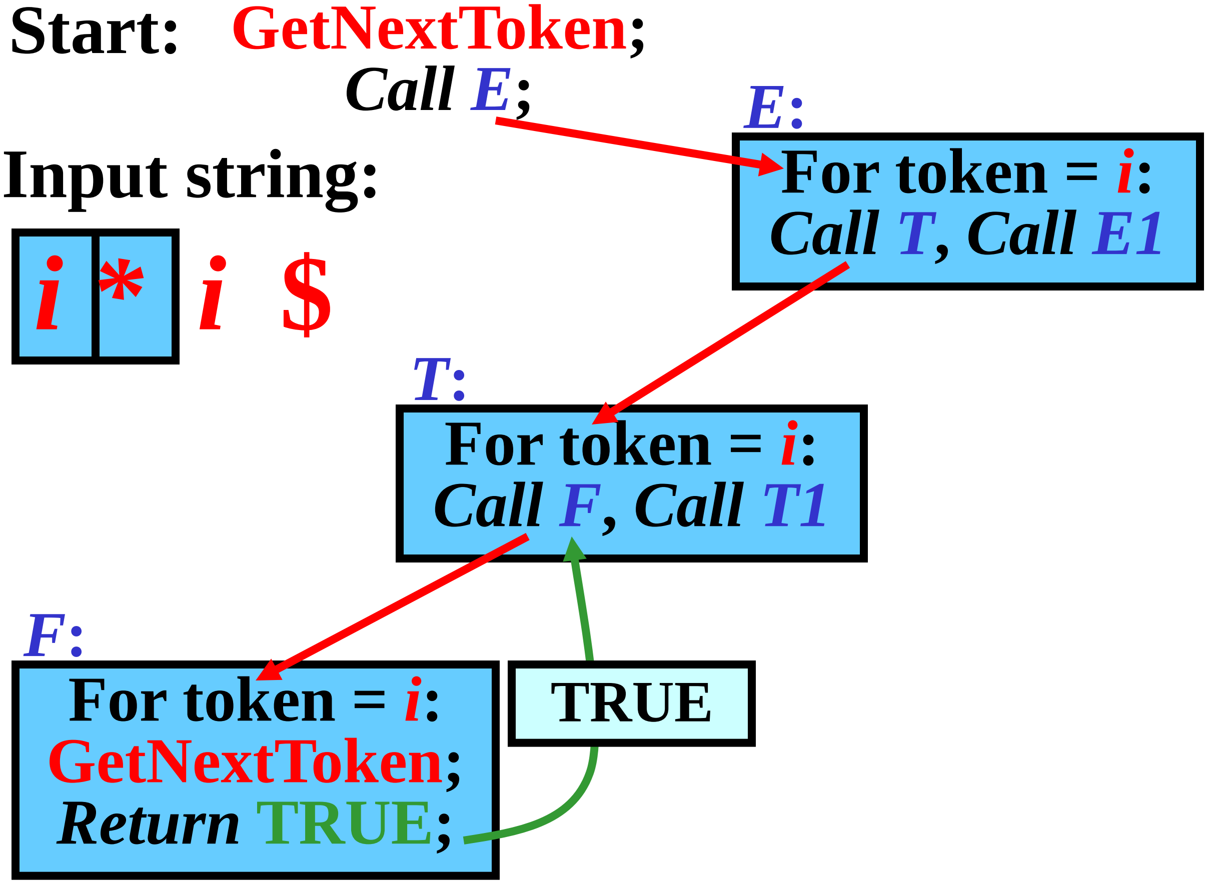
***T*:**

For token = *i*:
Call F*, *Call T1

***F*:**

For token = *i*:
GetNextToken;
***Return* TRUE;**

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



Start: **GetNextToken**;
Call E;

Input string:

i * *i* $

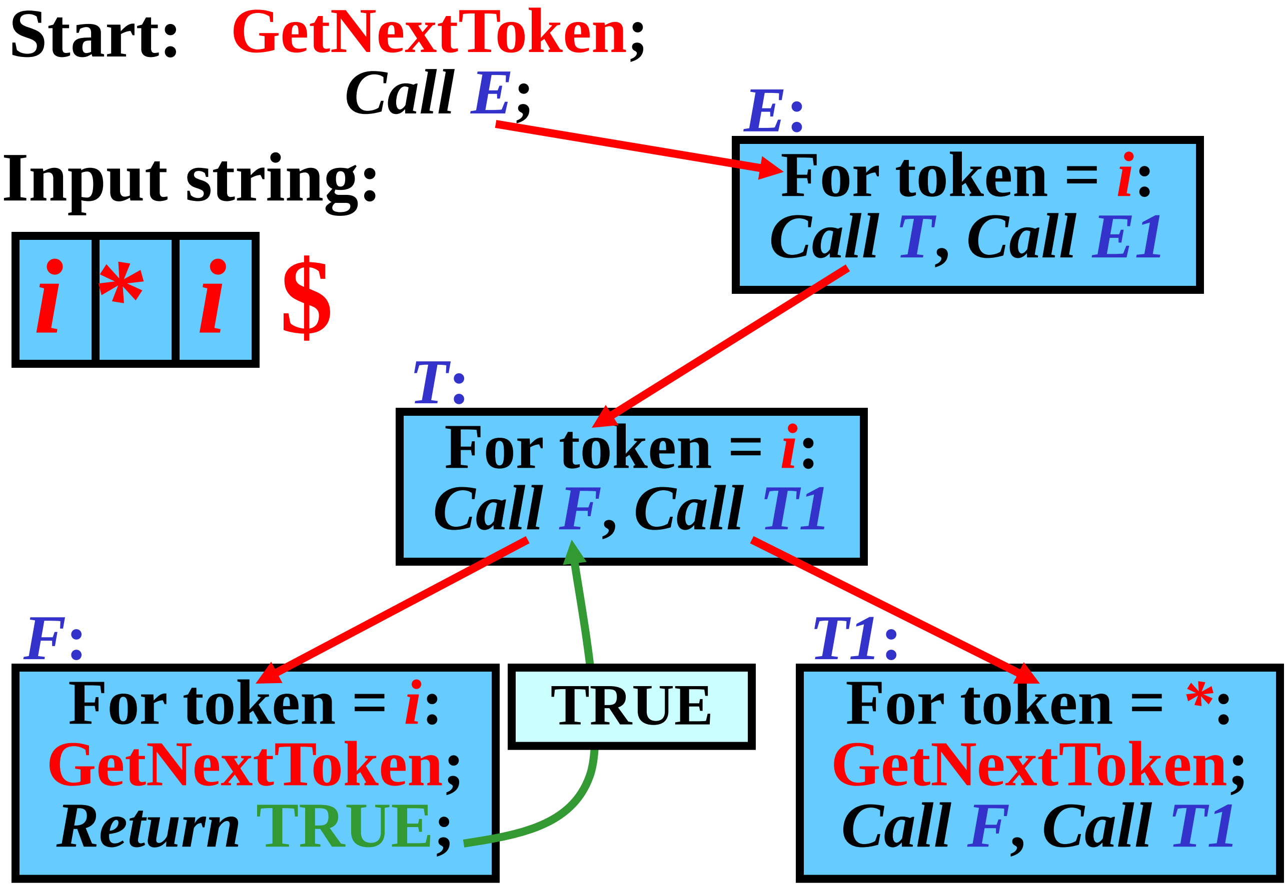
E:
For token = *i*:
Call T, *Call E1*

T:
For token = *i*:
Call F, *Call T1*

F:
For token = *i*:
GetNextToken;
Return **TRUE**;

TRUE

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



Start: **GetNextToken;**
Call E;

Input string:

i	*	*i*	$

***E*:**
For token = *i*:
Call T, *Call E1*

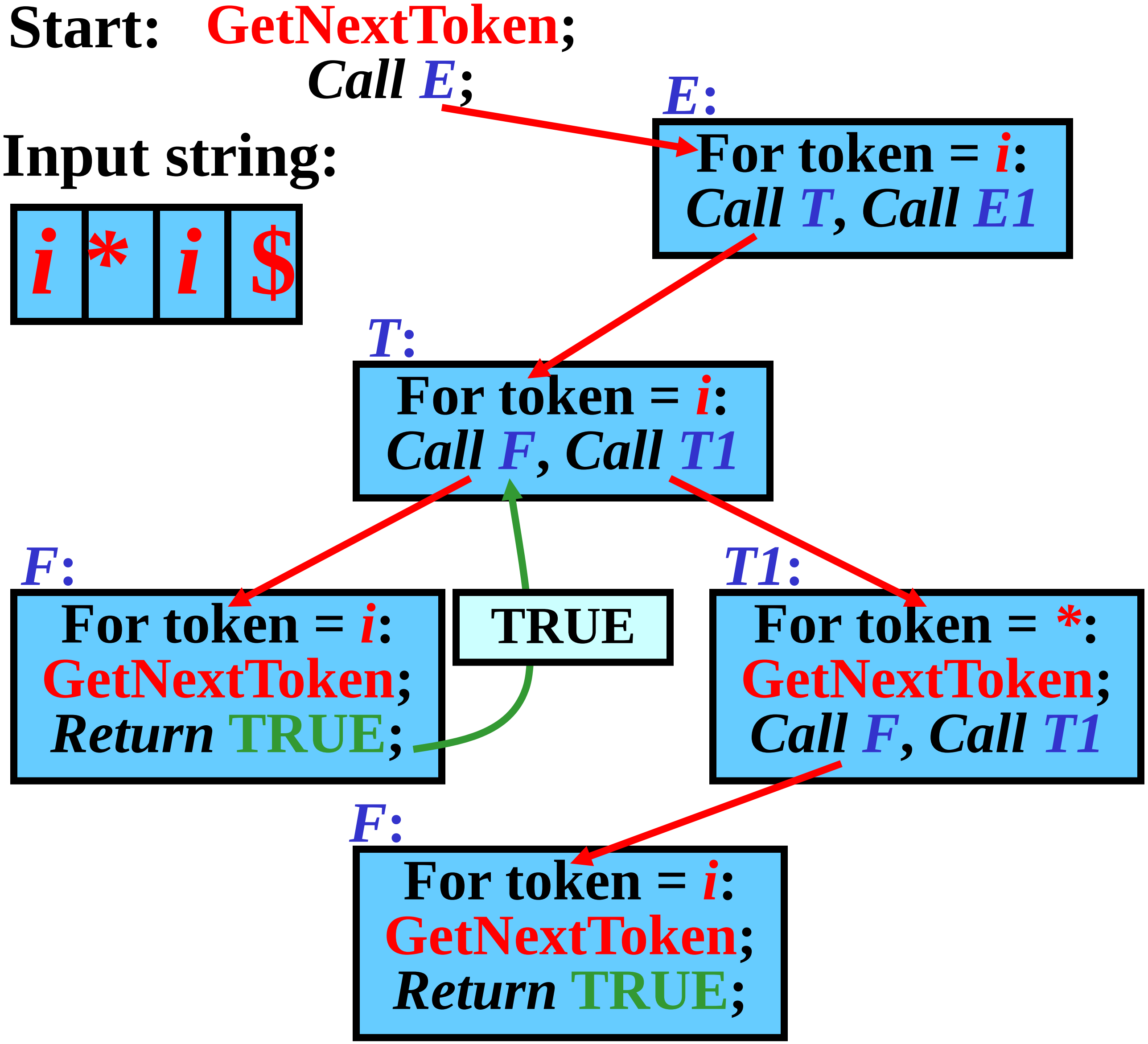
***T*:**
For token = *i*:
Call F, *Call T1*

***F*:**
For token = *i*:
GetNextToken;
***Return* TRUE;**

TRUE

***T1*:**
For token = *:
GetNextToken;
Call F, *Call T1*

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



Start: **GetNextToken;**
Call E;

Input string:

i	*****	***i***	**$**

E:
For token = *i*:
Call T, Call E1

T:
For token = *i*:
Call F, Call T1

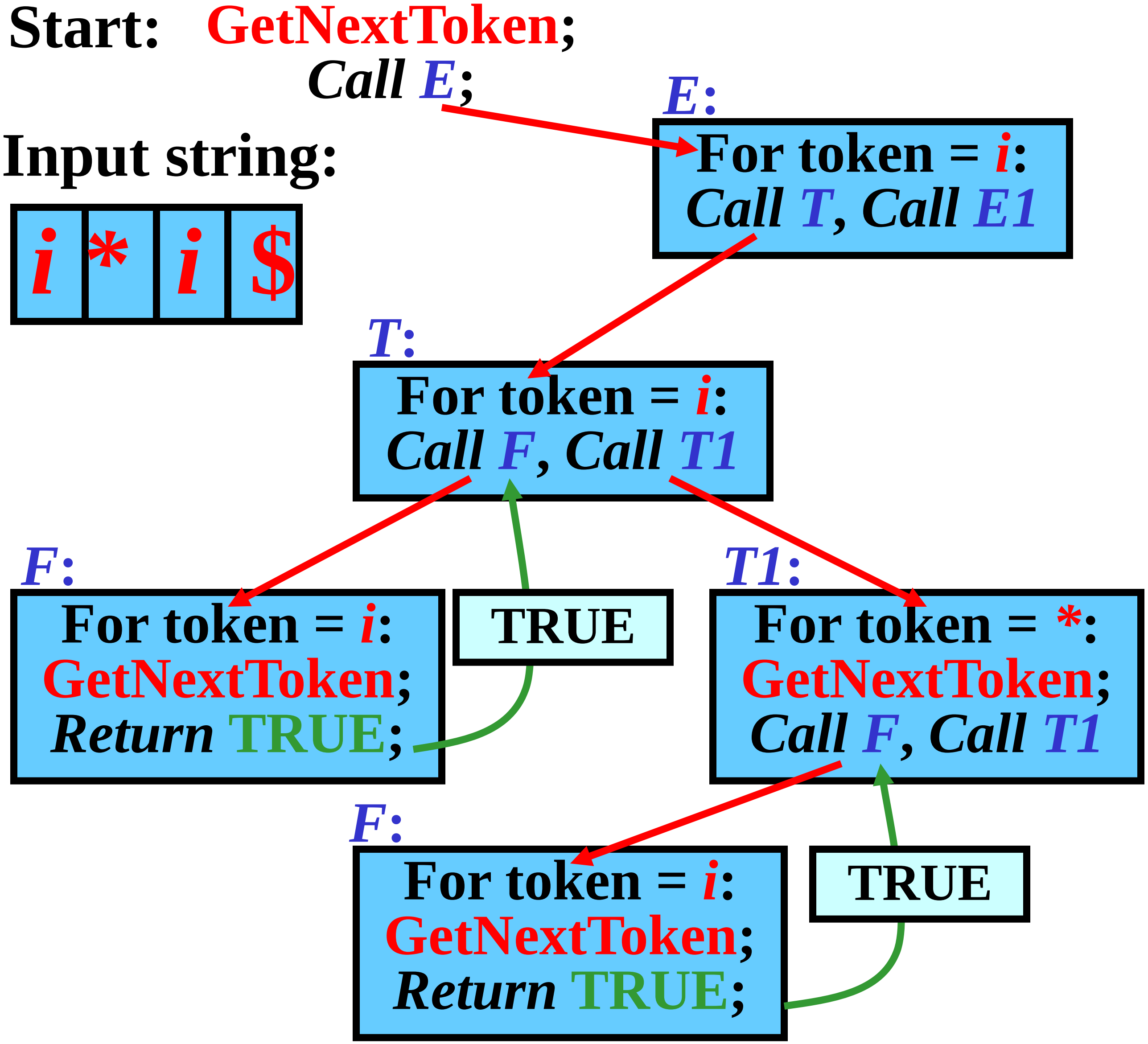
F:
For token = *i*:
GetNextToken;
Return **TRUE;**

TRUE

T1:
For token = *:
GetNextToken;
Call F, Call T1

F:
For token = *i*:
GetNextToken;
Return **TRUE;**

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



Start: **GetNextToken;**
Call E;

Input string:

i	*****	***i***	**$**

***E*:**
For token = *i*:
Call T, Call E1

***T*:**
For token = *i*:
Call F, Call T1

***F*:**
For token = *i*:
GetNextToken;
***Return* TRUE;**

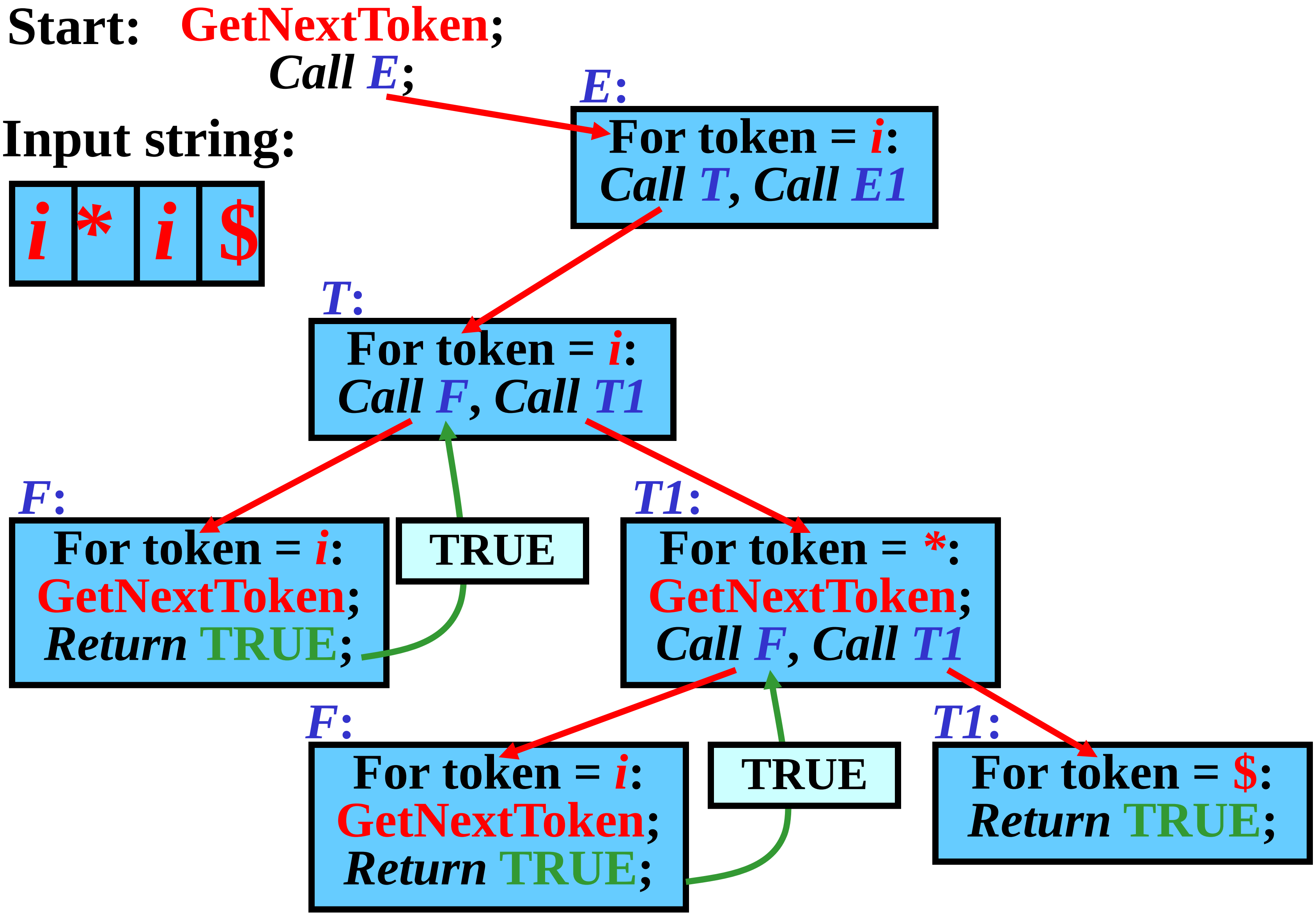
TRUE

***T1*:**
For token = *:
GetNextToken;
Call F, Call T1

***F*:**
For token = *i*:
GetNextToken;
***Return* TRUE;**

TRUE

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



Start: **GetNextToken;**
Call E;

Input string:

i	*****	***i***	**$**

***E*:**
For token = *i*:
Call T, Call E1

***T*:**
For token = *i*:
Call F, Call T1

***F*:**
For token = *i*:
GetNextToken;
***Return* TRUE;**

TRUE

***T1*:**
For token = *:
GetNextToken;
Call F, Call T1

***F*:**
For token = *i*:
GetNextToken;
***Return* TRUE;**

TRUE

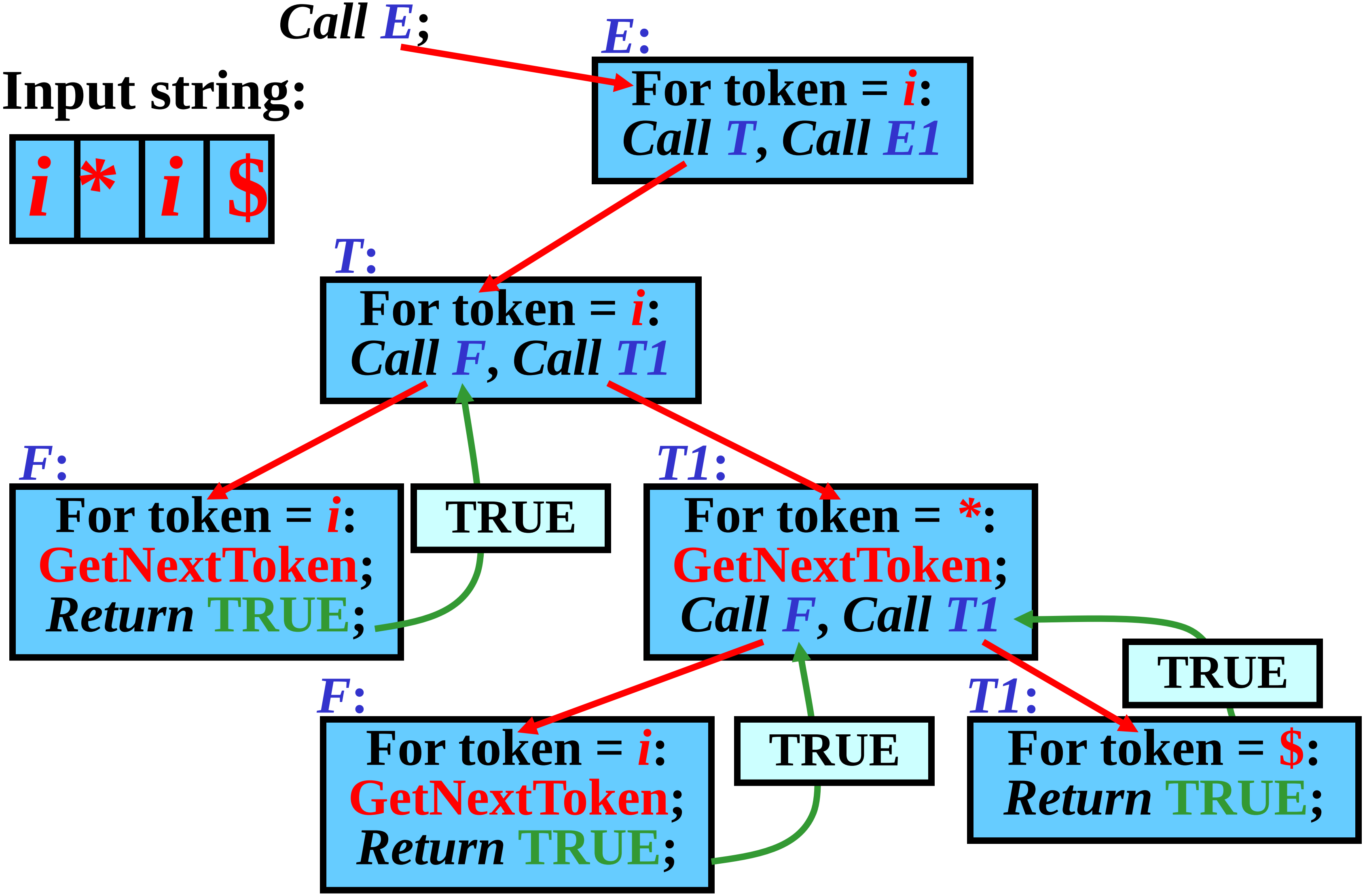
***T1*:**
For token = $:
***Return* TRUE;**

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

Start: **GetNextToken;**
Call E;

Input string:

i	*	*i*	$



E:
For token = *i*:
Call T, *Call E1*

T:
For token = *i*:
Call F, *Call T1*

F:
For token = *i*:
GetNextToken;
***Return* TRUE;**

TRUE

T1:
For token = *:
GetNextToken;
Call F, *Call T1*

F:
For token = *i*:
GetNextToken;
***Return* TRUE;**

TRUE

T1:
For token = $:
***Return* TRUE;**

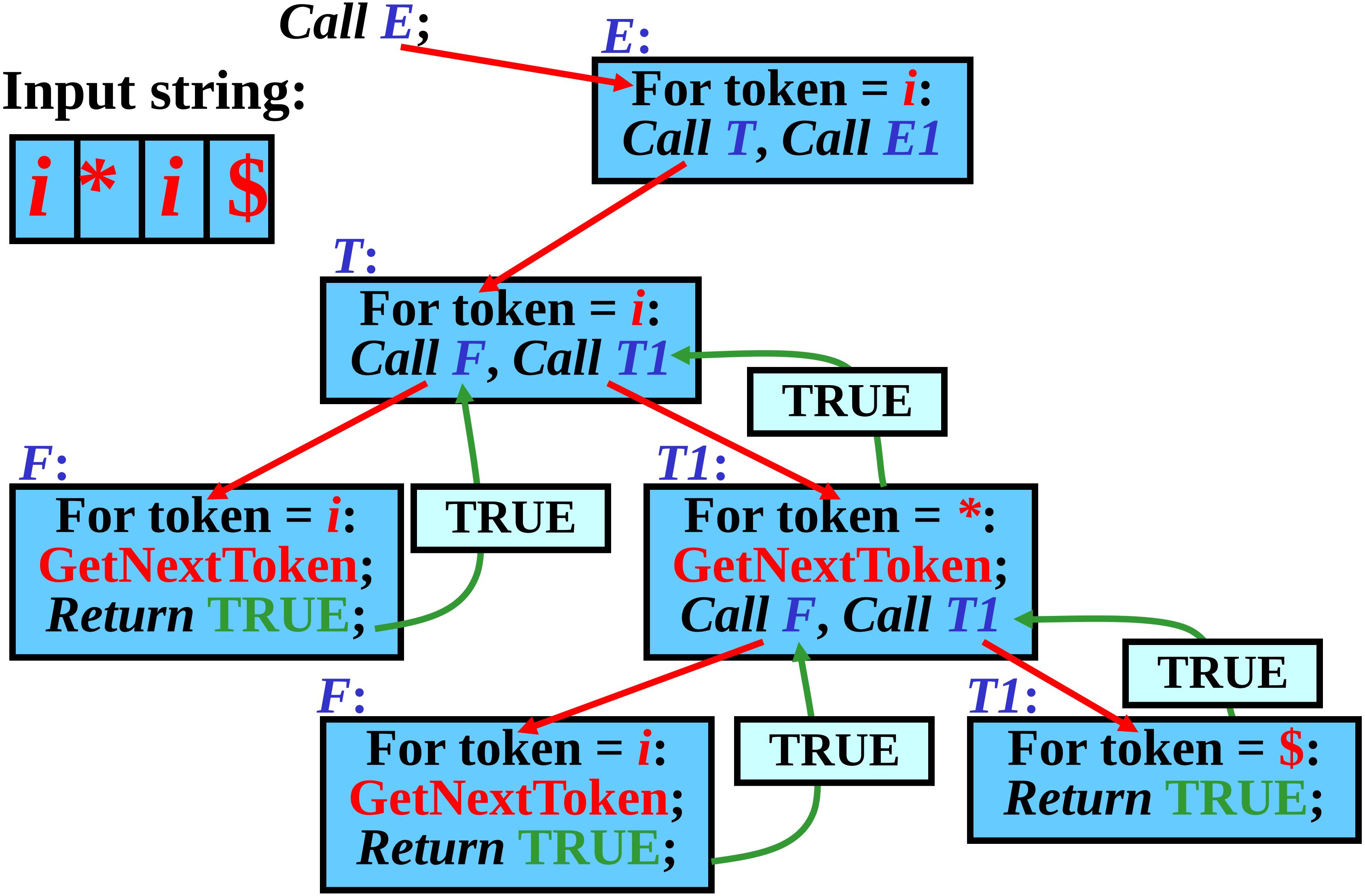
TRUE

Recursive Descent: Illustration for *i*i*$

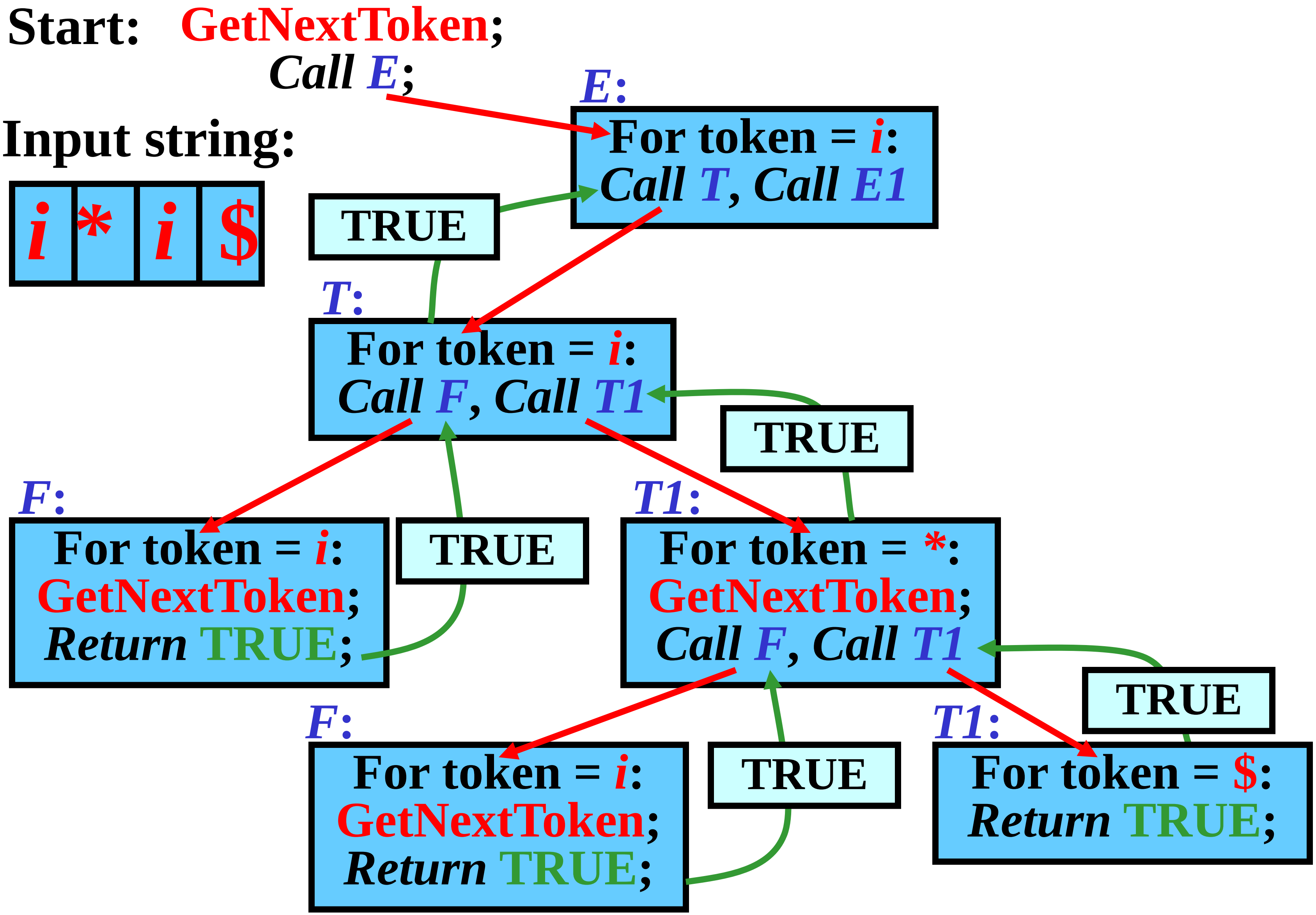
Start: **GetNextToken;**
Call E;

Input string:

i	*	*i*	$



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



Start: **GetNextToken;**
Call E;

Input string:

i	*****	***i***	**$**

E:
For token = *i*:
Call T, Call E1

TRUE

T:
For token = *i*:
Call F, Call T1

TRUE

F:
For token = *i*:
GetNextToken;
***Return* TRUE;**

TRUE

T1:
For token = *:
GetNextToken;
Call F, Call T1

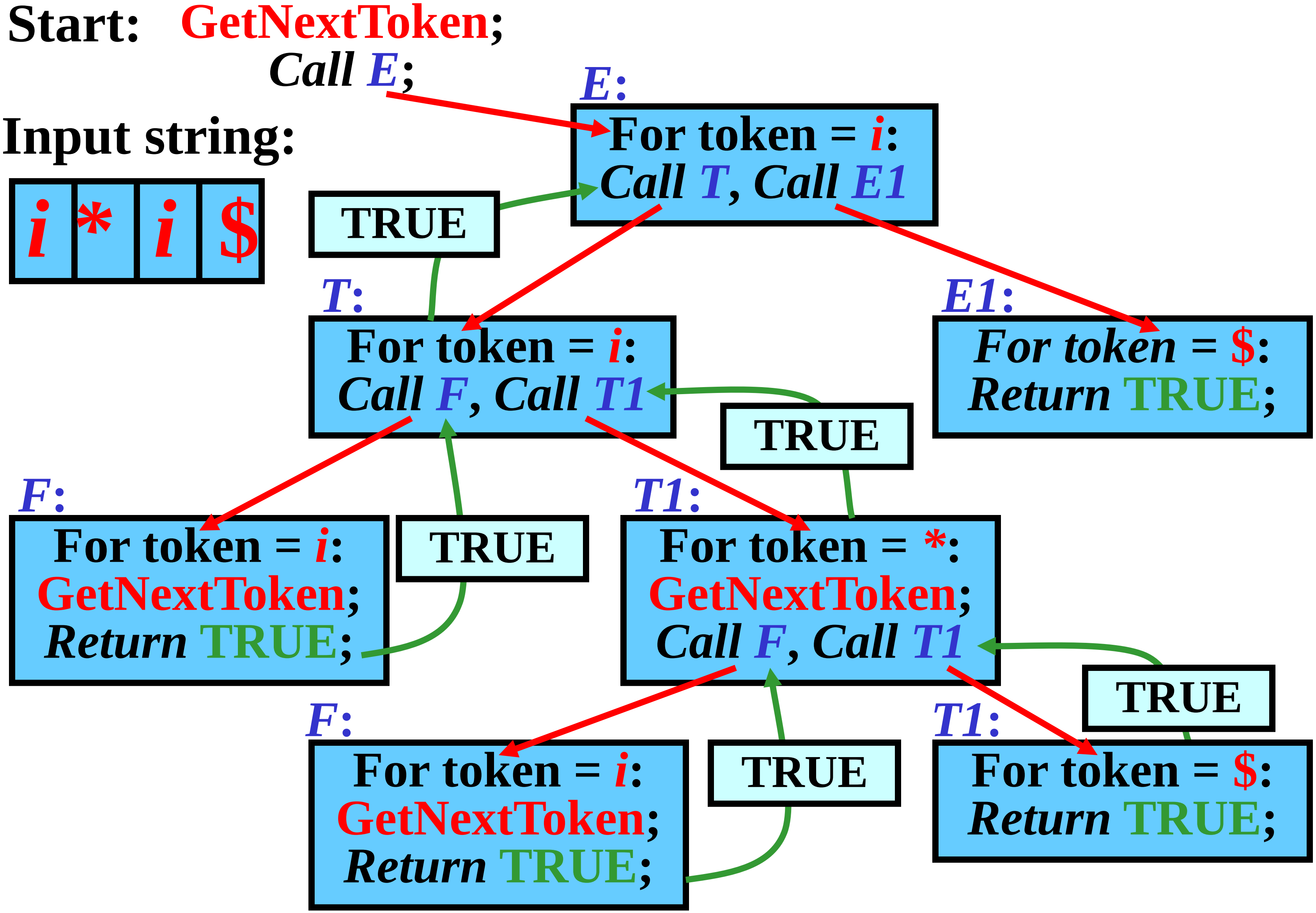
TRUE

F:
For token = *i*:
GetNextToken;
***Return* TRUE;**

TRUE

T1:
For token = $:
***Return* TRUE;**

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



Start: **GetNextToken;**
Call E;

Input string:

i	*	*i*	$

E:
For token = *i*:
Call T, Call E1

T:
For token = *i*:
Call F, Call T1

E1:
For token = $:
Return **TRUE;**

F:
For token = *i*:
GetNextToken;
Return **TRUE;**

T1:
For token = *:
GetNextToken;
Call F, Call T1

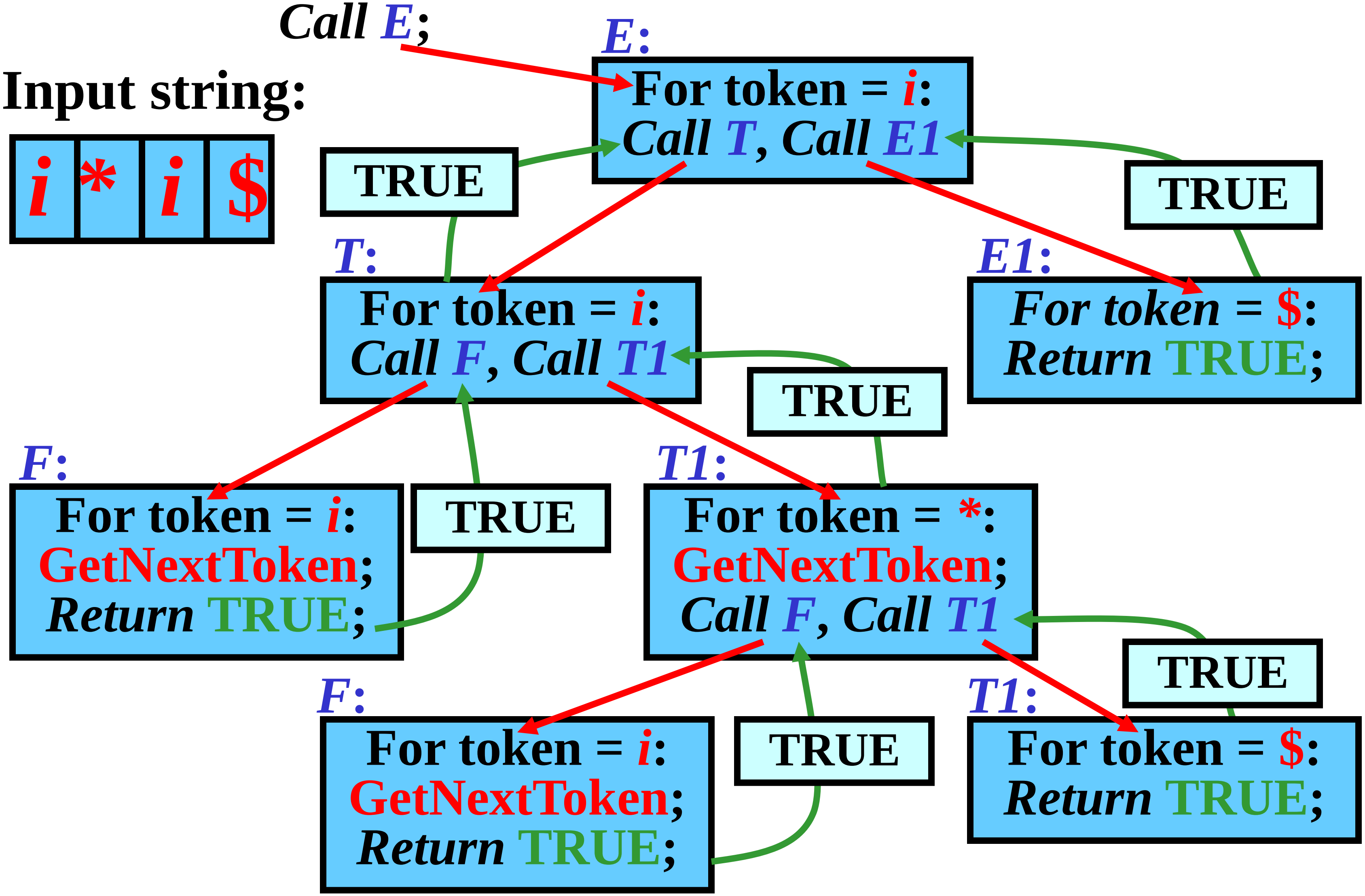
F:
For token = *i*:
GetNextToken;
Return **TRUE;**

T1:
For token = $:
Return **TRUE;**

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

Start: **GetNextToken;**
Call E;

Input string:



i	*	*i*	$

E:
For token = *i*:
Call T, Call E1

TRUE

T:
For token = *i*:
Call F, Call T1

TRUE

E1:
For token = $:
Return **TRUE**;

TRUE

F:
For token = *i*:
GetNextToken;
Return **TRUE**;

TRUE

T1:
For token = *:
GetNextToken;
Call F, Call T1

TRUE

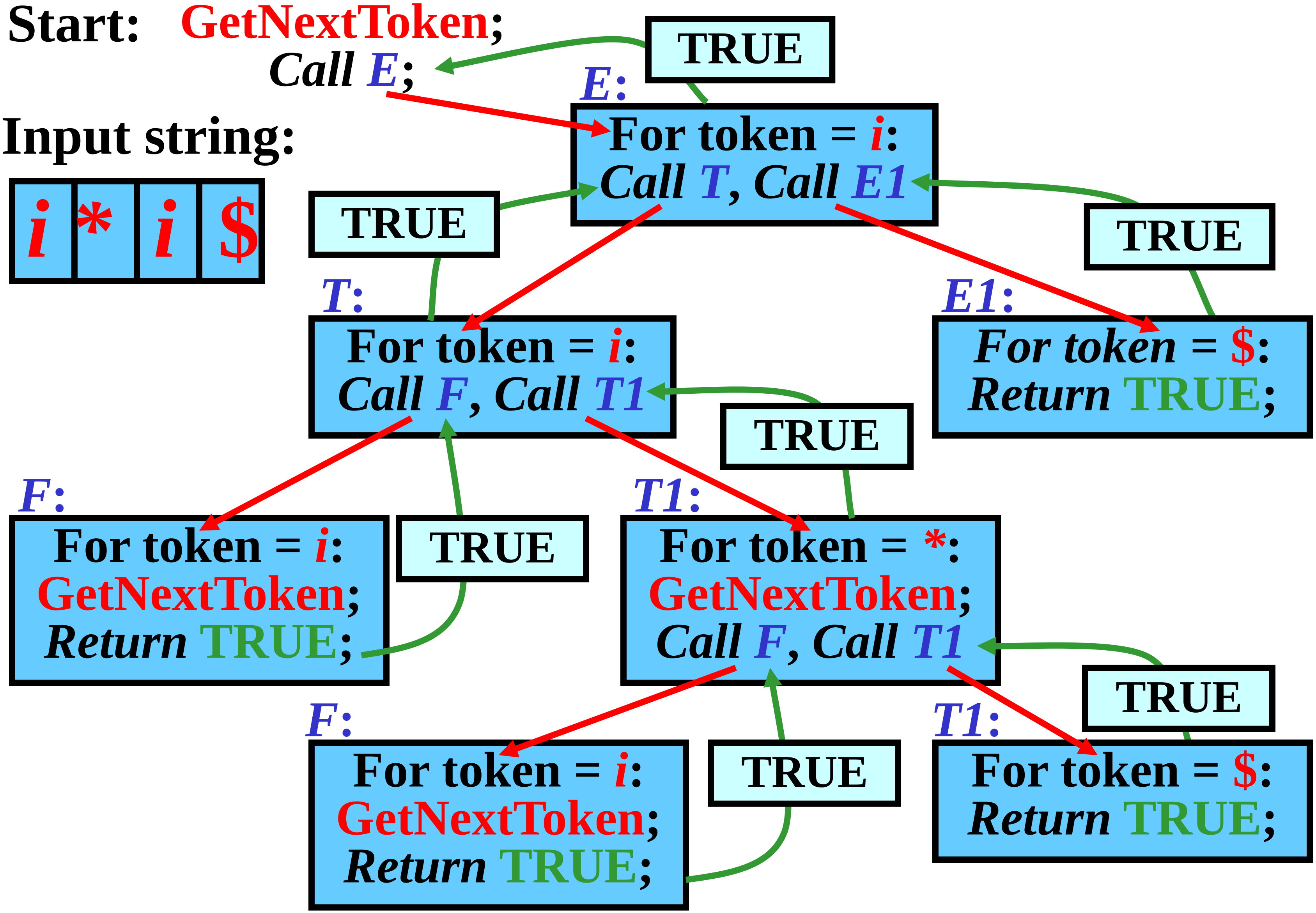
F:
For token = *i*:
GetNextToken;
Return **TRUE**;

TRUE

T1:
For token = $:
Return **TRUE**;

TRUE

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



Start: **GetNextToken;**
Call E;

Input string:

i	*	*i*	$

E:
For token = *i*:
Call T, Call E1

T:
For token = *i*:
Call F, Call T1

E1:
For token = $:
Return **TRUE;**

F:
For token = *i*:
GetNextToken;
Return **TRUE;**

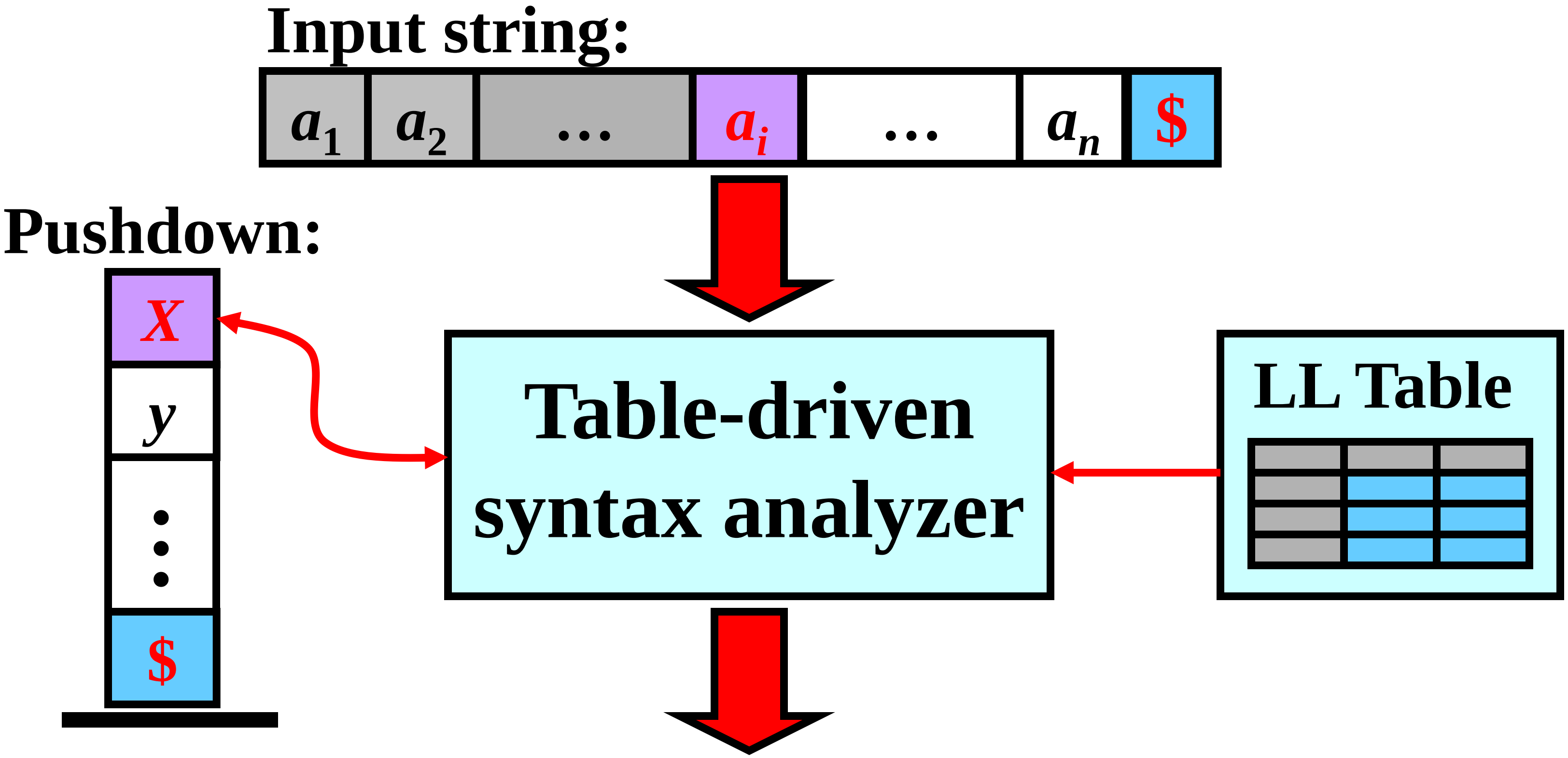
T1:
For token = *:
GetNextToken;
Call F, Call T1

F:
For token = *i*:
GetNextToken;
Return **TRUE;**

T1:
For token = $:
Return **TRUE;**

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 ∈ T:     if X = a then pop(X) & read next a from
                                   input string
                          else error;
        • X ∈ N:     if r: X → x ∈ 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* → *TE'*
2: *E'* → +*TE'*
3: *E'* → ε
4: *T* → *FT'*
5: *T'* → **FT'*
6: *T'* → ε
7: *F* → (*E*)
8: *F* → *i*

Input string: *i* * *i* $

Pushdown	**Input**	**Rule**	**Derivation**

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
$*E*	*i***i*$	1: $E \rightarrow TE'$	$\underline{E} \Rightarrow \underline{T}E'$

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
$E	*i***i*$	1: $E \rightarrow TE'$	$\underline{E} \Rightarrow \underline{T}E'$
$E'T	*i***i*$	4: $T \rightarrow FT'$	$\Rightarrow \underline{F}T'E'$

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 → TE’
2: E’ → +TE’
3: E’ → ε
4: T → FT’
5: T’ → *FT’
6: T’ → ε
7: F → (E)
8: F → i

Input string: i * i $

Pushdown	Input	Rule	Derivation
$E	i*i$	1: E → TE’	$\underline{E} \Rightarrow \underline{T}E'$
$E’T	i*i$	4: T → FT’	$\Rightarrow \underline{F}T'E'$
$E’T’F	i*i$	8: F → i	$\Rightarrow i\underline{T'}E'$

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* → *TE'*
2: *E'* → +*TE'*
3: *E'* → ε
4: *T* → *FT'*
5: *T'* → **FT'*
6: *T'* → ε
7: *F* → (*E*)
8: *F* → *i*

Input string: *i* * *i* $

Pushdown	Input	Rule	Derivation
$*E*	*i***i*$	1: *E* → *TE'*	$\underline{E} \Rightarrow \underline{T}E'$
$*E'T*	*i***i*$	4: *T* → *FT'*	$\Rightarrow \underline{F}T'E'$
$*E'T'F*	*i***i*$	8: *F* → *i*	$\Rightarrow i\underline{T'}E'$
$*E'T'i*	*i***i*$		

Table-Driven Parsing: Example

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

Input string: i * i $

Pushdown	Input	Rule	Derivation
$E	i*i$	1: $E \rightarrow TE'$	$\underline{E} \Rightarrow \underline{T}E'$
$E'T	i*i$	4: $T \rightarrow FT'$	$\Rightarrow \underline{F}T'E'$
$E'T'F	i*i$	8: $F \rightarrow i$	$\Rightarrow i\underline{T'}E'$
$E'T'i	i*i$		
$E'T'	*i$	5: $T' \rightarrow *FT'$	$\Rightarrow i*\underline{F}T'E'$

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: $F \rightarrow i$

Table-Driven Parsing: Example

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

Input string: *i* * *i* $

Pushdown	Input	Rule	Derivation
$*E*	*i***i*$	1: *E* → *TE'*	$\underline{E} \Rightarrow \underline{T}E'$
$*E'T*	*i***i*$	4: *T* → *FT'*	$\Rightarrow \underline{F}T'E'$
$*E'T'F*	*i***i*$	8: *F* → *i*	$\Rightarrow i\underline{T'}E'$
$*E'T'i*	*i***i*$		
$*E'T'*	**i*$	5: *T'* → **FT'*	$\Rightarrow i*\underline{F}T'E'$
$*E'T'F**	**i*$		

Rules:

1: *E* → *TE'*

2: *E'* → +*TE'*

3: *E'* → ε

4: *T* → *FT'*

5: *T'* → **FT'*

6: *T'* → ε

7: *F* → (*E*)

8: *F* → *i*

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 → TE'

2: E' → +TE'

3: E' → ε

4: T → FT'

5: T' → *FT'

6: T' → ε

7: F → (E)

8: F → i

Input string: i * i $

Pushdown	Input	Rule	Derivation
$E	i*i$	1: E → TE'	$\underline{E} \Rightarrow \underline{T}E'$
$E'T	i*i$	4: T → FT'	$\Rightarrow \underline{F}T'E'$
$E'T'F	i*i$	8: F → i	$\Rightarrow i\underline{T'}E'$
$E'T'i	i*i$		
$E'T'	*i$	5: T' → *FT'	$\Rightarrow i*\underline{F}T'E'$
$E'T'F*	*i$		
$E'T'F	i$	8: F → i	$\Rightarrow i*i\underline{T'}E'$

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 → TE'
2: E' → +TE'
3: E' → ε
4: T → FT'
5: T' → *FT'
6: T' → ε
7: F → (E)
8: F → i

Input string: i * i $

Pushdown	Input	Rule	Derivation
$E	i*i$	1: E → TE'	$\underline{E} \Rightarrow \underline{T}E'$
$E'T	i*i$	4: T → FT'	$\Rightarrow \underline{F}T'E'$
$E'T'F	i*i$	8: F → i	$\Rightarrow i\underline{T'}E'$
$E'T'i	i*i$		
$E'T'	*i$	5: T' → *FT'	$\Rightarrow i*\underline{F}T'E'$
$E'T'F*	*i$		
$E'T'F	i$	8: F → i	$\Rightarrow i*i\underline{T'}E'$
$E'T'i	i$		

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 → TE’
2: E’ → +TE’
3: E’ → ε
4: T → FT’
5: T’ → *FT’
6: T’ → ε
7: F → (E)
8: F → i

Input string: i * i $

Pushdown	Input	Rule	Derivation
$E	i*i$	1: E → TE’	$\underline{E} \Rightarrow \underline{T}E'$
$E’T	i*i$	4: T → FT’	$\Rightarrow \underline{F}T'E'$
$E’T’F	i*i$	8: F → i	$\Rightarrow i\underline{T'}E'$
$E’T’i	i*i$		
$E’T’	*i$	5: T’ → *FT’	$\Rightarrow i*\underline{F}T'E'$
$E’T’F*	*i$		
$E’T’F	i$	8: F → i	$\Rightarrow i*i\underline{T'}E'$
$E’T’i	i$		
$E’T’	$	6: T’ → ε	$\Rightarrow i*i\underline{E'}$

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 → TE’
2: E’ → +TE’
3: E’ → ε
4: T → FT’
5: T’ → *FT’
6: T’ → ε
7: F → (E)
8: F → i

Input string: i * i $

Pushdown	Input	Rule	Derivation
$E	i*i$	1: E → TE’	$\underline{E} \Rightarrow \underline{T}E'$
$E’T	i*i$	4: T → FT’	$\Rightarrow \underline{F}T'E'$
$E’T’F	i*i$	8: F → i	$\Rightarrow i\underline{T'}E'$
$E’T’i	i*i$		
$E’T’	*i$	5: T’ → *FT’	$\Rightarrow i*\underline{F}T'E'$
$E’T’F*	*i$		
$E’T’F	i$	8: F → i	$\Rightarrow i*i\underline{T'}E'$
$E’T’i	i$		
$E’T’	$	6: T’ → ε	$\Rightarrow i*i\underline{E'}$
$E’	$	3: E’ → ε	$\Rightarrow i*i$

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

Input string: $i * i \$$

Pushdown	Input	Rule	Derivation
$\$E$	$i*i\$$	1: $E \to TE'$	$\underline{E} \Rightarrow \underline{T}E'$
$\$E'T$	$i*i\$$	4: $T \to FT'$	$\Rightarrow \underline{F}T'E'$
$\$E'T'F$	$i*i\$$	8: $F \to i$	$\Rightarrow i\underline{T'}E'$
$\$E'T'i$	$i*i\$$		
$\$E'T'$	$*i\$$	5: $T' \to *FT'$	$\Rightarrow i*\underline{F}T'E'$
$\$E'T'F*$	$*i\$$		
$\$E'T'F$	$i\$$	8: $F \to i$	$\Rightarrow i*i\underline{T'}E'$
$\$E'T'i$	$i\$$		
$\$E'T'$	$\$$	6: $T' \to \varepsilon$	$\Rightarrow i*i\underline{E'}$
$\$E'$	$\$$	3: $E' \to \varepsilon$	$\Rightarrow i*i$
$\$$	$\$$		

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 → TE’
2: E’ → +TE’
3: E’ → ε
4: T → FT’
5: T’ → *FT’
6: T’ → ε
7: F → (E)
8: F → i

Input string: i * i $

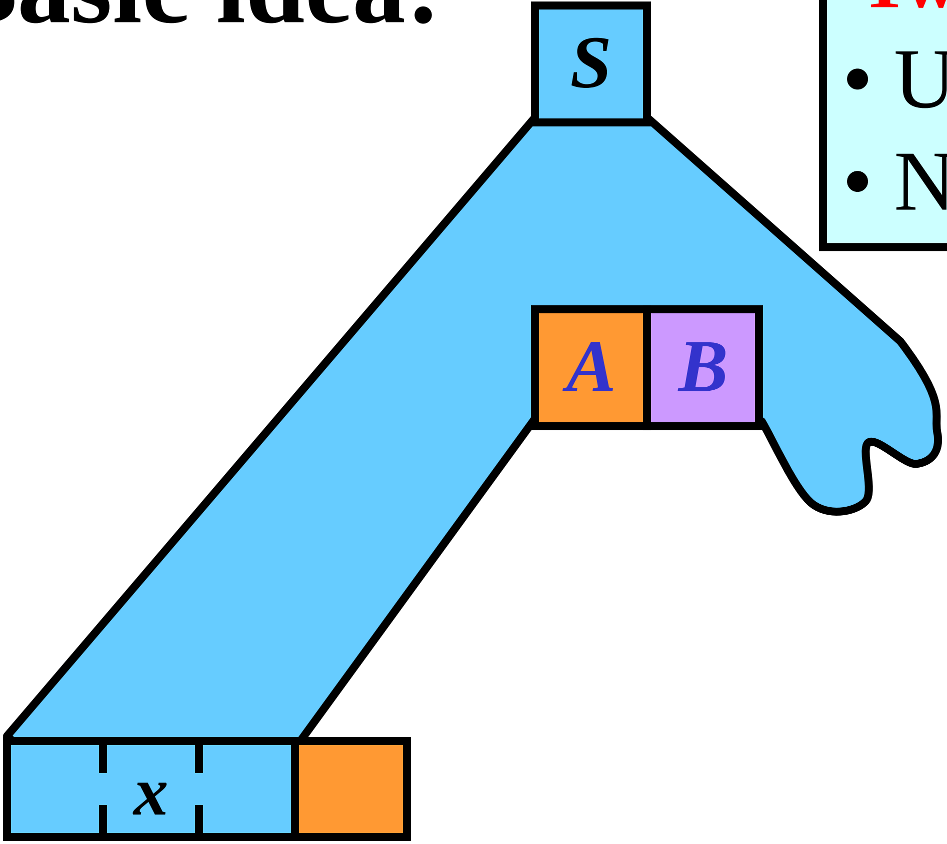
Pushdown	Input	Rule	Derivation
$E	i*i$	1: E → TE’	$\underline{E} \Rightarrow \underline{T}E'$
$E’T	i*i$	4: T → FT’	$\Rightarrow \underline{F}T'E'$
$E’T’F	i*i$	8: F → i	$\Rightarrow i\underline{T'}E'$
$E’T’i	i*i$		
$E’T’	*i$	5: T’ → *FT’	$\Rightarrow i*\underline{F}T'E'$
$E’T’F*	*i$		
$E’T’F	i$	8: F → i	$\Rightarrow i*i\underline{T'}E'$
$E’T’i	i$		
$E’T’	$	6: T’ → ε	$\Rightarrow i*i\underline{E'}$
$E’	$	3: E’ → ε	$\Rightarrow i*i$
$	$		

Success

Left parse: 1485863

Handling Errors: Introduction

Basic idea:

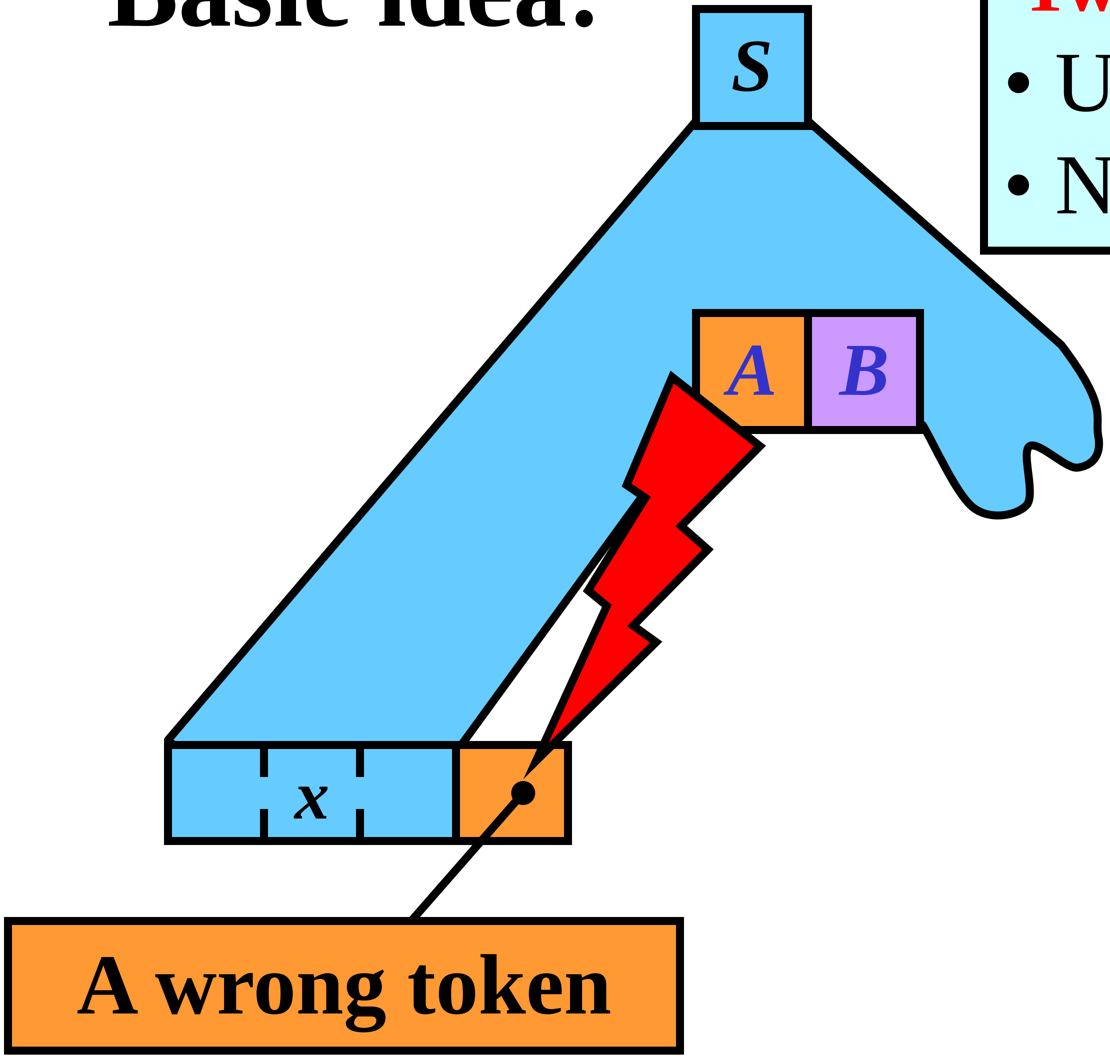


Two kinds of errors:

- Unexpected token
- No rule applicable

Handling Errors: Introduction

Basic idea:



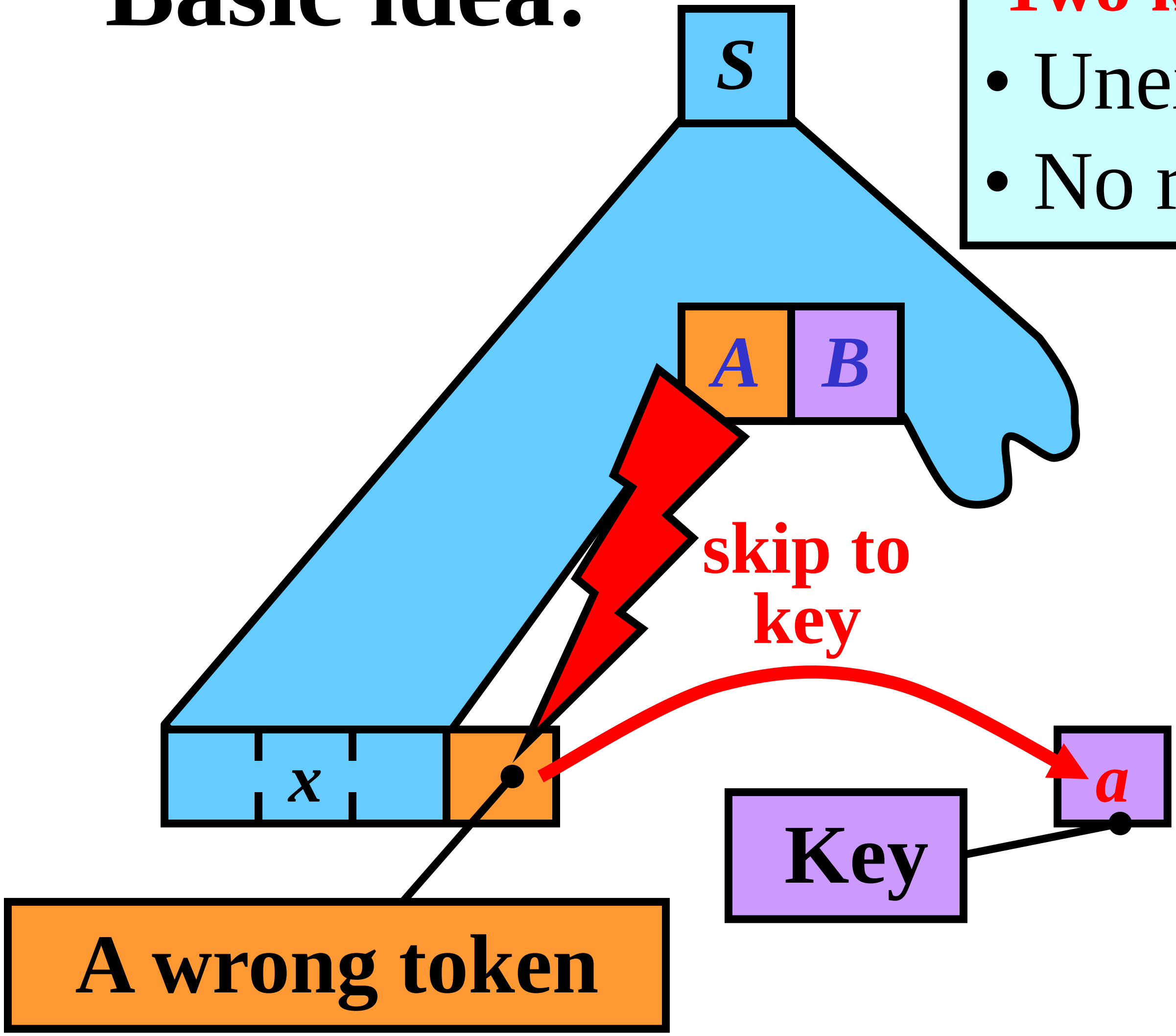
A wrong token

Two kinds of errors:

- Unexpected token
- No rule applicable

Handling Errors: Introduction

Basic idea:



Two kinds of errors:

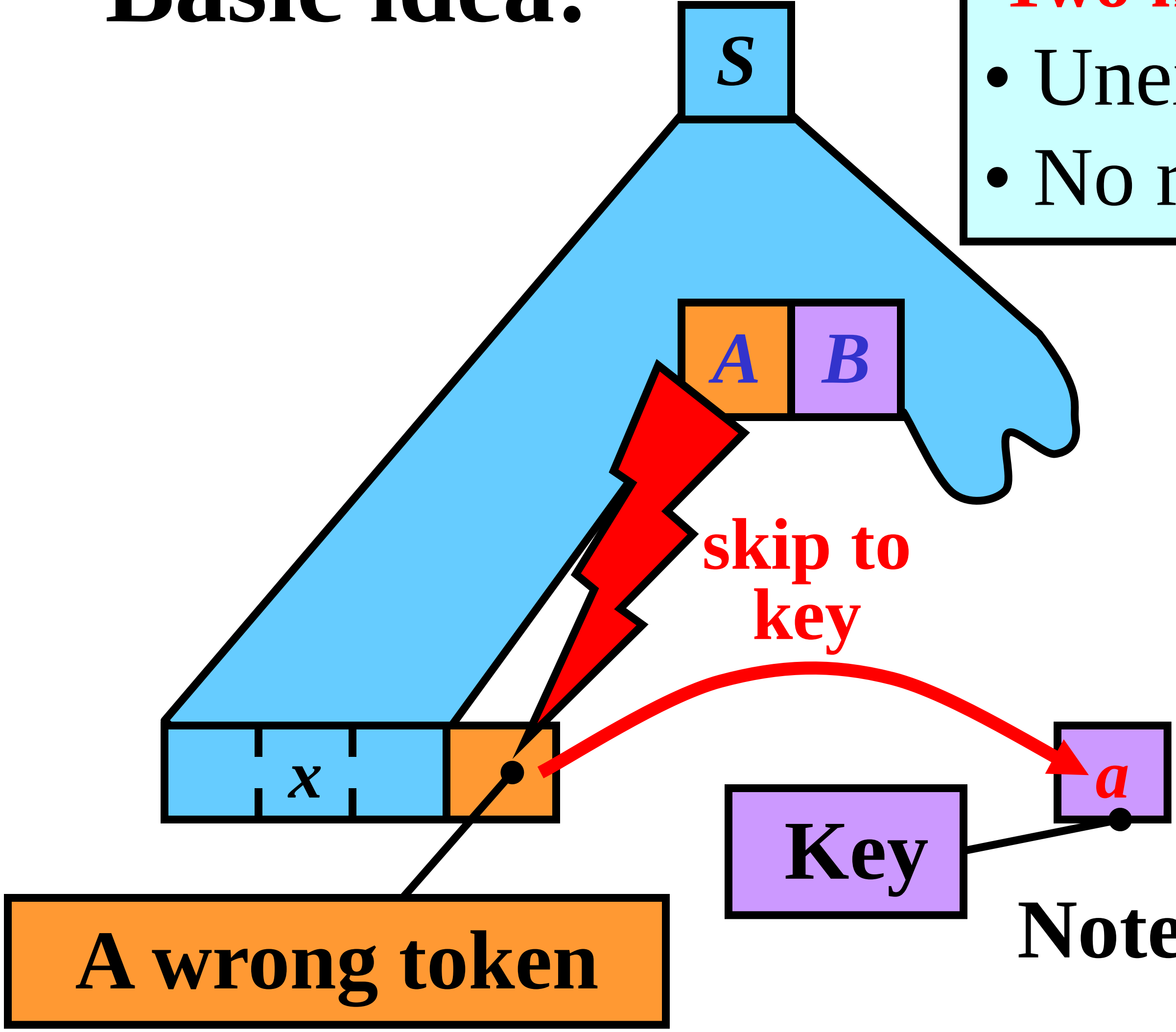
- Unexpected token
- No rule applicable

Handling Errors: Introduction

Basic idea:

Two kinds of errors:

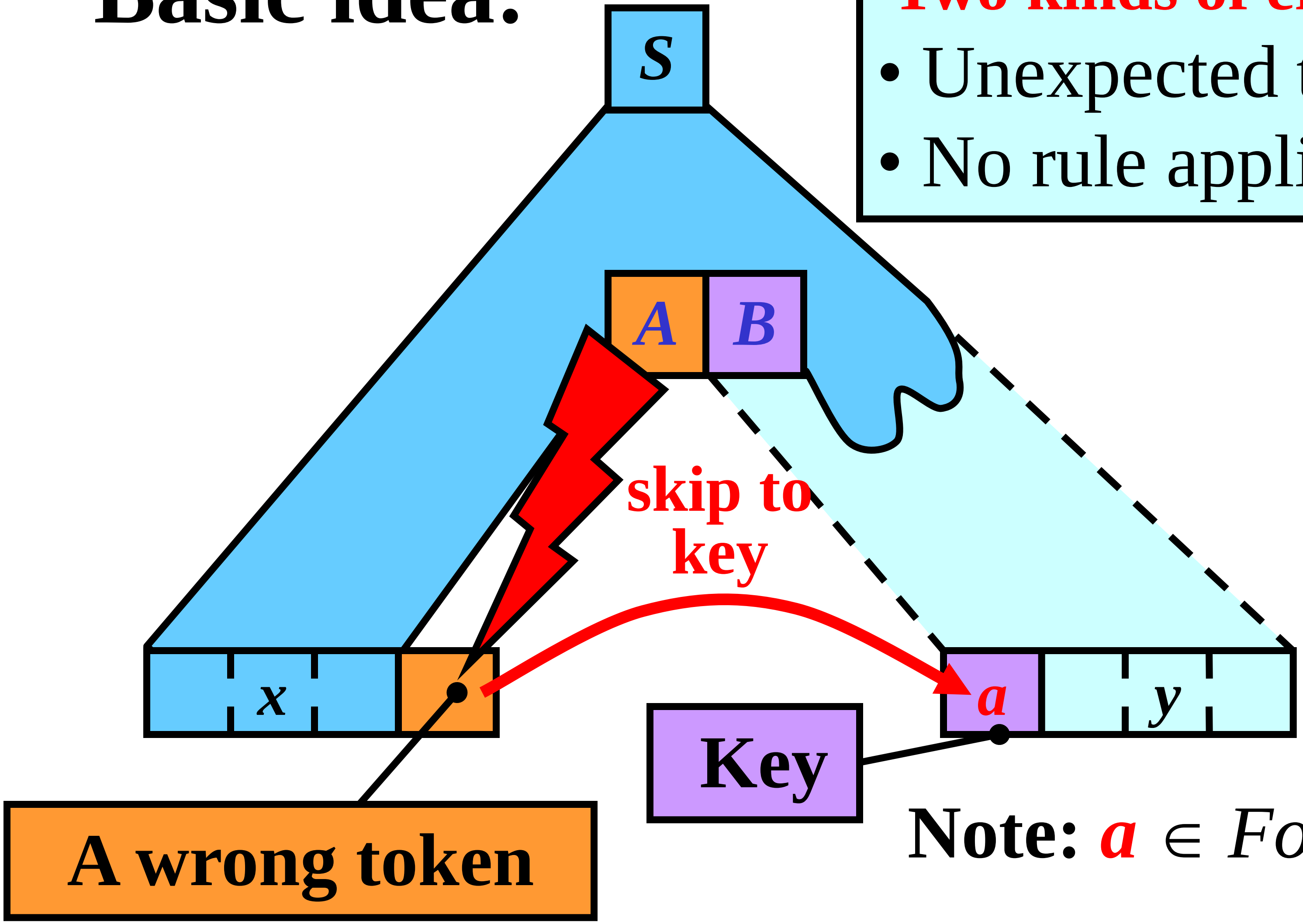
- Unexpected token
- No rule applicable



Note: $a \in Follow(A)$

Handling Errors: Introduction

Basic idea:

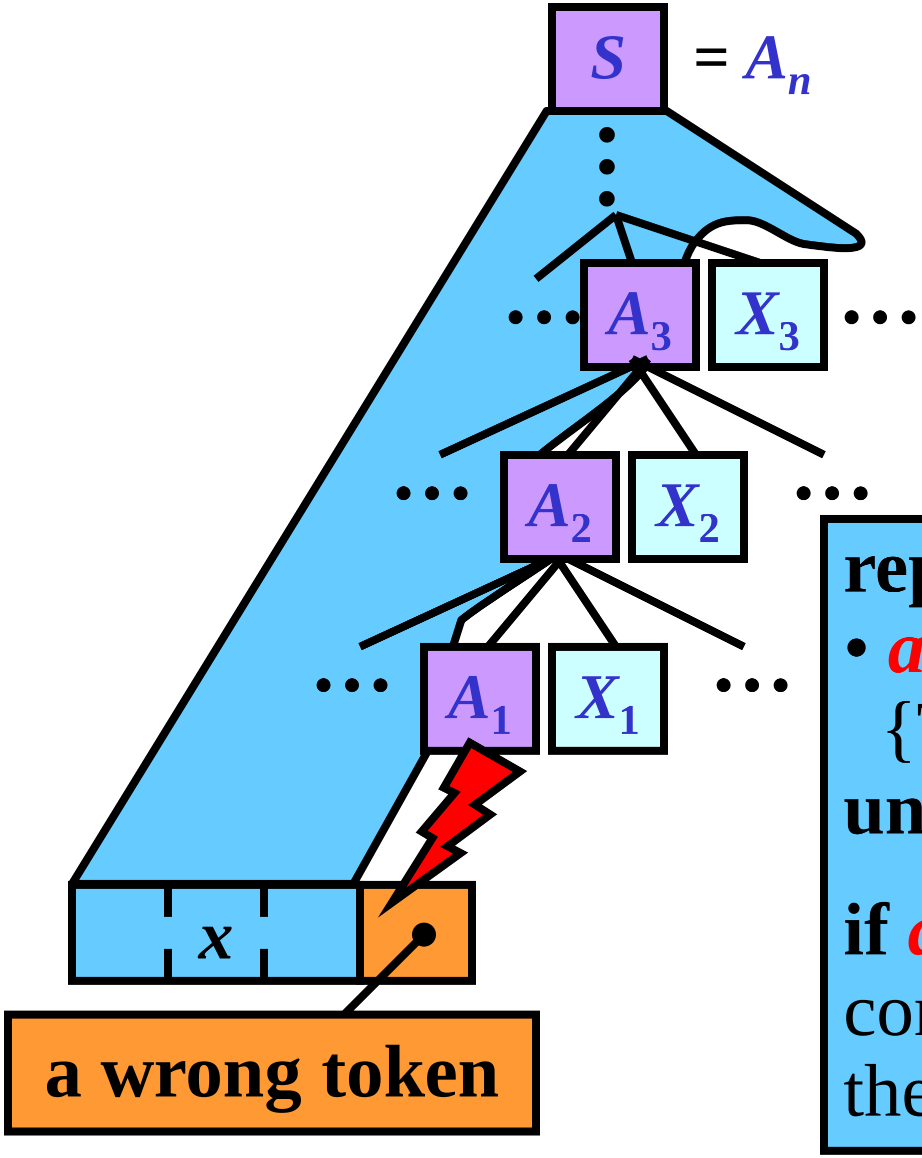


Two kinds of errors:

- Unexpected token
- No rule applicable

Note: $a \in Follow(A)$

Panic-Mode (Hartmann) Error Recovery

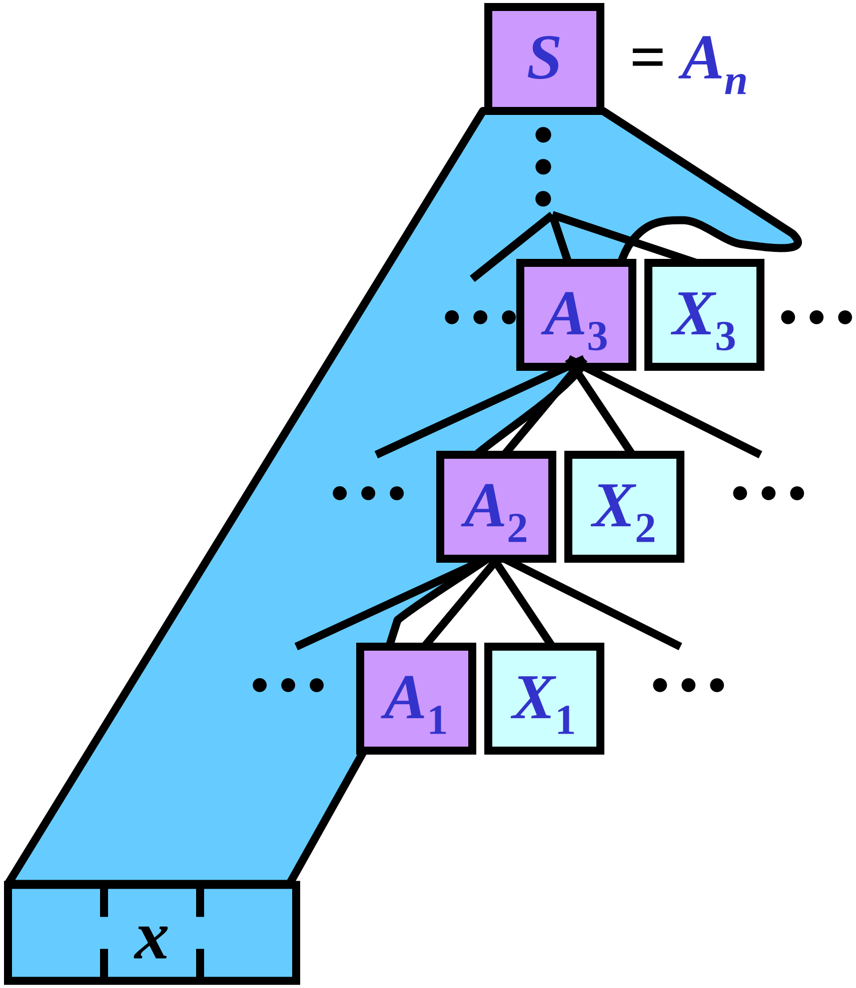


- Let $\mathit{Context}(A_1)$ =
  $\mathit{Follow}(A_1) \cup$
  $\mathit{Follow}(A_2) \cup$
  …
  $\mathit{Follow}(A_n)$

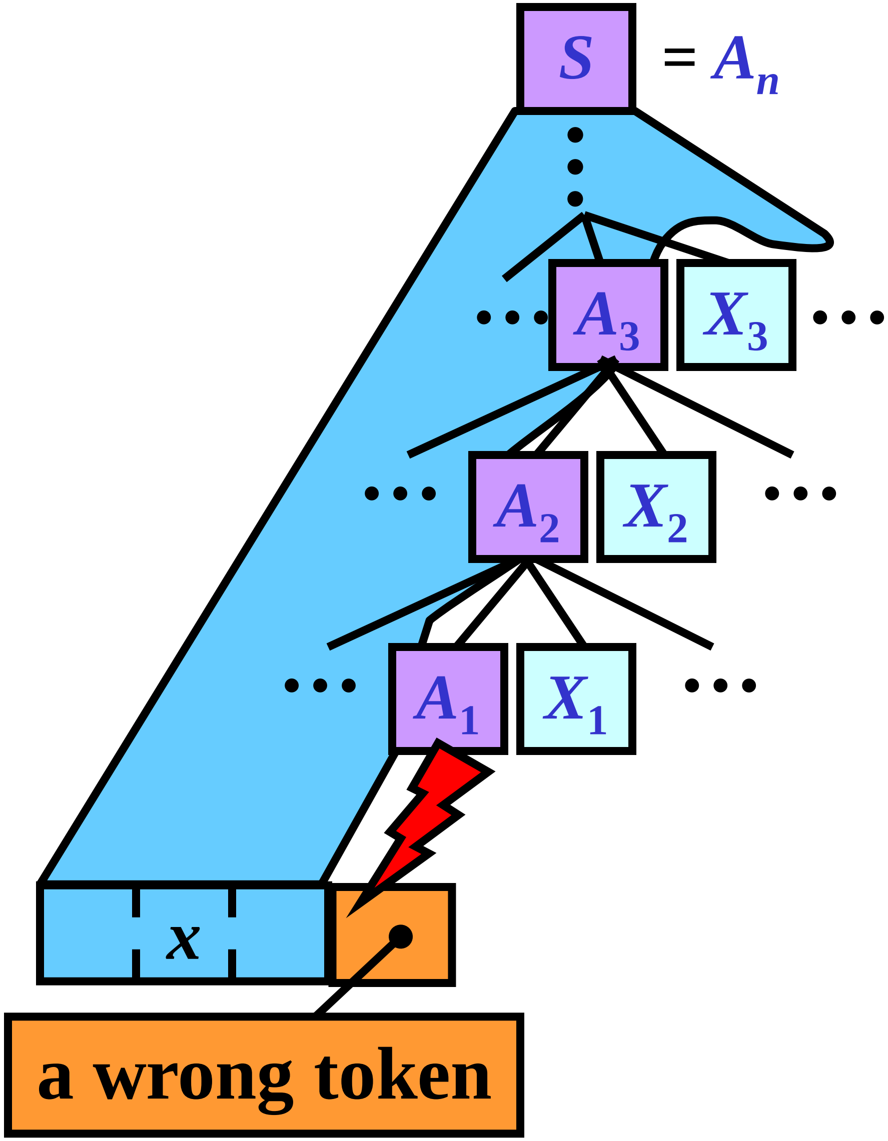
```
repeat
  • $a$ := GetNextToken;
    {These tokens are skipped}
until $a$ in $\mathit{Context}(A_1)$
```

if a in $\mathit{Follow}(A_i)$ **then** continue with parsing from the symbol X_i.

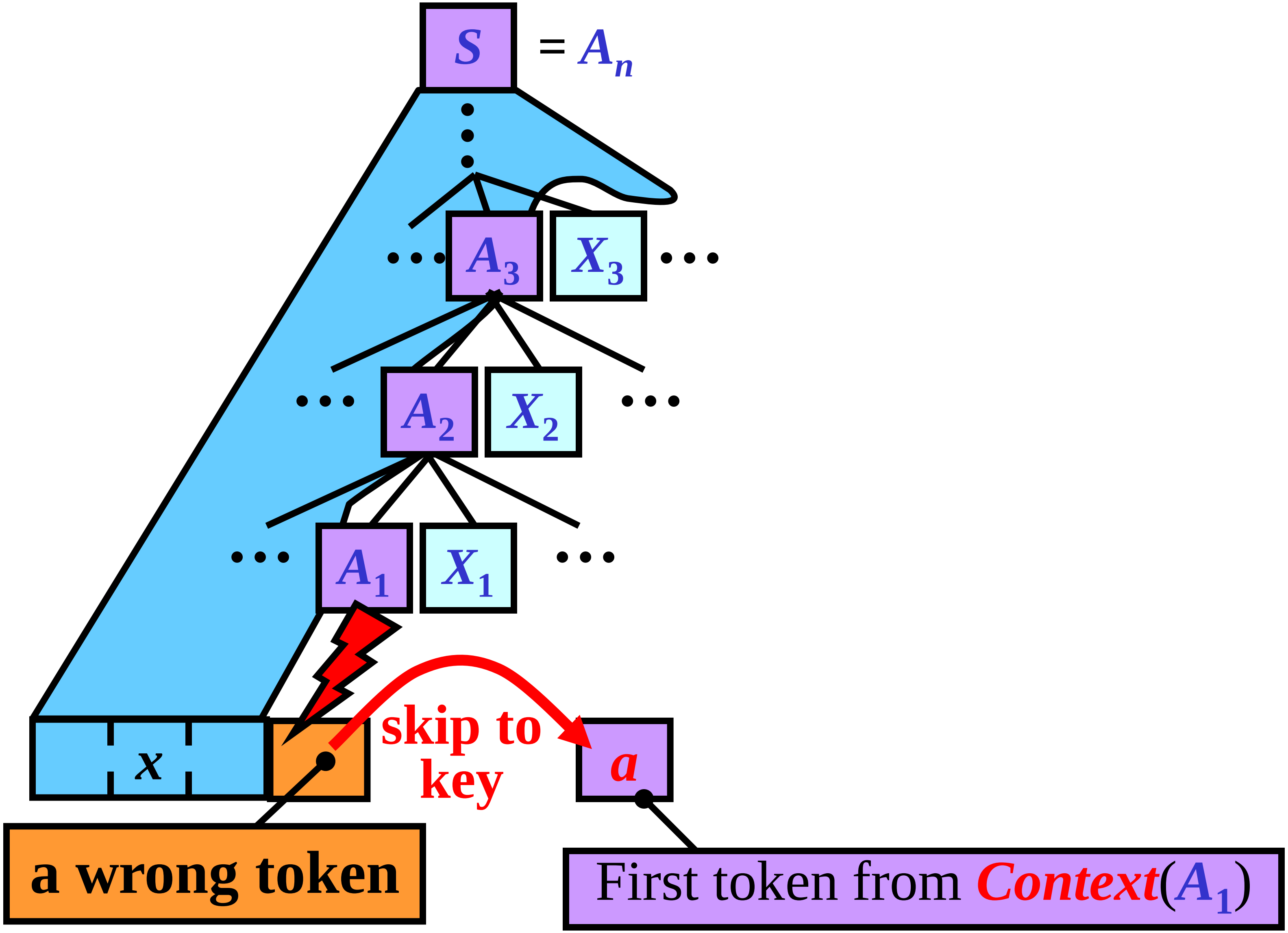
Panic-Mode Recovery: Illustration 1/2



Panic-Mode Recovery: Illustration 1/2

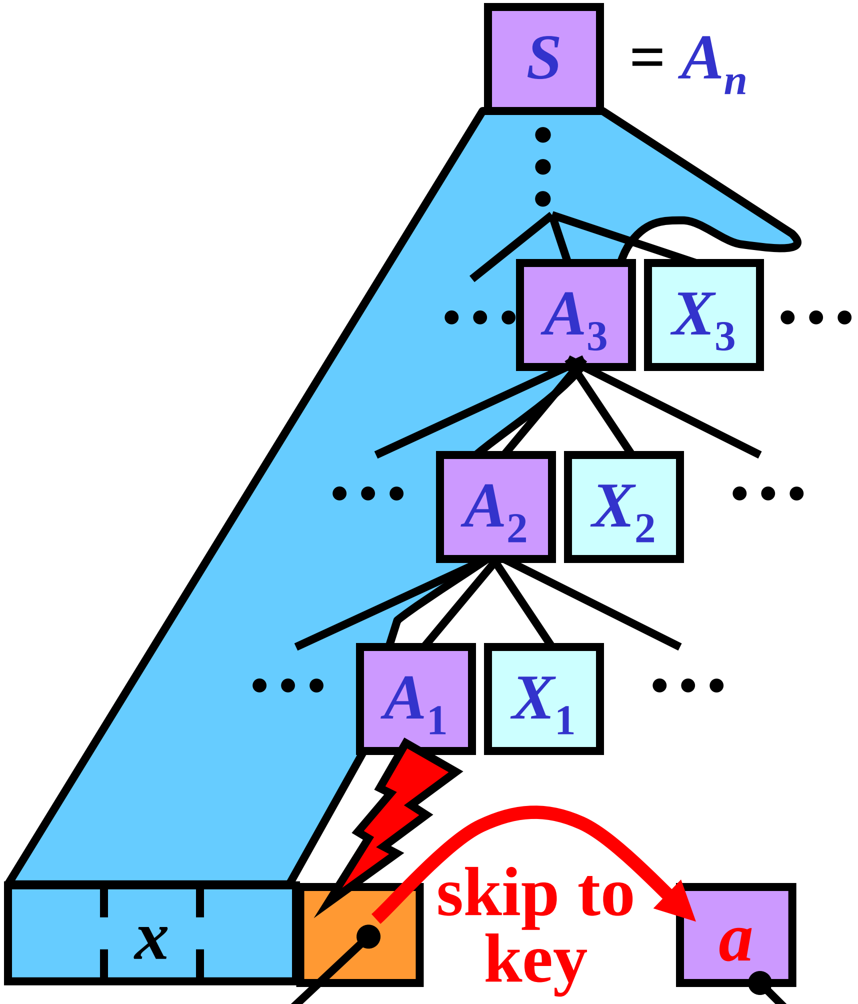


Panic-Mode Recovery: Illustration 1/2



Panic-Mode Recovery: Illustration 1/2

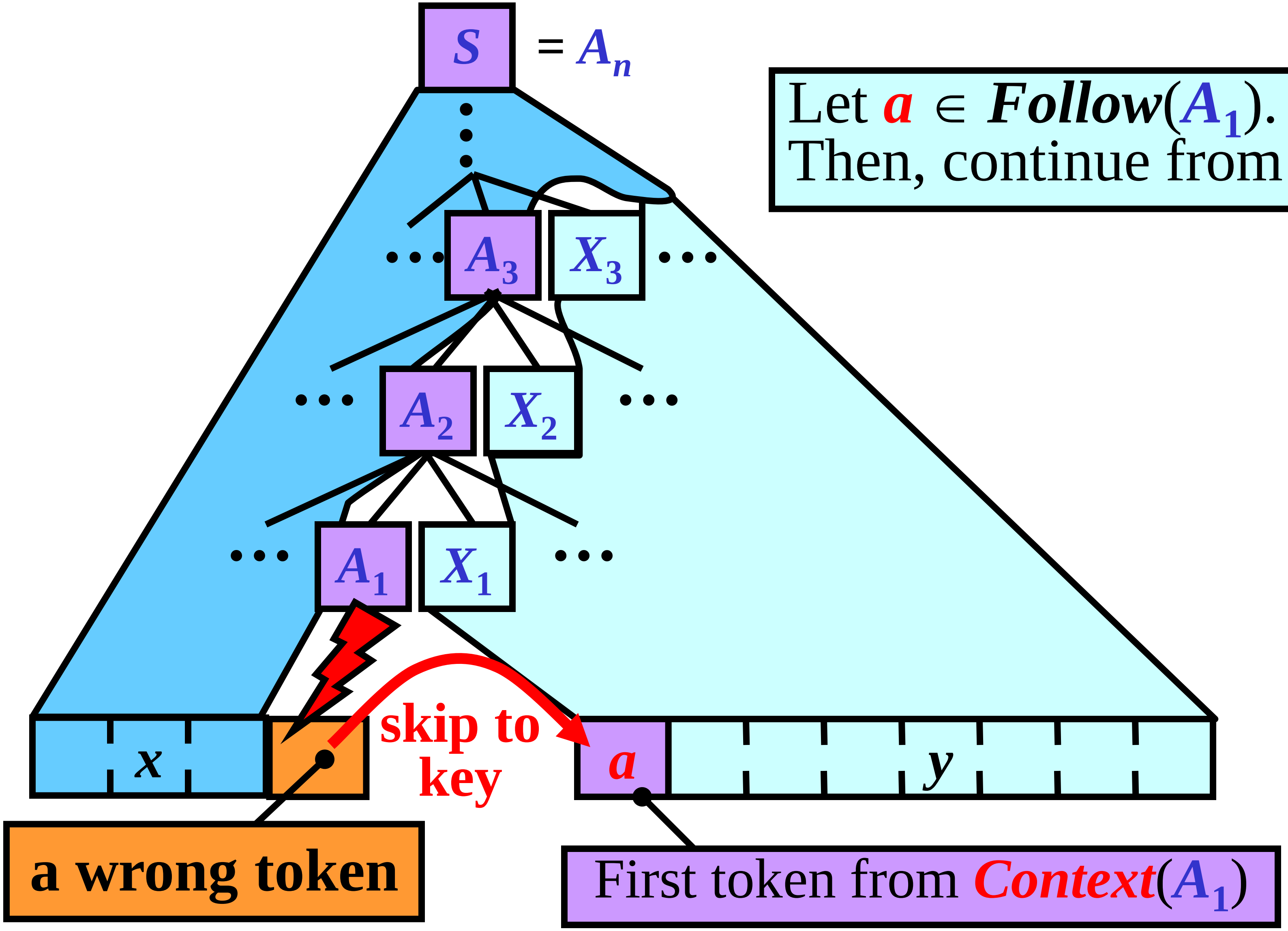
Let $\boldsymbol{a} \in \boldsymbol{Follow}(A_1)$.
Then, continue from X_1



a wrong token

First token from $\boldsymbol{Context}(A_1)$

Panic-Mode Recovery: Illustration 1/2

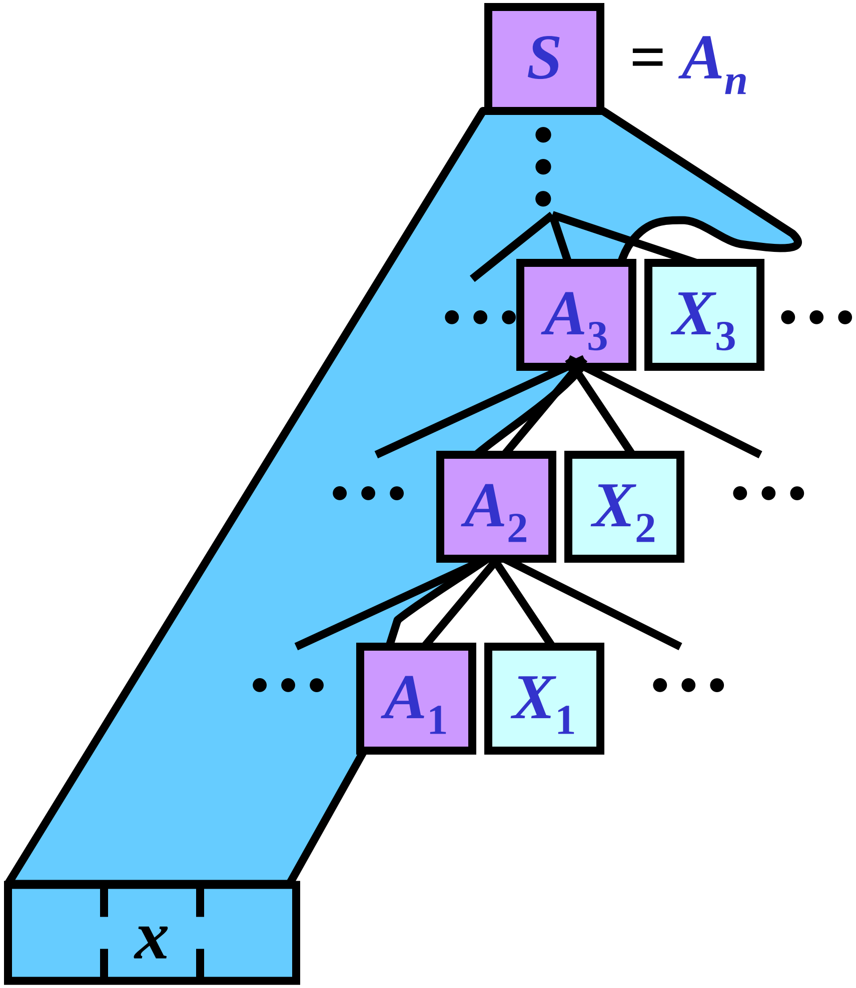


Let $a \in$ ***Follow***(A_1).
Then, continue from X_1

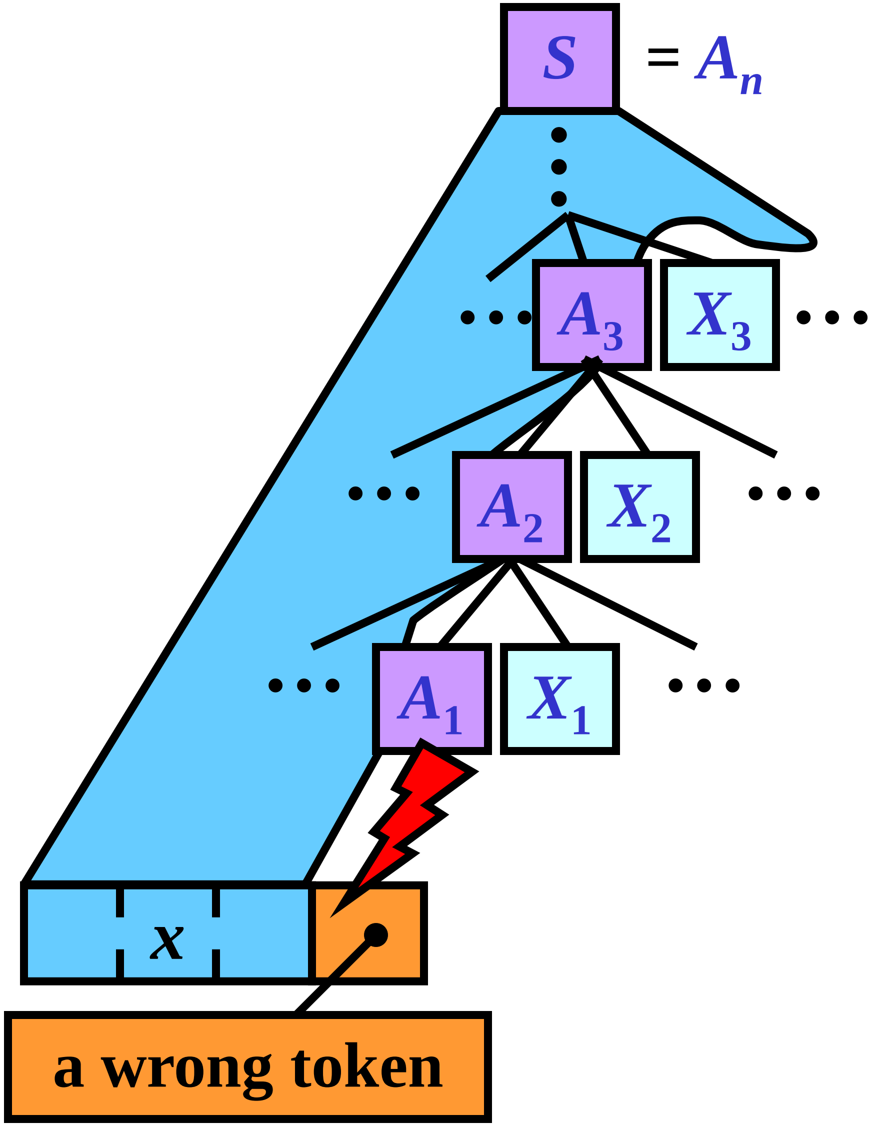
a wrong token

First token from ***Context***(A_1)

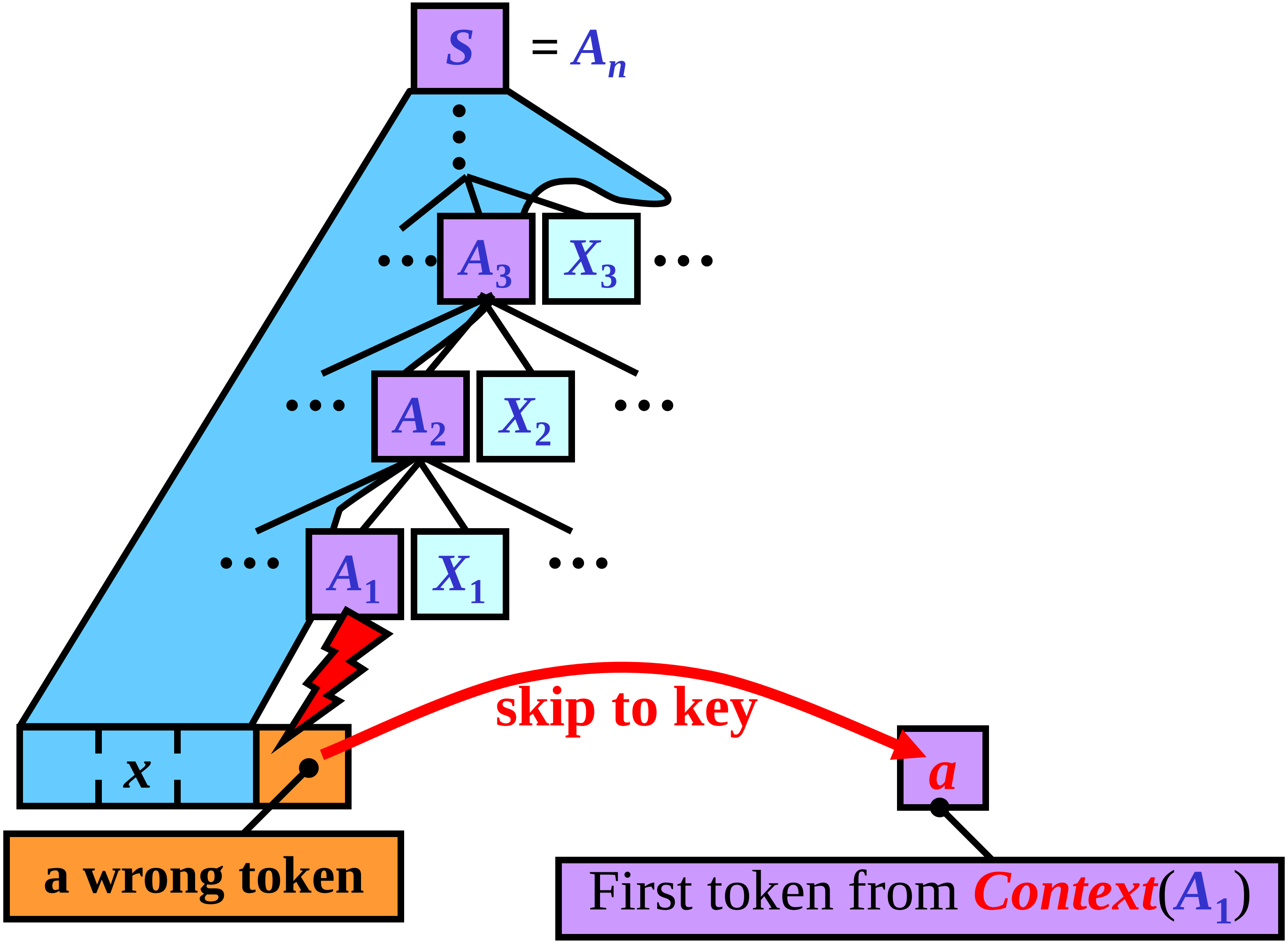
Panic-Mode Recovery: Illustration 2/2



Panic-Mode Recovery: Illustration 2/2



Panic-Mode Recovery: Illustration 2/2



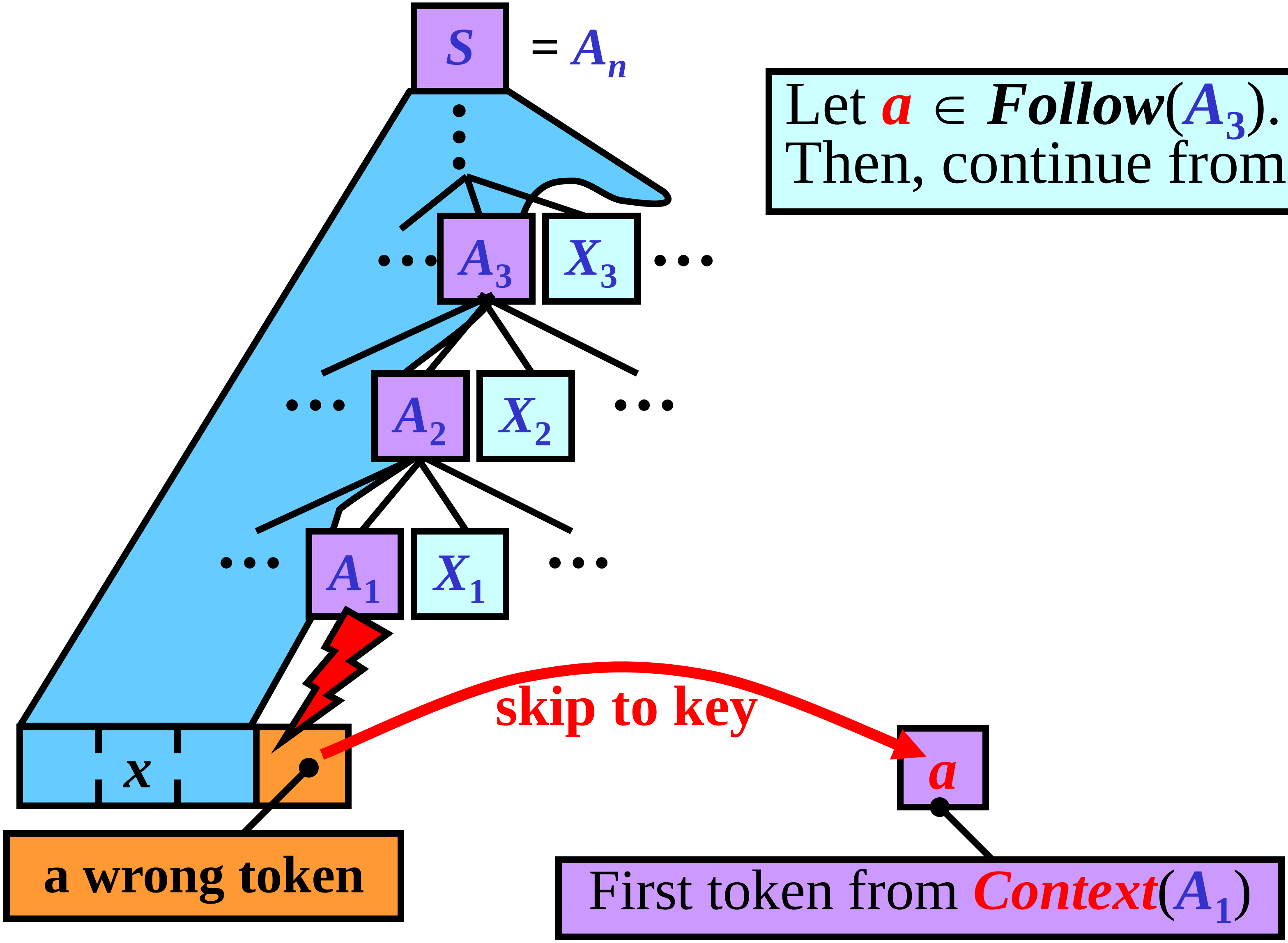
a wrong token

skip to key

First token from ***Context***(A_1)

Panic-Mode Recovery: Illustration 2/2

Let $a \in \textbf{\textit{Follow}}(A_3)$.
Then, continue from X_3

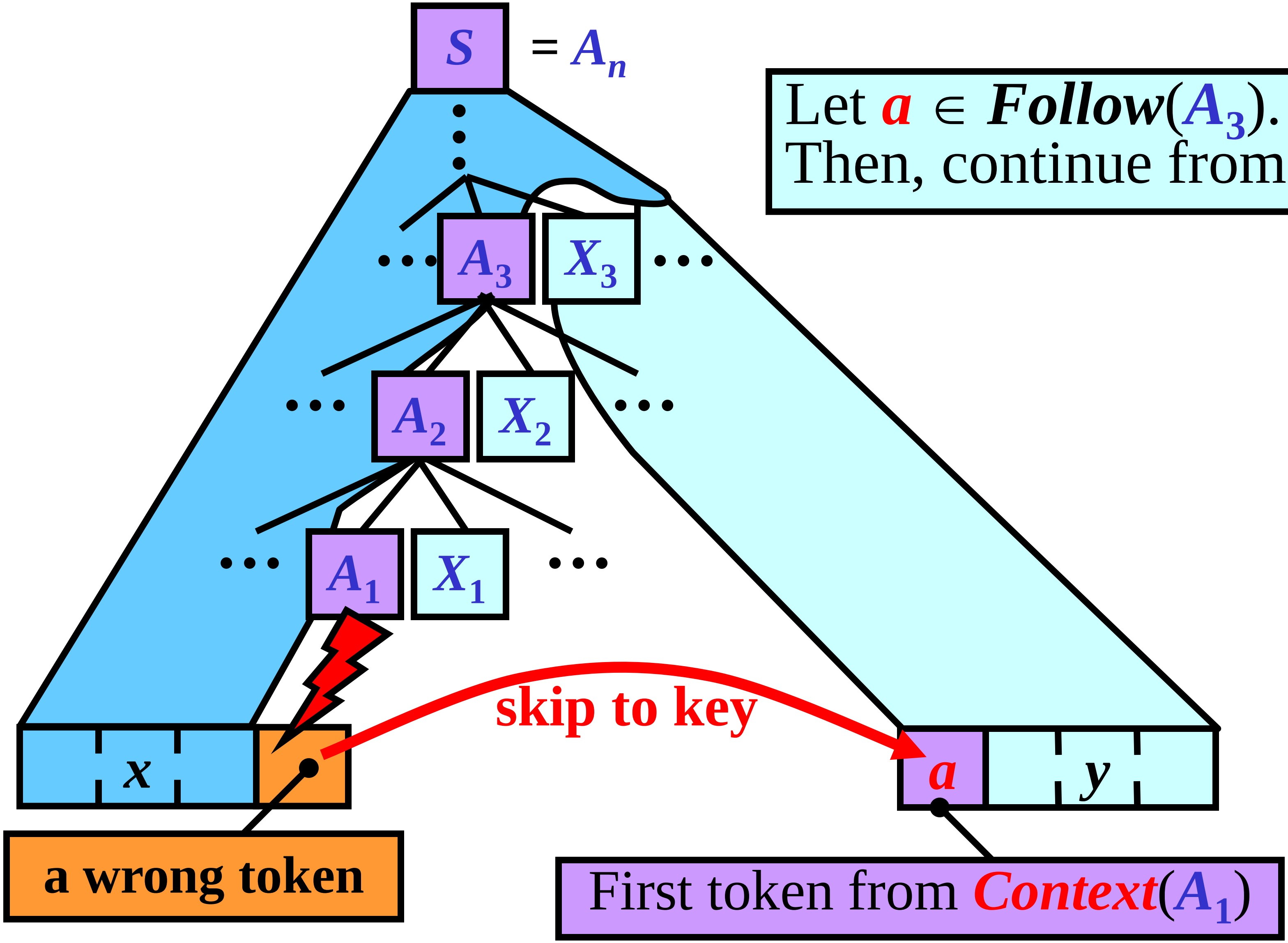


skip to key

a wrong token

First token from *Context*(A_1)

Panic-Mode Recovery: Illustration 2/2



Let $a \in$ ***Follow***(A_3).
Then, continue from X_3

a wrong token

First token from ***Context***(A_1)

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

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

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

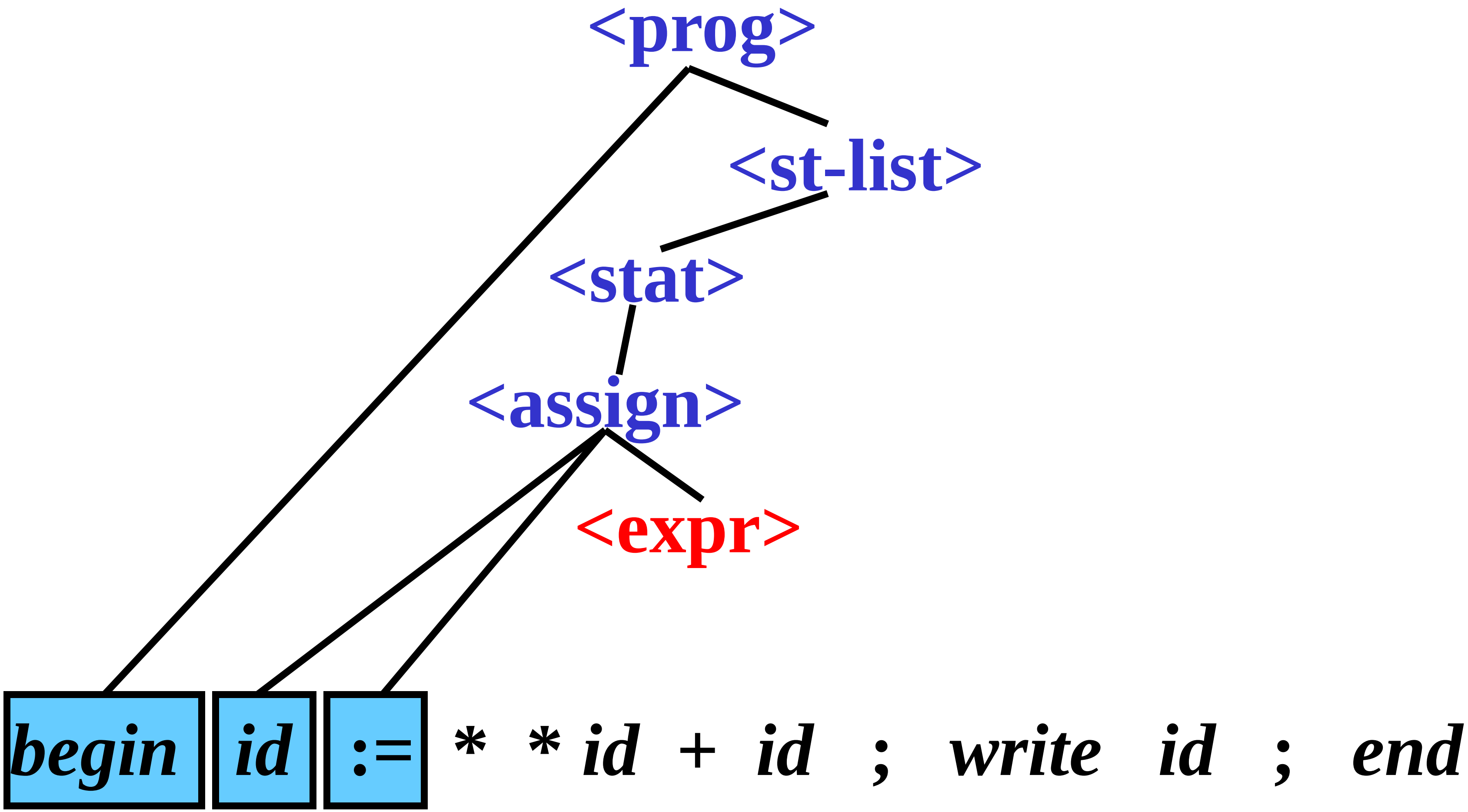
- **Method:**
- Let ***A*** be pushdown top & no rule is applicable:
- **repeat**

  a := GetNextToken;

  {These tokens are skipped}

  until ***a*** in ***Context*(*A*)**
- pop ***A*** from the pushdown;

Variant I: Example



<prog>

<st-list>

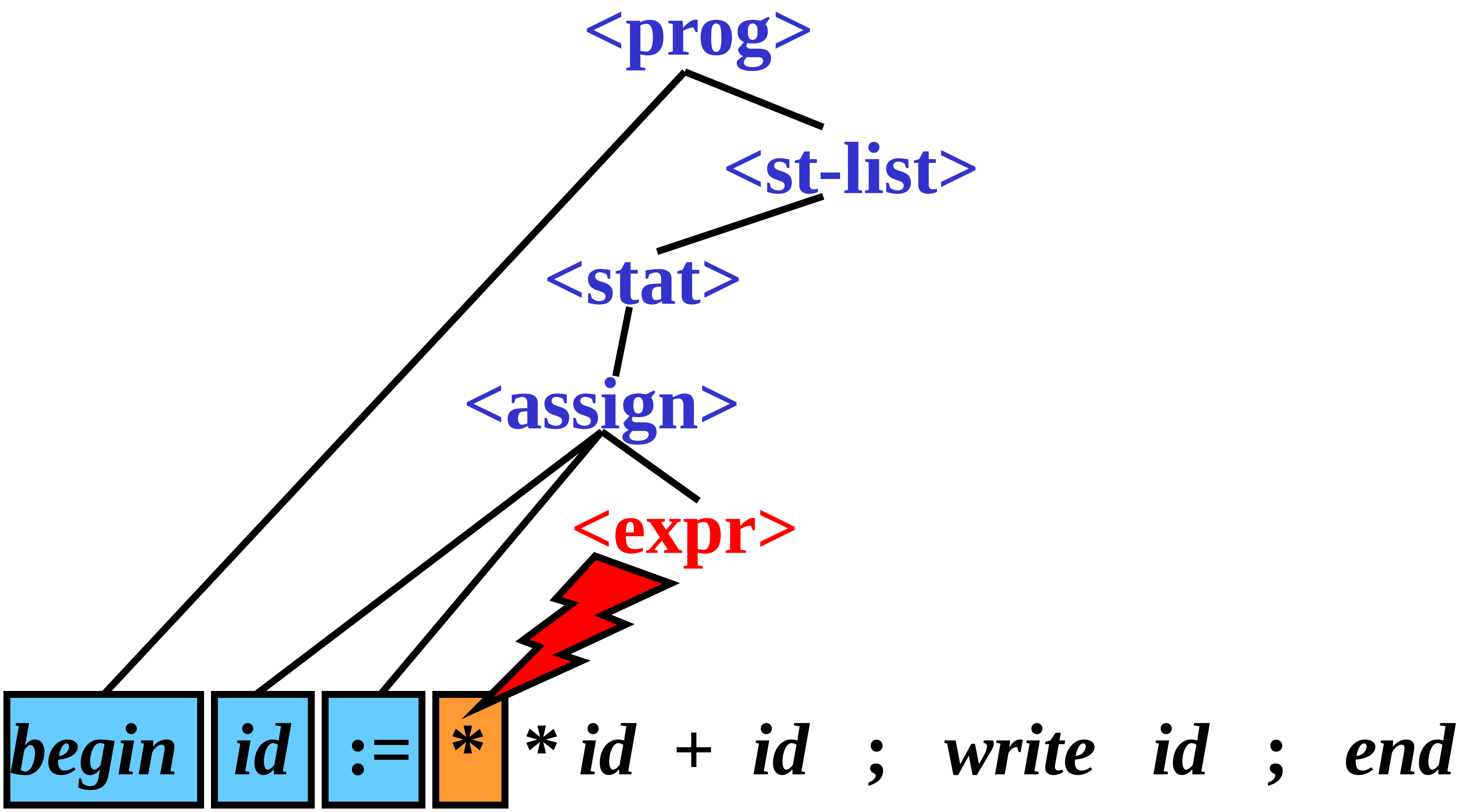
<stat>

<assign>

<expr>

begin ***id*** **:=** ***** ***** ***id*** **+** ***id*** **;** ***write*** ***id*** **;** ***end***

Variant I: Example



<prog>

<st-list>

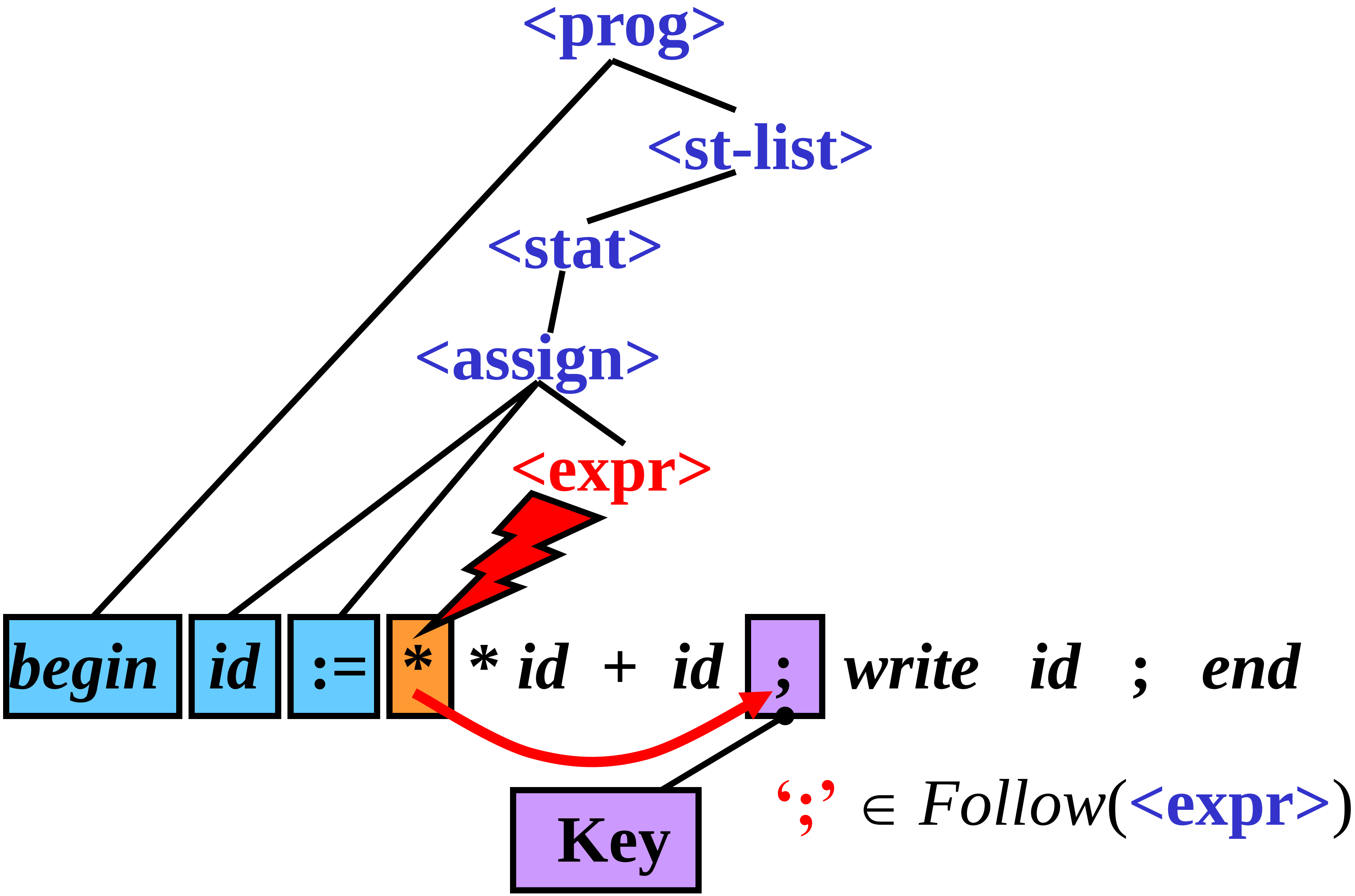
<stat>

<assign>

<expr>

begin ***id*** **:=** ***** ***** ***id*** **+** ***id*** **;** ***write*** ***id*** **;** ***end***

Variant I: Example



<prog>

<st-list>

<stat>

<assign>

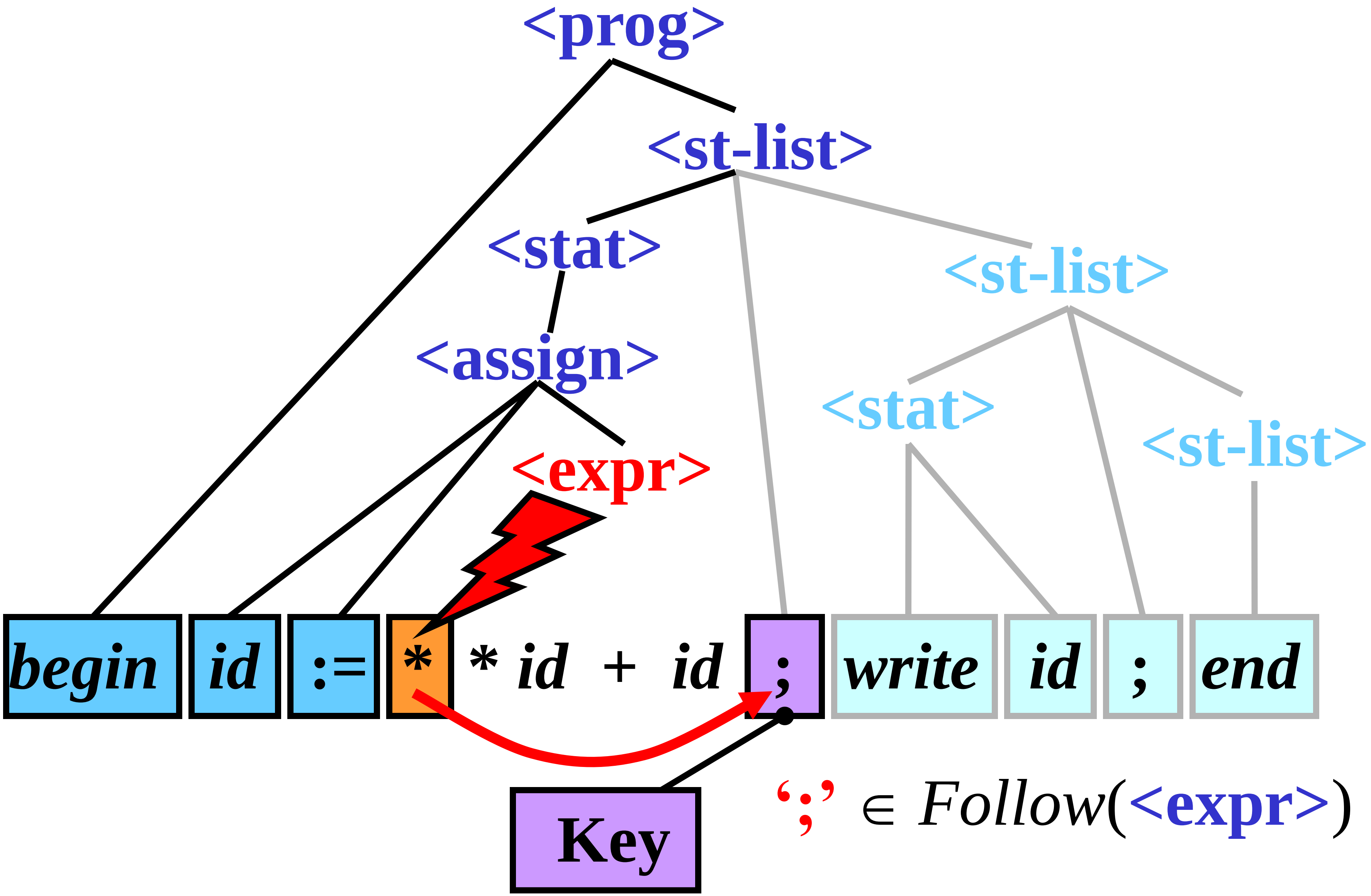
<expr>

begin *id* := * * *id* + *id* ; *write* *id* ; *end*

Key

‘;’ ∈ *Follow*(**<expr>**)

Variant I: Example



';' ∈ *Follow*(**<expr>**)

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

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

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

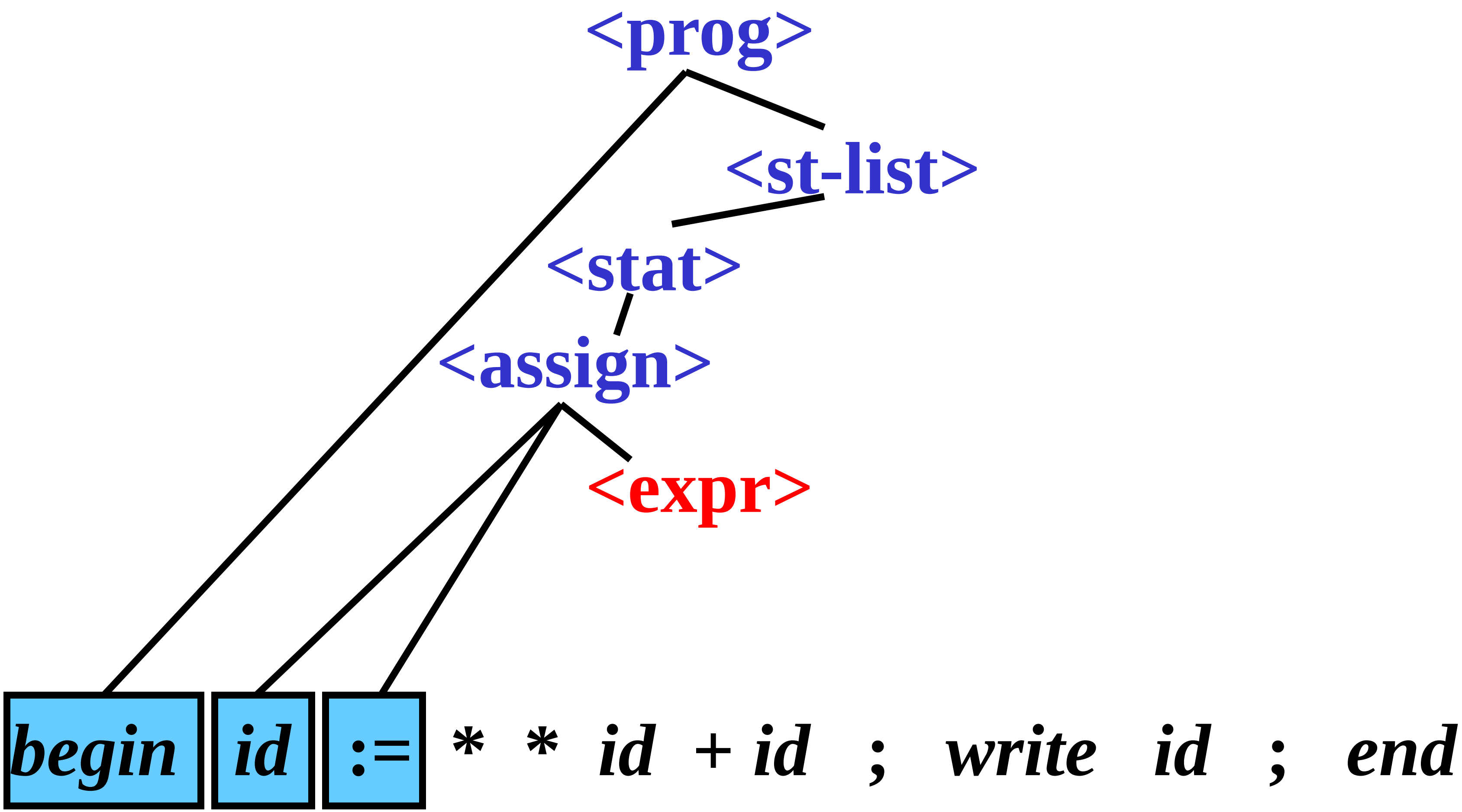
- **Method:**
- Let ***A*** be pushdown top & no rule is applicable:
- **repeat**

```
    a := GetNextToken;
    {These tokens are skipped}
  until a in Context(A)
```

- **if** ***a*** ∈ *First*(***A***) **then** resume according to ***A***

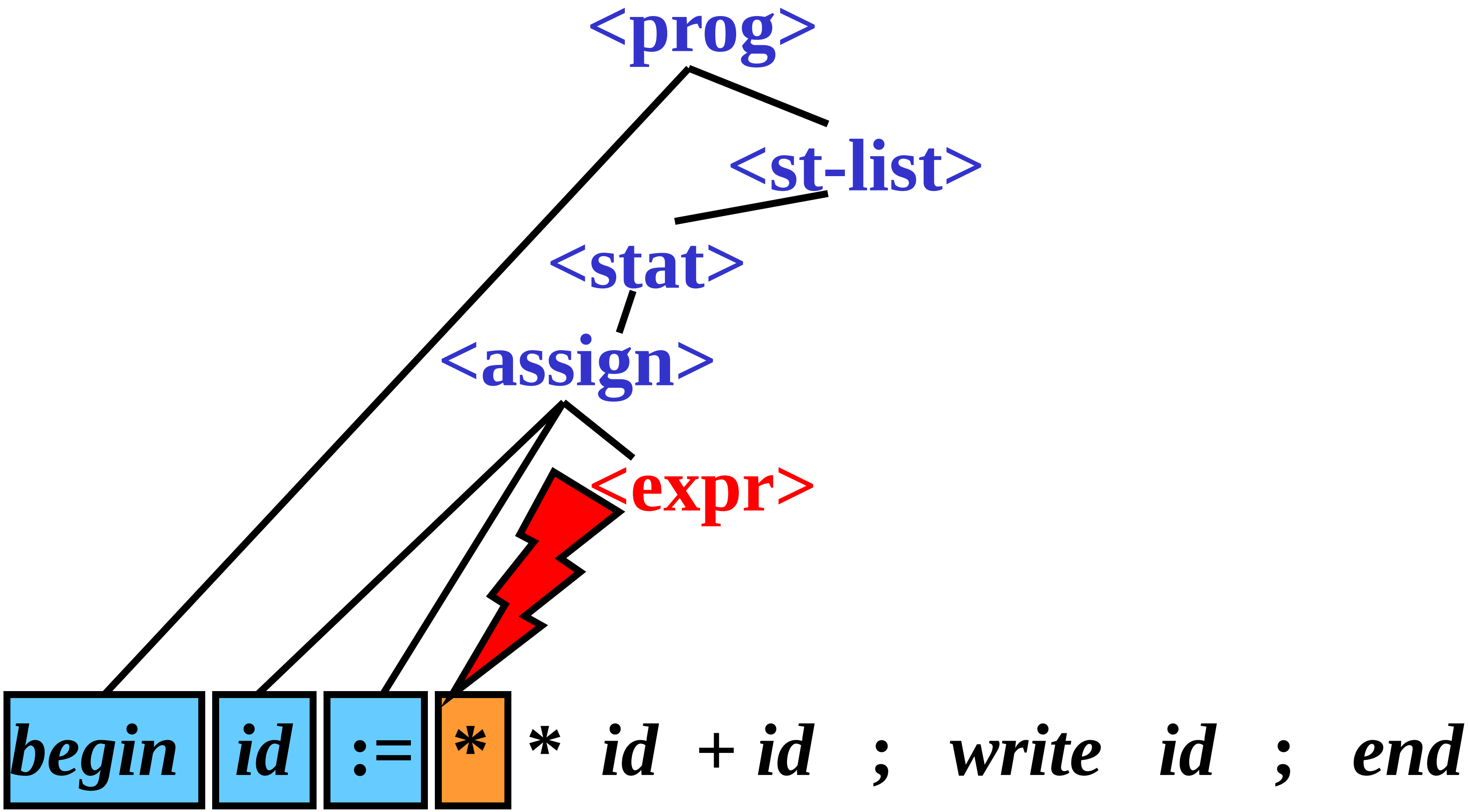
  else pop ***A*** from the pushdown // ***a*** ∈ *Follow*(***A***)

Variant II: Example

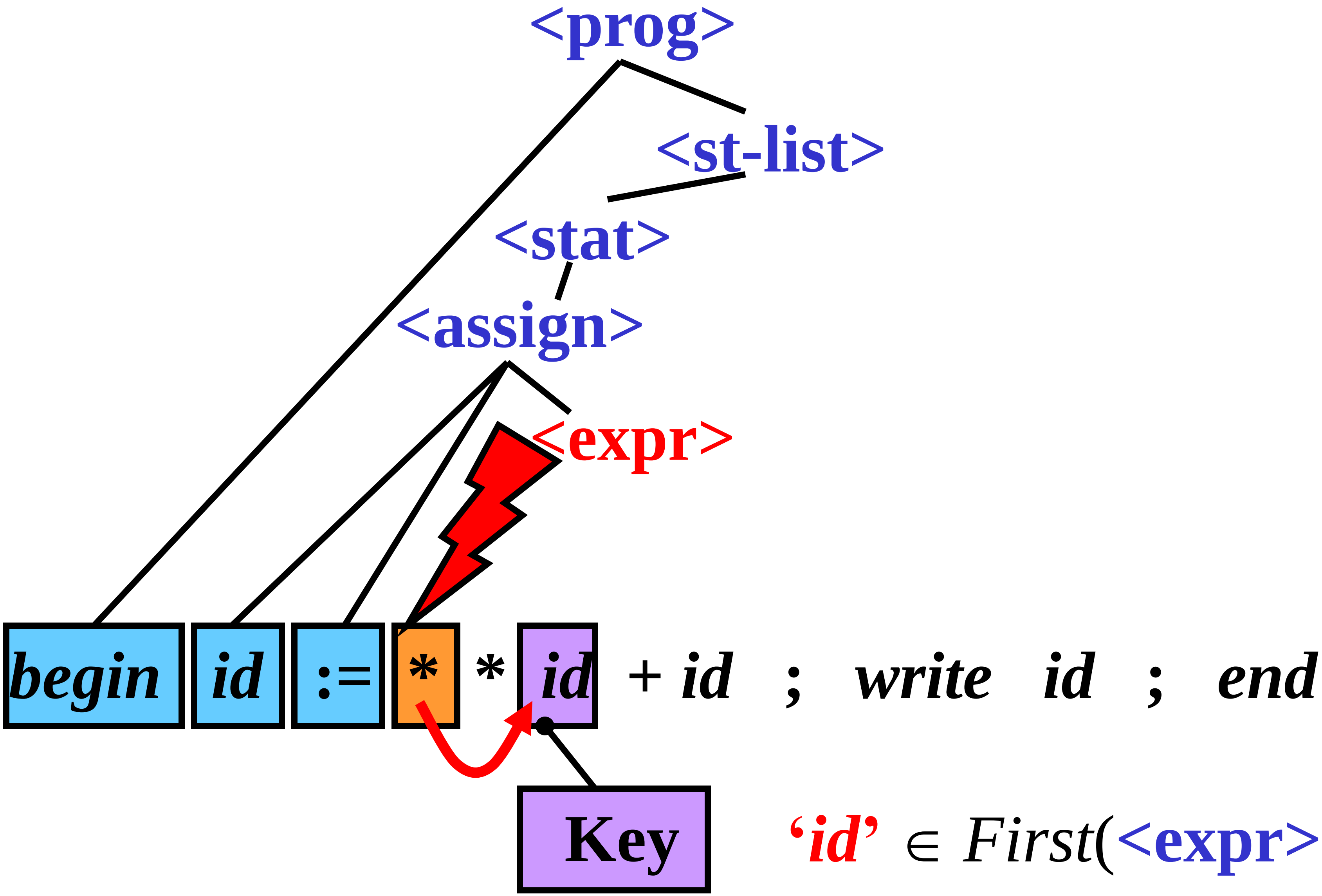


begin* *id* *:=* * * *id* + *id* ; *write* *id* ; *end

Variant II: Example



Variant II: Example



<prog>

<st-list>

<stat>

<assign>

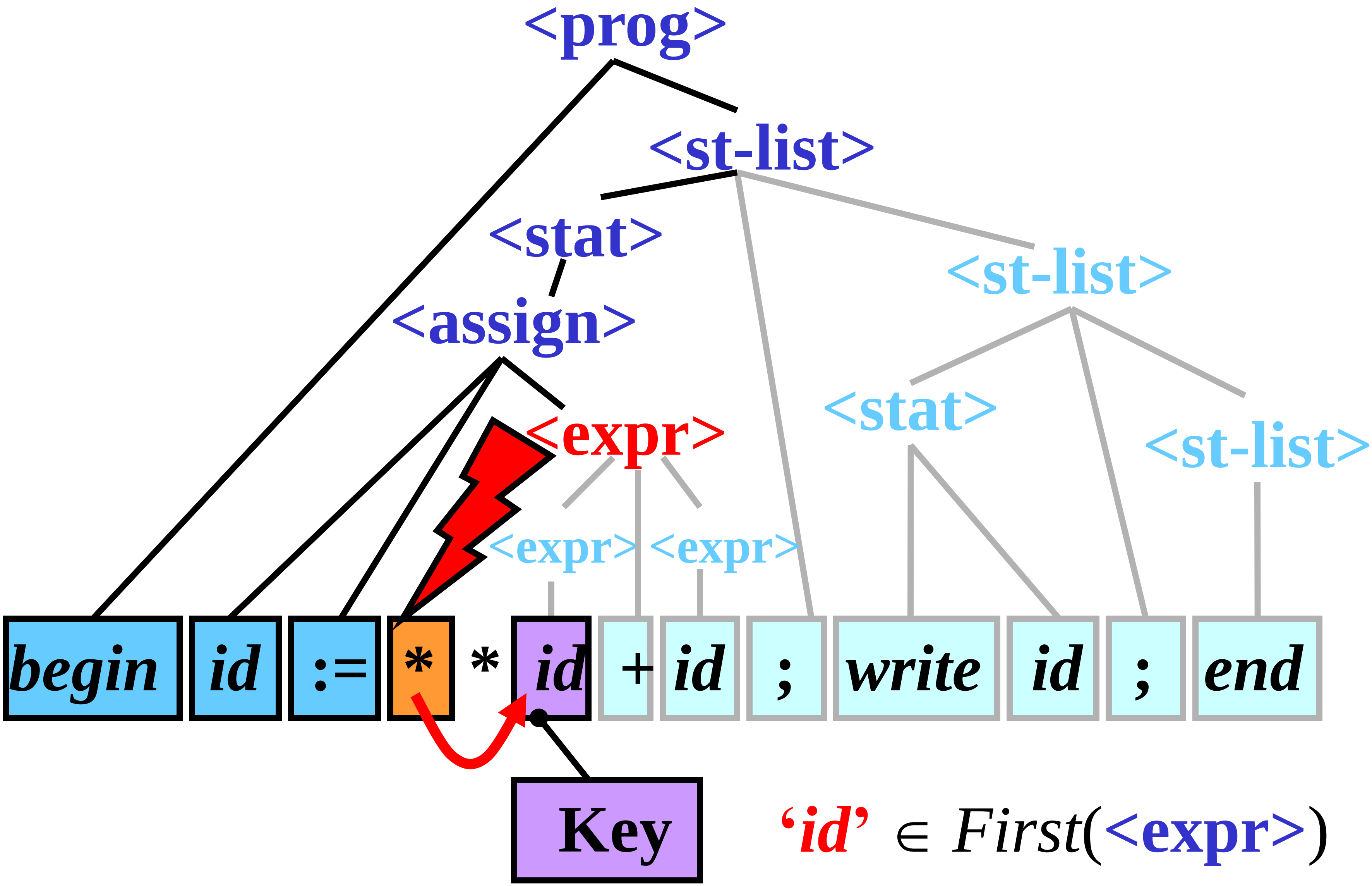
<expr>

begin ***id*** **:=** ***** ***** ***id*** **+** ***id*** **;** ***write*** ***id*** **;** ***end***

Key

‘id’ ∈ *First*(**<expr>**)

Variant II: Example



‘*id*’ ∈ *First*(**<expr>**)