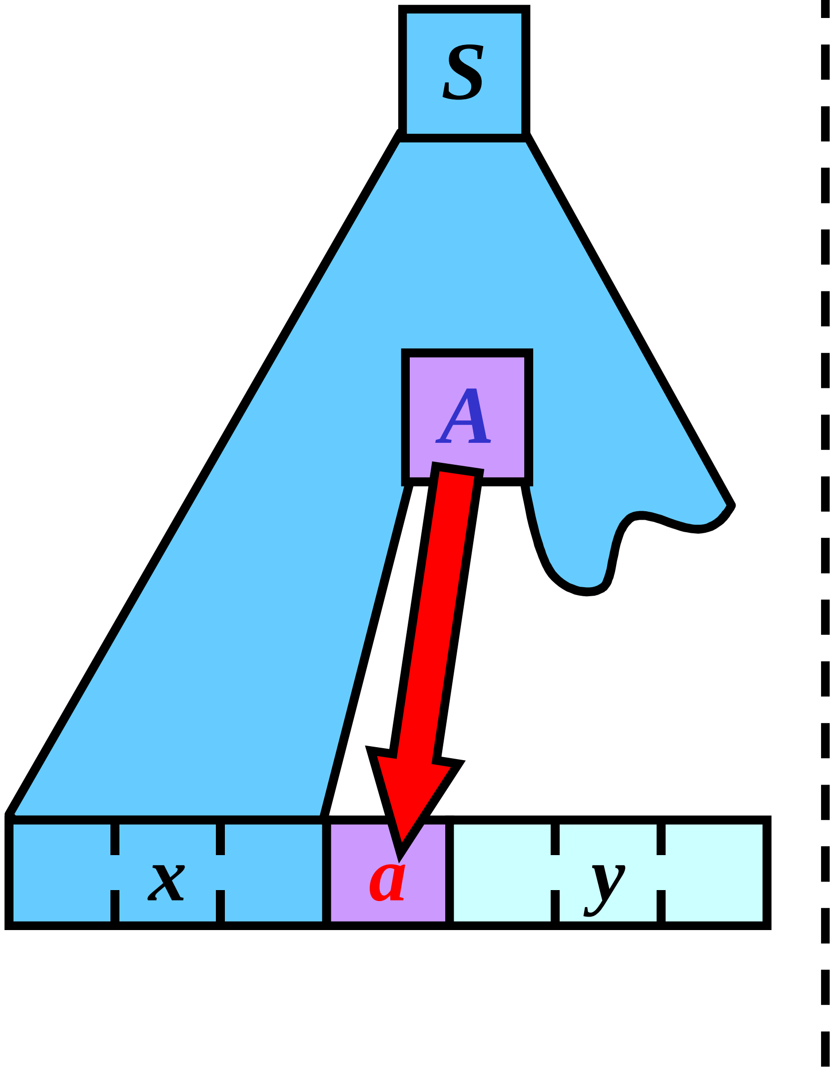
Part VII.
Top-Down Parsing

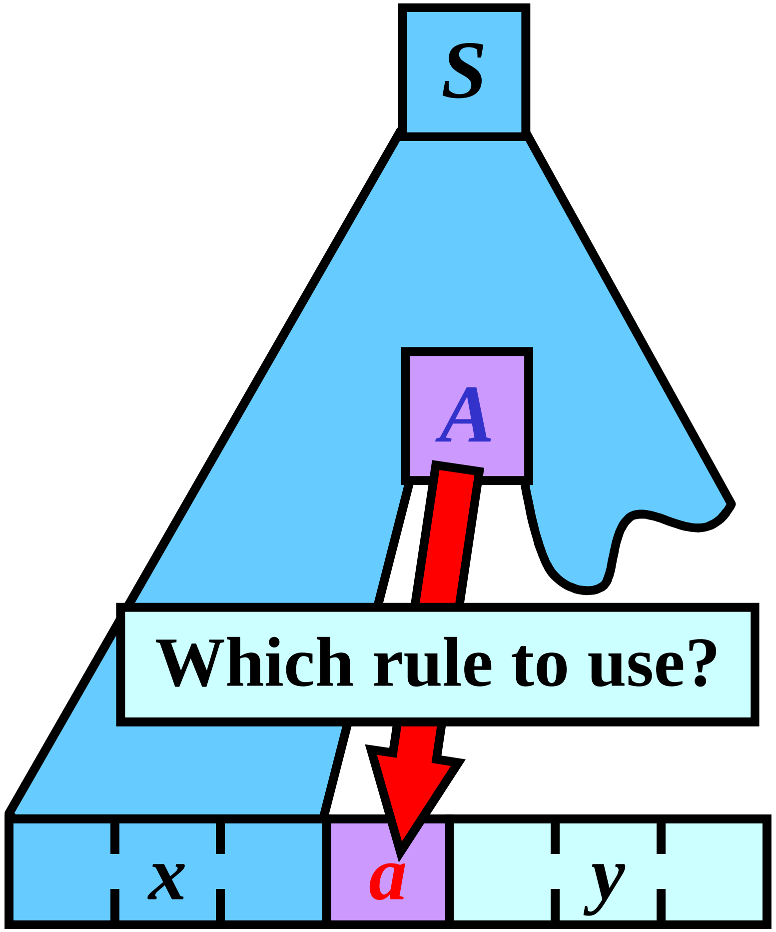
Top-Down Parsing: Introduction

Problem:



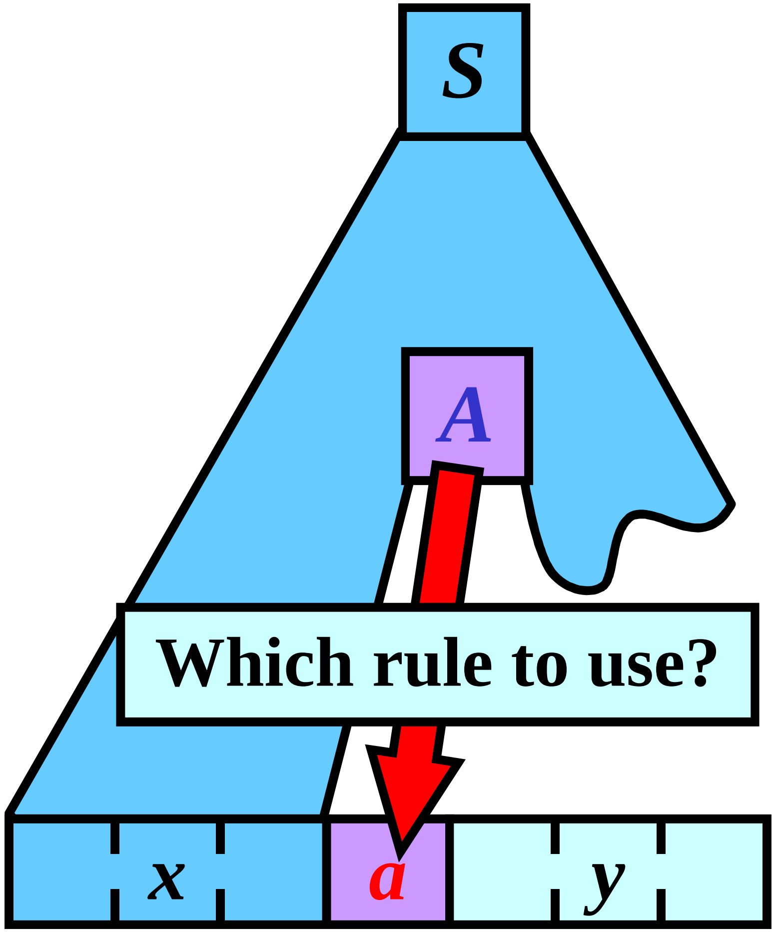
Top-Down Parsing: Introduction

Problem:



Top-Down Parsing: Introduction

Problem:



Basic idea:

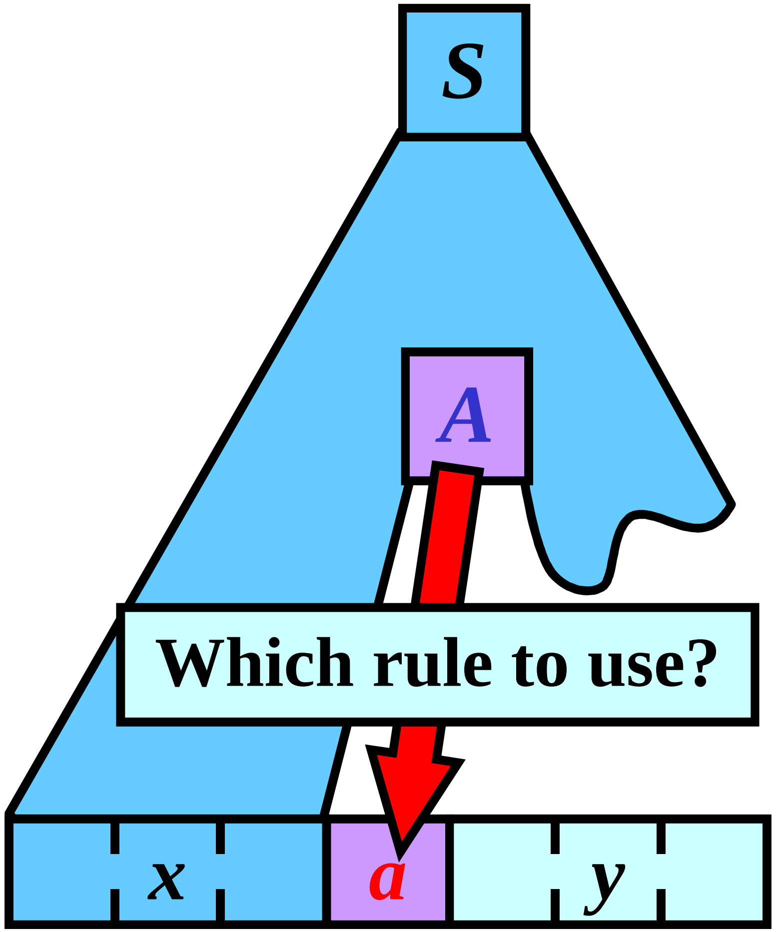
Table:

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

Use rule r: $A \rightarrow x$

Top-Down Parsing: Introduction

Problem:



Basic idea:

Table:

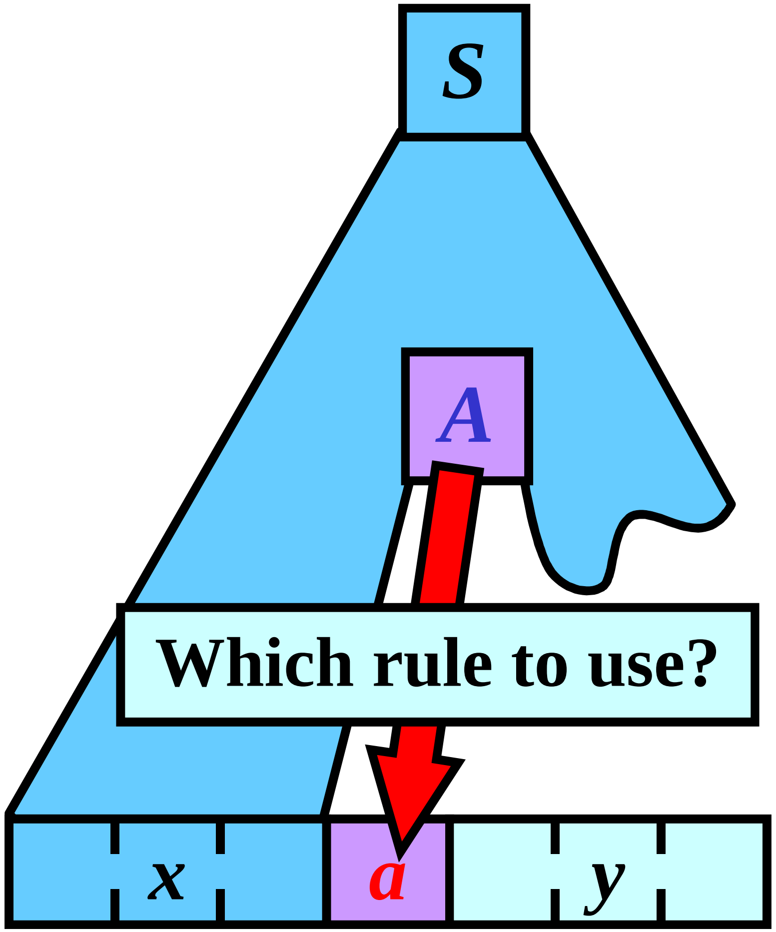
α	...	***a***	...
...			
A		α(***A***, ***a***)	
...			

Use rule ***r***: ***A*** → ***x***

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

Top-Down Parsing: Introduction

Problem:



Basic idea:

Table:

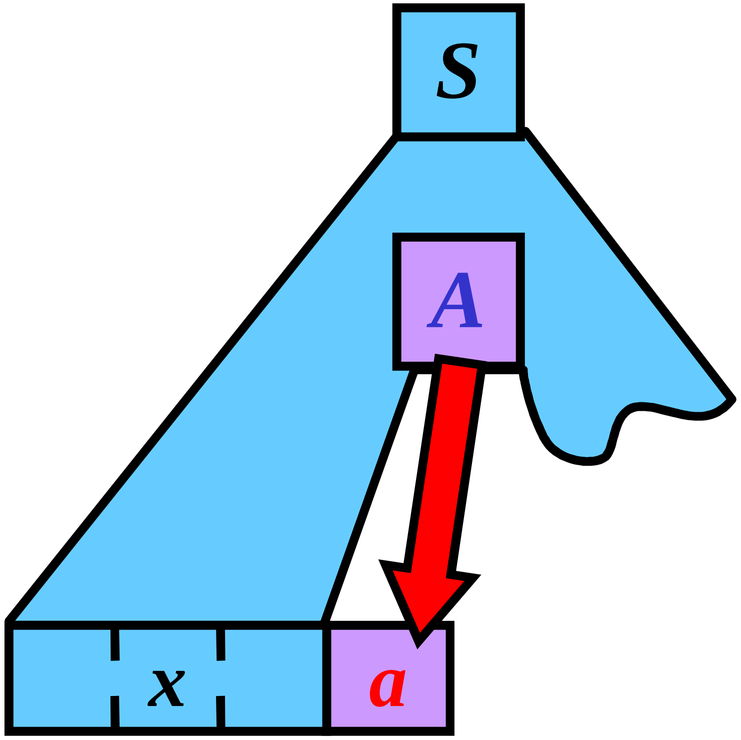
α	...	*a*	...
...			
A		α(*A*, *a*)	
...			

Use rule r: $A \to x$

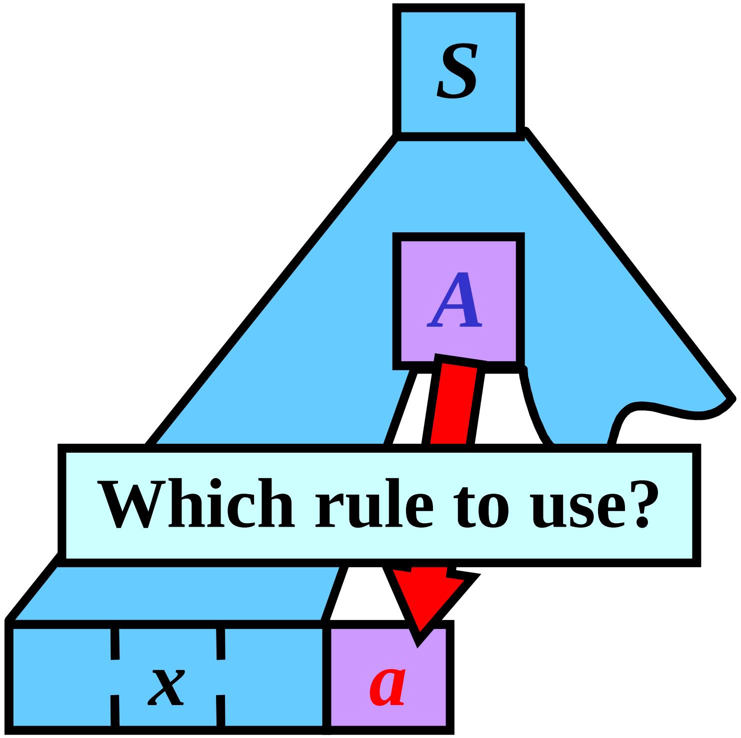
Question: Could you construct this table for **any** CFG?

Answer: NO

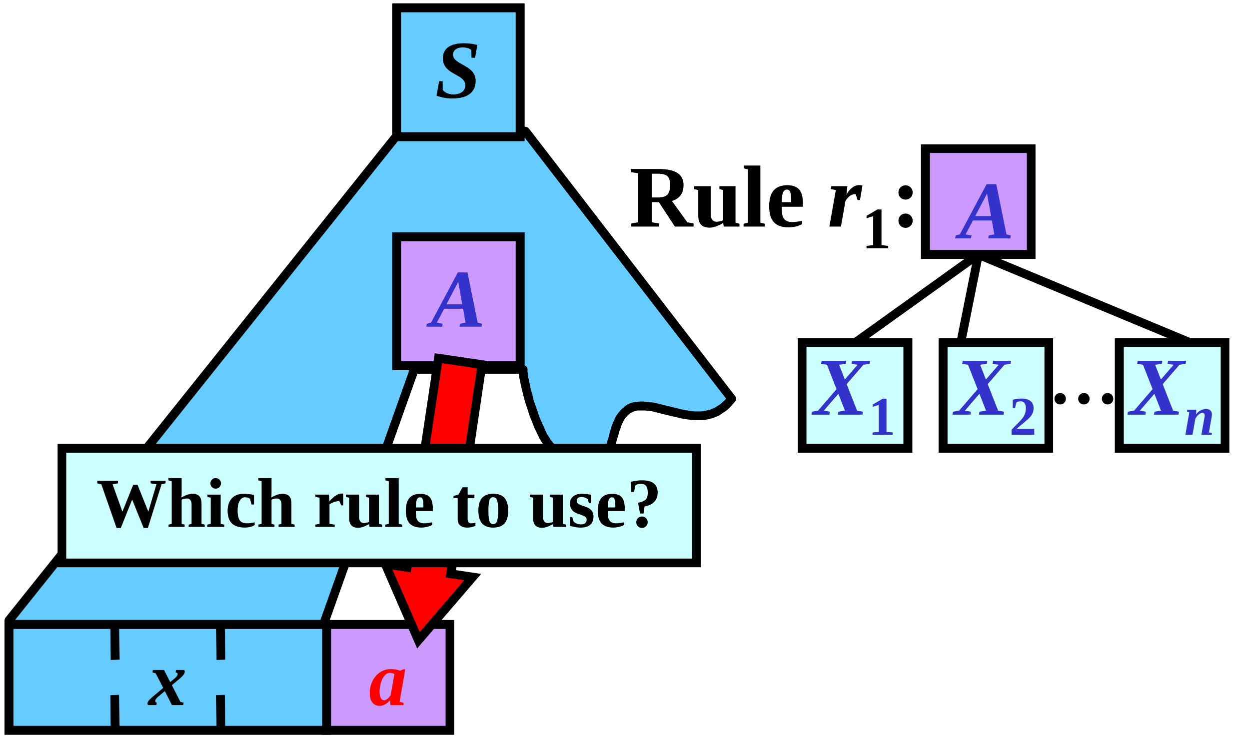
A Table-Based Selection of a Rule



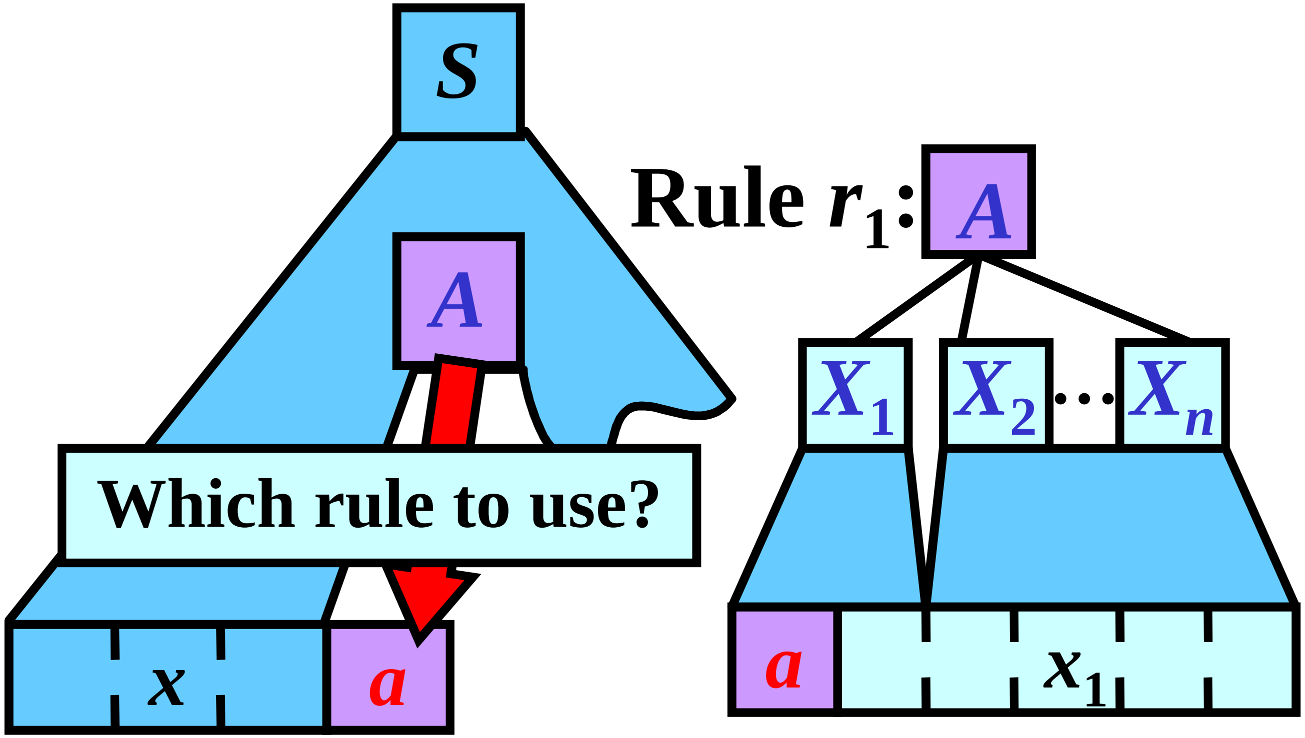
A Table-Based Selection of a Rule



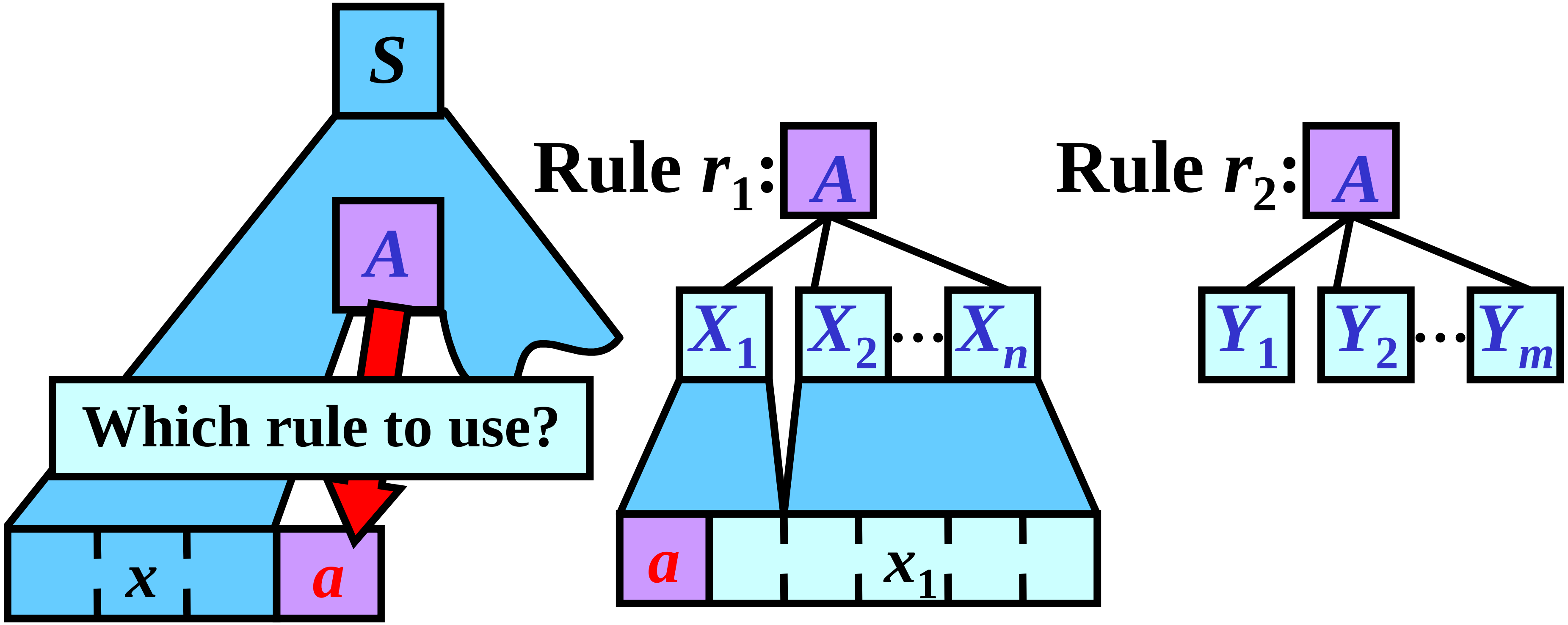
A Table-Based Selection of a Rule



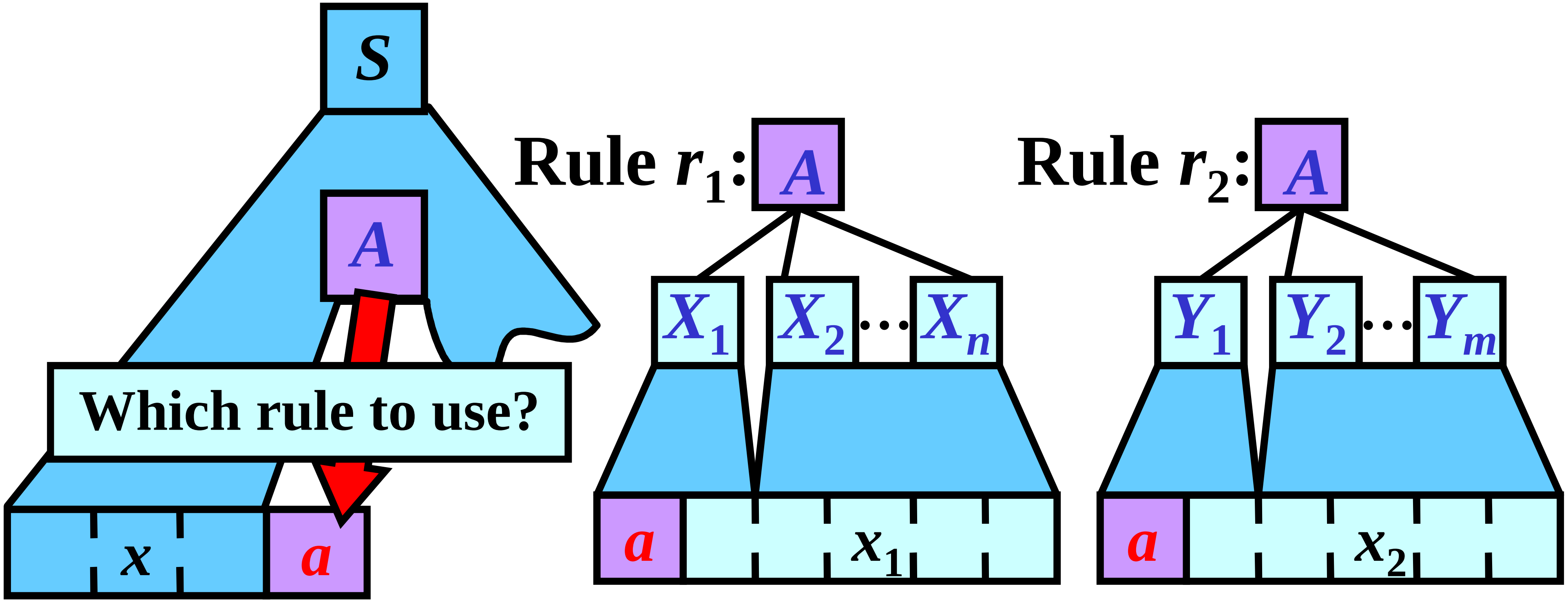
A Table-Based Selection of a Rule



A Table-Based Selection of a Rule



A Table-Based Selection of a Rule



A Table-Based Selection of a Rule

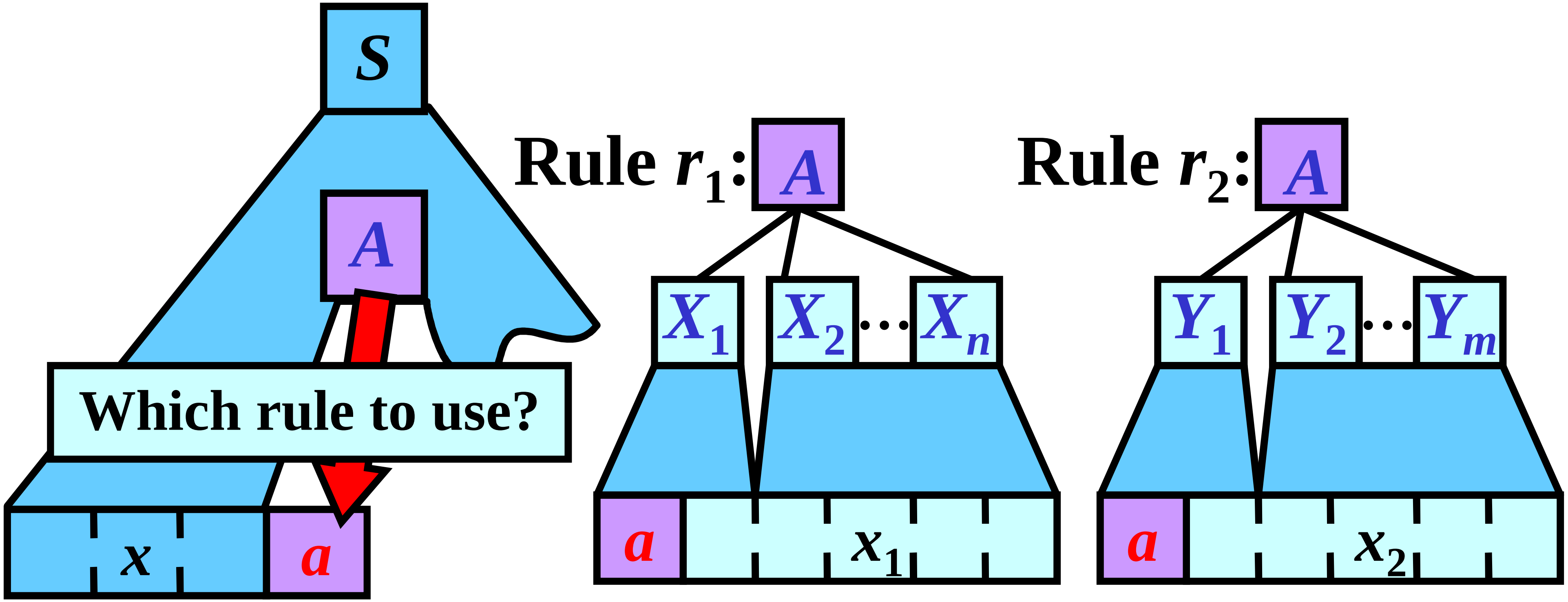


Table:

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

A Table-Based Selection of a Rule

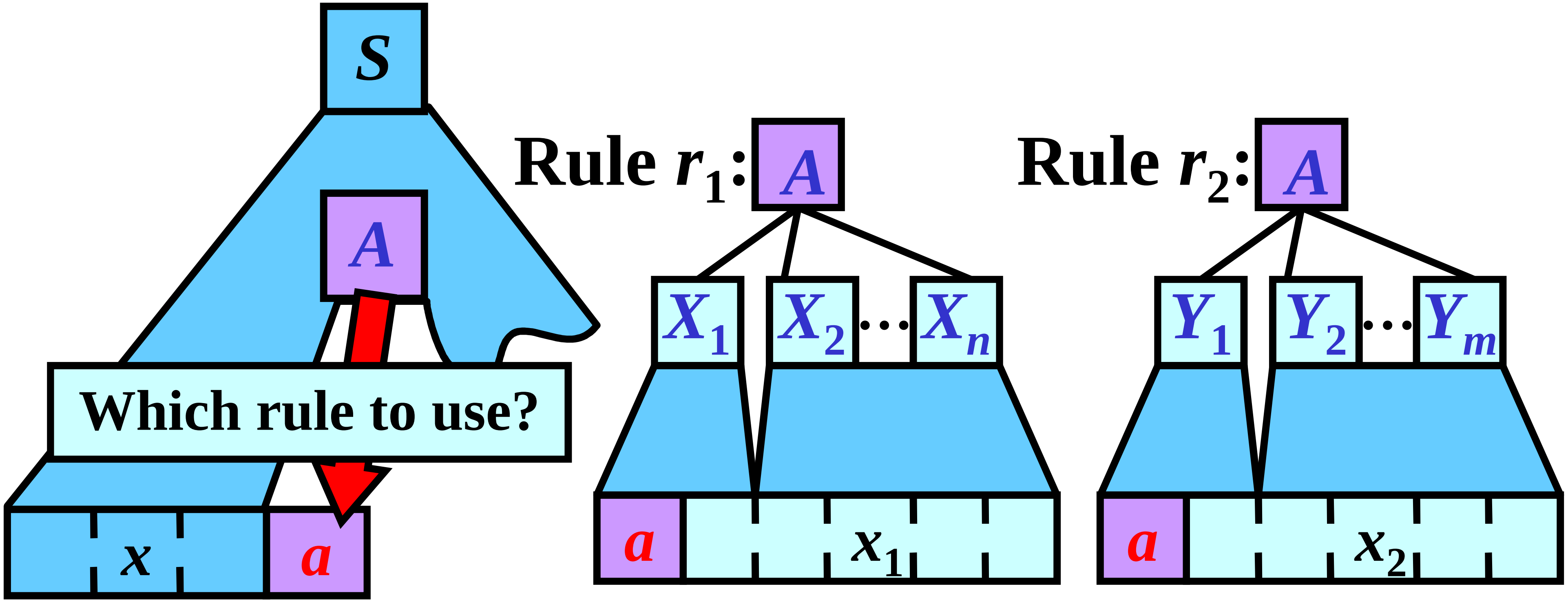
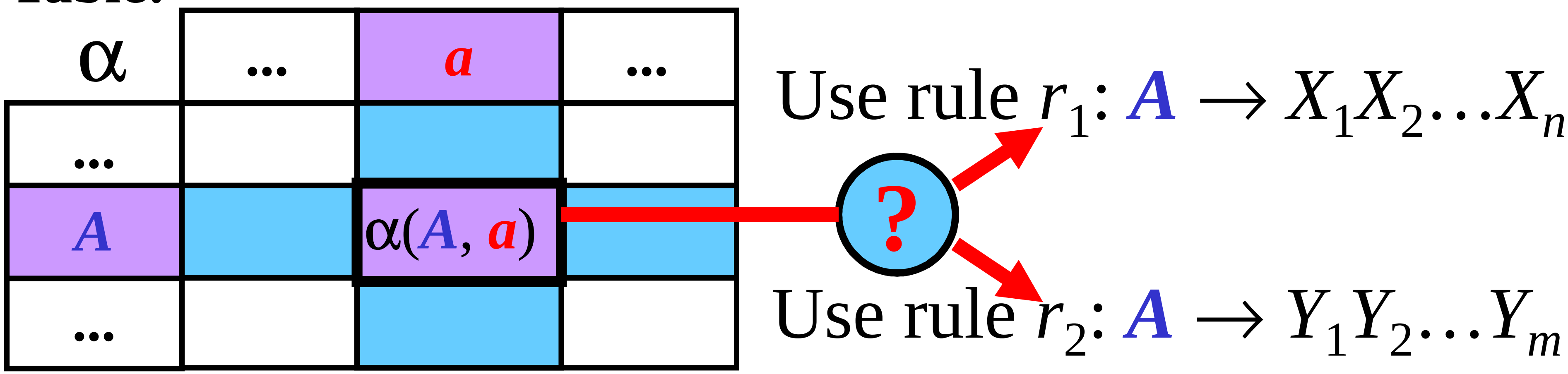


Table:



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 $\boldsymbol{x} \in (N \cup T)^*$,we define the set $First(\boldsymbol{x})$ as $First(\boldsymbol{x}) = \{\boldsymbol{a}: \boldsymbol{a} \in T, \boldsymbol{x} \Rightarrow^* \boldsymbol{a}y; y \in (N \cup T)^*\}$.

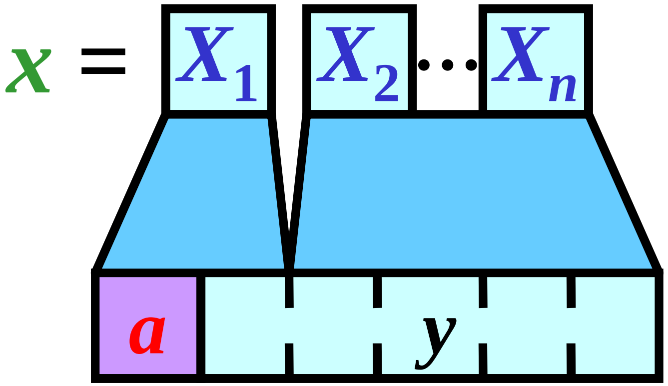
Illustration: $\boldsymbol{x} = \boxed{X_1}\ \boxed{X_2} \dots \boxed{X_n}$

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 $x \in (N \cup T)^*$, we define the set $First(x)$ as $First(x) = \{a: a \in T, x \Rightarrow^* ay; y \in (N \cup T)^*\}$.

Illustration:

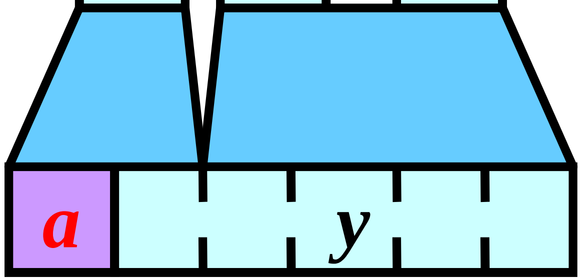


Set *First*

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 $First(x) = \{a\colon a \in T, x \Rightarrow^* ay; y \in (N \cup T)^*\}$.

Illustration: $x = X_1X_2 \ldots X_n$



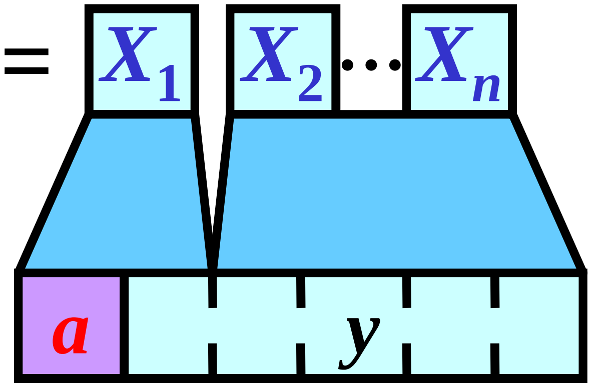
$x = X_1X_2 \ldots X_n \Rightarrow^* ay$

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: $x =$



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

$$a \in First(x)$$

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

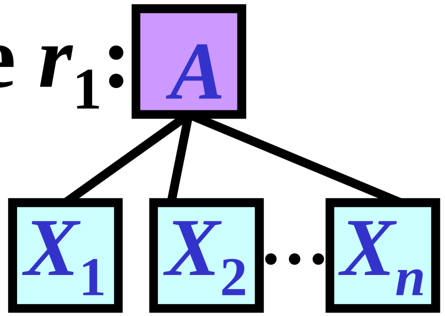
Illustration:

LL Grammars without ε-rules

Definition: Let $G = (N, T, P, S)$ be a CFG <u>without ε-rules</u>. 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:

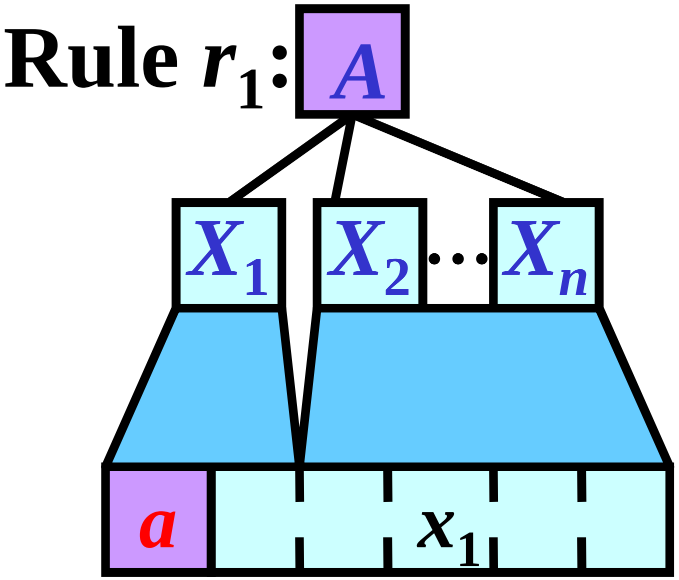
Rule r_1:



LL Grammars without ε-rules

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

Illustration:

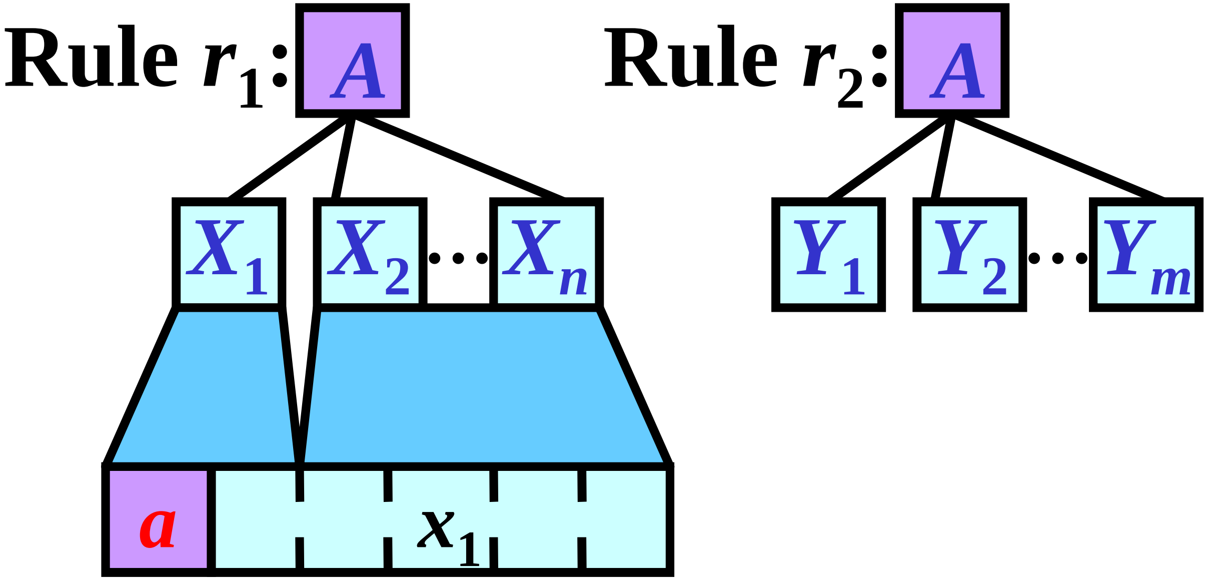


$a \in First(X_1X_2...X_n)$

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:

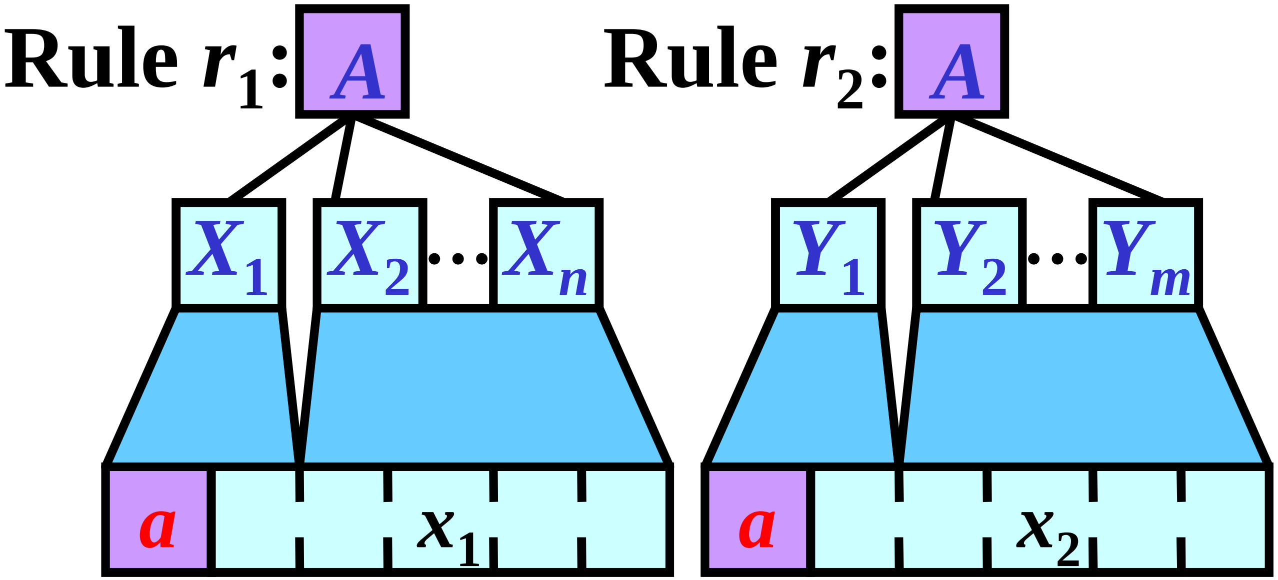


$a \in First(X_1X_2...X_n)$

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:

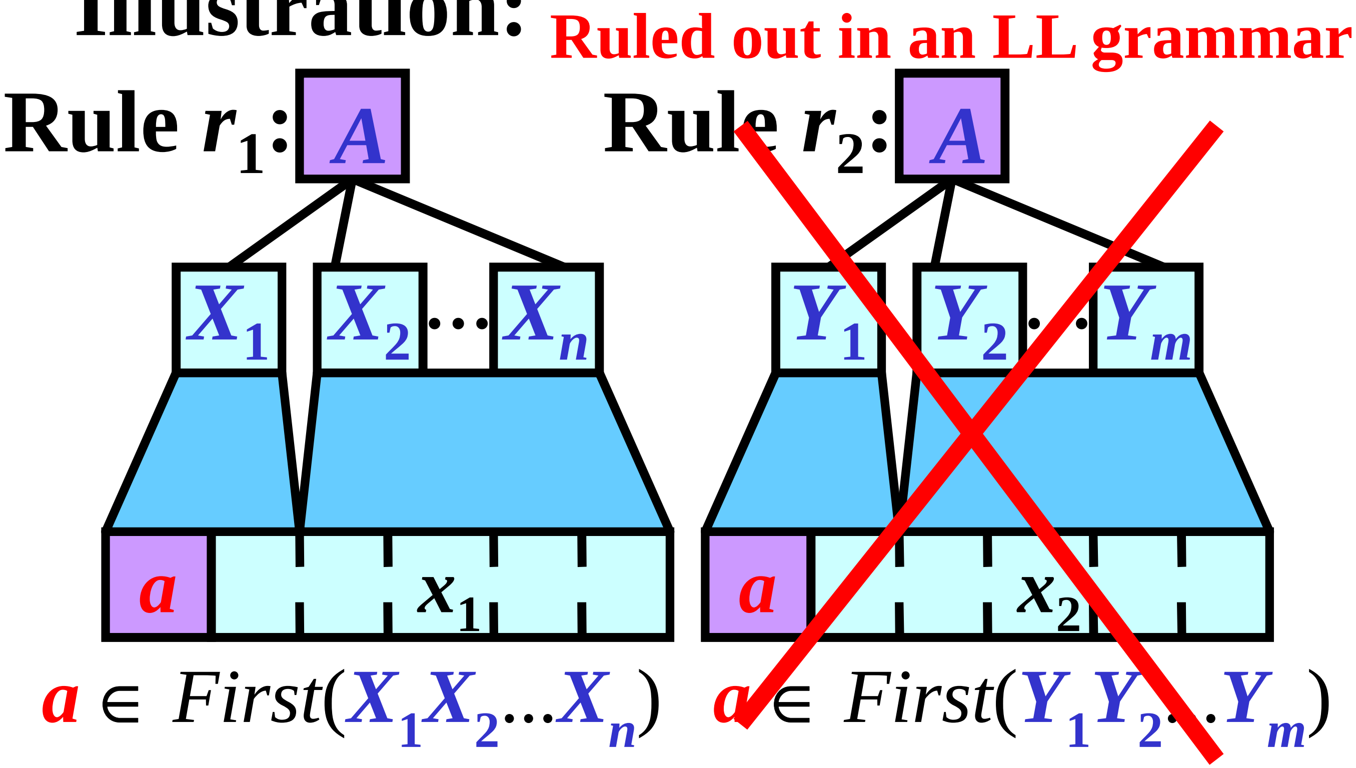


$a \in First(X_1X_2...X_n)$ $a \in First(Y_1Y_2...Y_m)$

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

Illustration:

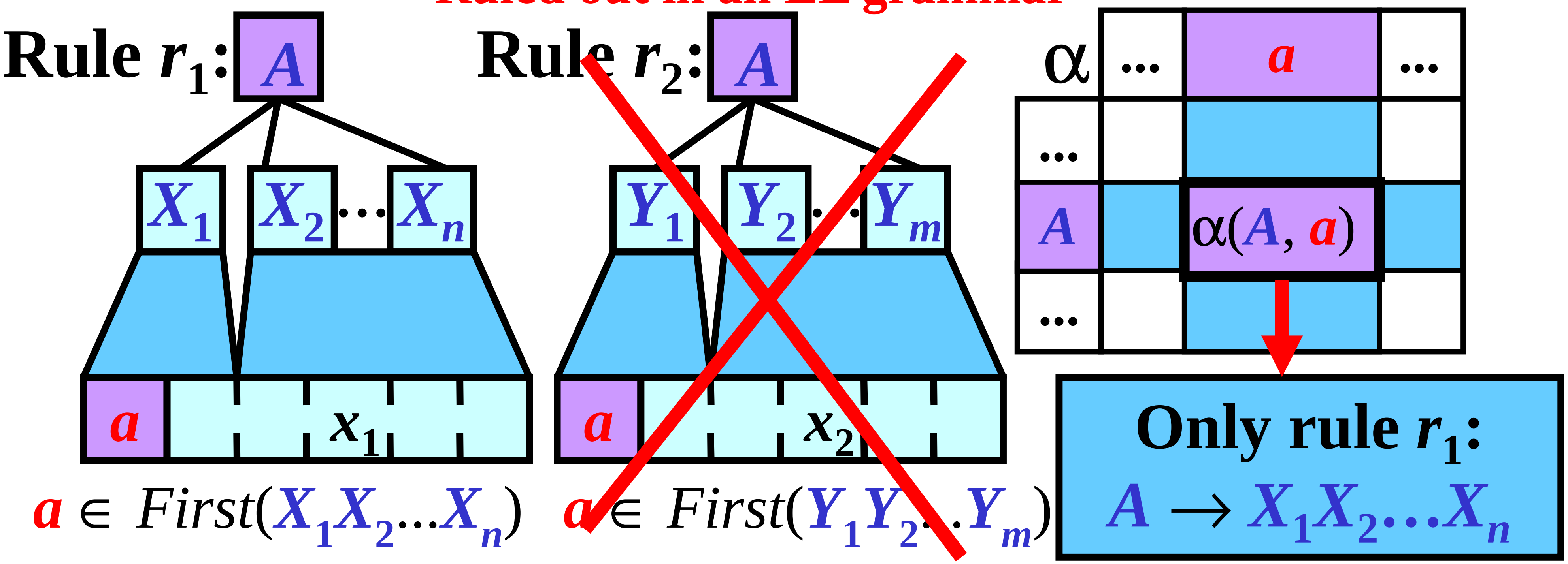


LL Grammars without ε-rules

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

Illustration:

Table:



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

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

Illustration:

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)

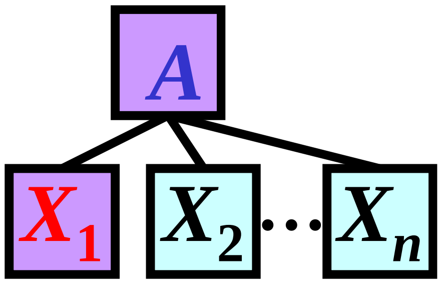
- **Input:** $G = (N, T, P, S)$ **without ε-rules**
- **Output:** *First*(X) for every $X \in N \cup T$

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

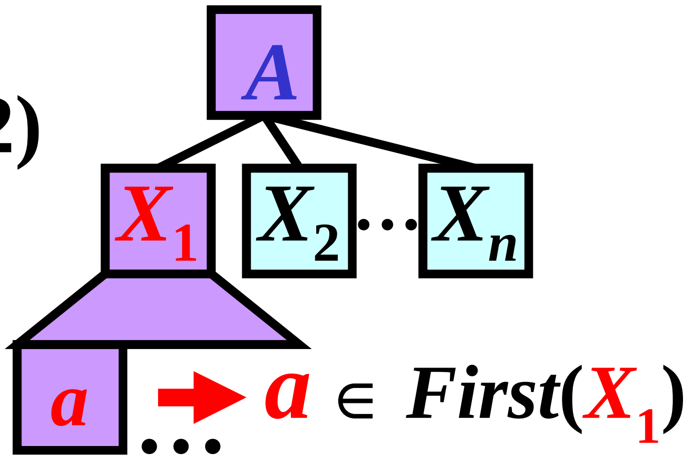
Illustration:

1) for each $a \in T$:

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

because $a \Rightarrow^0 a$

2)



Algorithm: *First*(X)

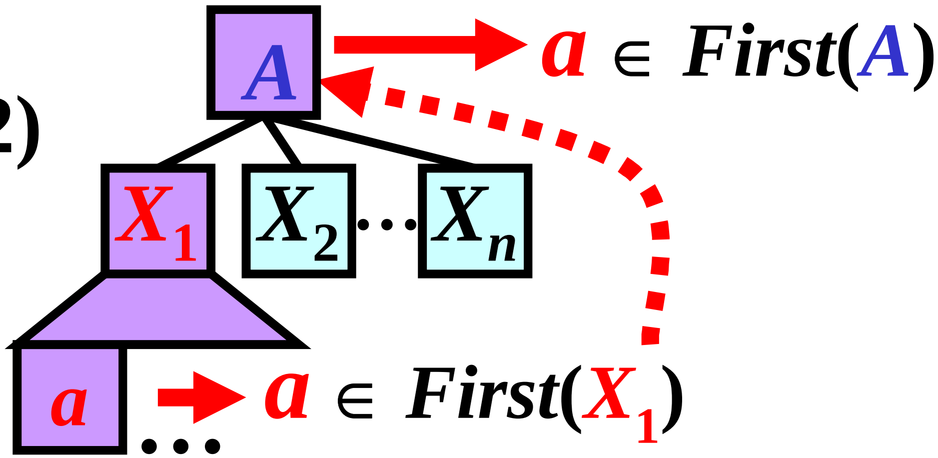
- **Input:** $G = (N, T, P, S)$ **without ε-rules**
- **Output:** $First(X)$ for every $X \in N \cup T$

- **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)



First(*X*) for SPL: Example

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

First(*X*) for SPL: Example

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

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

First(*X*) for SPL: Example

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

<item> $\rightarrow$ **id** $\in P$**:** **add** *First*(**id**) **to** *First*(**<item>**)

<item> $\rightarrow$ **int** $\in P$**:** **add** *First*(**int**) **to** *First*(**<item>**)

Summary: *First*(**<item>**) **=** {**id**, **int**}

<it-list> $\rightarrow$ **)** $\in P$**:** **add** *First*(**)**) **to** *First*(**<it-list>**)

<it-list> $\rightarrow$ **,** … $\in P$**:** **add** *First*(**,**) **to** *First*(**<it-list>**)

Summary: *First*(**<it-list>**) **=** {**)**, **,**}

First(*X*) for SPL: Example

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

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

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

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

First(*X*) for SPL: Example

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

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

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

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

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

First(*X*) for SPL: Example

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

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

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

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

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

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

Construction of LL Table

α	…	*a*	…
…			
A		α(*A*, *a*)	
…			

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 $\Rightarrow$ **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

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 $\Rightarrow$ **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		$\alpha(A, a)$	
…			

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

Task: LL table for SPL

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

id $\in$ *First*(<stat>)

id $\in$ *First*(id)

Rule r**:** $A \rightarrow X_1X_2\ldots X_n$	$First(X_1)$
1: <prog> $\rightarrow$ begin …	{begin}
2: <st-list> $\rightarrow$ <stat>…	{id, write, read}
3: <st-list> $\rightarrow$ end	{end}
4: <stat> $\rightarrow$ read …	{read}
5: <stat> $\rightarrow$ write …	{write}
6: <stat> $\rightarrow$ id …	{id}
7: <it-list> $\rightarrow$, …	{,}
8: <it-list> $\rightarrow$)	{)}
9: <item> $\rightarrow$ int	{int}
10: <item> $\rightarrow$ id	{id}

Construction of LL Table

α	…	a	…
…			
A		$\alpha(A, a)$	
…			

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

Task: LL table for SPL

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

2: id ∈ *First*(<stat>)

6: id ∈ *First*(id)

10: 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		$\alpha(A, a)$	
…			

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

Task: LL table for SPL

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

id $\in$ *First*(<stat>)

id $\in$ *First*(id)

id $\in$ *First*(id)

Construct the rest analogically.

Rule r: $A \rightarrow X_1X_2\ldots X_n$	$First(X_1)$
1: <prog> $\rightarrow$ begin …	{begin}
2: <st-list> $\rightarrow$ <stat>…	{id, write, read}
3: <st-list> $\rightarrow$ end	{end}
4: <stat> $\rightarrow$ read …	{read}
5: <stat> $\rightarrow$ write …	{write}
6: <stat> $\rightarrow$ id …	{id}
7: <it-list> $\rightarrow$, …	{,}
8: <it-list> $\rightarrow$)	{)}
9: <item> $\rightarrow$ int	{int}
10: <item> $\rightarrow$ id	{id}

Parsing Based on LL Table: Example

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

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

<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
```

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						

<prog>

Source program:

`begin write 25; end`

Lexical Analyzer

begin

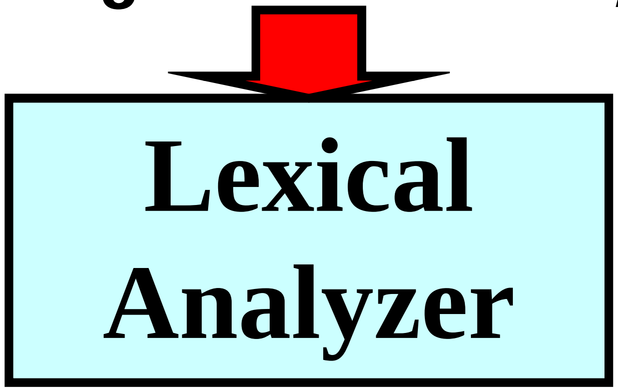
Parsing Based on LL Table: Example

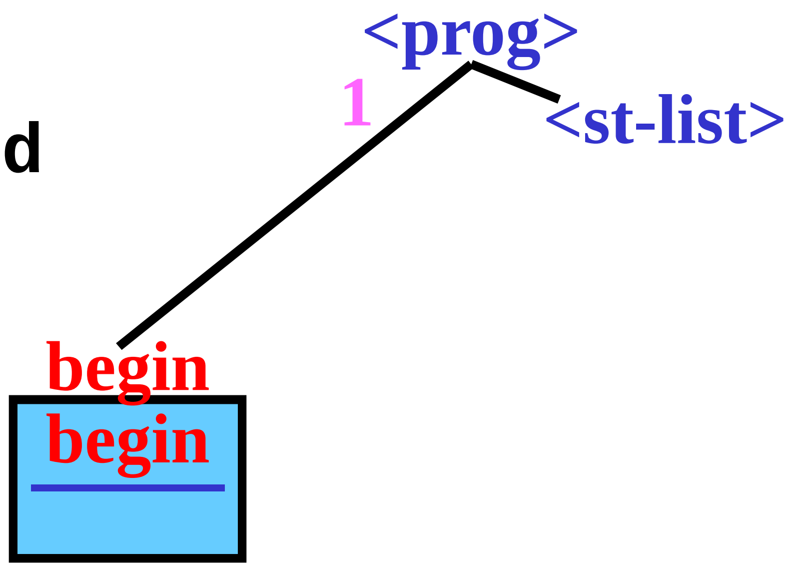
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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
```





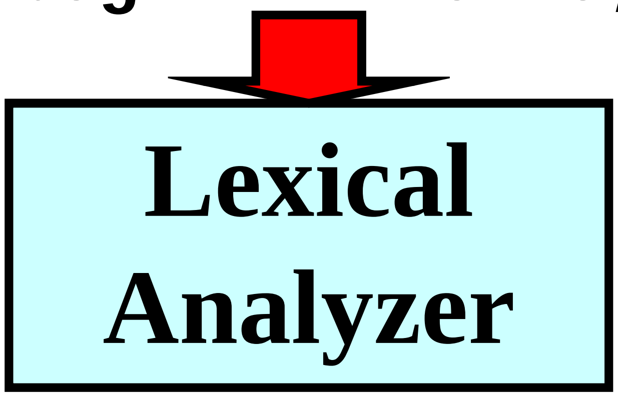
Parsing Based on LL Table: Example

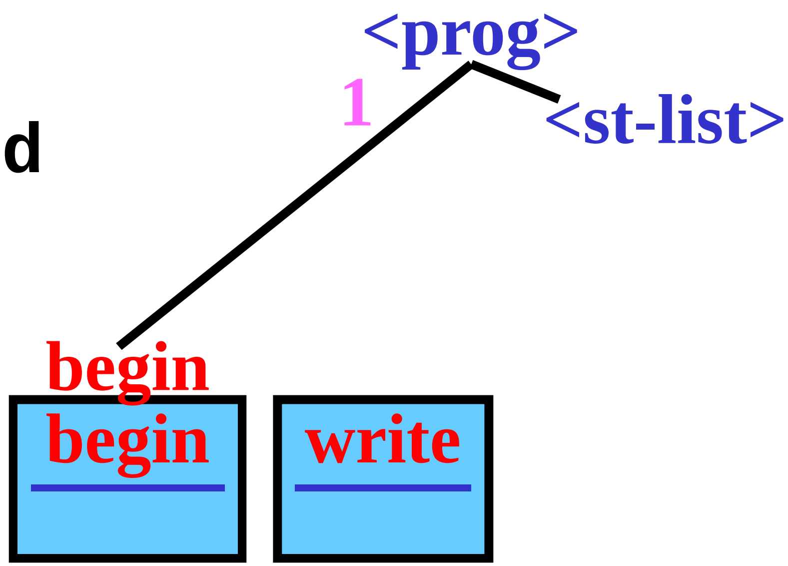
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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`





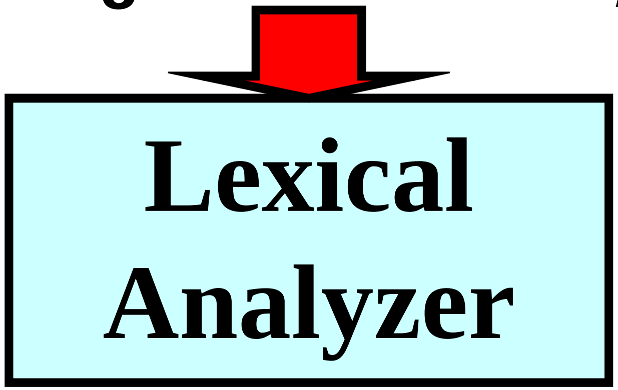
Parsing Based on LL Table: Example

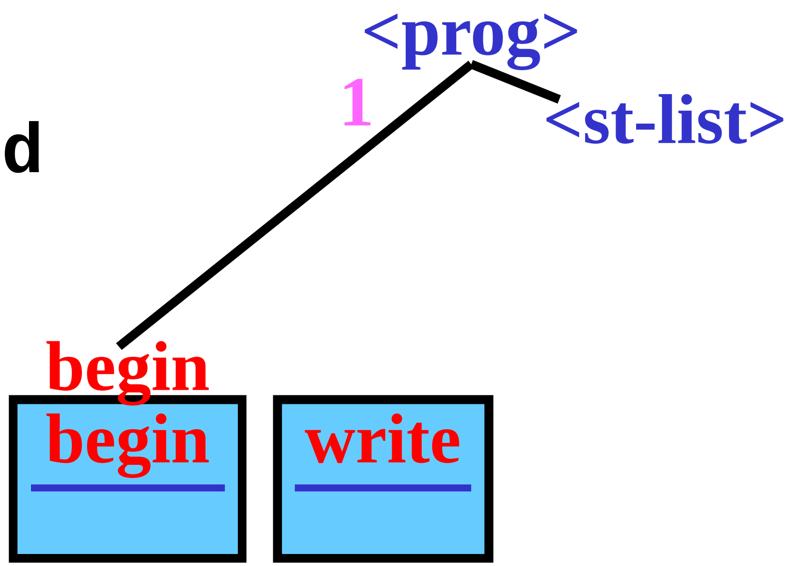
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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
```





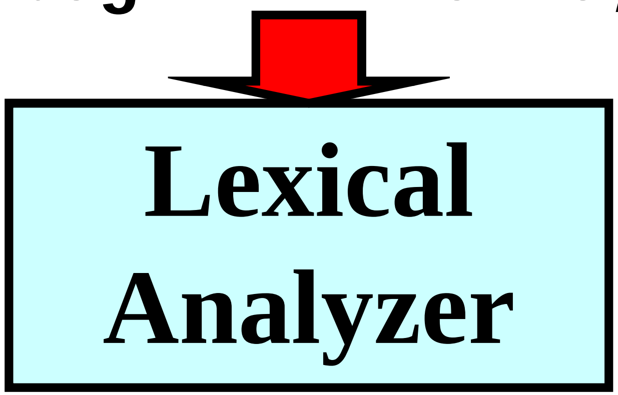
Parsing Based on LL Table: Example

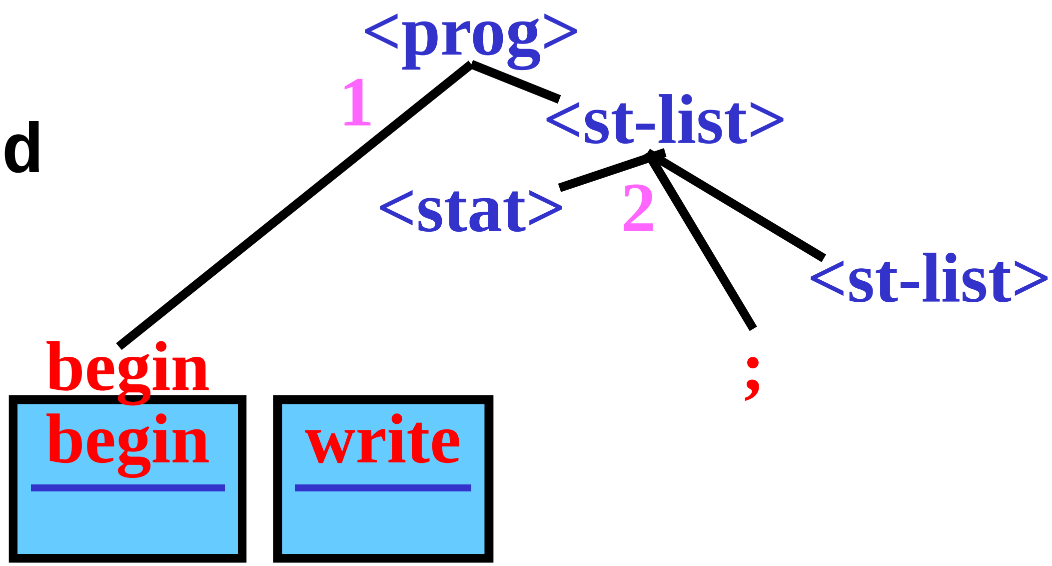
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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
```





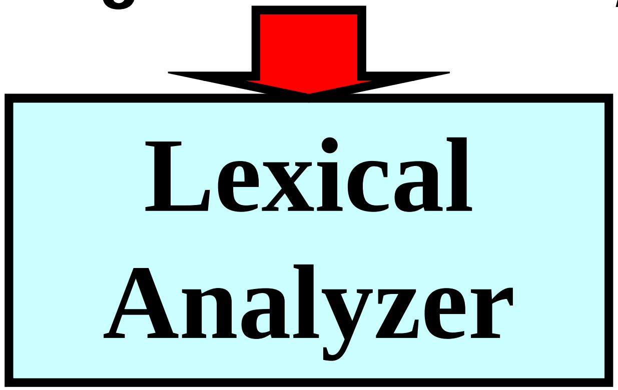
Parsing Based on LL Table: Example

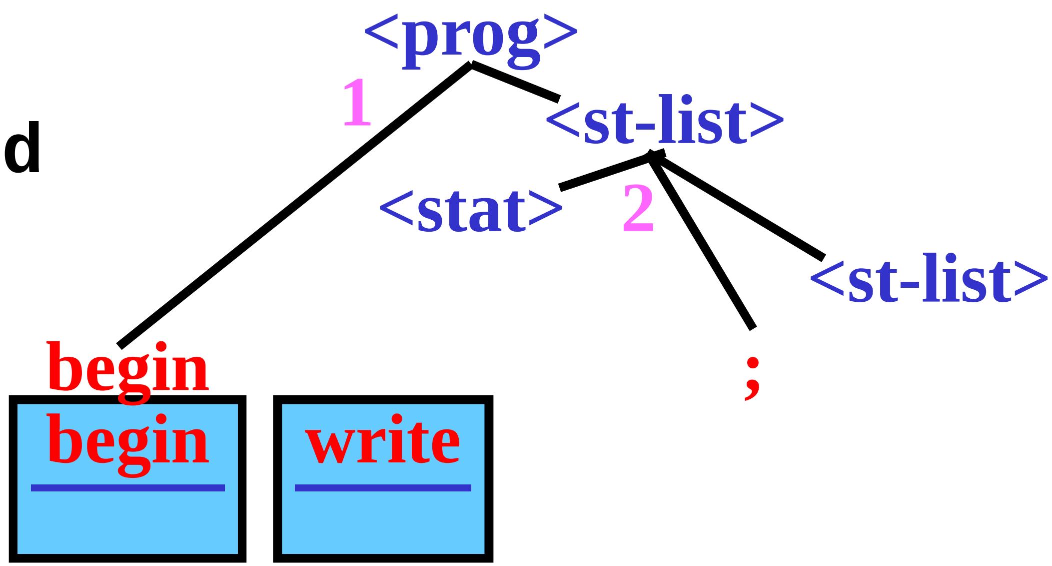
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

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

Source program:

```
begin write 25; end
```





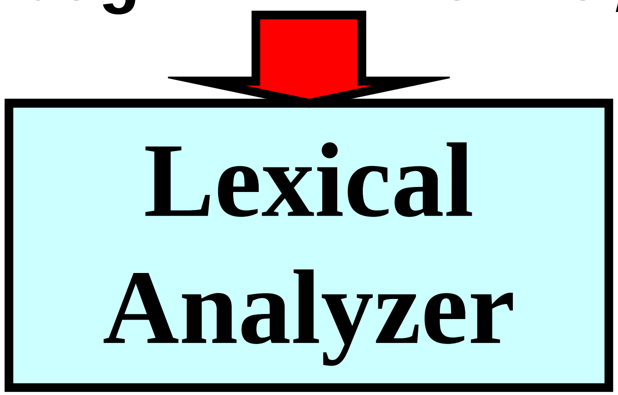
Parsing Based on LL Table: Example

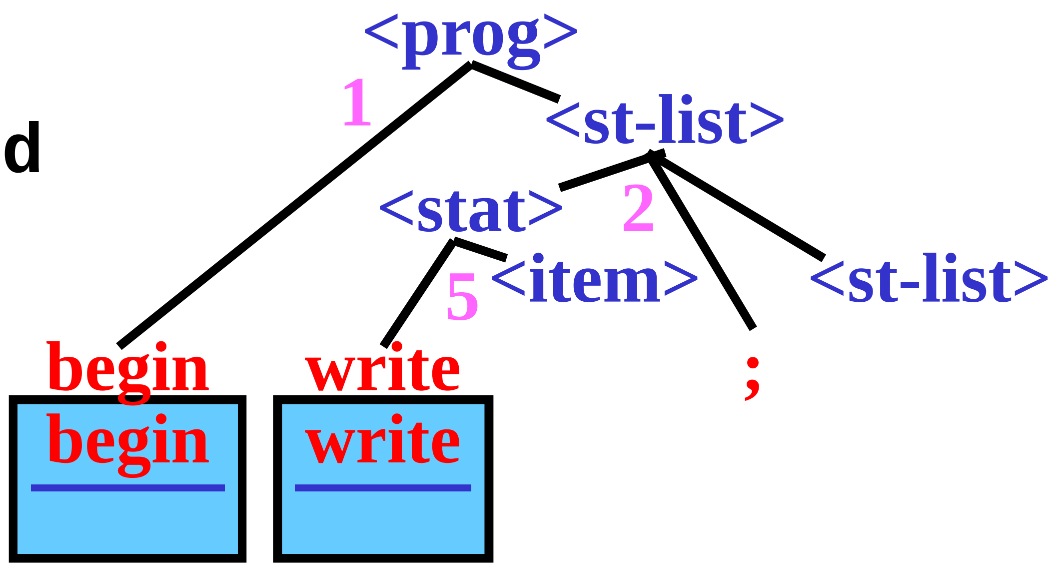
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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
```





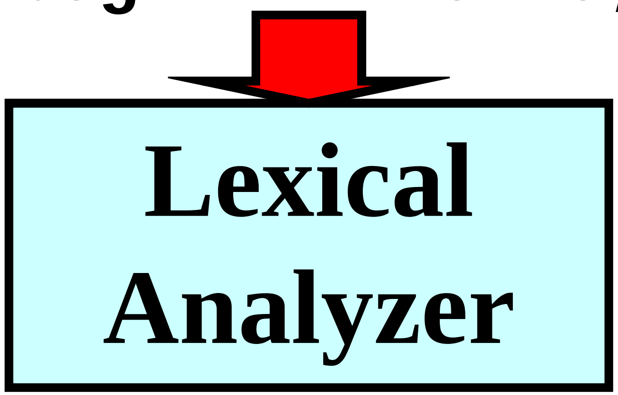
Parsing Based on LL Table: Example

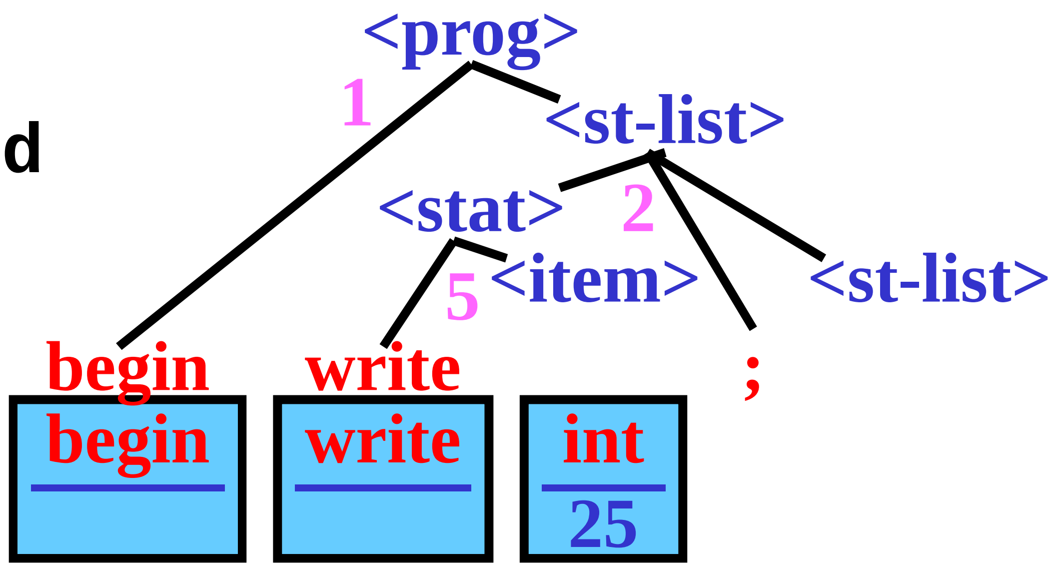
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read id**
5: <stat> → **write** <item>
6: <stat> → **id := add (** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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`





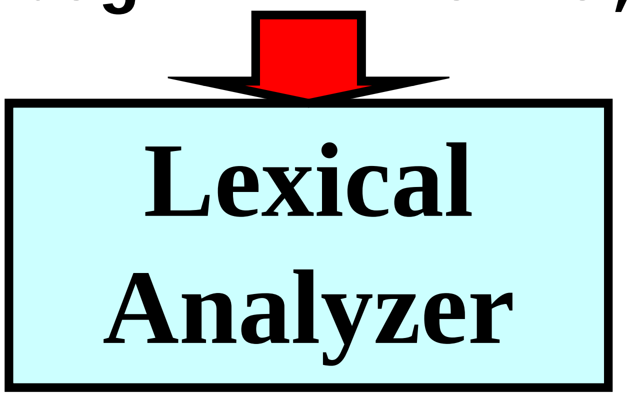
Parsing Based on LL Table: Example

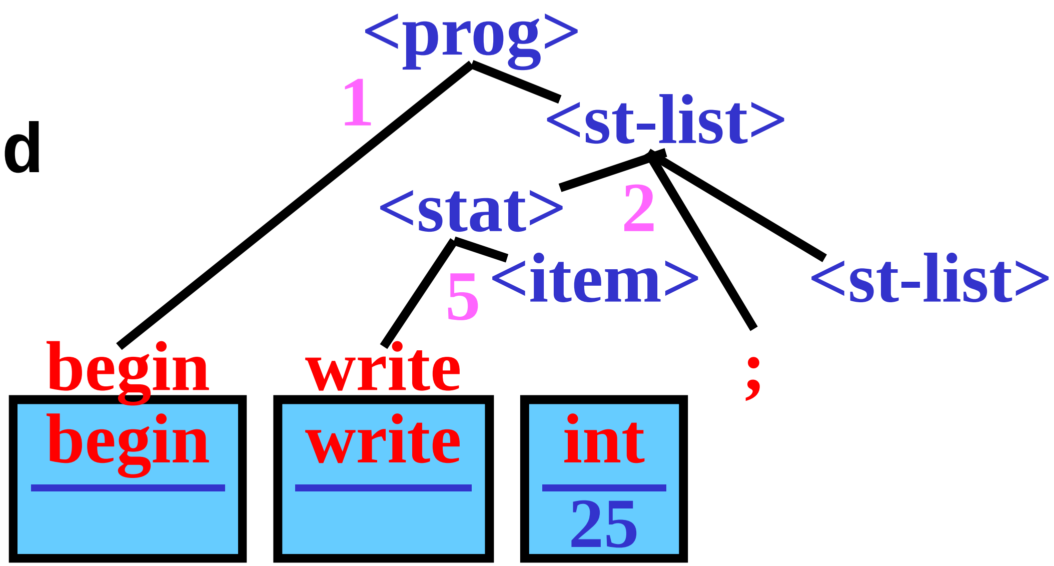
1: <prog> → begin <st-list>
2: <st-list> → <stat> ; <st-list>
3: <st-list> → end
4: <stat> → read id
5: <stat> → write <item>
6: <stat> → id := add (…
7: <it-list> → , <item> <it-list>
8: <it-list> →)
9: <item> → int
10: <item> → id

	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
```





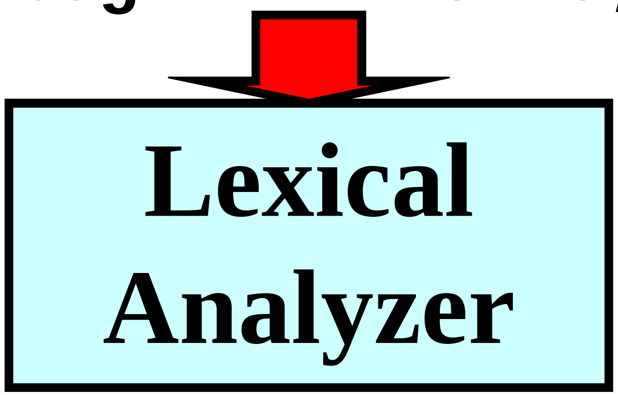
Parsing Based on LL Table: Example

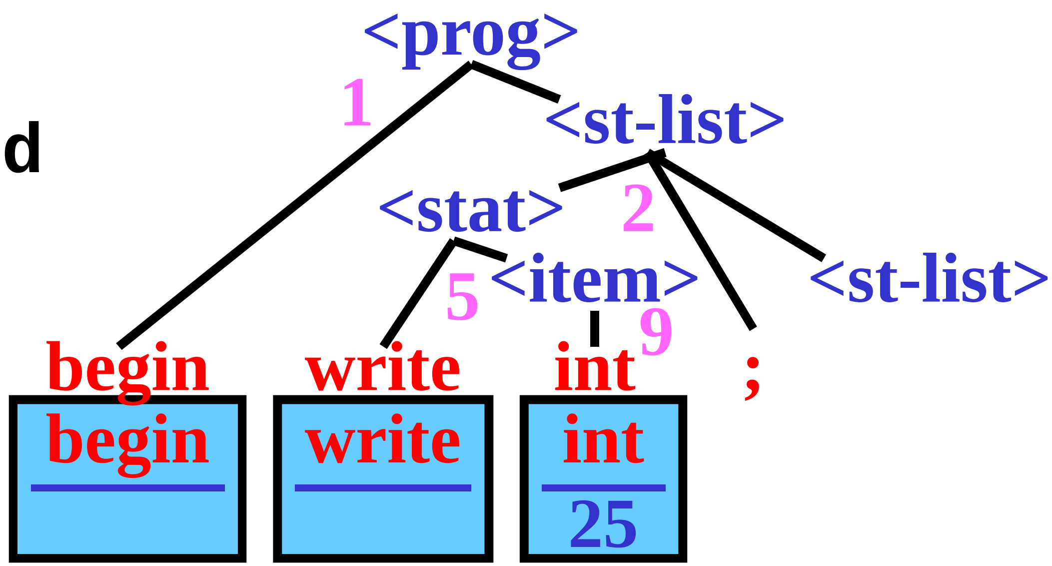
1: <prog> → begin <st-list>
2: <st-list> → <stat> ; <st-list>
3: <st-list> → end
4: <stat> → read id
5: <stat> → write <item>
6: <stat> → id := add (…
7: <it-list> → , <item> <it-list>
8: <it-list> →)
9: <item> → int
10: <item> → id

	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
```





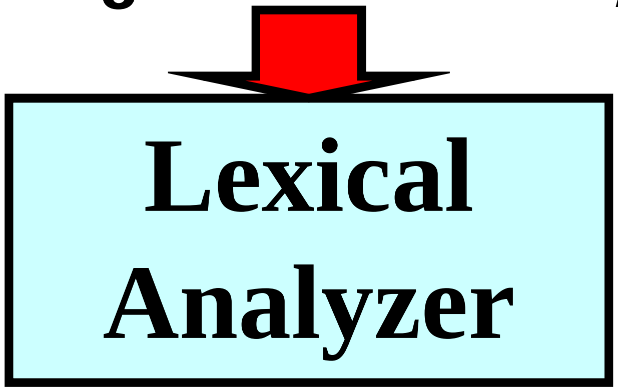
Parsing Based on LL Table: Example

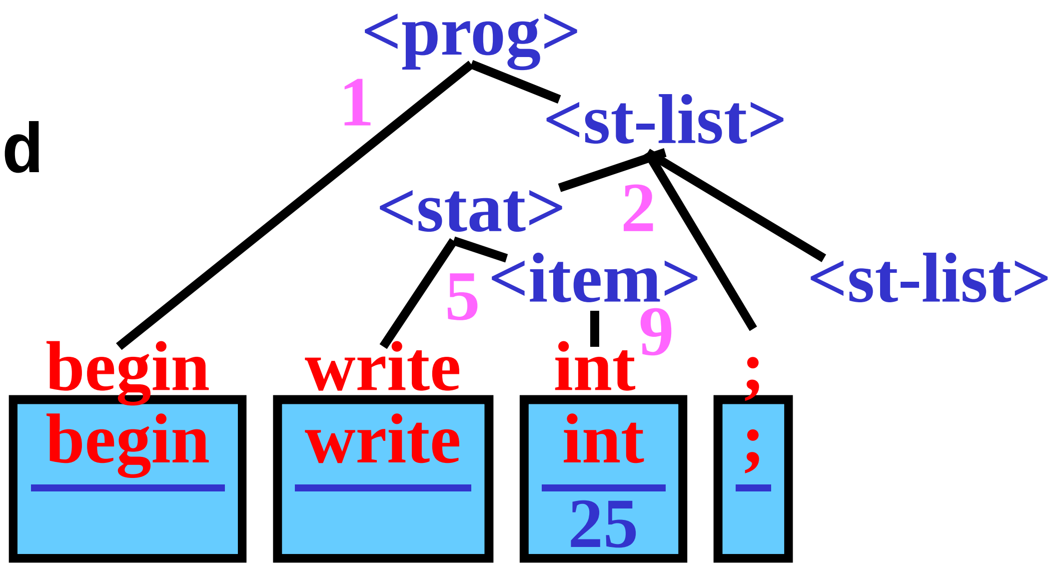
1: <prog> → **begin** <st-list>		6: <stat> → **id** **:=** **add** **(** …	
2: <st-list> → <stat> **;** <st-list>		7: <it-list> → **,** <item> <it-list>	
3: <st-list> → **end**		8: <it-list> → **)**	
4: <stat> → **read** **id**		9: <item> → **int**	
5: <stat> → **write** <item>		10: <item> → **id**	

	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
```





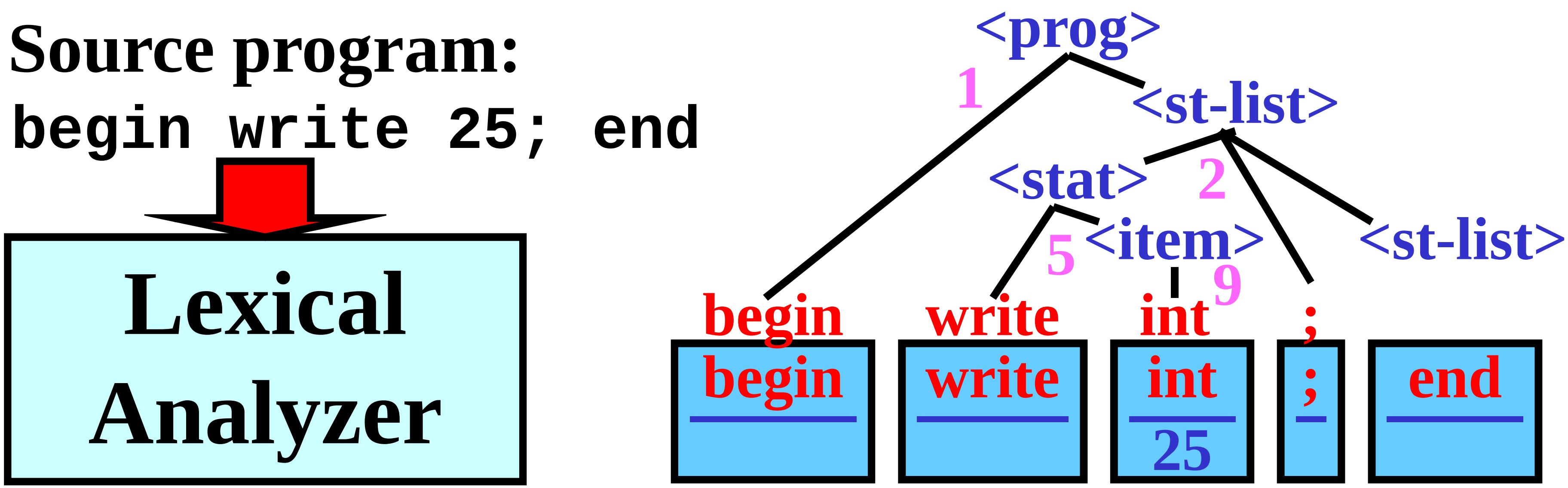
Parsing Based on LL Table: Example

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`



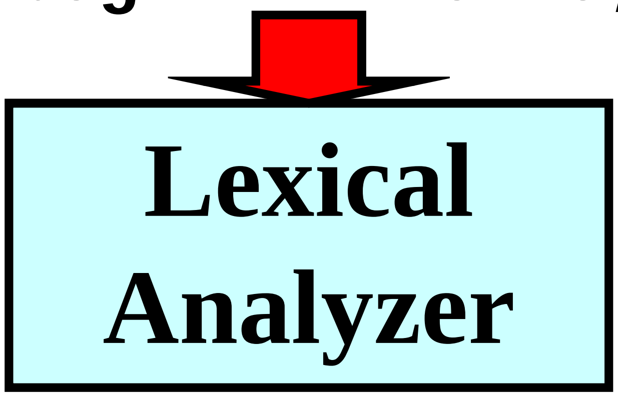
Parsing Based on LL Table: Example

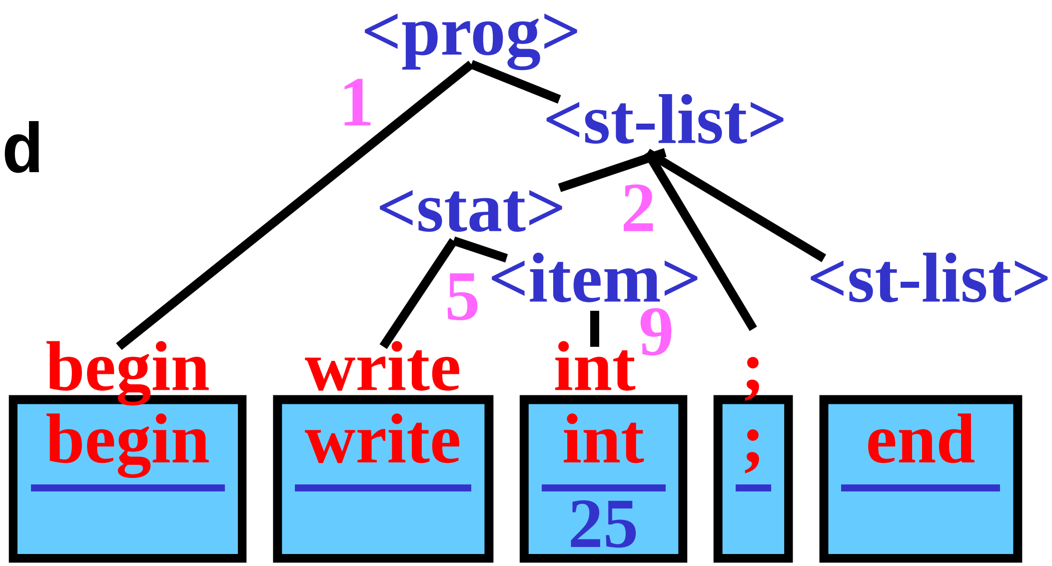
1: <prog> → **begin** <st-list>
2: <st-list> → <stat> **;** <st-list>
3: <st-list> → **end**
4: <stat> → **read** **id**
5: <stat> → **write** <item>
6: <stat> → **id** **:=** **add** **(** …
7: <it-list> → **,** <item> <it-list>
8: <it-list> → **)**
9: <item> → **int**
10: <item> → **id**

	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`





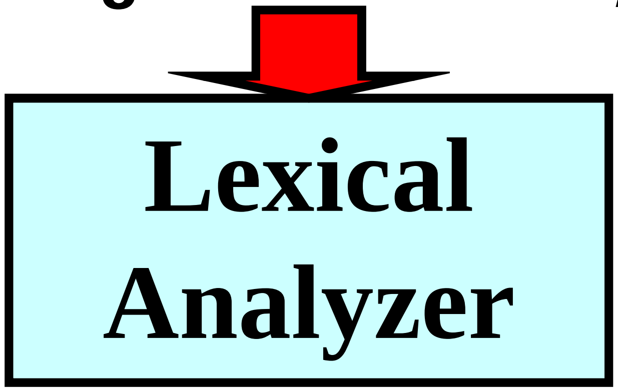
Parsing Based on LL Table: Example

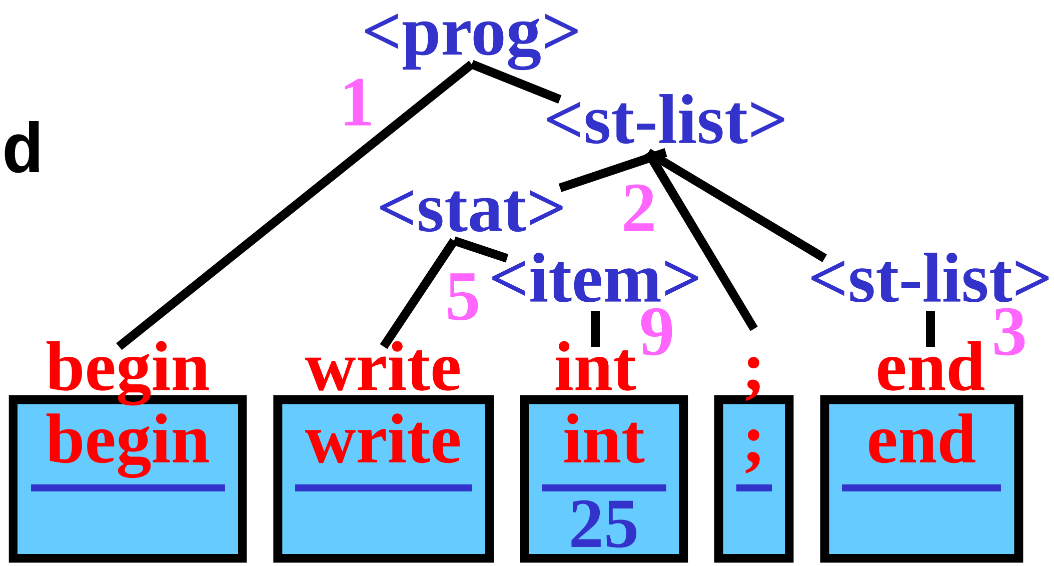
1: <prog> → begin <st-list>
2: <st-list> → <stat> ; <st-list>
3: <st-list> → end
4: <stat> → read id
5: <stat> → write <item>
6: <stat> → id := add (…
7: <it-list> → , <item> <it-list>
8: <it-list> →)
9: <item> → int
10: <item> → id

	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
```

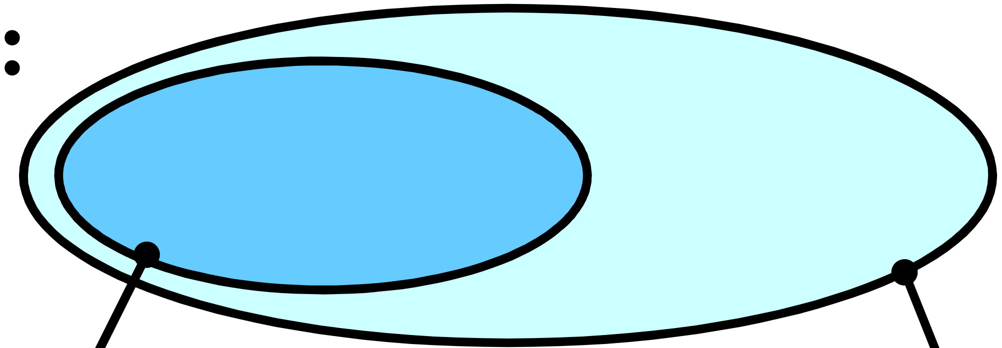




LL Grammars: Useful Transformations

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 $\boldsymbol{A} \rightarrow \boldsymbol{Ax}$, where $\boldsymbol{A} \in N$, $\boldsymbol{x} \in (N \cup T)^*$ is called a *left recursive rule*.

Factorization

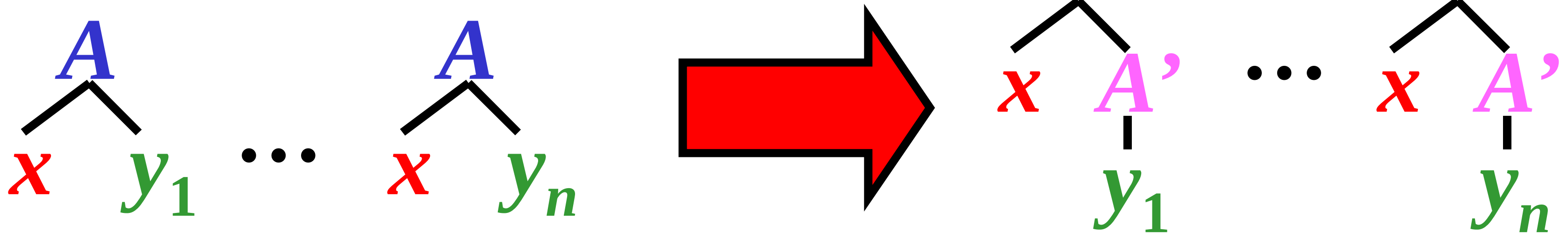
Idea: Replace rules of the form

$$A \rightarrow xy_1,\ A \rightarrow xy_2,\ \ldots,\ A \rightarrow xy_n \text{ with}$$

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

where A' is a new nonterminal

Illustration:



Factorization

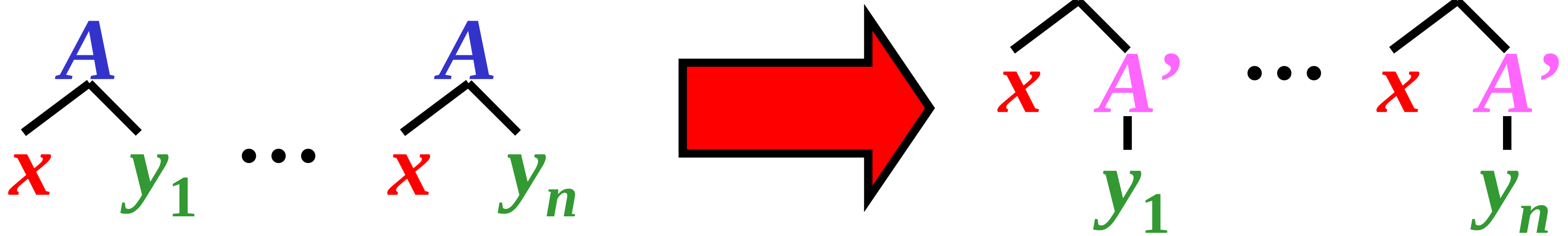
Idea: Replace rules of the form

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

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

where A' is a new nonterminal

Illustration:



Example:

<stat> → **write id**
<stat> → **write int**

Factorization

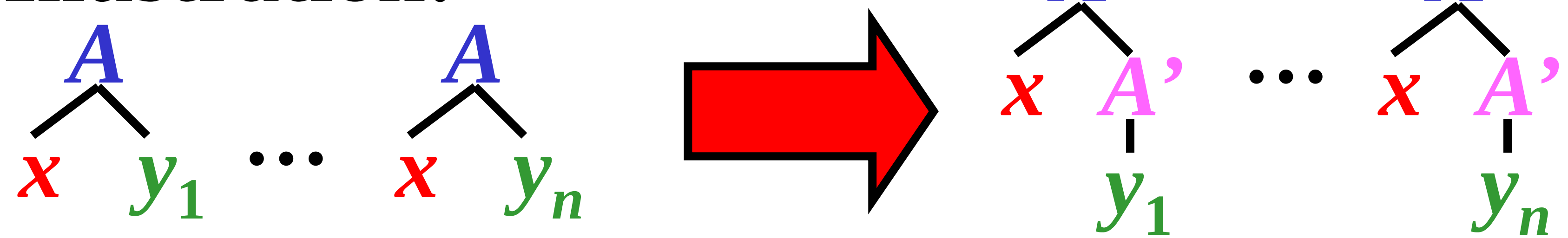
Idea: Replace rules of the form

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

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

where A' is a new nonterminal

Illustration:



Example:

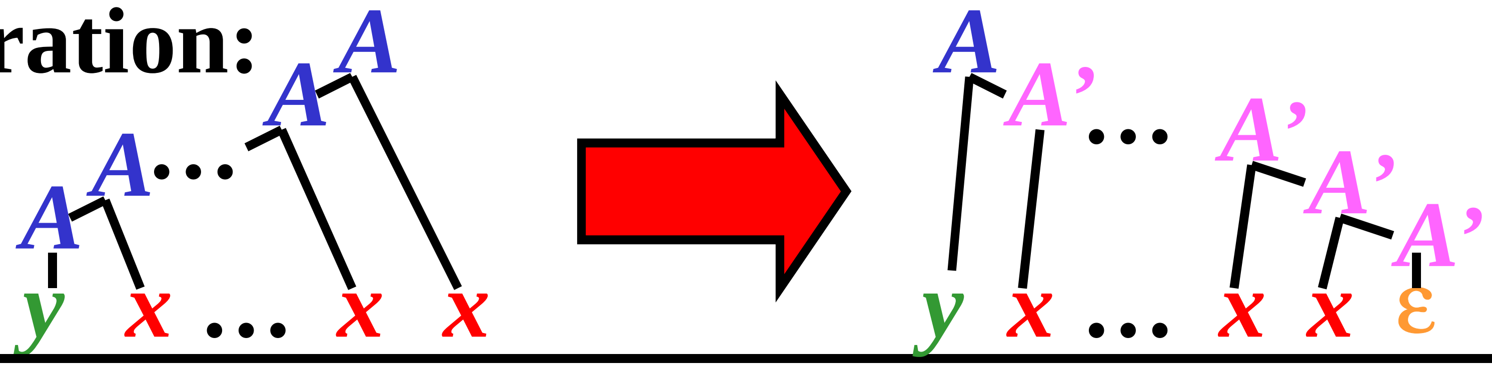
<stat> → write id
<stat> → write int

<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.

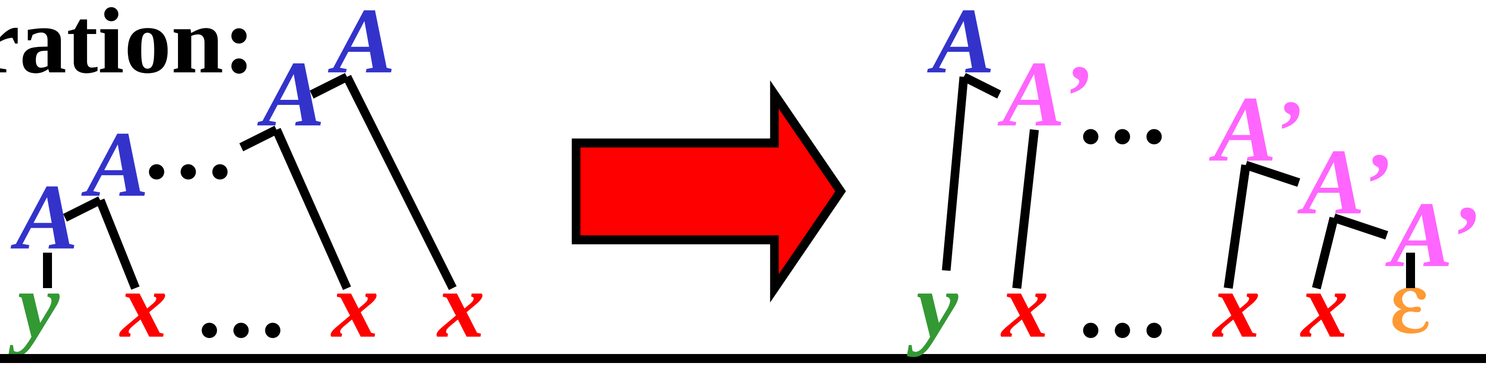
Illustration:



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:

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F$

$T \rightarrow F$

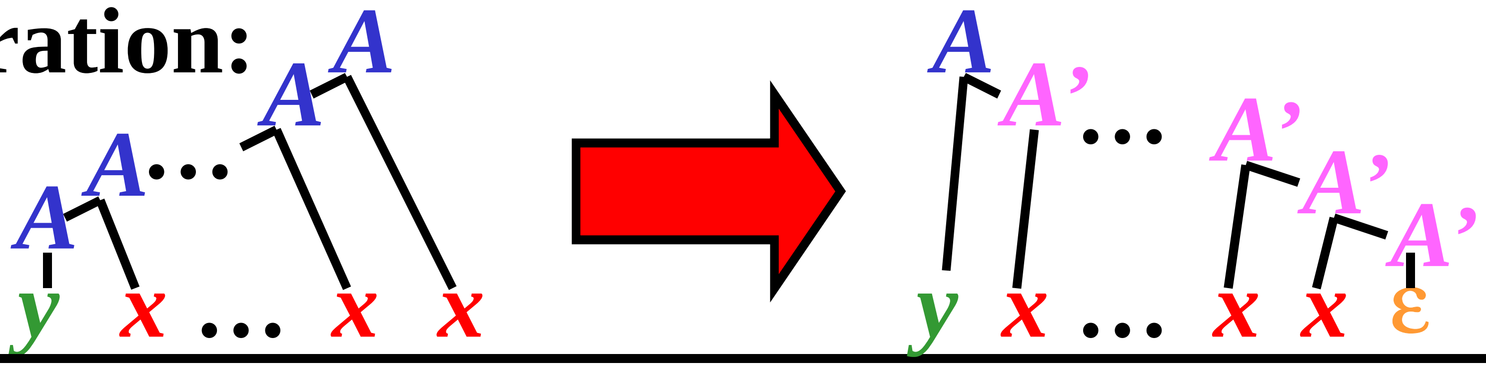
$F \rightarrow (E)$

$F \rightarrow i$

Left Recursion Replacement

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

Illustration:



Example:

$E \rightarrow E+T$
$E \rightarrow T$

$E \rightarrow TE'$, $E' \rightarrow +TE'$, $E' \rightarrow \varepsilon$

$T \rightarrow T*F$
$T \rightarrow F$

$T \rightarrow FT'$, $T' \rightarrow *FT'$, $T' \rightarrow \varepsilon$

$F \rightarrow (E)$ $\qquad$ $F \rightarrow (E)$
$F \rightarrow i$ $\qquad$ $F \rightarrow i$

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:

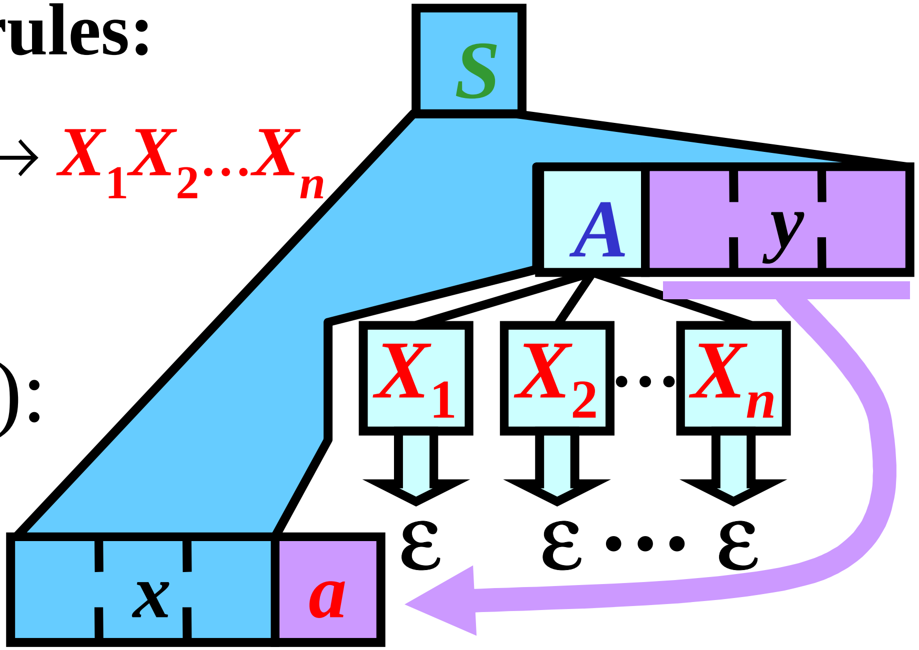
Assume that every input string of tokens ends with $.

Note: **$** acts as an *end marker*.

Main problem with ε-rules:

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

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



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

Grammar for Arithmetical Expressions

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

Example:

$$(\boldsymbol{i} + \boldsymbol{i})*(\boldsymbol{i} + \boldsymbol{i}) \in \boldsymbol{L}(\boldsymbol{G}_{expr3})$$

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.
$Empty(x) = \{\varepsilon\}$ if $x \Rightarrow^* \varepsilon$; otherwise,
$Empty(x) = \emptyset$, where $x \in (N \cup T)^*$.

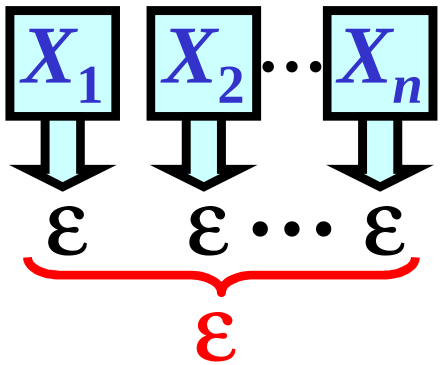
Illustration: $x = \boxed{X_1}\ \boxed{X_2} \cdots \boxed{X_n}$

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

$$\begin{array}{cccc} \Downarrow & \Downarrow & & \Downarrow \\ \end{array}$$

$$\underbrace{\varepsilon \quad \varepsilon \cdots \varepsilon}_{\varepsilon}$$

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

$$\Downarrow$$

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

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) := $\emptyset$
- **for each** $A \in N$:

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

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

Previous Algorithm: Illustration

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

Previous Algorithm: Illustration

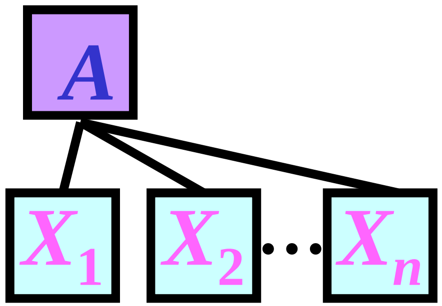
1) for each $a \in T$: *Empty*(a) := $\varnothing$ because $a \not\Rightarrow^* \varepsilon$

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



Previous Algorithm: Illustration

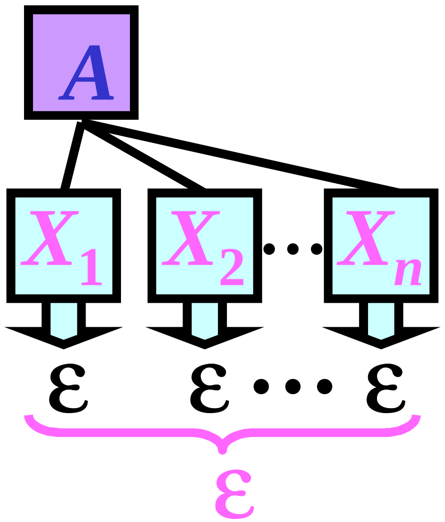
1) for each $\boldsymbol{a} \in T$: $Empty(\boldsymbol{a}) := \emptyset$ because $\boldsymbol{a} \not\Rightarrow^* \varepsilon$

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

---

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

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

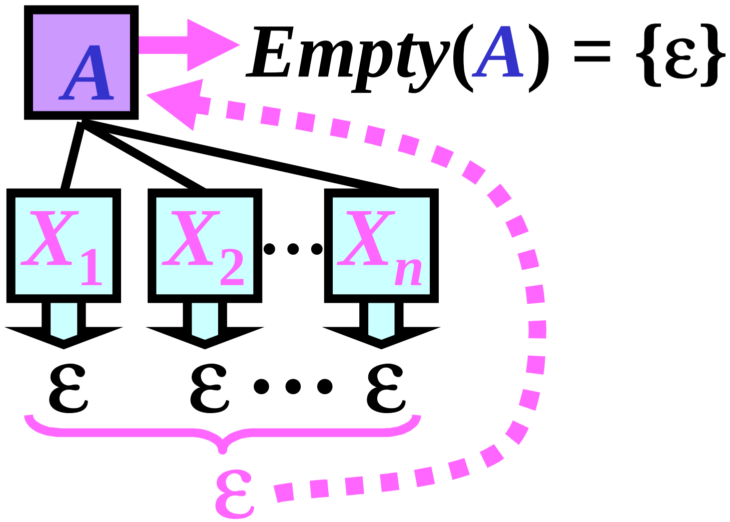


Previous Algorithm: Illustration

1) for each $\boldsymbol{a} \in T$: $Empty(\boldsymbol{a}) := \emptyset$ because $\boldsymbol{a} \not\Rightarrow^{*} \varepsilon$
2) for each $\boldsymbol{r}$: $\boldsymbol{A} \rightarrow \varepsilon \in \boldsymbol{P}$: $Empty(\boldsymbol{A}) := \{\varepsilon\}$ because $\boldsymbol{A} \Rightarrow^{1} \varepsilon$ **[*r*]**

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

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



$Empty(X)$ 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:

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

- **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) := $\emptyset$
- **Apply the following rule until no *First* set can be changed:**
- **if** $A \to X_1X_2\ldots X_{k-1}X_k\ldots X_n \in P$ **then**
  - add all symbols from *First*(X_1) to *First*(A)
  - **if** *Empty*(X_i) = $\{\varepsilon\}$ 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 $\boldsymbol{a} \in T$: $First(\boldsymbol{a}) := \{\boldsymbol{a}\}$ because $\boldsymbol{a} \Rightarrow^{\mathbf{0}} \boldsymbol{a}$

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

---

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

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

Previous Algorithm: Illustration

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

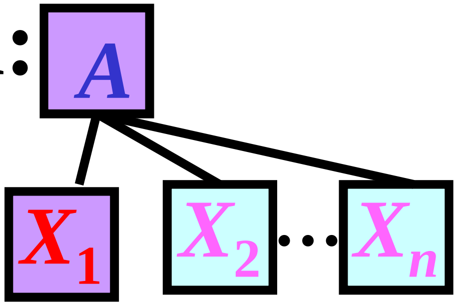
2) for each $\boldsymbol{A} \in N$: $First(\boldsymbol{A}) := \emptyset$ (inicialization)

---

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

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

  3a) add all symbols from $First(\boldsymbol{X_1})$ to $First(\boldsymbol{A})$

3a: 

Previous Algorithm: Illustration

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

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

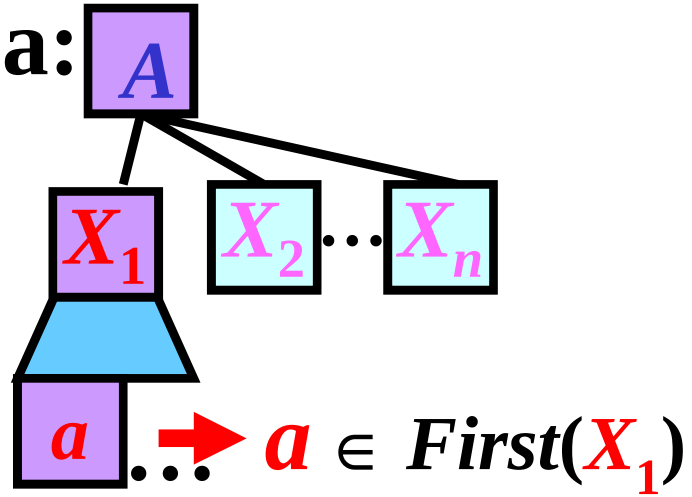
---

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

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

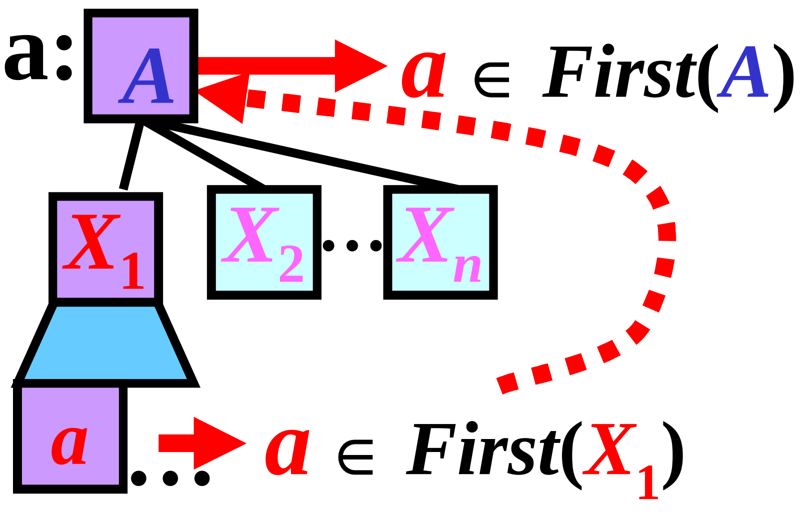
---

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

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

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

3a:



Previous Algorithm: Illustration

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

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

---

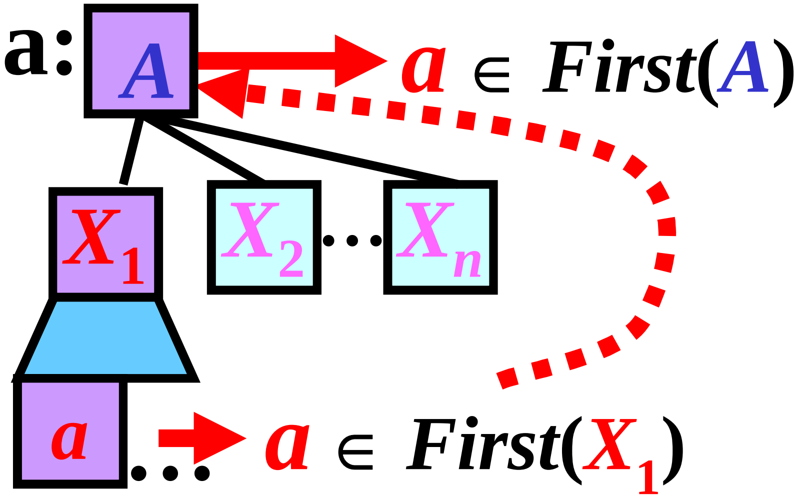
3) Apply the following rules until no *First* set or *Empty* set can be changed:

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

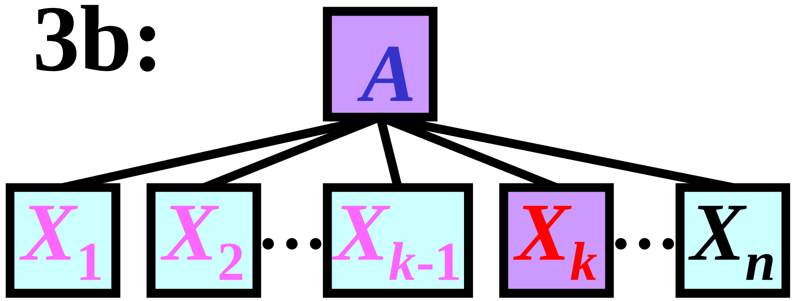
3a) add all symbols from $First(X_1)$ to $First(A)$

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 $First(A)$:

3a:



3b:



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)

---

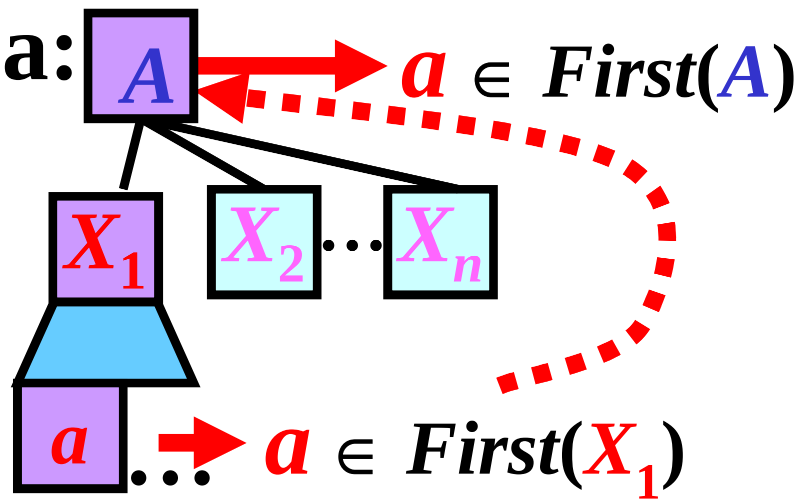
3) Apply the following rules until no *First* set or *Empty* set can be changed:

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

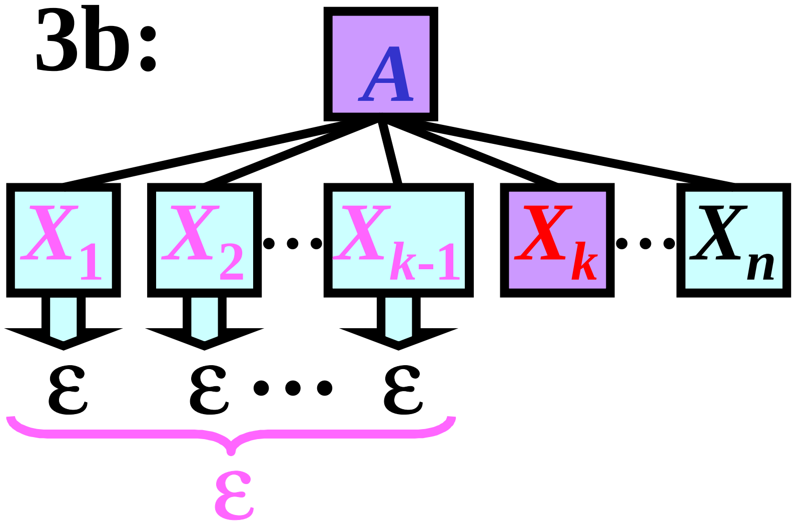
  3a) add all symbols from $First(X_1)$ to $First(A)$

  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 $First(A)$:

3a:



3b:



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)

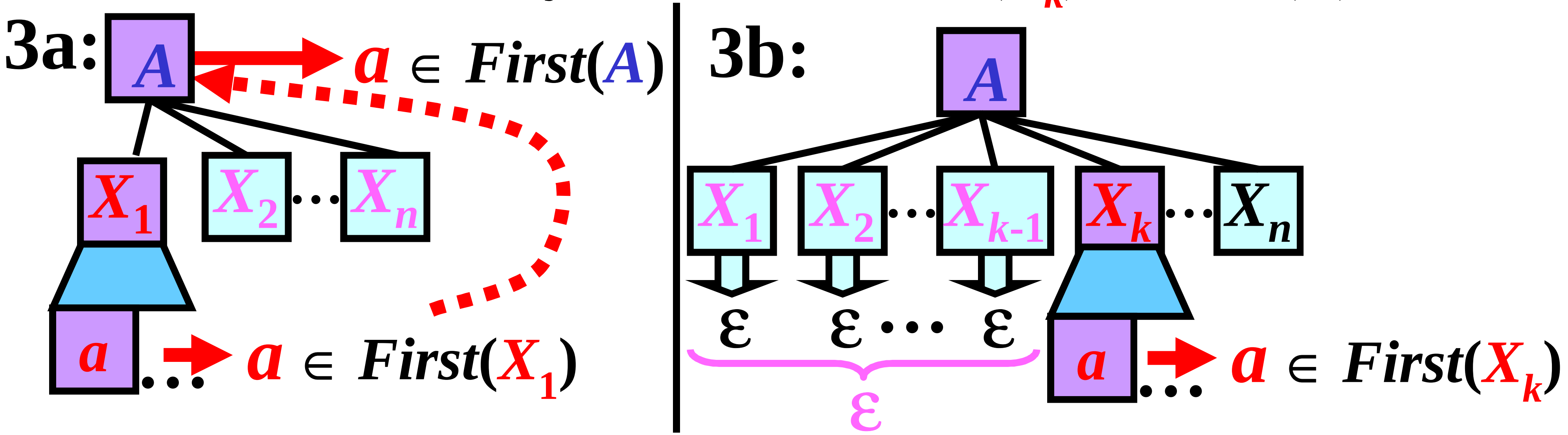
---

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

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

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

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



Previous Algorithm: Illustration

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

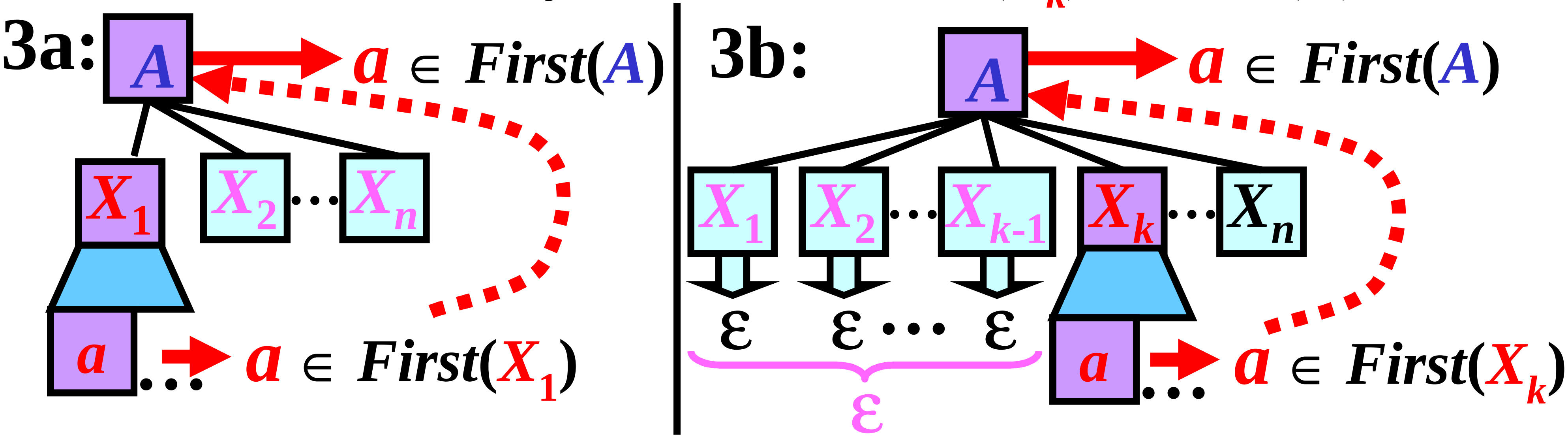
2) for each $\boldsymbol{A} \in N$: $First(\boldsymbol{A}) := \emptyset$ (inicialization)

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 $First(A)$

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



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

Initialization:

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

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

Initialization:

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

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

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

Summary: $First(F) = \{i, (\}$

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'*) = {*}

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

Initialization:

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

$F \rightarrow i \in P$: **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: $First(T) = \{i, (\}$

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

Initialization:

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

$F \rightarrow i \in P$: **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*(***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*, (}

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

Initialization:

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

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

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

Summary: $First(T) = \{i, (\}$

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

Summary: $First(E') = \{+\}$

$E \to TE' \in P$: **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: $E \to TE'$, 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 $X \in N \cup T$:

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

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

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

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

Algorithm: $First(X_1X_2\ldots X_n)$

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

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

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

Illustration:

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

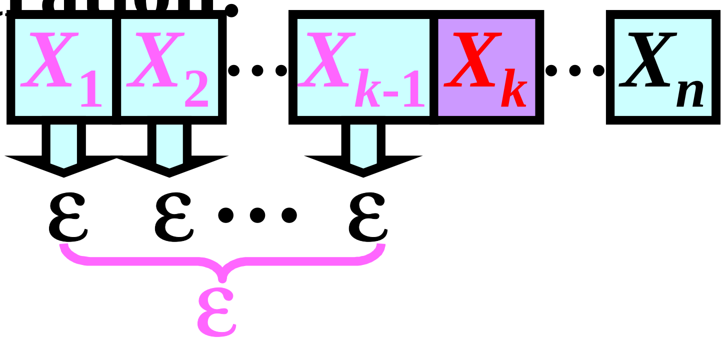
Algorithm: $First(X_1X_2\ldots X_n)$

- **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 $i = 1,\ldots,k$-1, where $k \leq n$ **then** add all symbols from $First(\boldsymbol{X_k})$ to $First(X_1X_2\ldots X_n)$

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

Illustration:



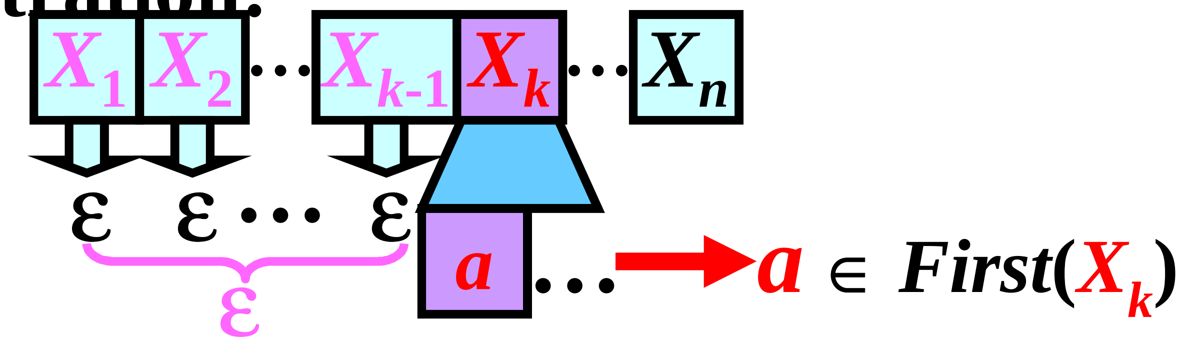
Algorithm: *First*$(X_1X_2\ldots X_n)$

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

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

Illustration:



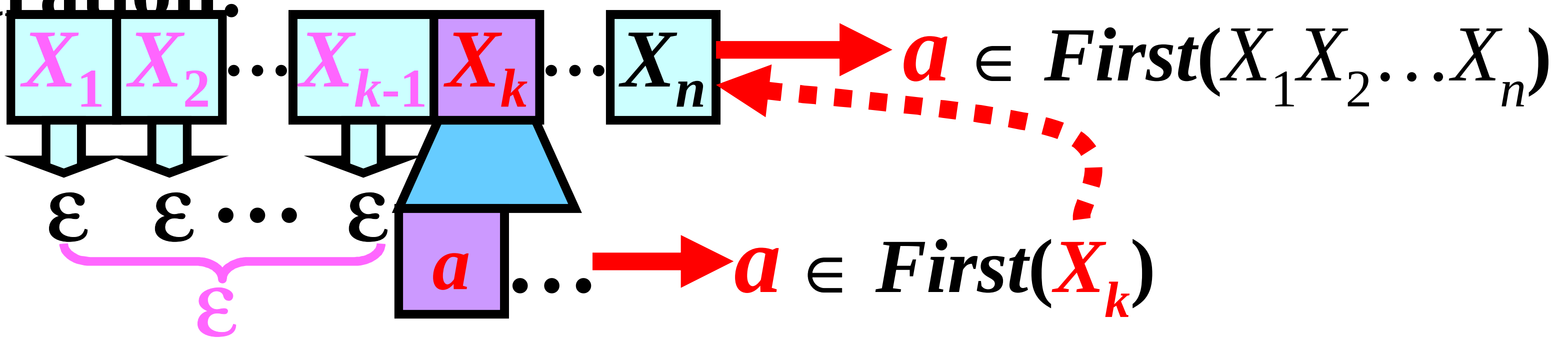
Algorithm: $First(X_1X_2\ldots X_n)$

- **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** $\boldsymbol{First(X_1X_2 \ldots X_{k-1} X_k \ldots X_n)}$**:**
  - **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:



$First(X_1X_2\ldots X_n)$: 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: $F \rightarrow i$ $\}$

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) $First(E'T'\underline{F}ET)$: 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: $\boxed{X_1}\boxed{X_2}\ldots\boxed{X_n}$

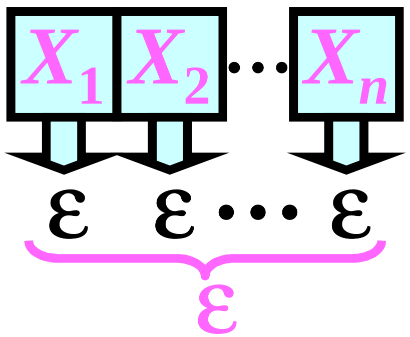
Algorithm: $Empty(X_1X_2\ldots X_n)$

- **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: $Empty(X_1X_2\dots X_n)$

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

  $x = X_1X_2\dots X_n$, where $x \in (N \cup T)^+$
- **Output:** $Empty(X_1X_2\dots X_n)$

- **Method:**
- **if** $Empty(X_i) = \{\varepsilon\}$ for all $i = 1,\dots,n$ **then**

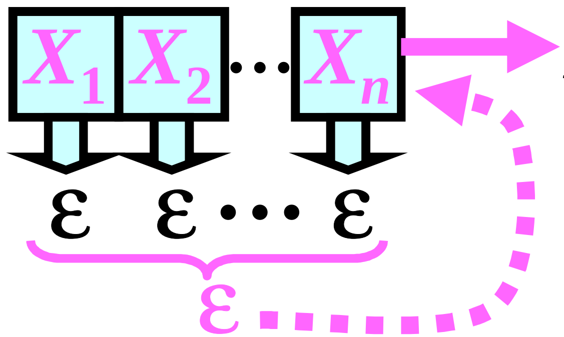
  $Empty(X_1X_2\dots X_n) := \{\varepsilon\}$

  else

  $Empty(X_1X_2\dots X_n) := \varnothing$

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

Illustration:



$Empty(X_1X_2\dots X_n) = \{\varepsilon\}$

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

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

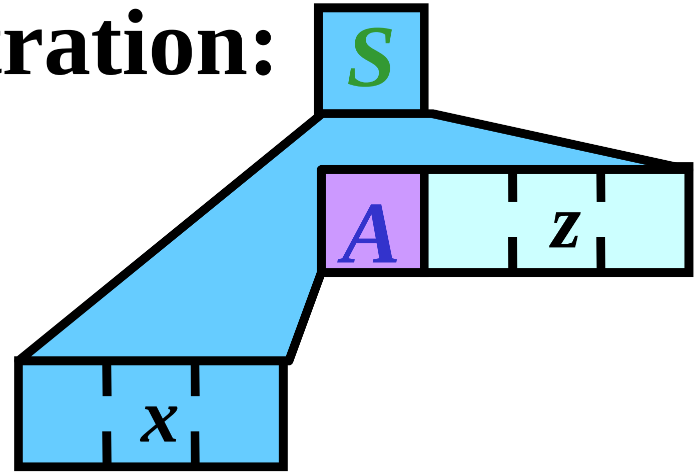
Illustration:

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:



$S \Rightarrow^* xAz$

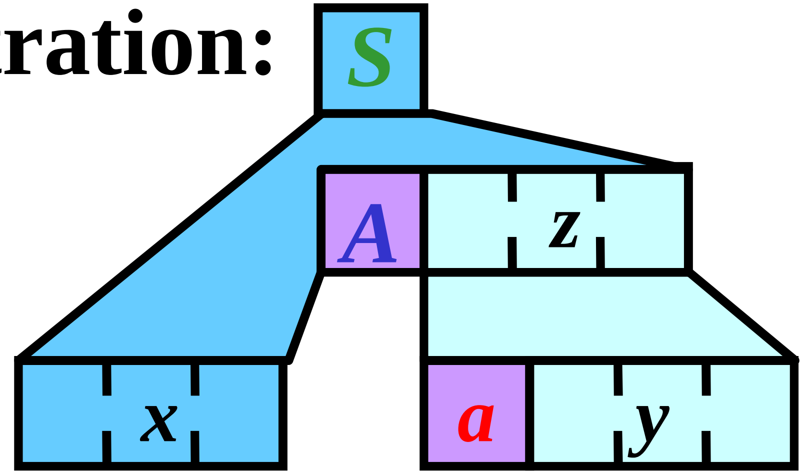
Set *Follow*

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:



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

$$a \in Follow(A)$$

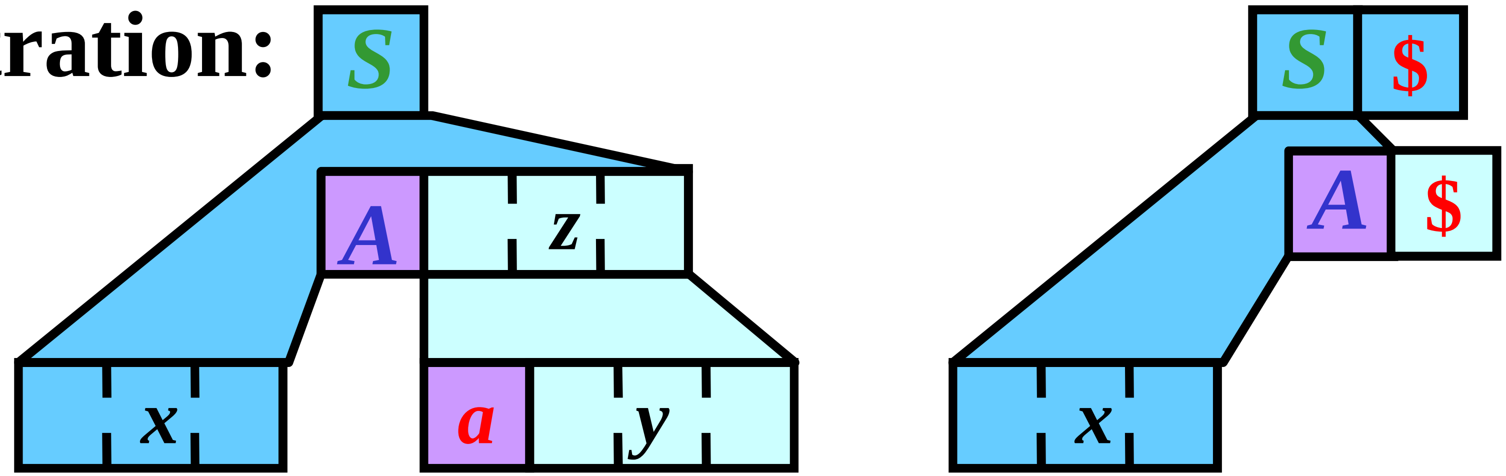
Set *Follow*

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:



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

$a \in Follow(A)$

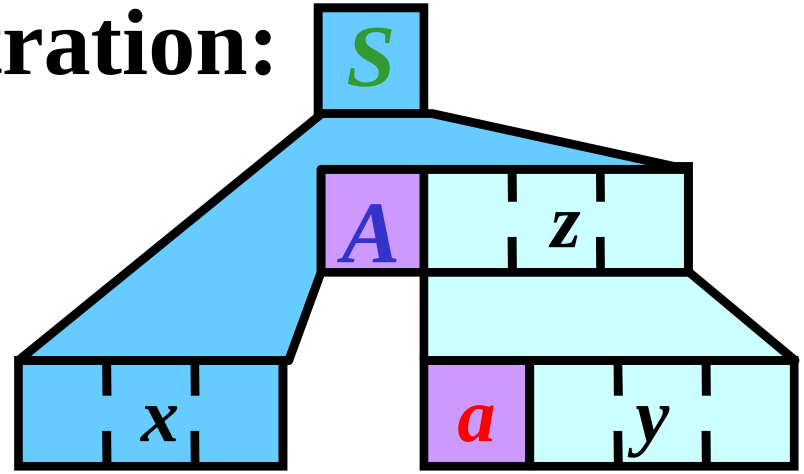
$S \Rightarrow^* xA$

Set *Follow*

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

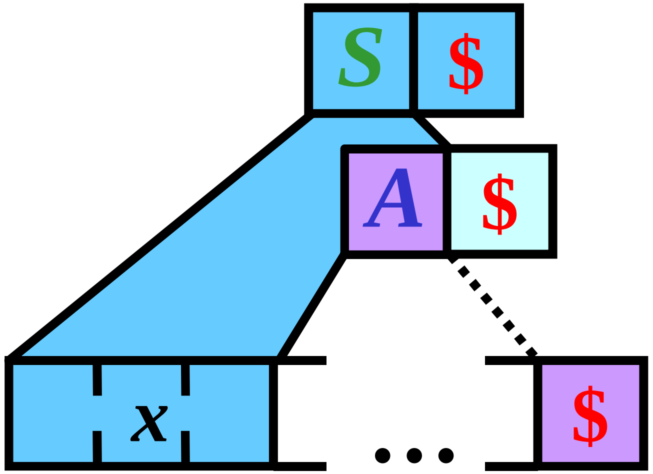
Definition: Let $G = (N, T, P, S)$ be a CFG. For every $A \in N$, we define the set $Follow(A)$ as
$$Follow(A) = \{a : a \in T, S \Rightarrow^* xAay, x, y \in (N \cup T)^*\}$$
$$\cup \{\$: S \Rightarrow^* xA, x \in (N \cup T)^*\}$$

Illustration:



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

$a \in Follow(A)$



$S \Rightarrow^* xA$

$\$ \in Follow(A)$

Algorithm: *Follow*(*A*)

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

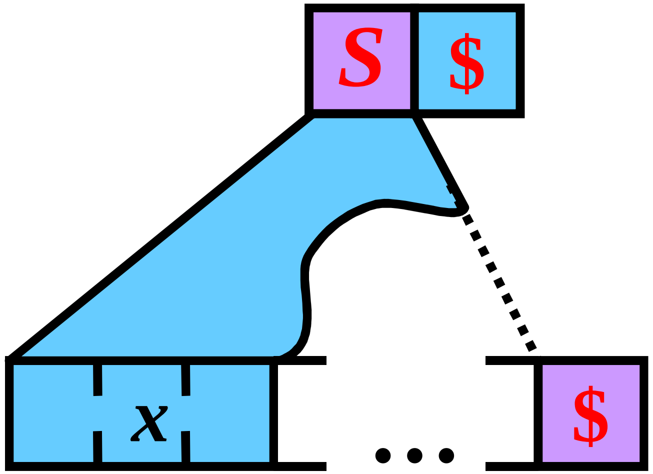
- **Method:**
- *Follow*(***S***) := {**$**};
- **Apply the following rules until no *Follow* set can be changed:**
- **if** $A \rightarrow xBy \in P$ **then**
  - **if** $y \neq \varepsilon$ **then**

    add all symbols from *First*(*y*) to *Follow*(***B***);
  - **if** *Empty*(*y*) = {ε} **then**

    add all symbols from *Follow*(***A***) to *Follow*(***B***);

Previous Algorithm: Illustration

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

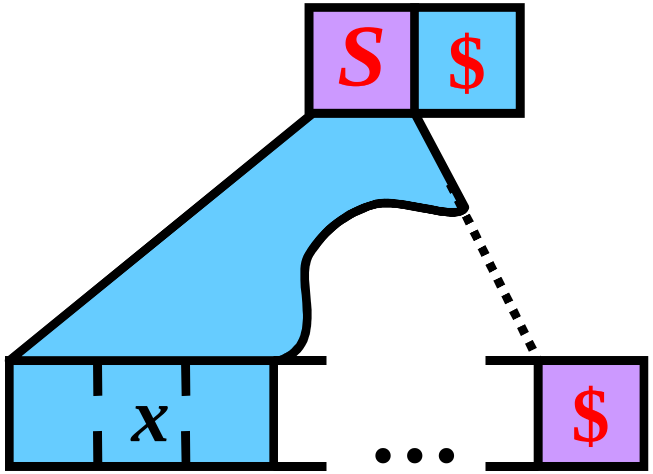


2) Apply the following rules until no *Follow* set can be changed:

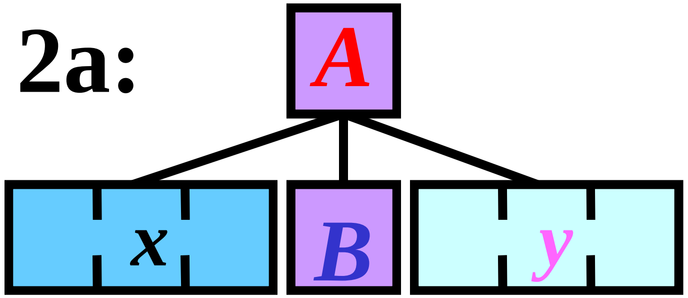
- **if** $A \rightarrow xBy \in P$ **then**

Previous Algorithm: Illustration

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



2a:



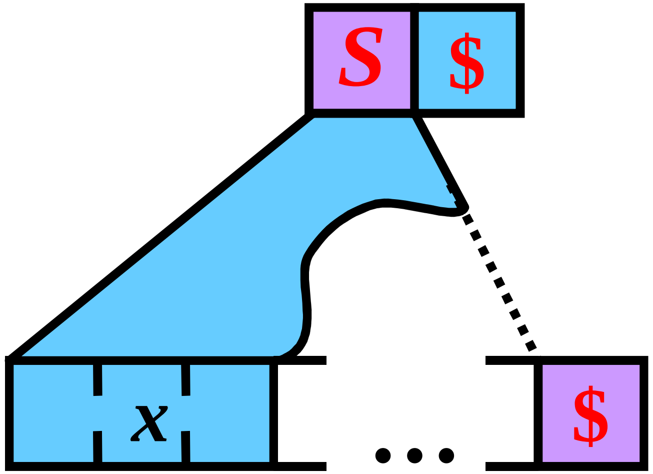
2) Apply the following rules until no *Follow* set can be changed:

- **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

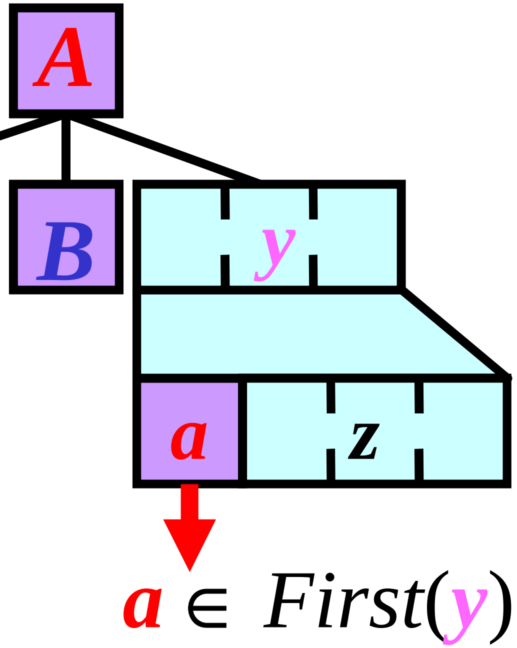
1) *Follow*(***S***) := {**$**}



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

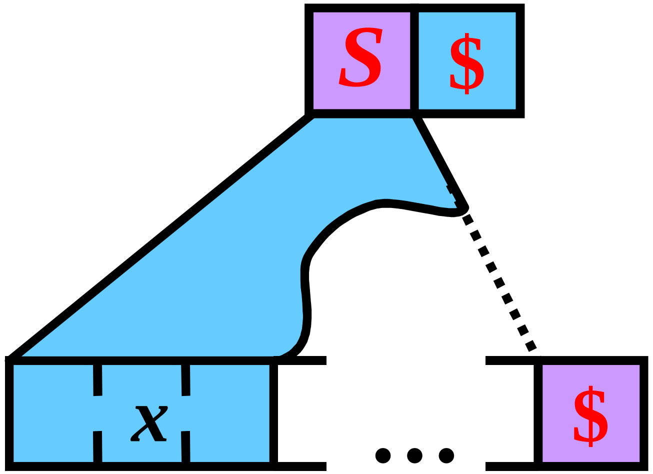
- **if** ***A*** $\rightarrow$ ***xBy*** $\in$ *P* **then**
  2a) if ***y*** $\neq \varepsilon$ **then** add all symbols from *First*(***y***) to *Follow*(***B***)

2a:

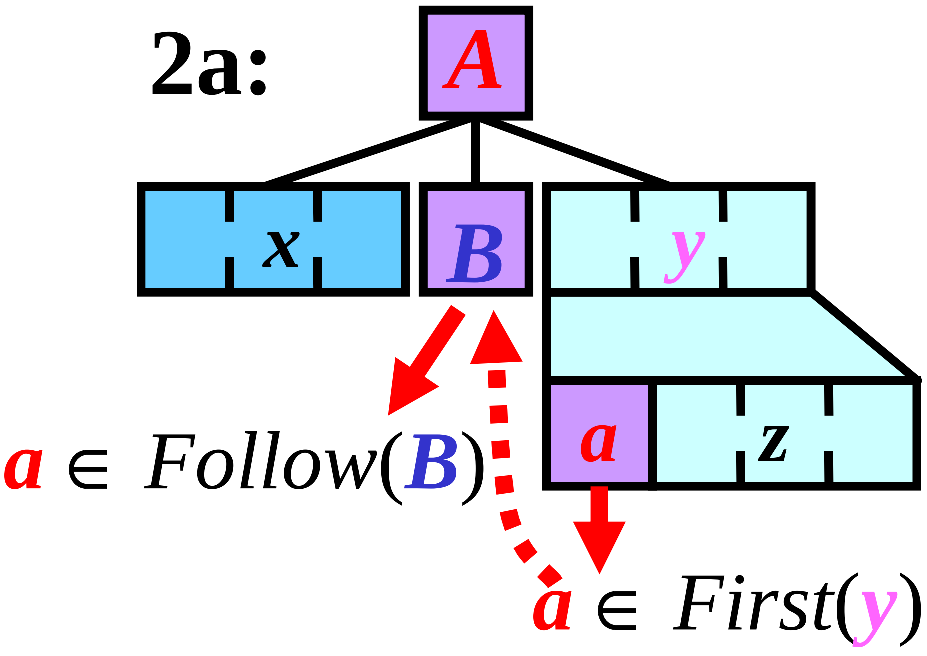


Previous Algorithm: Illustration

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



2a:

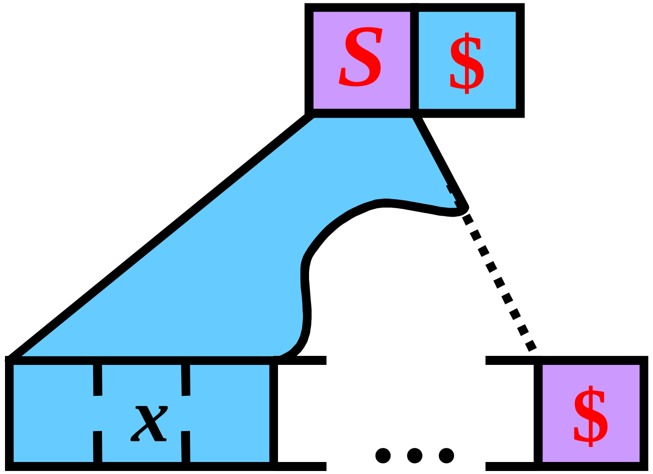


2) Apply the following rules until no *Follow* set can be changed:

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

Previous Algorithm: Illustration

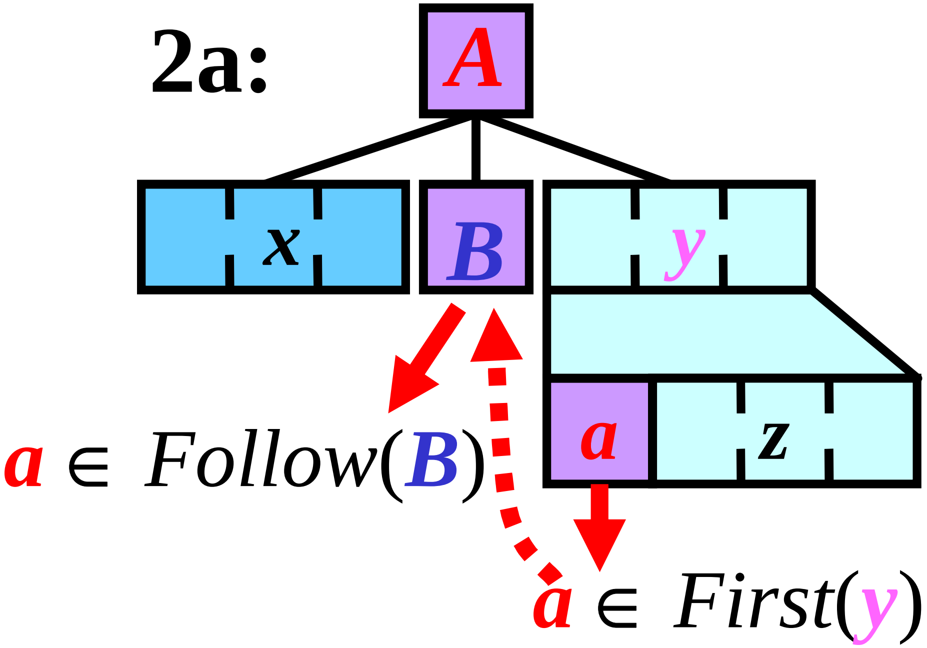
1) *Follow*(***S***) := {**$**}



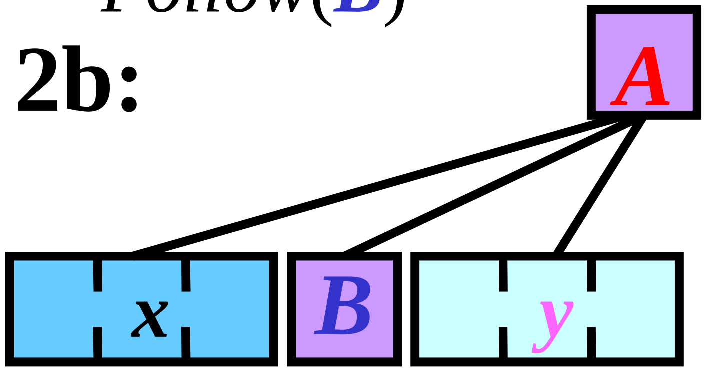
2) Apply the following rules until no *Follow* set can be changed:

- **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:

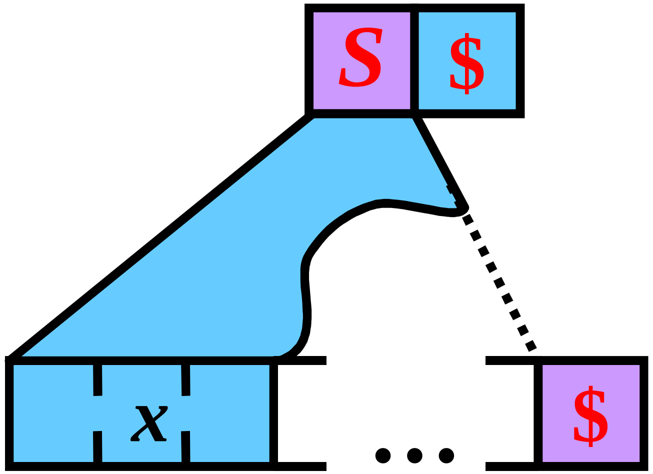


2b:



Previous Algorithm: Illustration

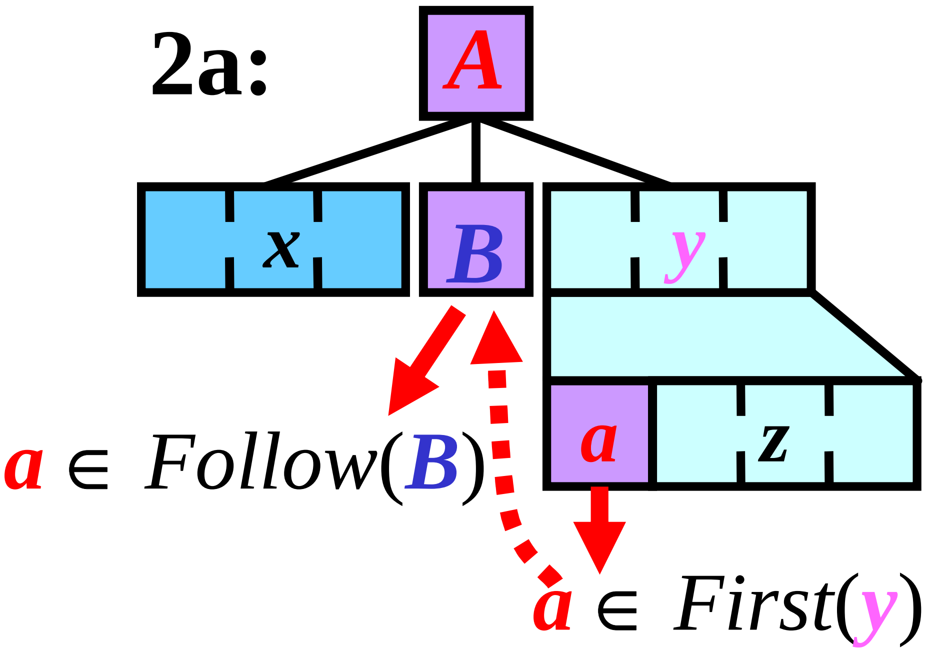
1) *Follow*(***S***) := {**$**}



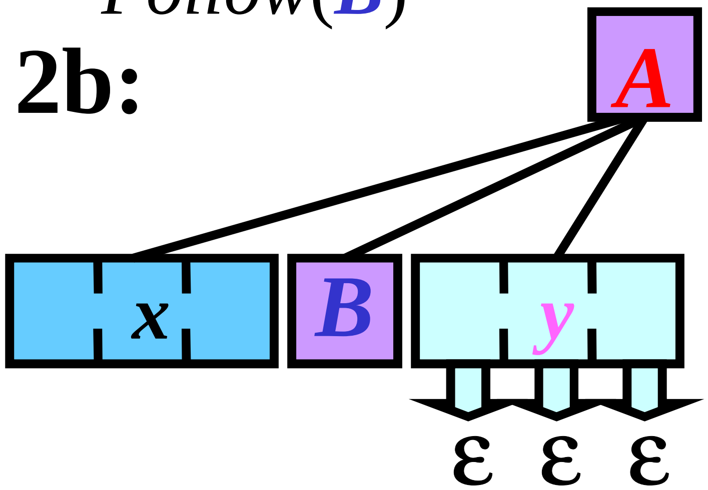
2) Apply the following rules until no *Follow* set can be changed:

- **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:

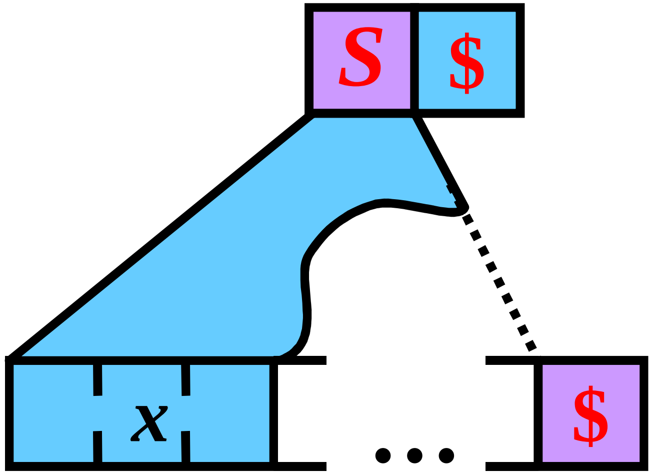


2b:



Previous Algorithm: Illustration

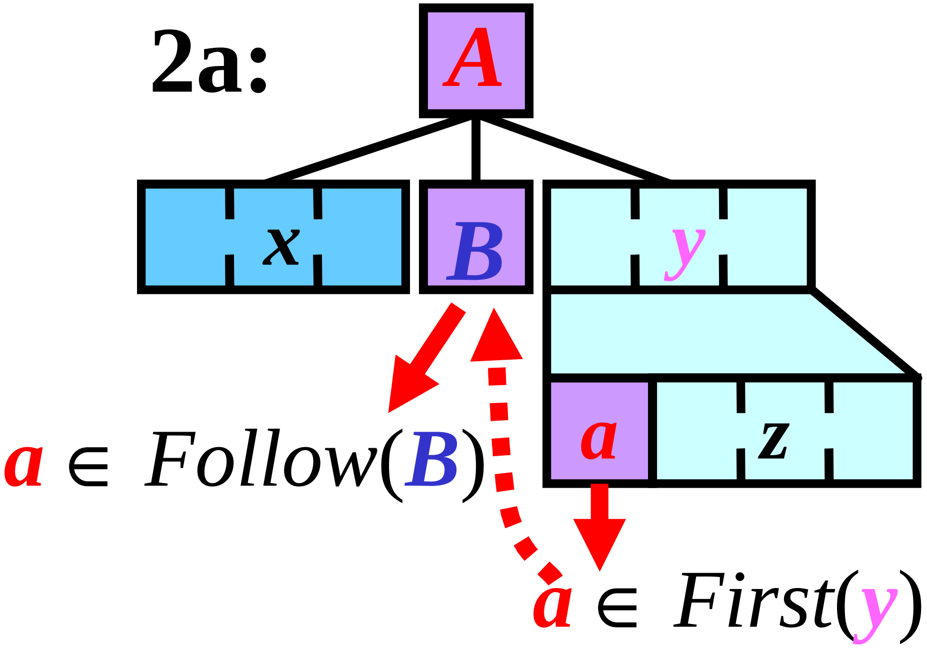
1) *Follow*(***S***) := {**$**}



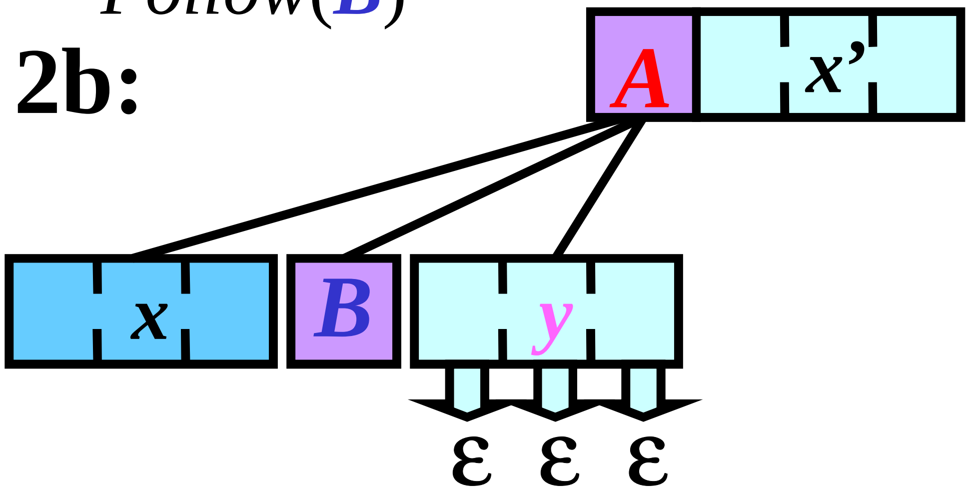
2) Apply the following rules until no *Follow* set can be changed:

- **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:

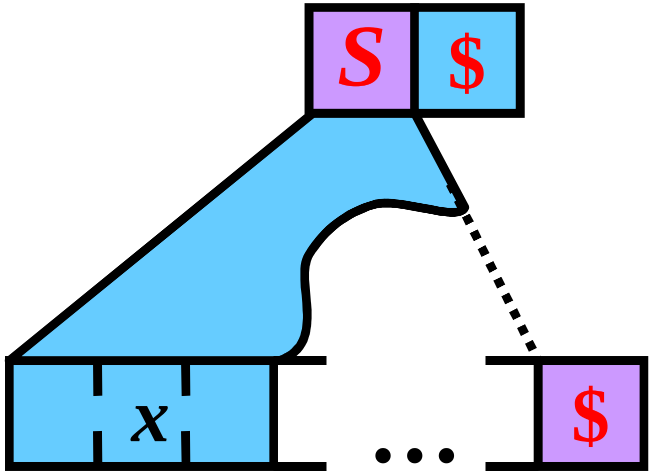


2b:



Previous Algorithm: Illustration

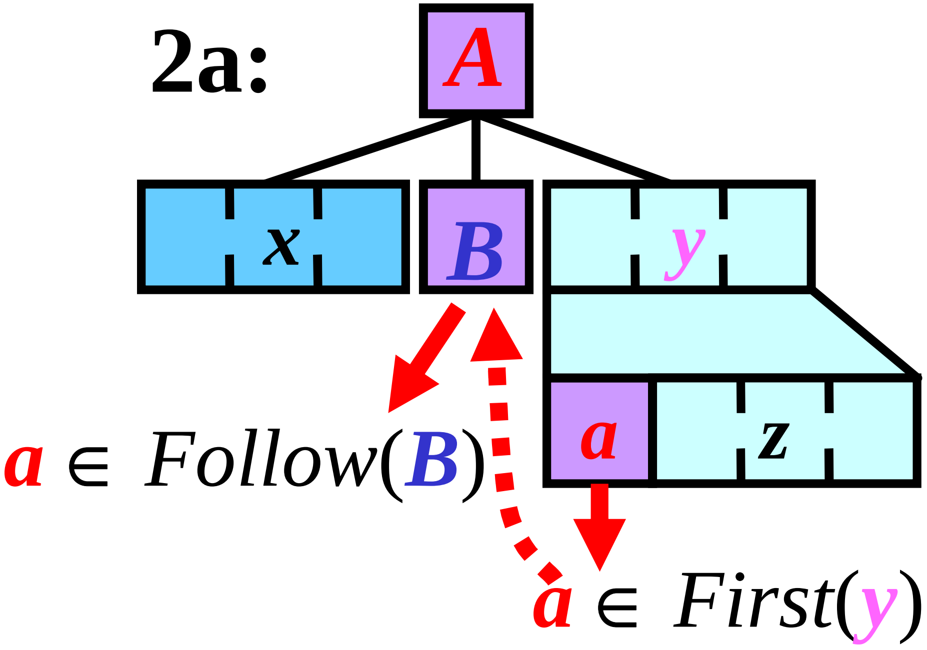
1) *Follow*(***S***) := {**$**}



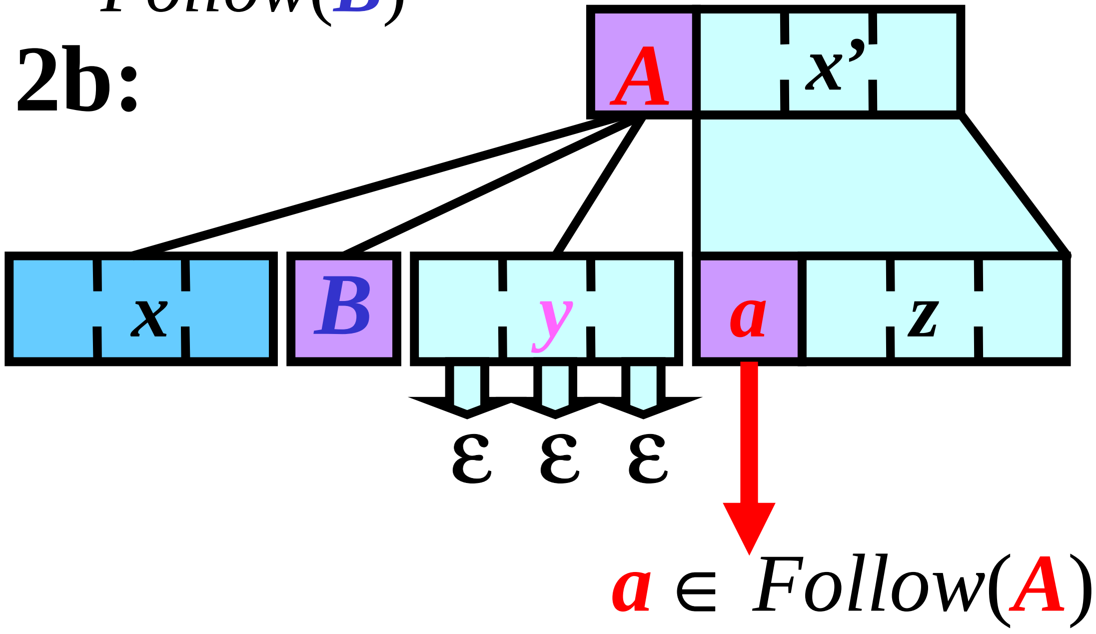
2) Apply the following rules until no *Follow* set can be changed:

- **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:

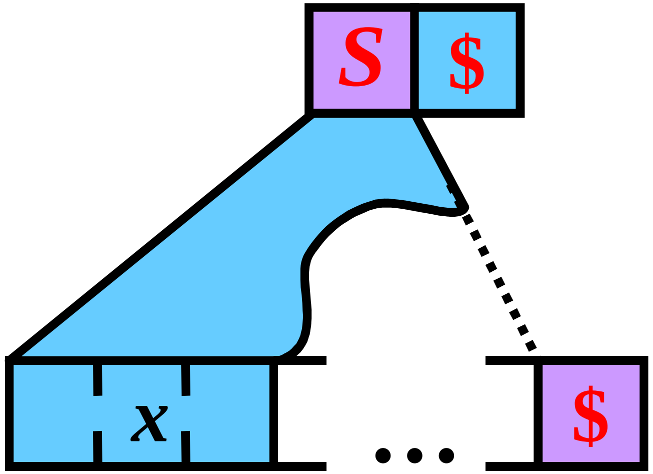


2b:



Previous Algorithm: Illustration

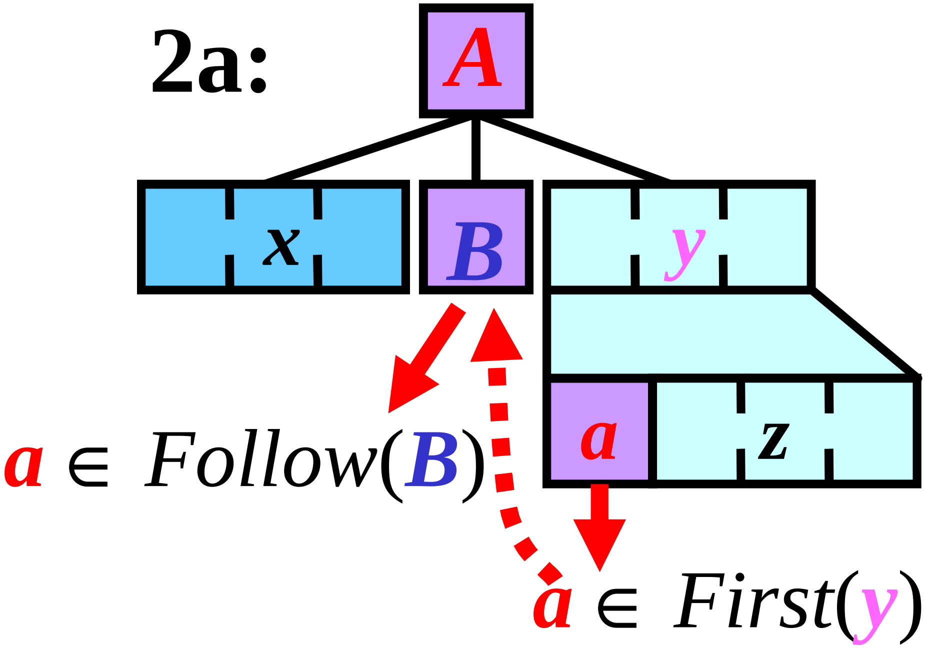
1) *Follow*(***S***) := {**$**}



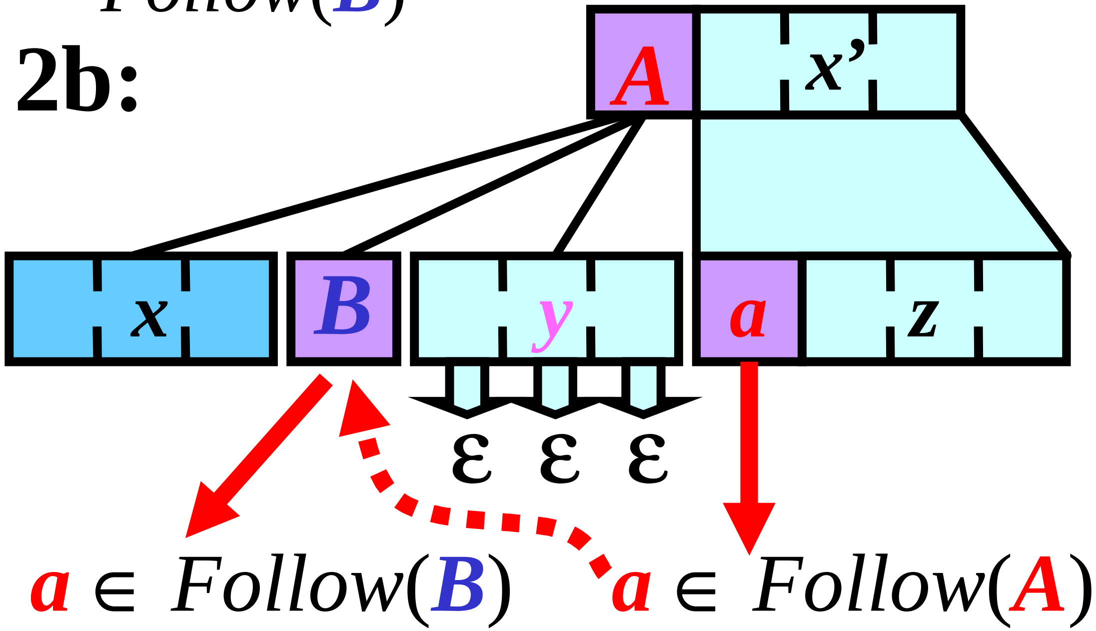
2) Apply the following rules until no *Follow* set can be changed:

- **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:



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

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

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

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

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

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

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

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

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

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

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

---

2) $E \rightarrow T \underbrace{E'}_{\varepsilon:\ Empty(\varepsilon) = \{\varepsilon\}} \in P$:

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'*)

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

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

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

ε: *Empty*(ε) = {ε}

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

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

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

ε: *Empty*(ε) = {ε}

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

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

Empty(***E'***) = {ε}

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** $Follow(T)$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

End: Nothing is changed.

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

ε: *Empty*(ε) = {ε}

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

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

Empty(***T'***) = {ε}

End: Nothing is changed.

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

Set *Predict*

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

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

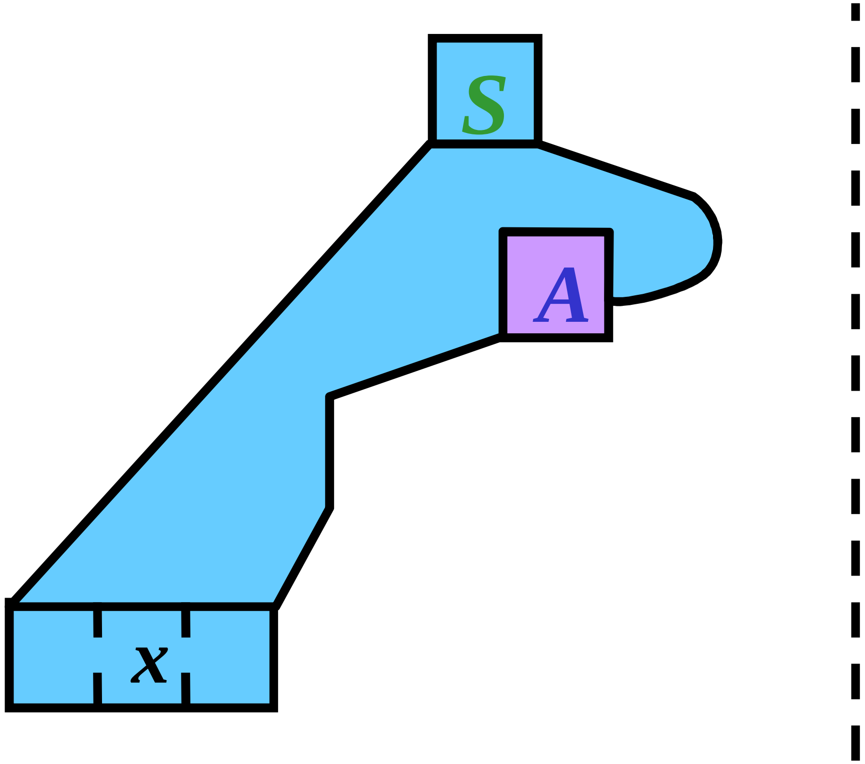
- if $Empty(x) = \{\varepsilon\}$ then
  $$Predict(A \rightarrow x) = First(x) \cup Follow(A)$$
- if $Empty(x) = \emptyset$ then
  $$Predict(A \rightarrow x) = First(x)$$

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

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

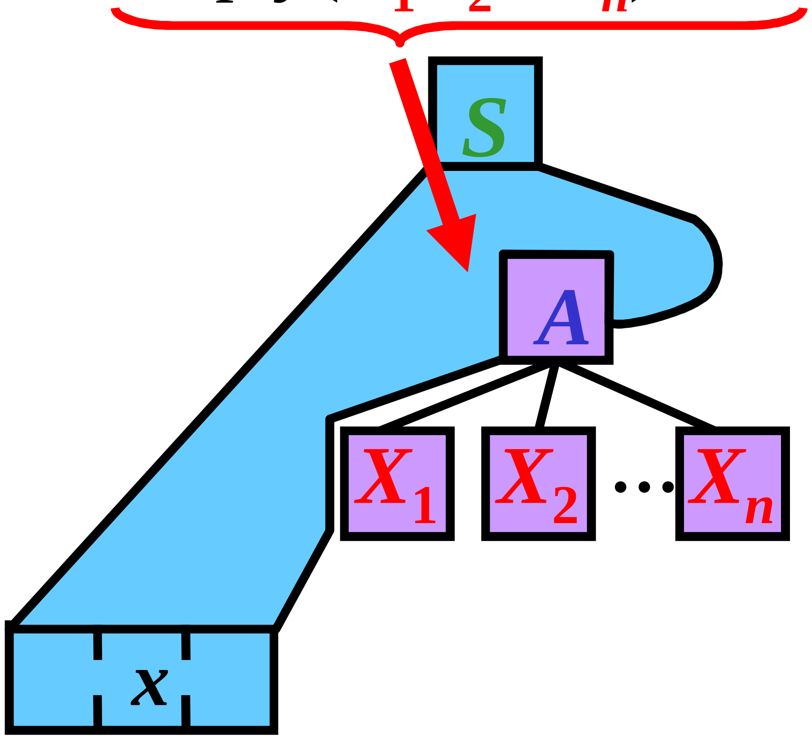
Set $Predict(A \to X_1X_2...X_n)$: Illustration

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



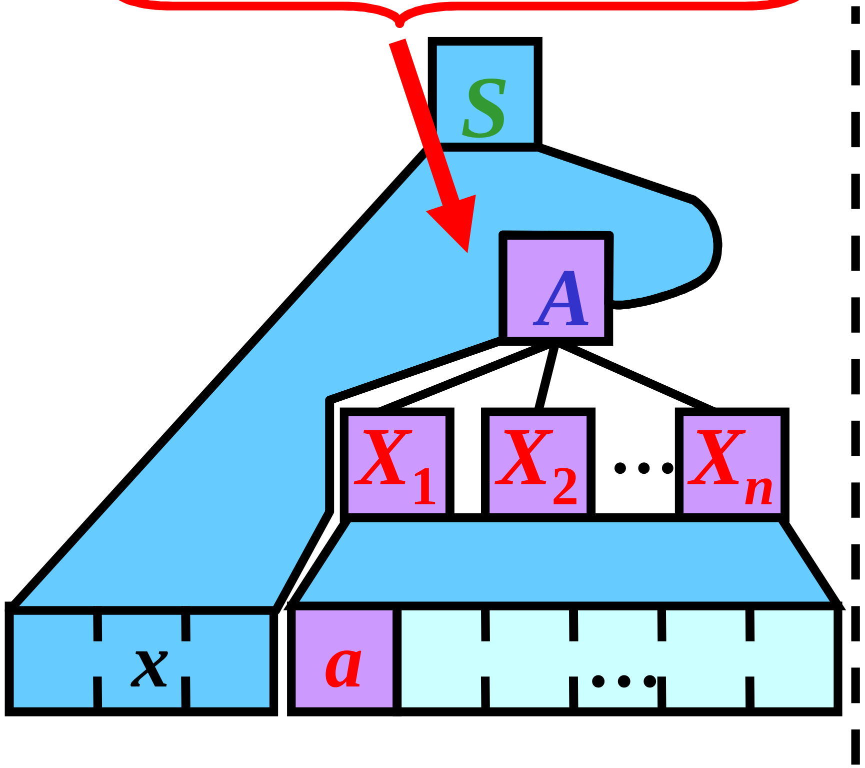
Set *Predict*($A \rightarrow X_1X_2...X_n$): Illustration

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



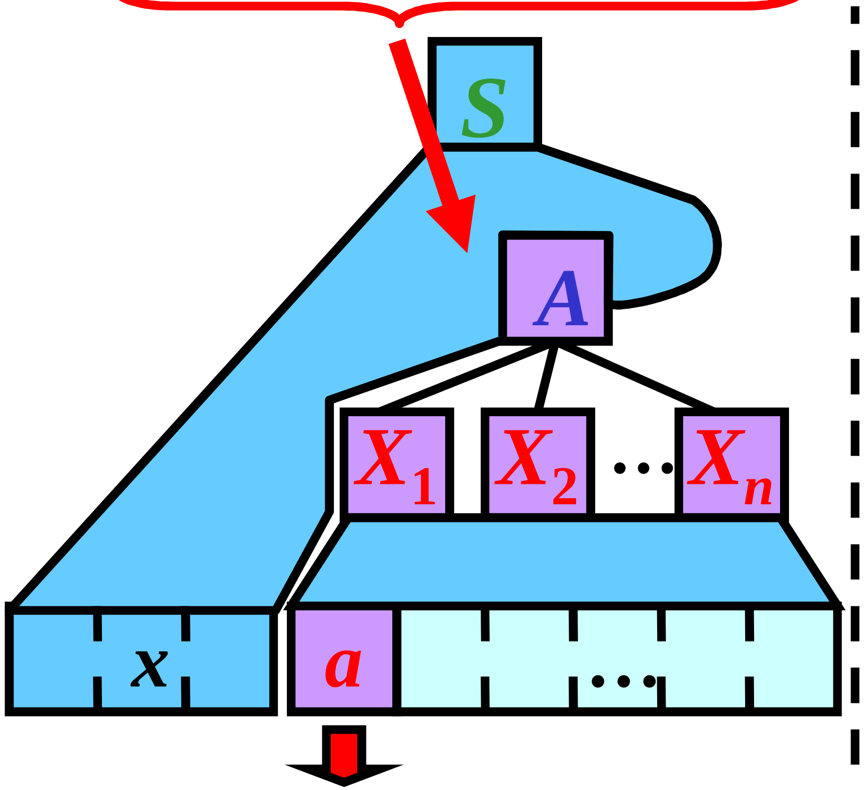
Set *Predict*($A \to X_1X_2\ldots X_n$): Illustration

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



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

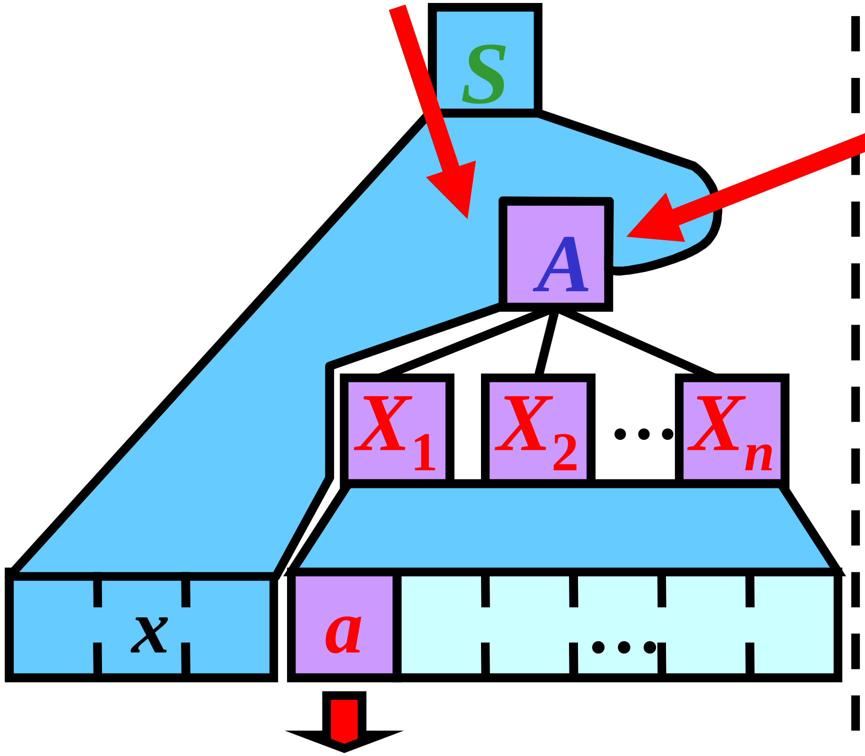
$Empty(X_1X_2\ldots X_n) = \emptyset$ vs. $Empty(X_1X_2\ldots X_n) = \{\varepsilon\}$



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

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

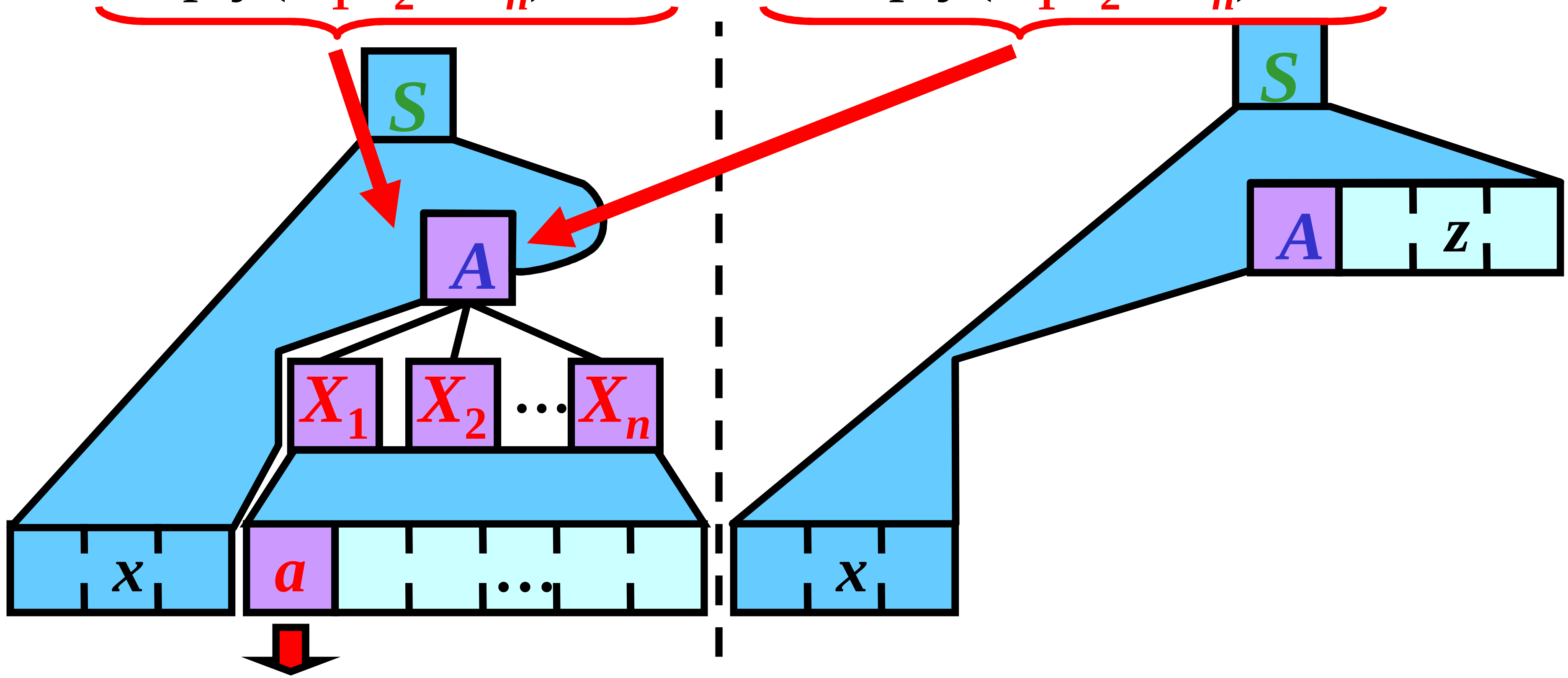
$\underbrace{Empty(X_1X_2...X_n) = \emptyset}$ vs. $\underbrace{Empty(X_1X_2...X_n) = \{\varepsilon\}}$



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

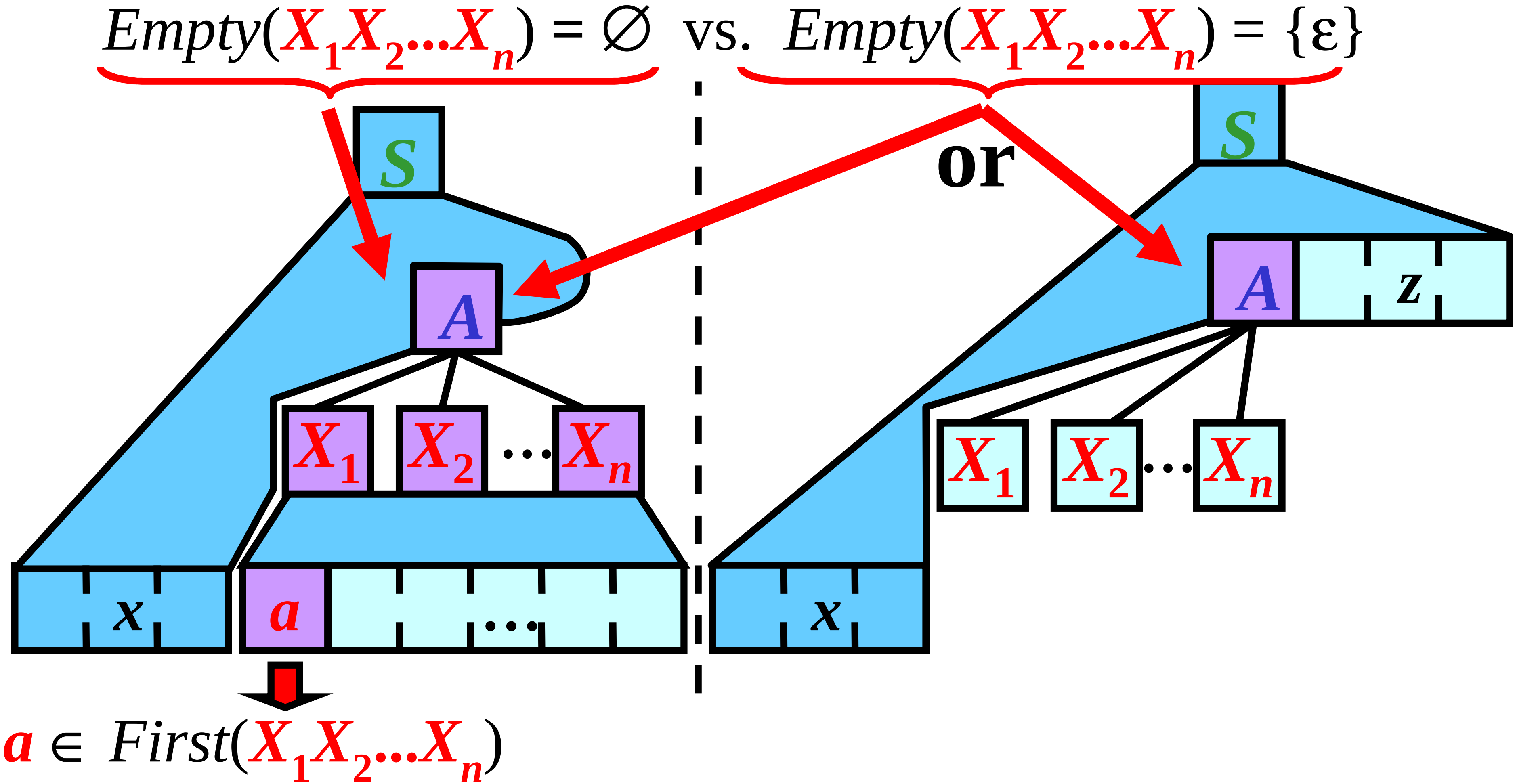
Set $Predict(A \rightarrow X_1X_2...X_n)$: Illustration

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

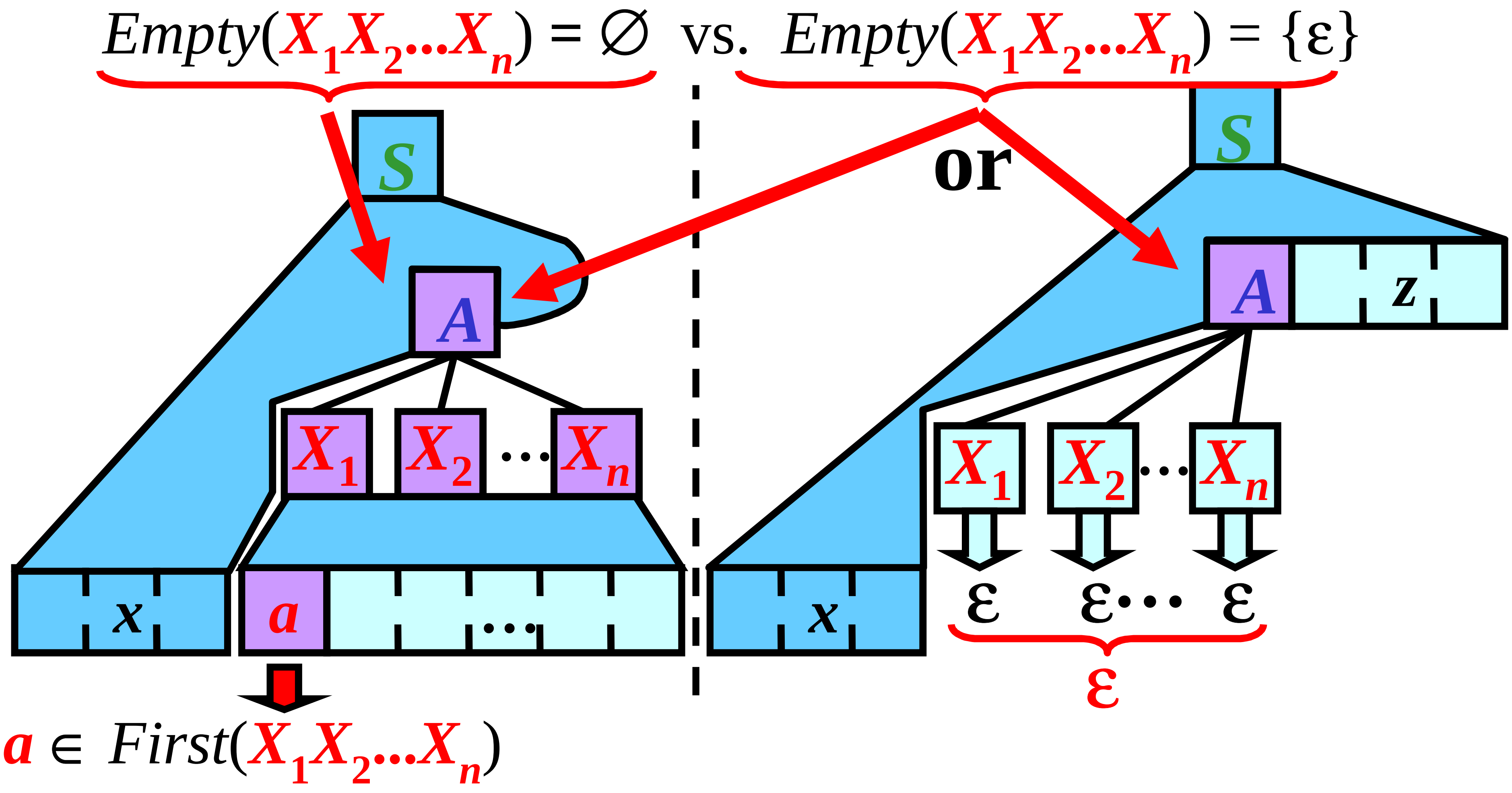


$a \in First(X_1X_2...X_n)$

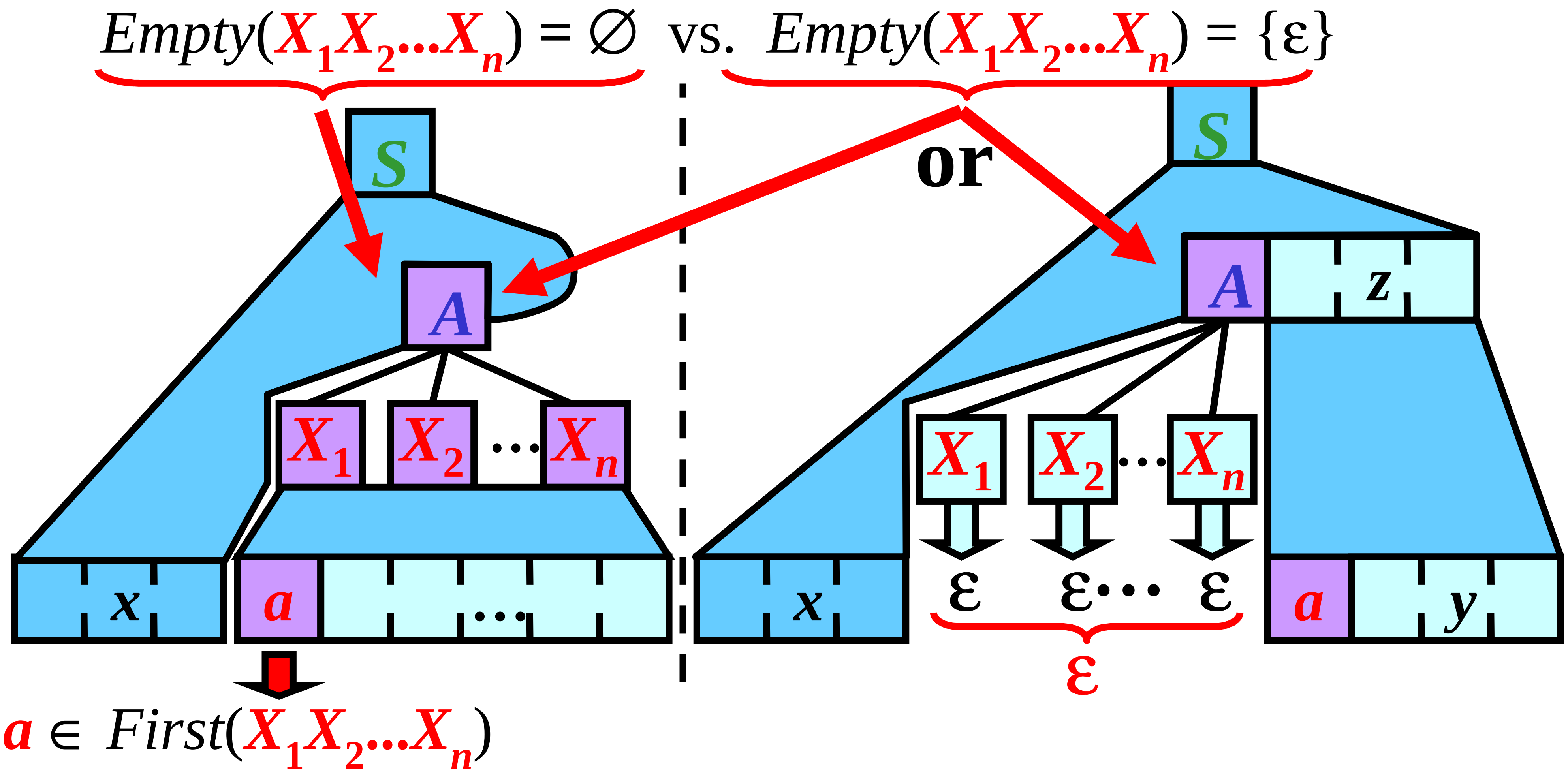
Set *Predict*($A \rightarrow X_1X_2...X_n$): Illustration



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

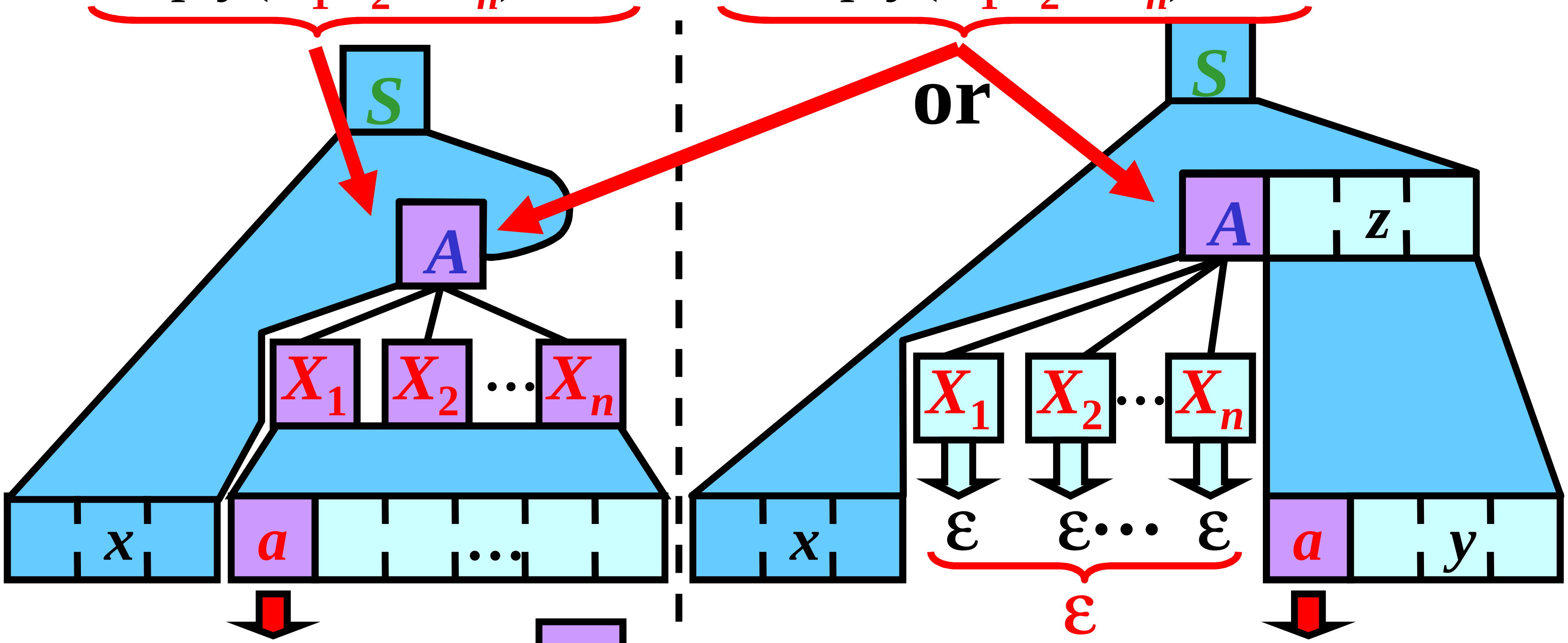


Set *Predict*($A \to X_1X_2...X_n$): Illustration



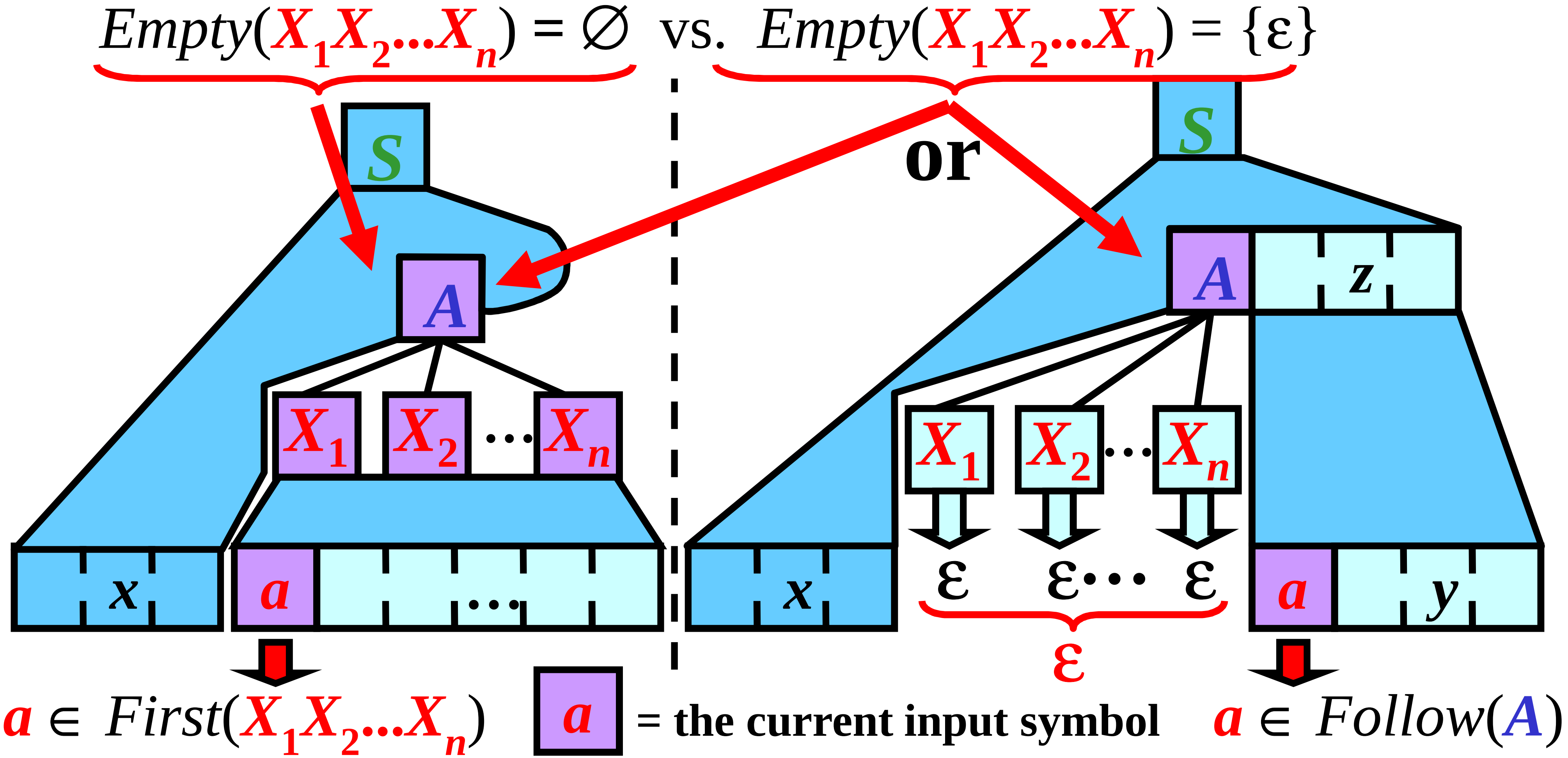
Set $Predict(A \rightarrow X_1X_2...X_n)$: Illustration

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



$a \in First(X_1X_2...X_n)$ **a** = the current input symbol $a \in Follow(A)$

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



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

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

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

Predict($A \to x$) 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***, **(**}

$Predict(A \rightarrow x)$ 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 $Empty(T) = \emptyset$

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

2: $E' \rightarrow +TE'$

$Empty(+TE') = \emptyset$ because $Empty(+) = \emptyset$

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

$Predict(A \rightarrow x)$ 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 $Empty(T) = \emptyset$

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

2: $E' \rightarrow +TE'$

$Empty(+TE') = \emptyset$ because $Empty(+) = \emptyset$

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

3: $E' \rightarrow \varepsilon$

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

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

$Predict(A \rightarrow x)$ 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 $Empty(T) = \emptyset$

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

2: $E' \rightarrow +TE'$

$Empty(+TE') = \emptyset$ because $Empty(+) = \emptyset$

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

3: $E' \rightarrow \varepsilon$

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

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

4: $T \rightarrow FT'$

$Empty(FT') = \emptyset$ because $Empty(F) = \emptyset$

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

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

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

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

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

5: $T' \to *FT'$

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

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

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

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

5: $T' \to {*}FT'$

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

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

6: $T' \to \varepsilon$

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

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

$Predict(A \to x)$ 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: $T' \to *FT'$

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

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

6: $T' \to \varepsilon$

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

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

7: $F \to (E)$

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

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

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

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

5: ***T'*** → ***FT'**

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

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

6: ***T'*** → ε

Empty(ε) = {ε}

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

7: ***F*** → **(*E*)**

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

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

8: ***F*** → ***i***

Empty(***i***) = ∅

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

Construction of LL Table

α	…	*a*	…
…			
A		α(*A*, *a*)	
…			

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		$\alpha(A, a)$	
…			

$\alpha(A, a) = A \rightarrow X_1X_2\ldots X_n \in P$ if $a \in Predict(A \rightarrow X_1X_2\ldots X_n)$; otherwise, $\alpha(A, a)$ 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: $E \rightarrow TE'$	{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*)	
...			

α(*A*, *a*) = $A \rightarrow X_1X_2\ldots X_n \in P$ if *a* ∈ *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 ∈ ***Predict***(1)

i ∈ ***Predict***(4)

i ∈ ***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*)	
...			

α(*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, α(*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)

Construct the rest analogically.

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

Parsing Based on LL Table: Example

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

1: $E \to TE'$
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: $i * i \in L(G_{expr3})$?

E

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

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: $E \to TE'$
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: $i * i \in L(G_{expr3})$?

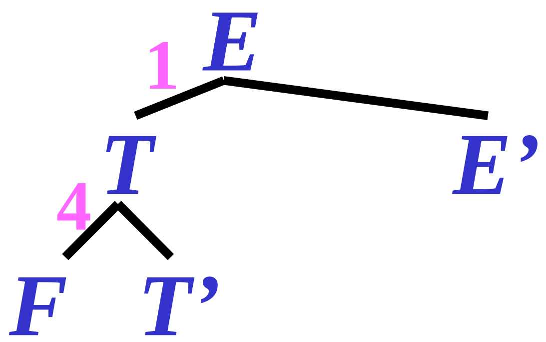
$i \quad * \quad i \quad \boxed{\$}$

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: $E \to TE'$
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: $i * i \in L(G_{expr3})$**?**



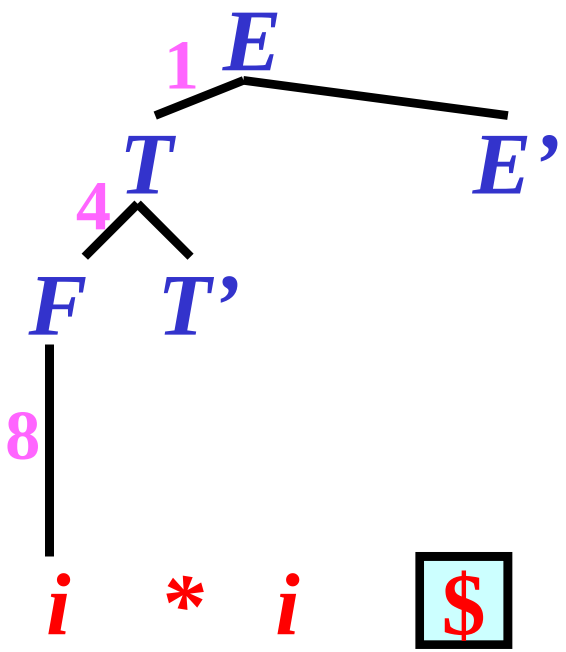
$i \quad * \quad i \quad \boxed{\$}$

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: $E \to TE'$
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: $i * i \in \mathrm{L}(G_{expr3})$?

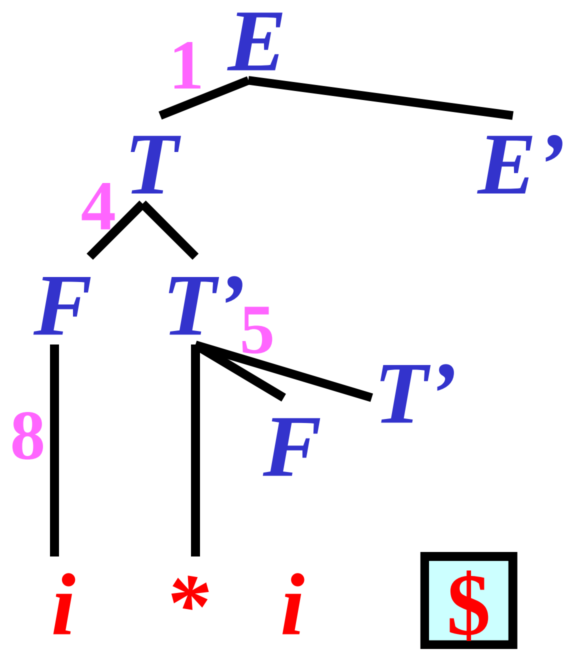


Parsing Based on LL Table: Example

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

1: $E \to TE'$
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: $i * i \in L(G_{expr3})$?

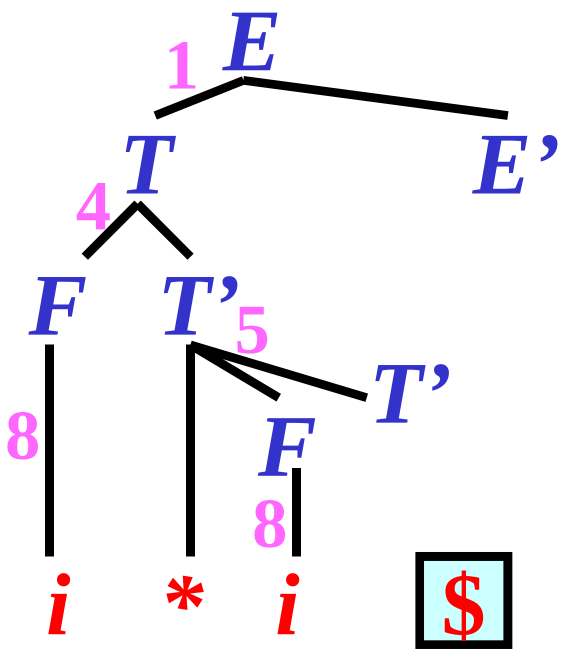


Parsing Based on LL Table: Example

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

1: $E \to TE'$
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: $i * i \in L(G_{expr3})$?

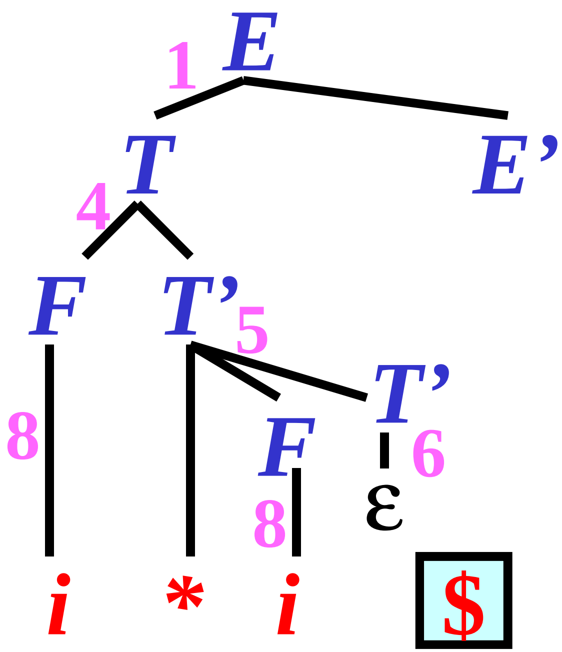


Parsing Based on LL Table: Example

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

1: $E \to TE'$
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: $i * i \in L(G_{expr3})$?

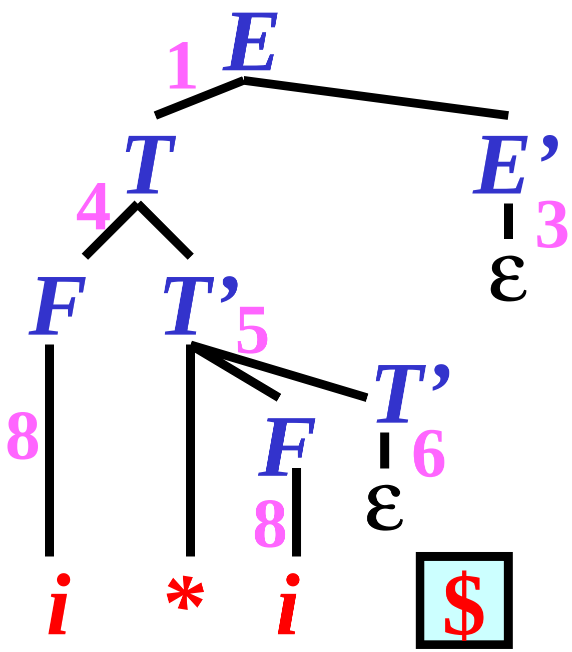


Parsing Based on LL Table: Example

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

1: $E \to TE'$
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: $i * i \in L(G_{expr3})$?



LL Grammars with ε-rules: Definition

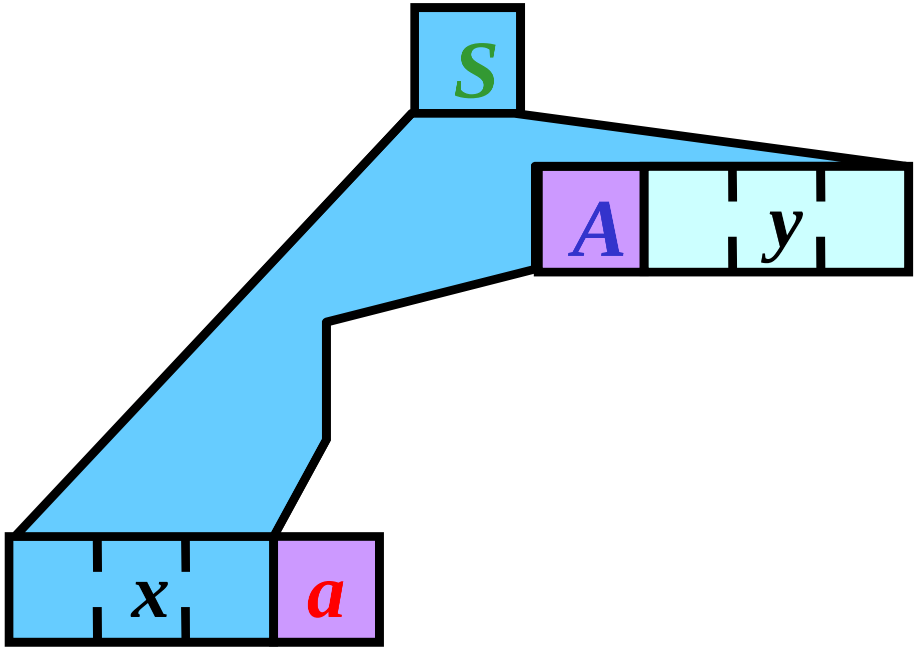
Definition: Let $G = (N, T, P, S)$ be a CFG. G is an *LL grammar* if for every $a \in T$ and every $A \in N$ there is **no more than one** A-rule $A \rightarrow X_1X_2…X_n \in P$ such that $a \in Predict(A \rightarrow X_1X_2…X_n)$

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

Illustration:



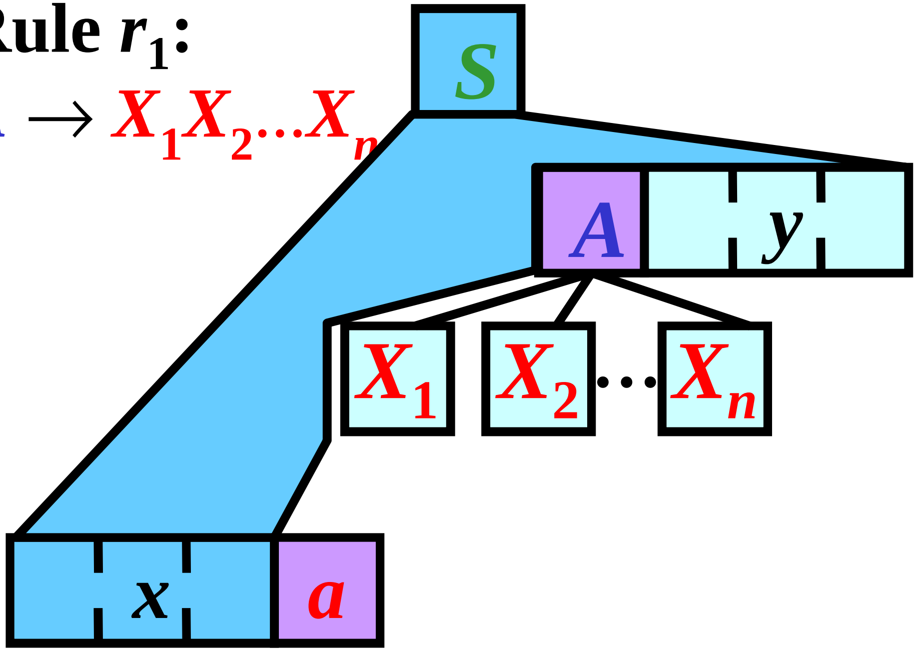
LL Grammars with ε-rules: Definition

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:

$A \rightarrow X_1X_2…X_n$



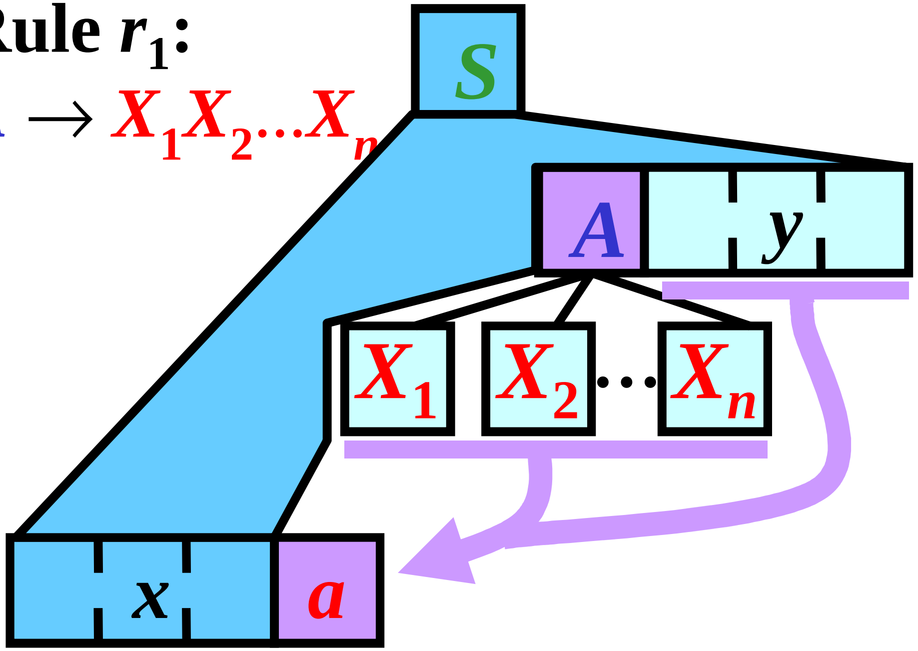
LL Grammars with ε-rules: Definition

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:

$A \rightarrow X_1X_2\ldots X_n$



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

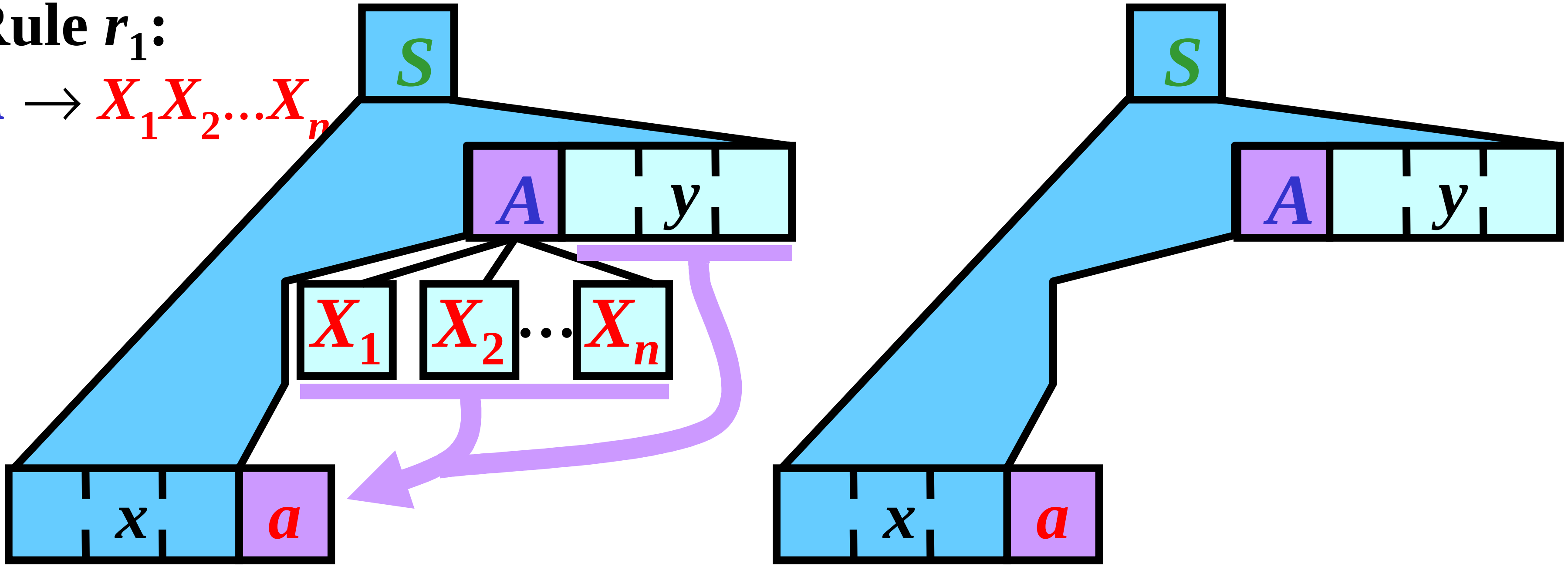
LL Grammars with ε-rules: Definition

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

Illustration:

Rule r_1:

$A \rightarrow X_1X_2\ldots X_n$

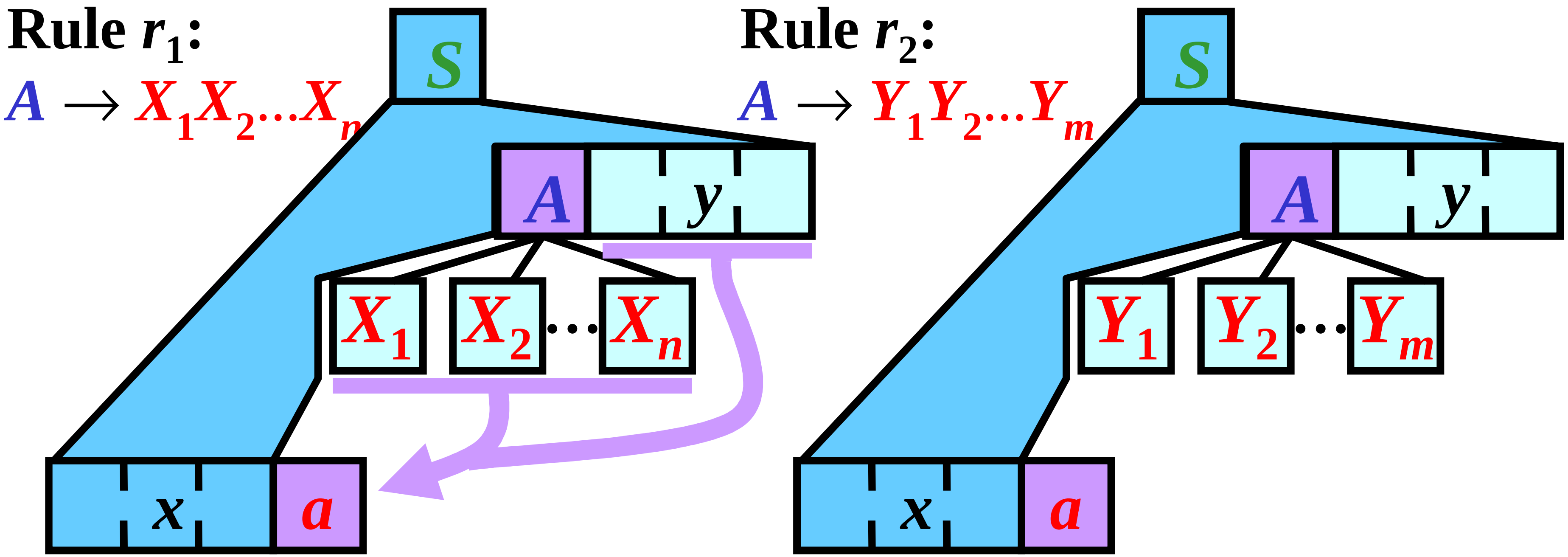


$a \in \mathit{Predict}(A \rightarrow X_1X_2\ldots X_n)$

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

Illustration:



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

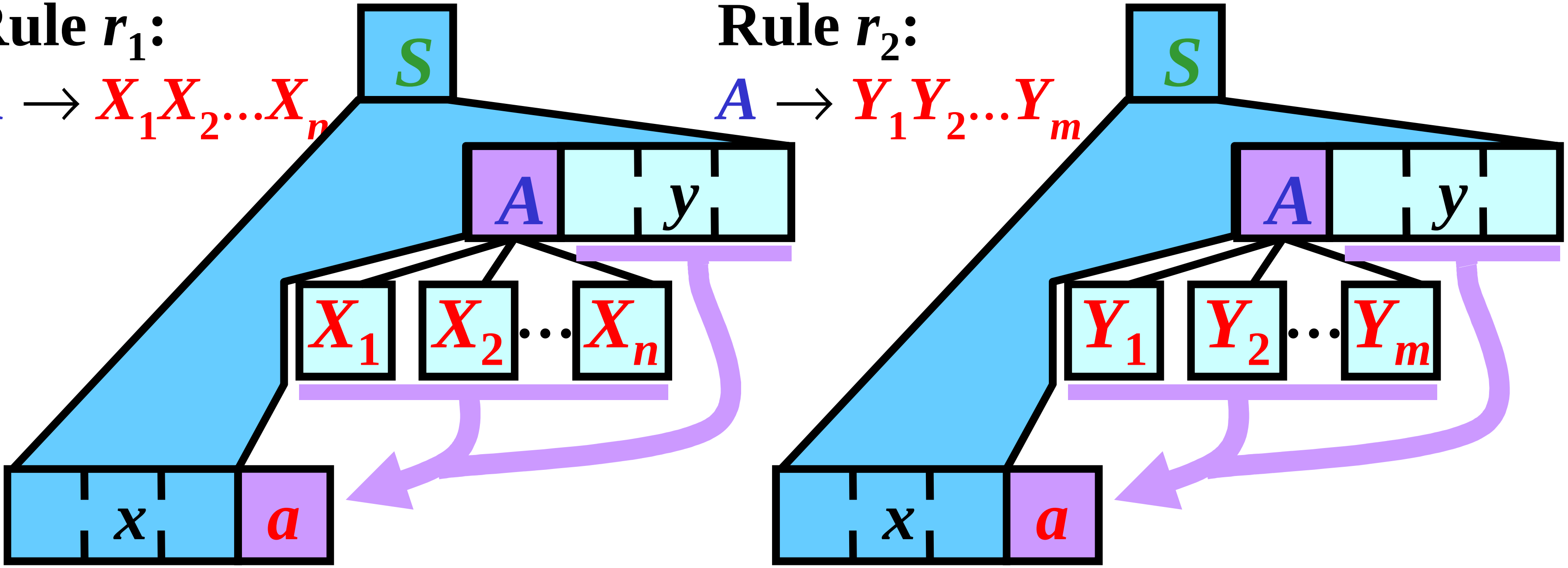
LL Grammars with ε-rules: Definition

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

Illustration:

Rule r_1: $A \to X_1X_2\ldots X_n$

Rule r_2: $A \to Y_1Y_2\ldots Y_m$



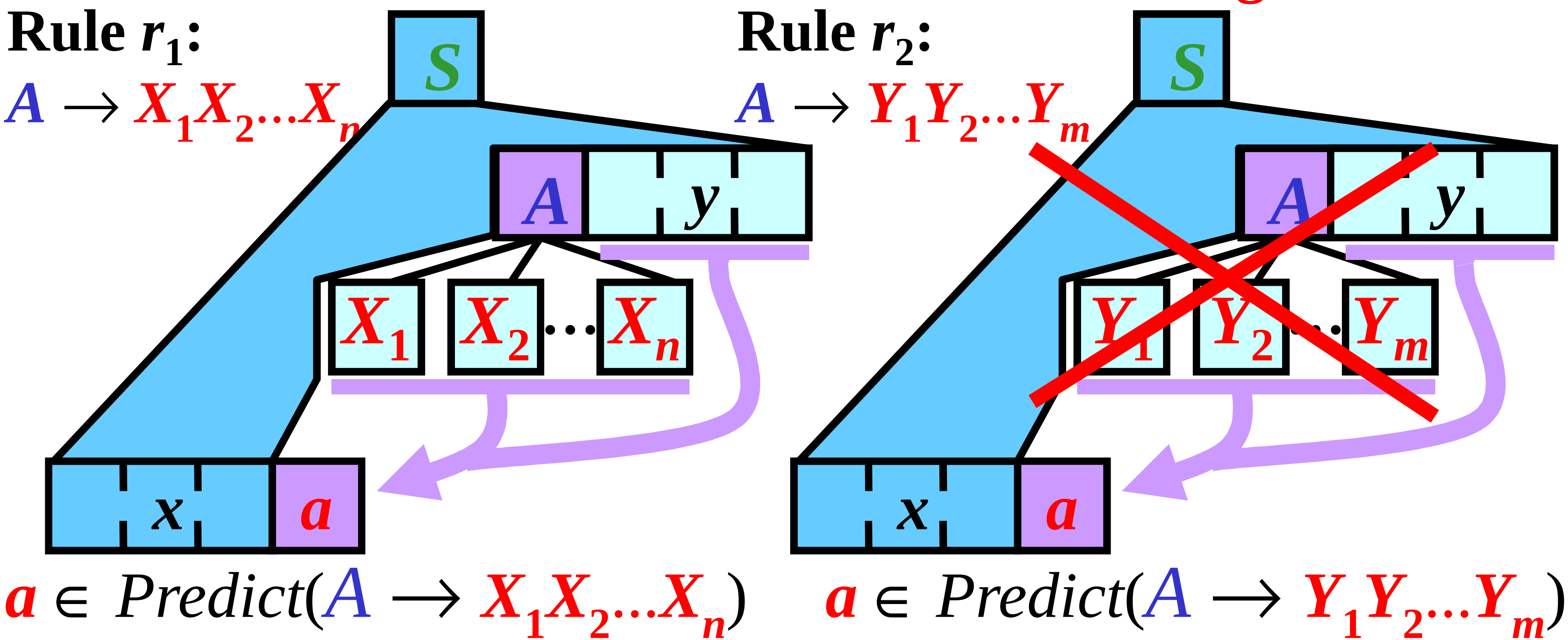
$a \in Predict(A \to X_1X_2\ldots X_n)$ $a \in Predict(A \to Y_1Y_2\ldots Y_m)$

LL 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:

Ruled out in an LL grammar



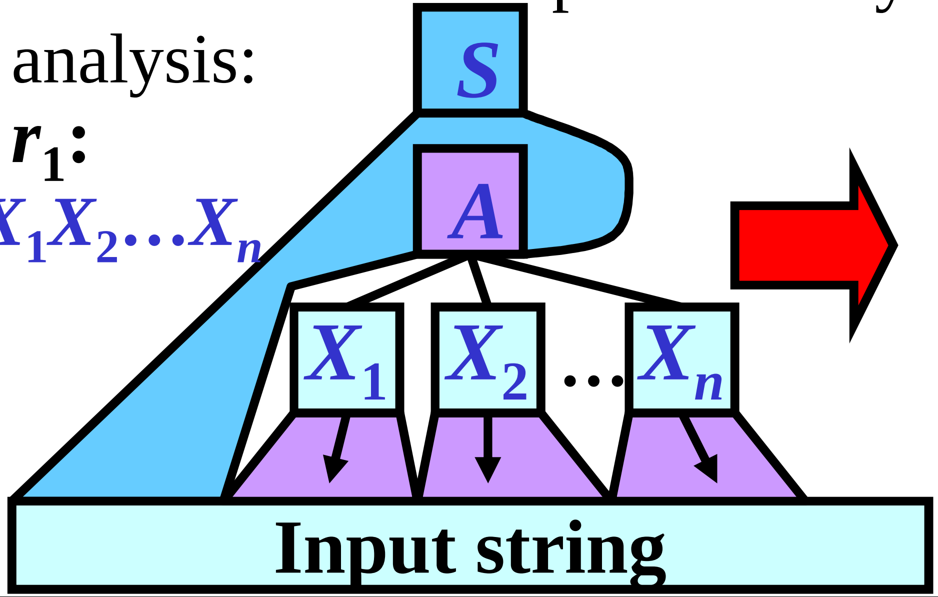
LL Analyzer Implementation

1) Recursive-Descent Parsing

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

Rule r_1:

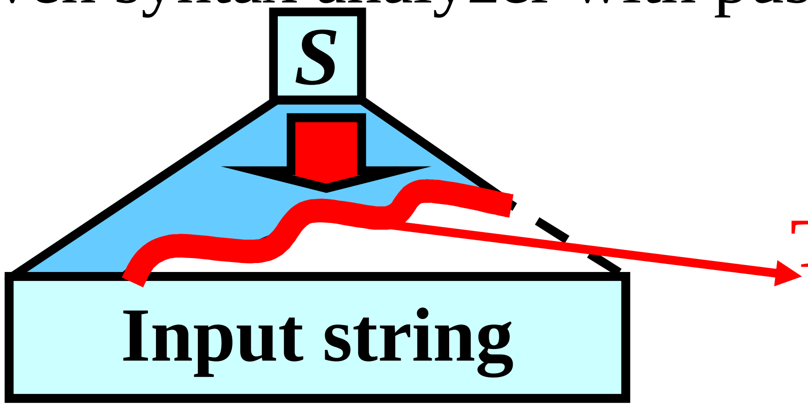
$A \rightarrow X_1X_2\ldots X_n$




```
function A: boolean;
begin
  {X1 analysis}
  {X2 analysis}
        ...
  {Xn 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'* → +*TE'*, 3: *E'* → ε

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

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

Recursive Descent: Example 3/4

- For *T'* ∈ *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 $F \in N$: Rules 7: $F \to (E)$, 8: $F \to i$

```
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:

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

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

Start: **GetNextToken;**
Call E;

Input string:

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

***E*:**

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

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

Start: **GetNextToken;**

Call E;

Input string:

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

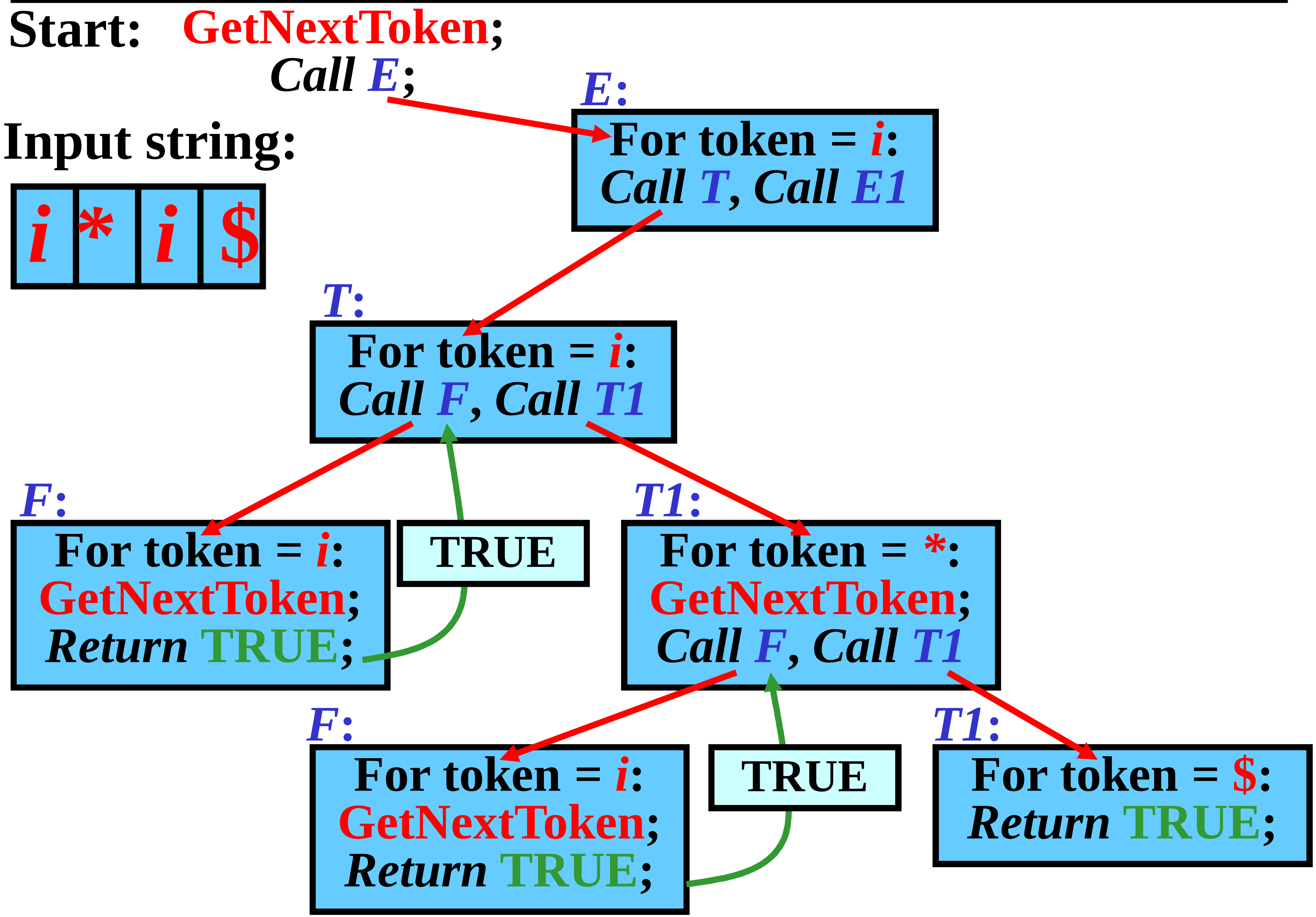
TRUE

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

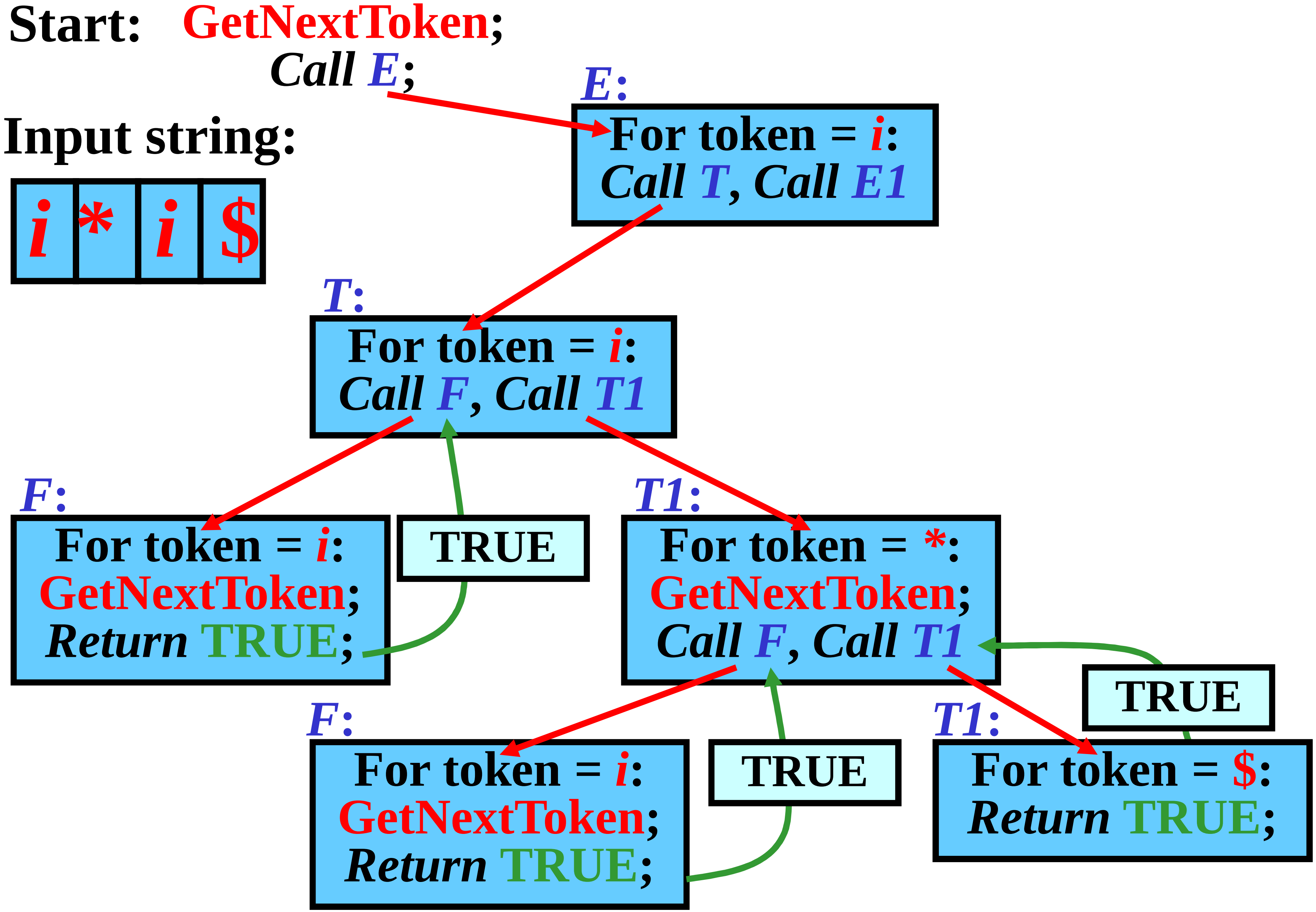
***F*:**
For token = *i*:
GetNextToken;
***Return* TRUE;**

TRUE

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

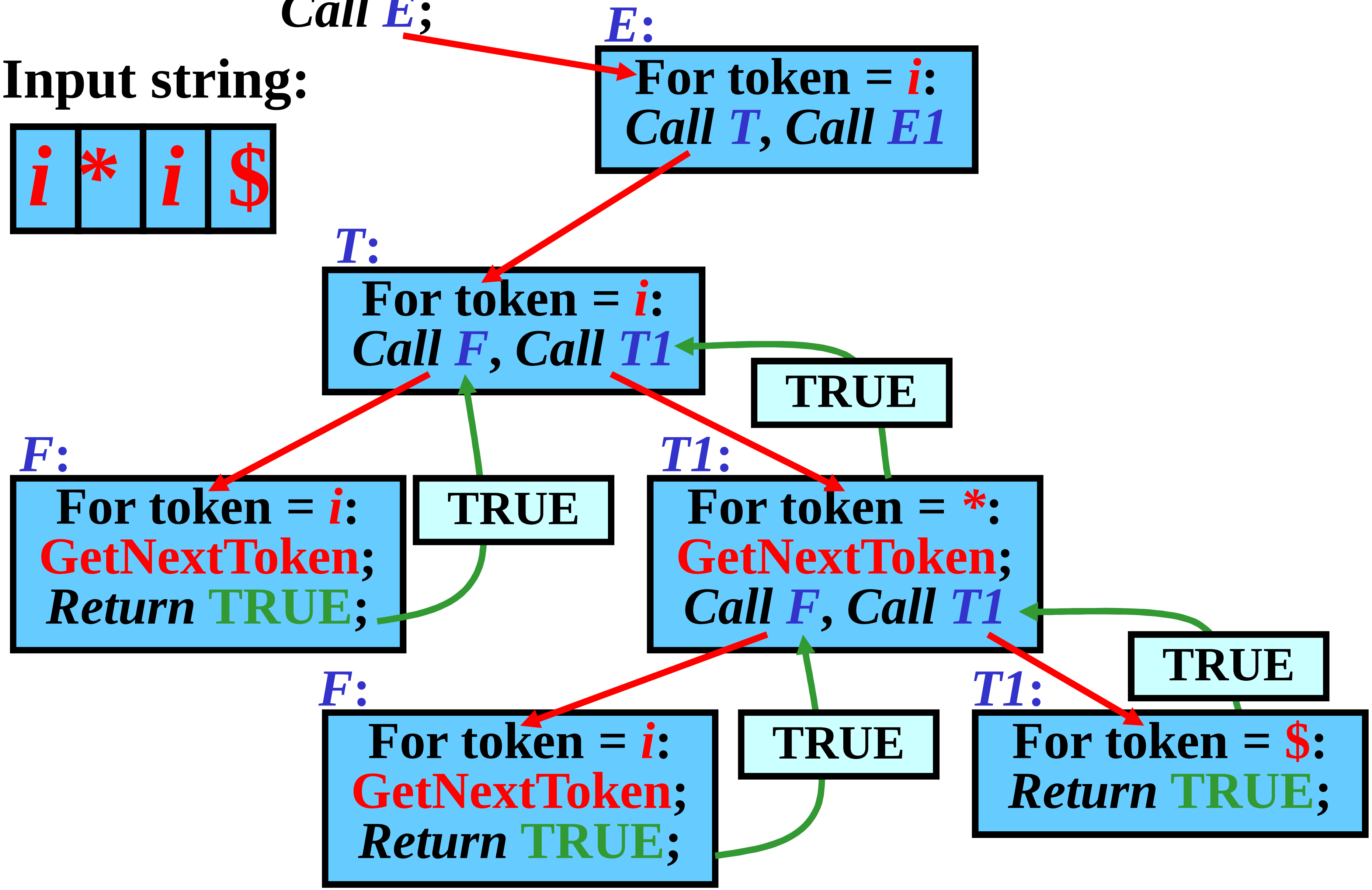


Recursive Descent: Illustration for *i*i$*



Recursive Descent: Illustration for *i*i$*

Start: **GetNextToken;**
Call E;

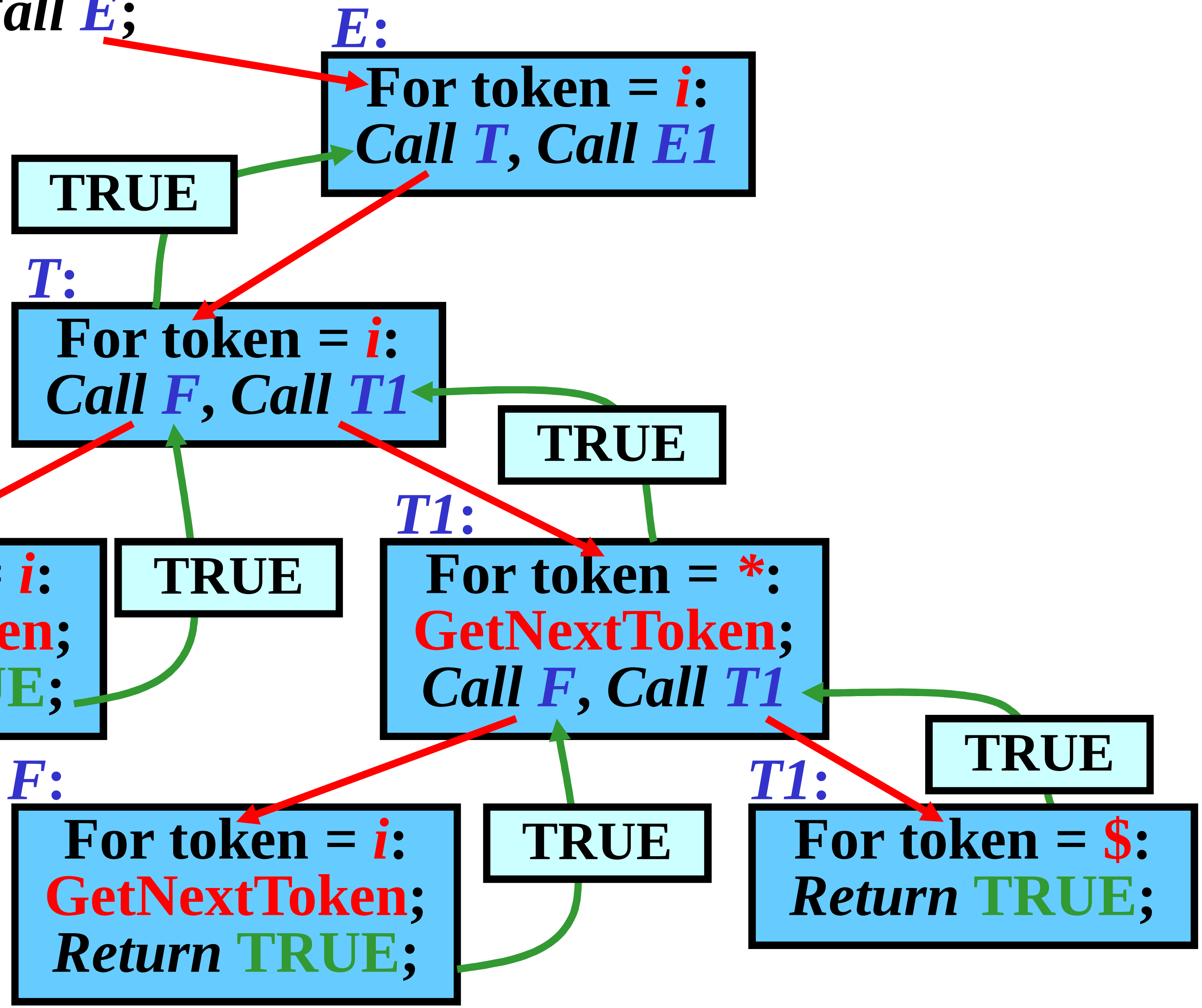


Recursive Descent: Illustration for *i***i*$

Start: **GetNextToken**;
Call E;

Input string:

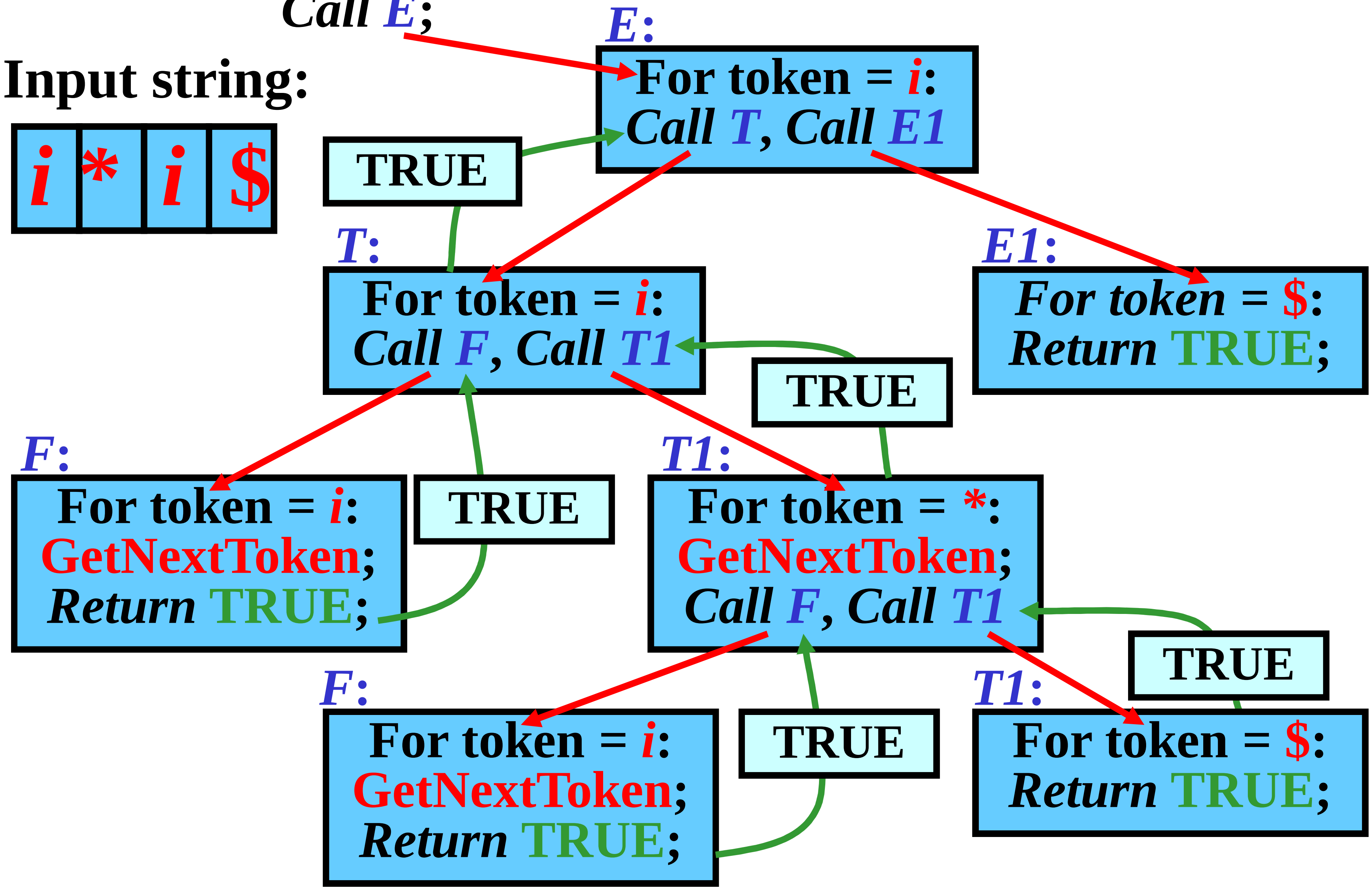
i	*	*i*	$



Recursive Descent: Illustration for *i*i$*

Start: **GetNextToken;**
Call E;

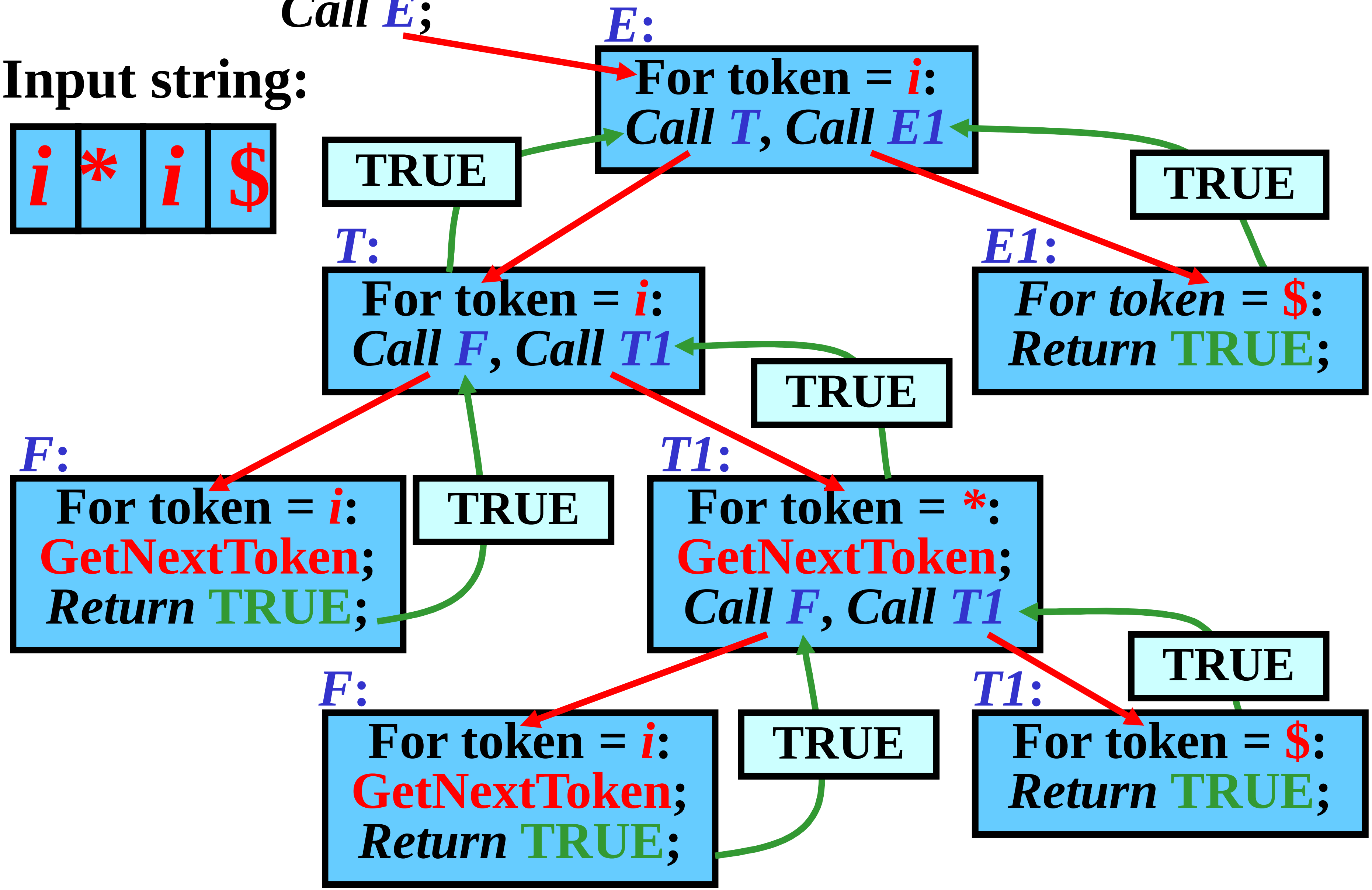
Input string:



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

Start: **GetNextToken;**
Call E;

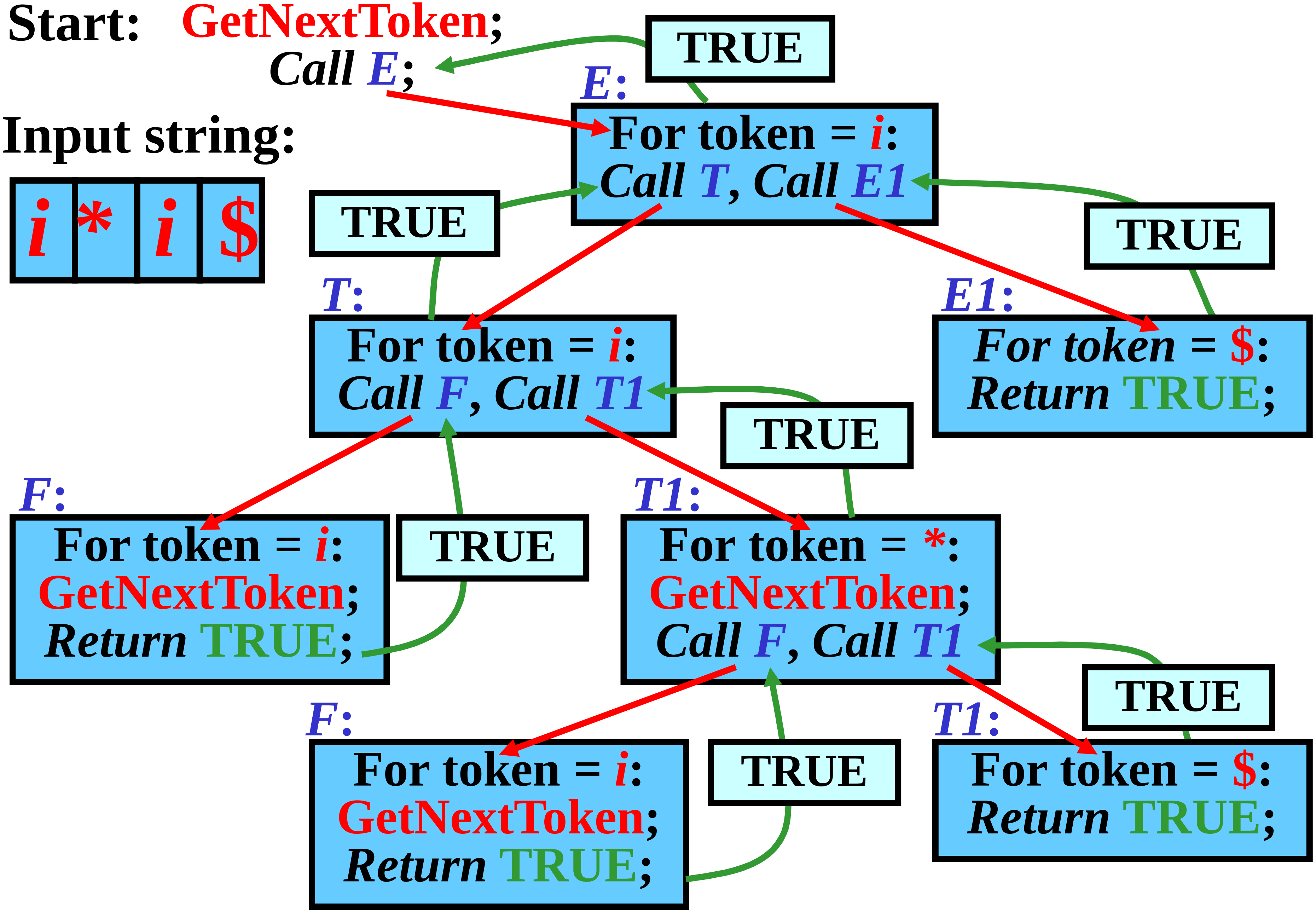
Input string:



Recursive Descent: Illustration for *i*i$*

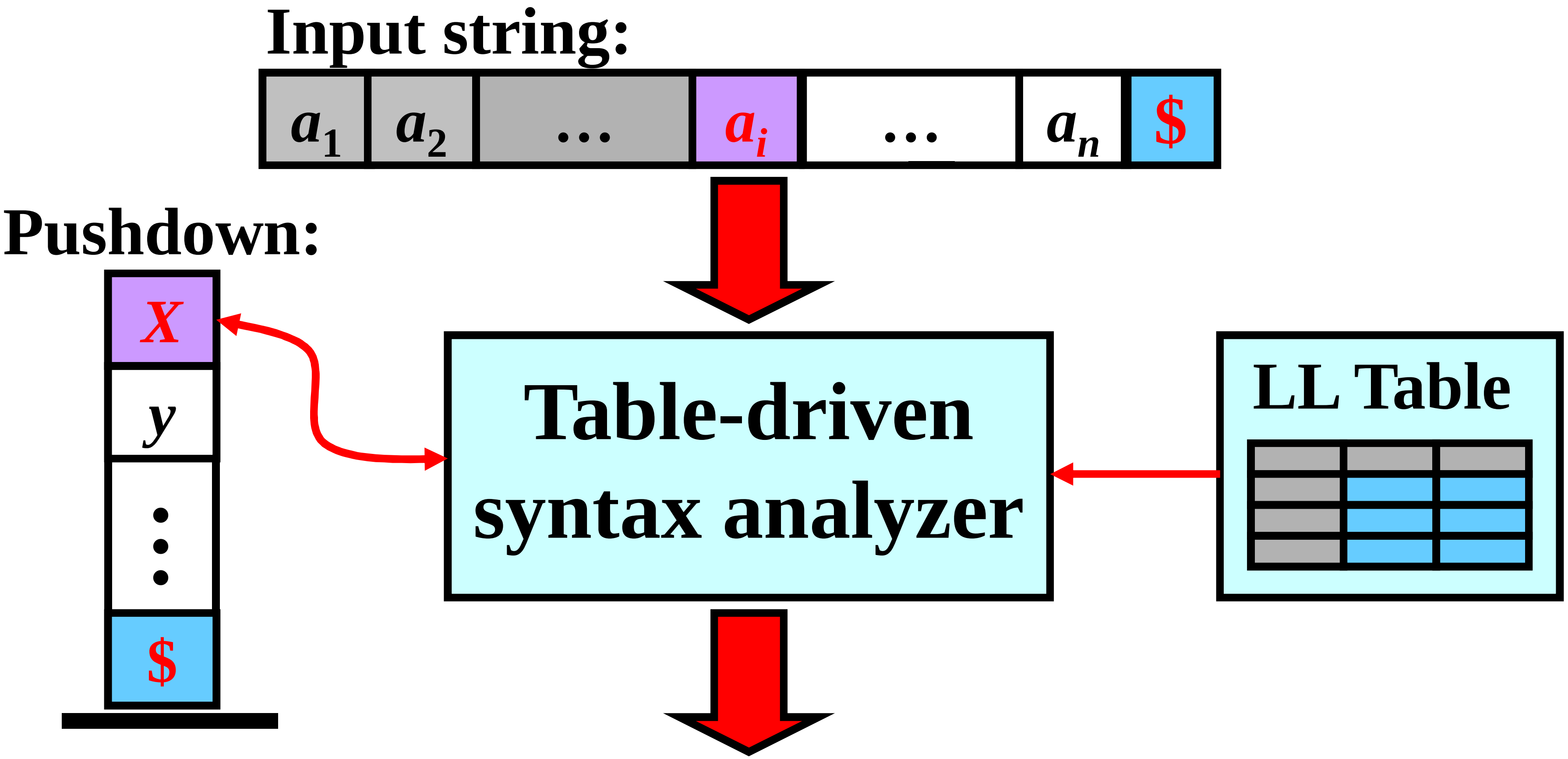
Start: **GetNextToken**;
Call ***E***;

Input string:



Predictive Parsing

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



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

Table-Driven Parsing: Algorithm

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

---

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

end

Table-Driven Parsing: Example

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

Input string: *i* * *i* $

Pushdown	Input	Rule	Derivation

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: $F \to i$

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

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

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: $E \to TE'$	$\underline{E} \Rightarrow \underline{T}E'$
$E'T	i*i$	4: $T \to FT'$	$\Rightarrow \underline{F}T'E'$
$E'T'F	i*i$	8: $F \to i$	$\Rightarrow i\underline{T'}E'$

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: $F \to i$

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: $T \rightarrow FT'$	$\Rightarrow \underline{F}T'E'$
$E'T'F	i*i$	8: $F \rightarrow i$	$\Rightarrow i\underline{T'}E'$
$E'T'i	i*i$		

Table-Driven Parsing: Example

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

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: $T \rightarrow FT'$	$\Rightarrow \underline{F}T'E'$
$E’T’F	i*i$	8: $F \rightarrow i$	$\Rightarrow i\underline{T'}E'$
$E’T’i	i*i$		
$E’T’	*i$	5: $T' \rightarrow *FT'$	$\Rightarrow i*\underline{F}T'E'$

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

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: $E \to TE'$	$\underline{E} \Rightarrow \underline{T}E'$
$E'T	i*i$	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: $F \to i$	$\Rightarrow i*i\underline{T'}E'$

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: $F \to i$

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’*	*E* ⇒ *TE’*
$*E’T*	*i***i*$	4: *T* → *FT’*	⇒ *FT’E’*
$*E’T’F*	*i***i*$	8: *F* → *i*	⇒ *iT’E’*
$*E’T’i*	*i***i*$		
$*E’T’*	**i*$	5: *T’* → **FT’*	⇒ *i***FT’E’*
$*E’T’F**	**i*$		
$*E’T’F*	*i*$	8: *F* → *i*	⇒ *i***iT’E’*
$*E’T’i*	*i*$		

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: $E \rightarrow TE'$
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: $T \rightarrow FT'$	$\Rightarrow \underline{F}T'E'$
$E'T'F	i*i$	8: $F \rightarrow i$	$\Rightarrow i\underline{T'}E'$
$E'T'i	i*i$		
$E'T'	*i$	5: $T' \rightarrow *FT'$	$\Rightarrow i*\underline{F}T'E'$
$E'T'F*	*i$		
$E'T'F	i$	8: $F \rightarrow i$	$\Rightarrow i*i\underline{T'}E'$
$E'T'i	i$		
$E'T'	$	6: $T' \rightarrow \varepsilon$	$\Rightarrow i*i\underline{E'}$

Table-Driven Parsing: Example

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

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: $E' \to \varepsilon$	$\Rightarrow i*i$

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: $F \to i$

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'	E ⇒ TE'
$E'T	i*i$	4: T → FT'	⇒ FT'E'
$E'T'F	i*i$	8: F → i	⇒ iT'E'
$E'T'i	i*i$		
$E'T'	*i$	5: T' → *FT'	⇒ i*FT'E'
$E'T'F*	*i$		
$E'T'F	i$	8: F → i	⇒ i*iT'E'
$E'T'i	i$		
$E'T'	$	6: T' → ε	⇒ i*iE'
$E'	$	3: E' → ε	⇒ 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: $E \to TE'$	$\underline{E} \Rightarrow \underline{T}E'$
$\$E'T$	$i*i\$$	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: $E' \to \varepsilon$	$\Rightarrow i*i$
$\$$	$\$$		

Rules:

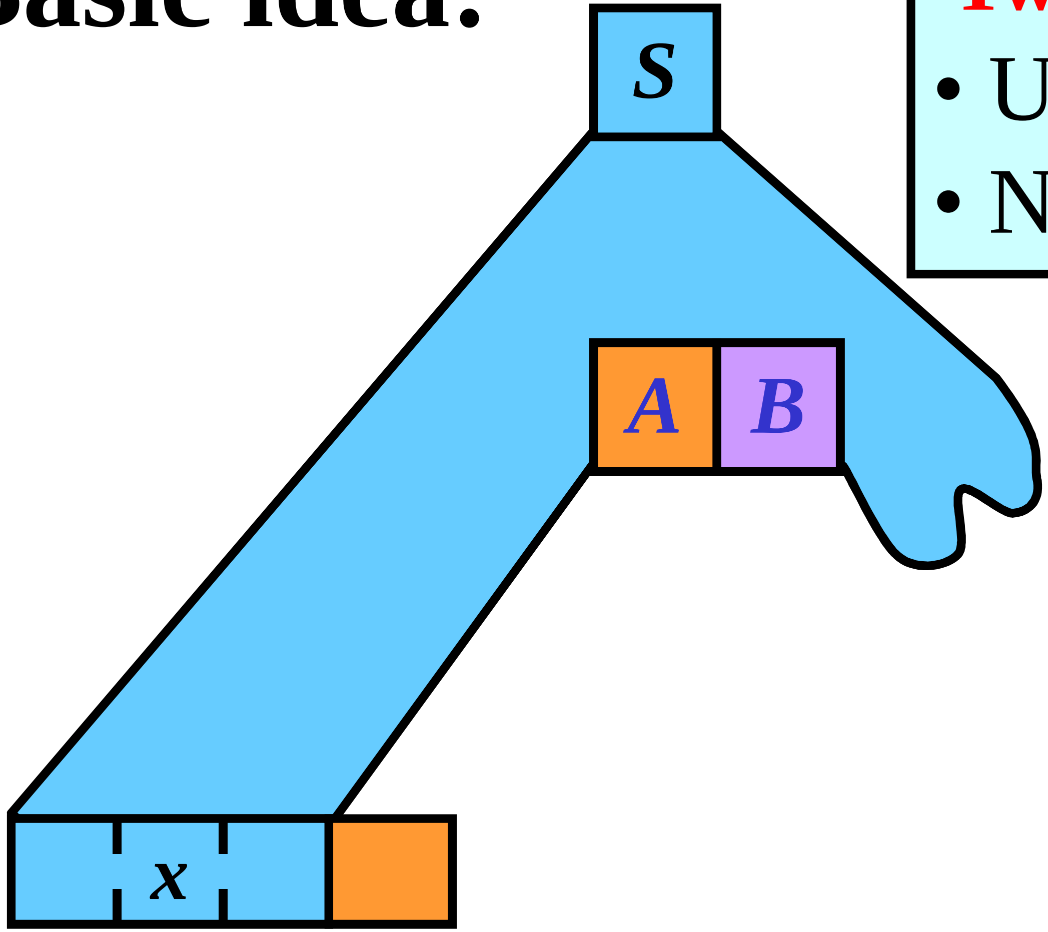
1: $E \to TE'$
2: $E' \to +TE'$
3: $E' \to \varepsilon$
4: $T \to FT'$
5: $T' \to *FT'$
6: $T' \to \varepsilon$
7: $F \to (E)$
8: $F \to i$

Success

Left parse: 1485863

Handling Errors: Introduction

Basic idea:

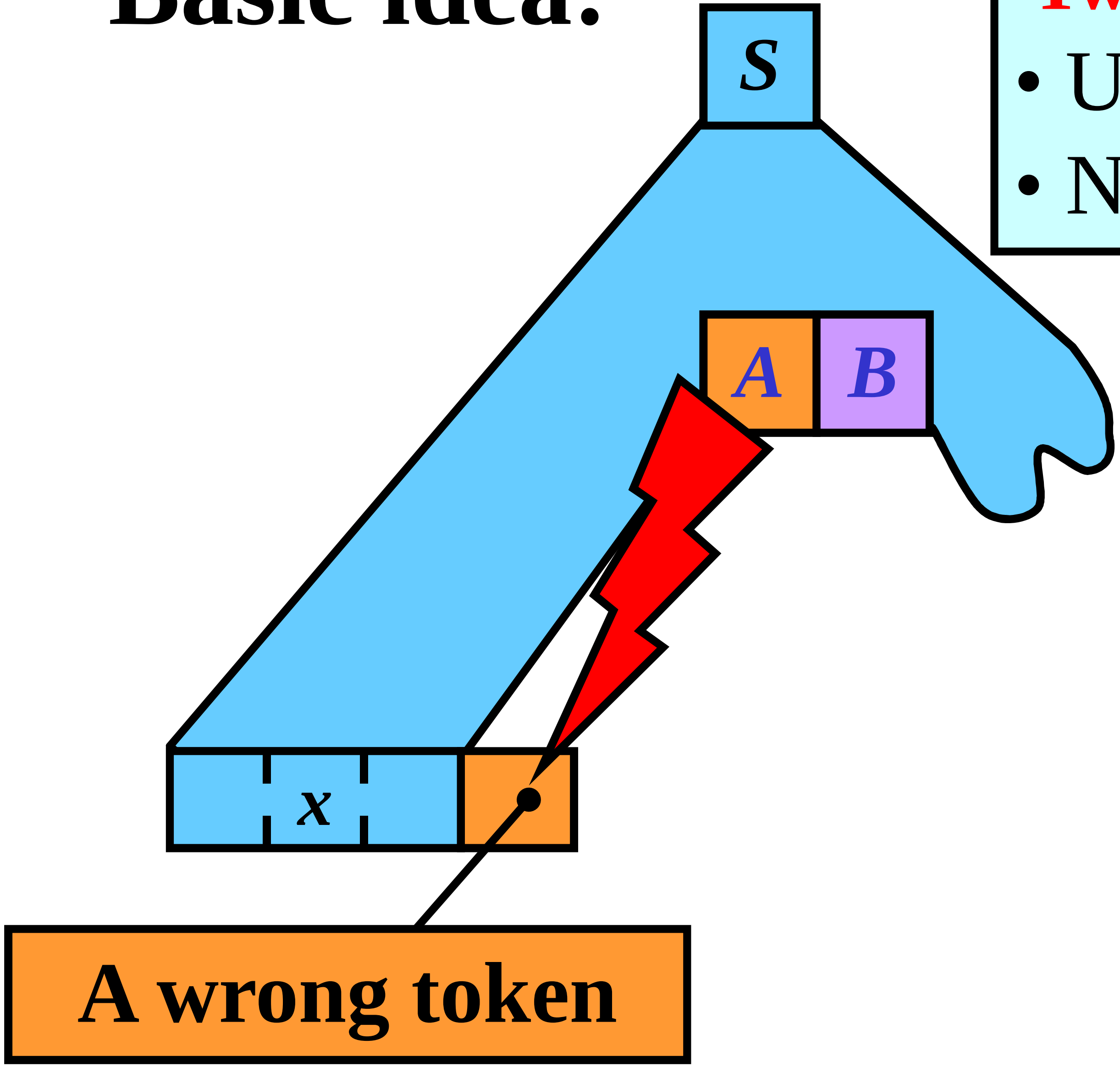


Two kinds of errors:

- Unexpected token
- No rule applicable

Handling Errors: Introduction

Basic idea:

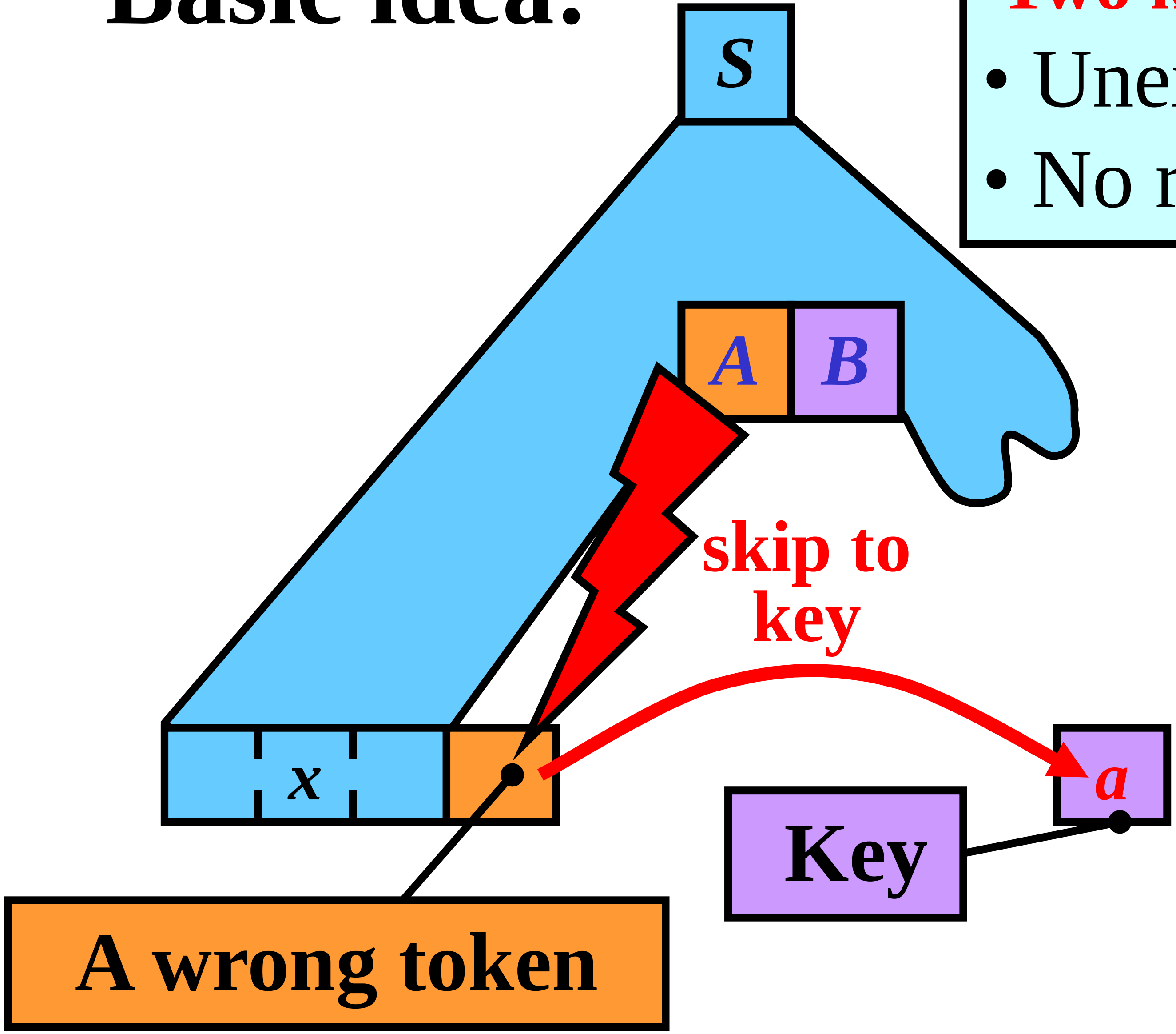


Two kinds of errors:

- Unexpected token
- No rule applicable

Handling Errors: Introduction

Basic idea:

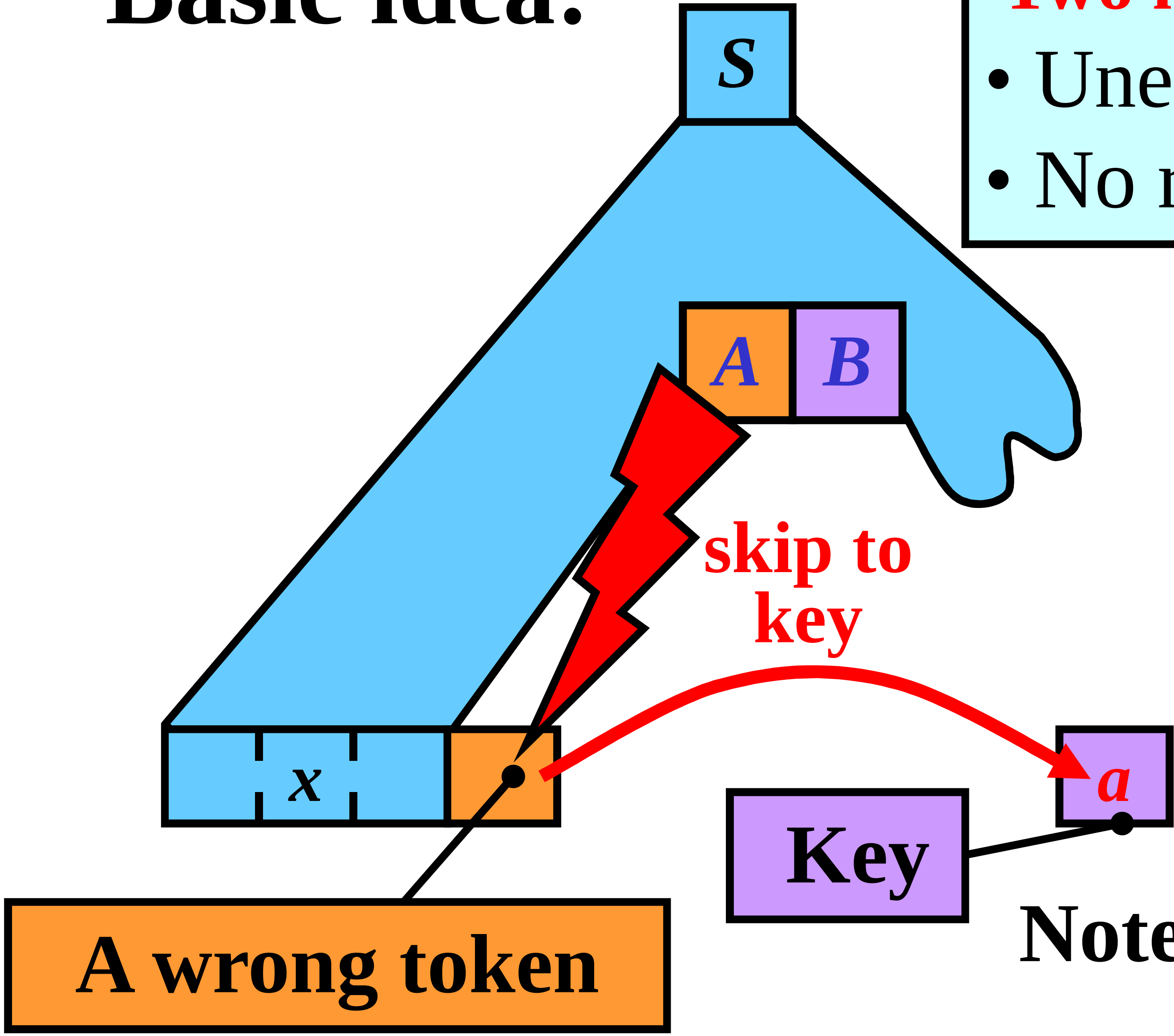


Two kinds of errors:

- Unexpected token
- No rule applicable

Handling Errors: Introduction

Basic idea:



Two kinds of errors:

- Unexpected token
- No rule applicable

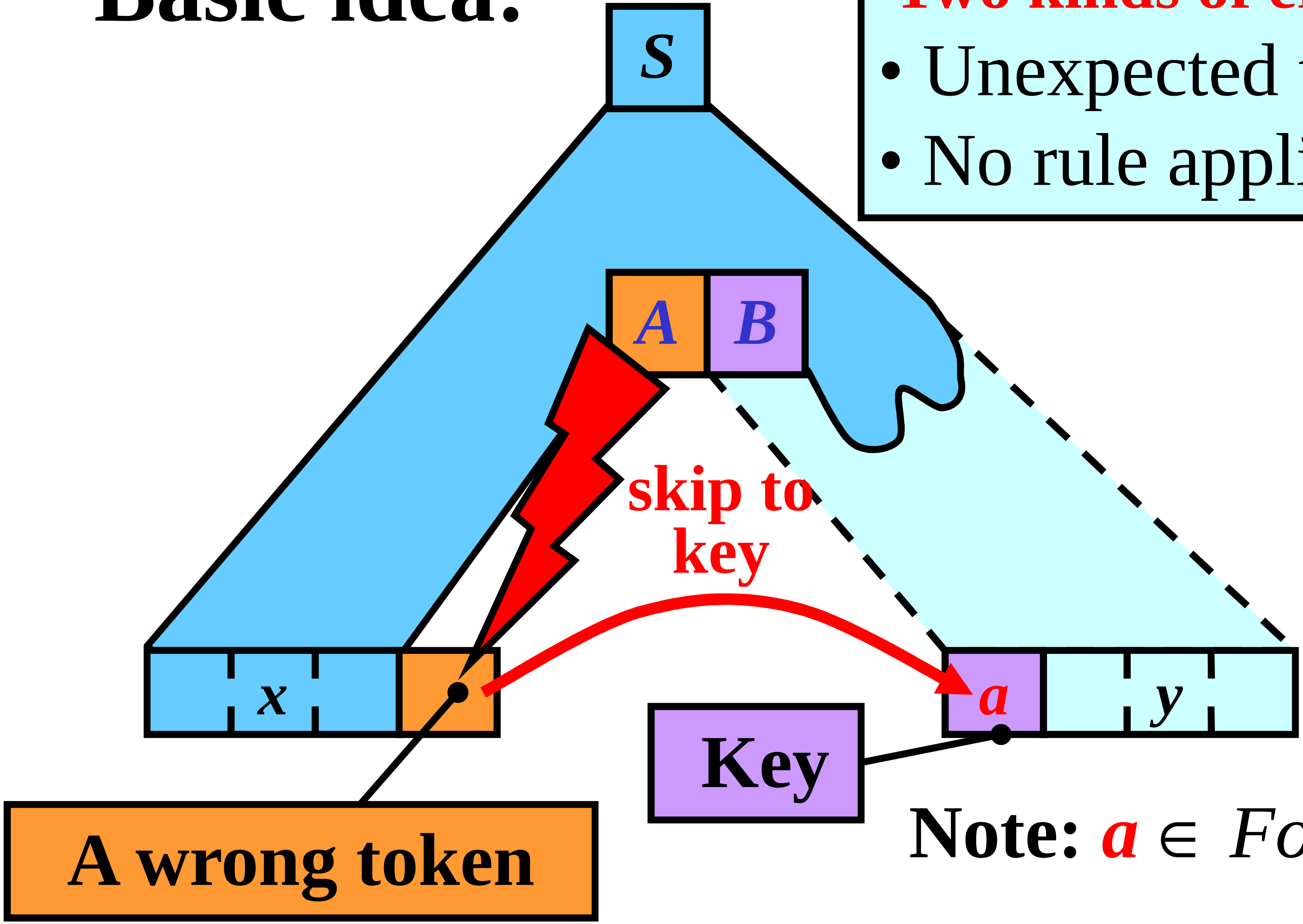
Note: $a \in \mathit{Follow}(A)$

Handling Errors: Introduction

Basic idea:

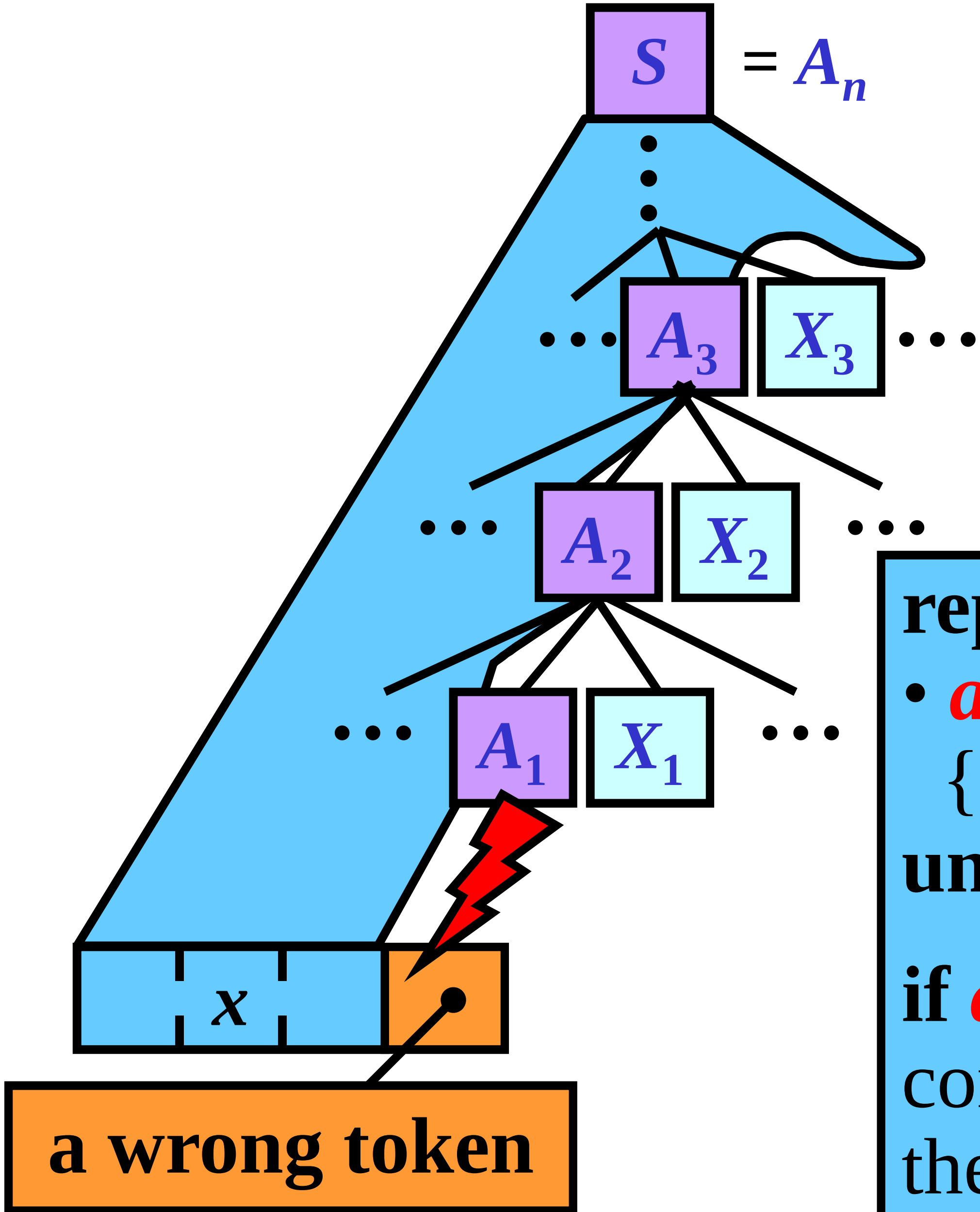
Two kinds of errors:

- Unexpected token
- No rule applicable



Note: $a \in \mathit{Follow}(A)$

Panic-Mode (Hartmann) Error Recovery



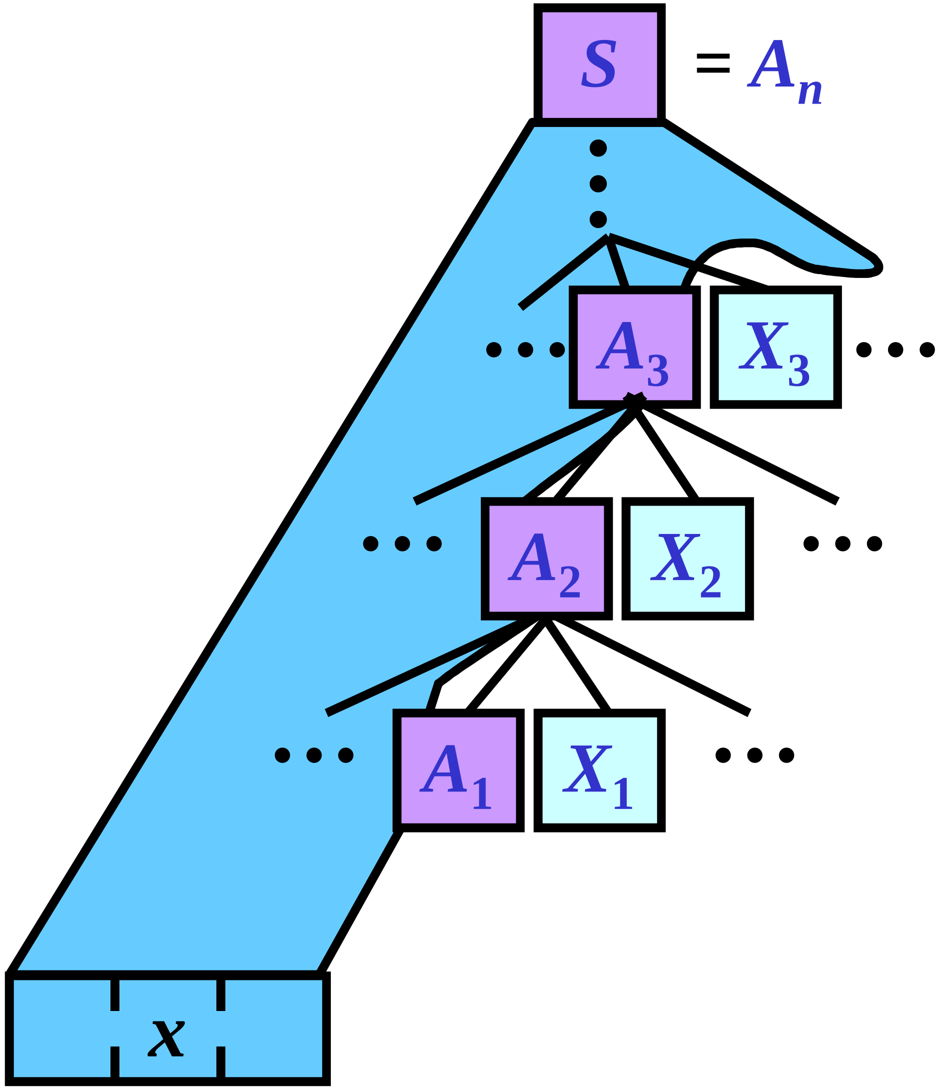
- Let ***Context***(A_1) = ***Follow***$(A_1) \cup$ ***Follow***$(A_2) \cup$ … ***Follow***(A_n)

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

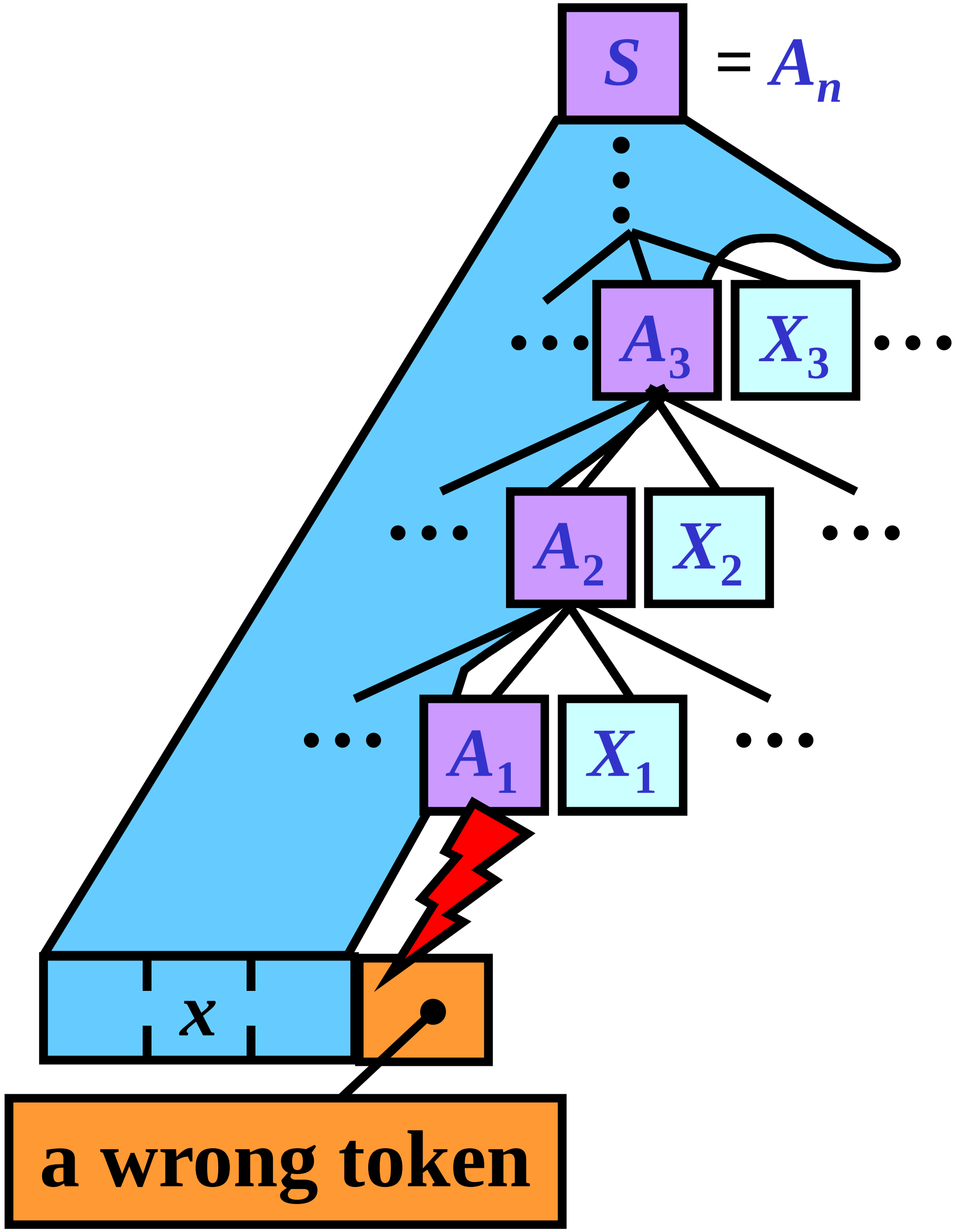
until a in ***Context***(A_1)

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

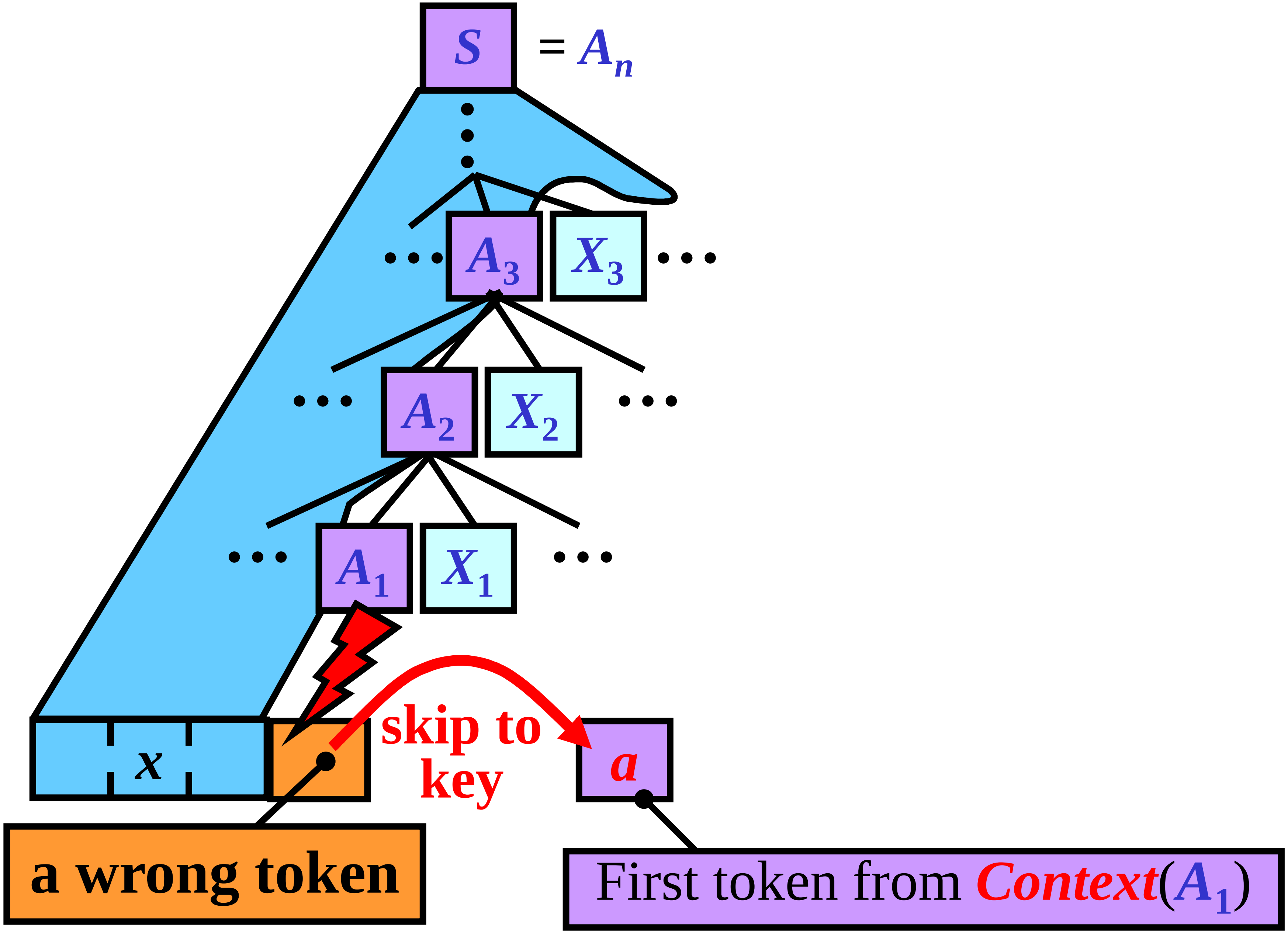
Panic-Mode Recovery: Illustration 1/2



Panic-Mode Recovery: Illustration 1/2

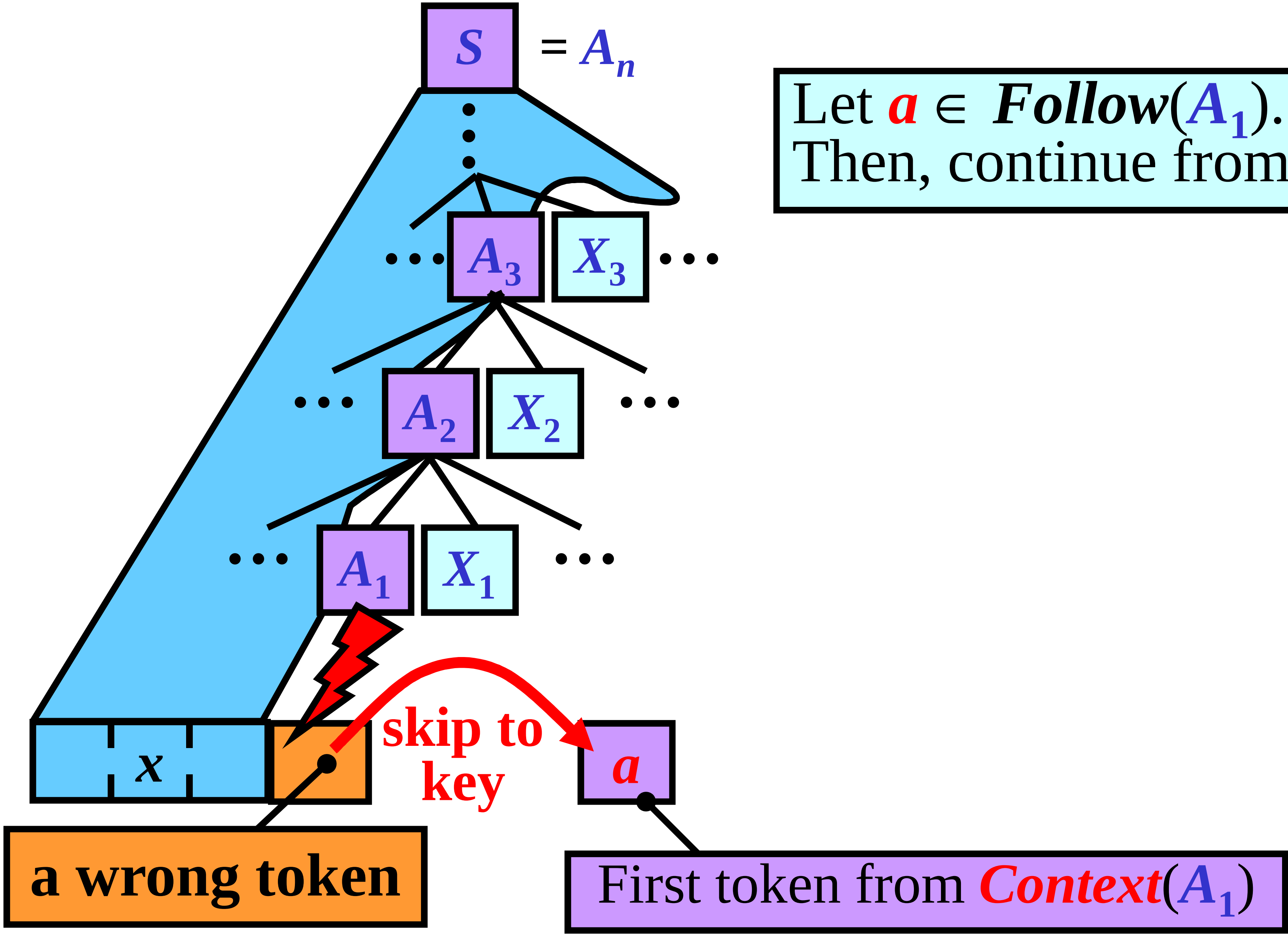


Panic-Mode Recovery: Illustration 1/2



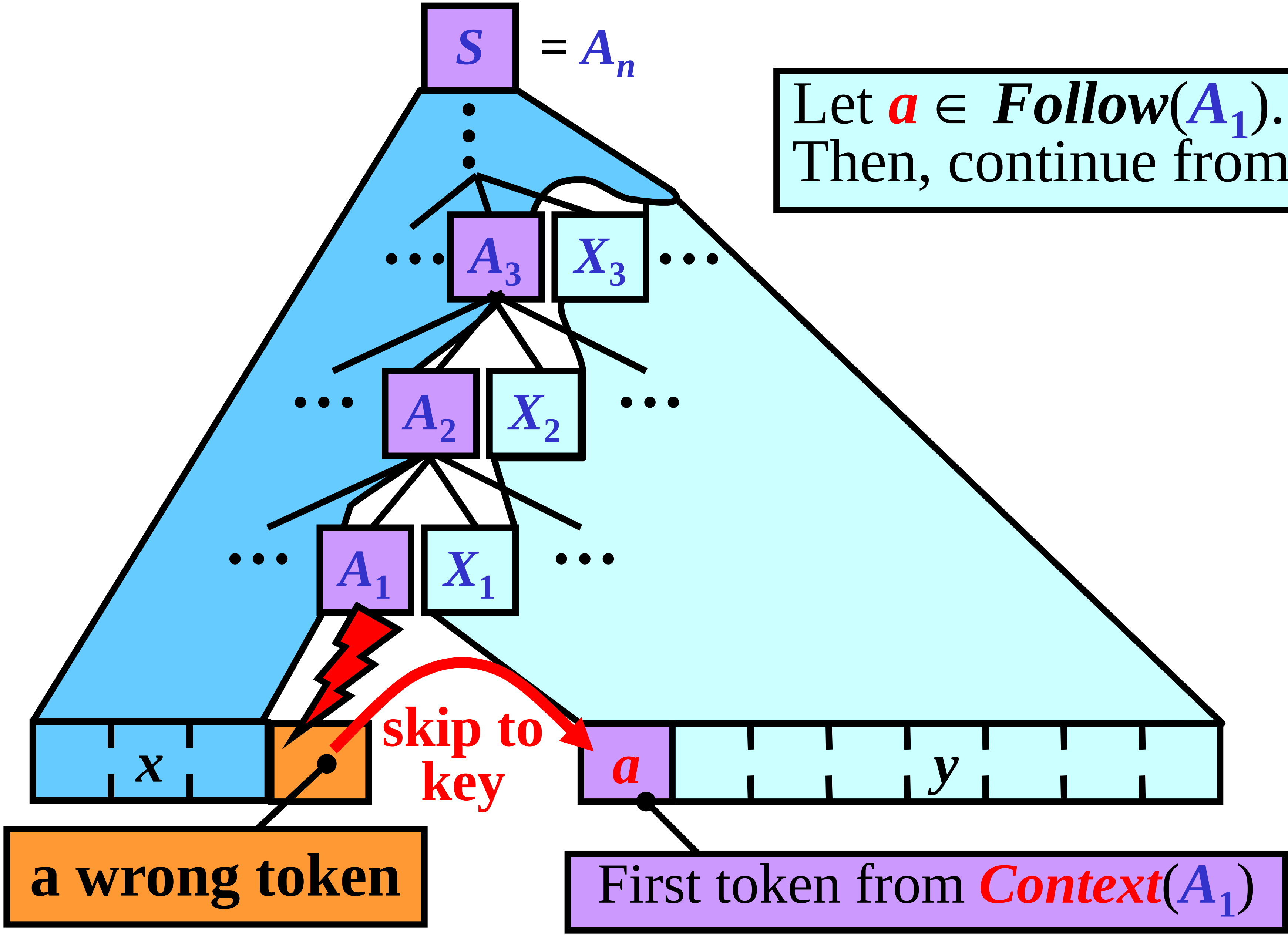
Panic-Mode Recovery: Illustration 1/2

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

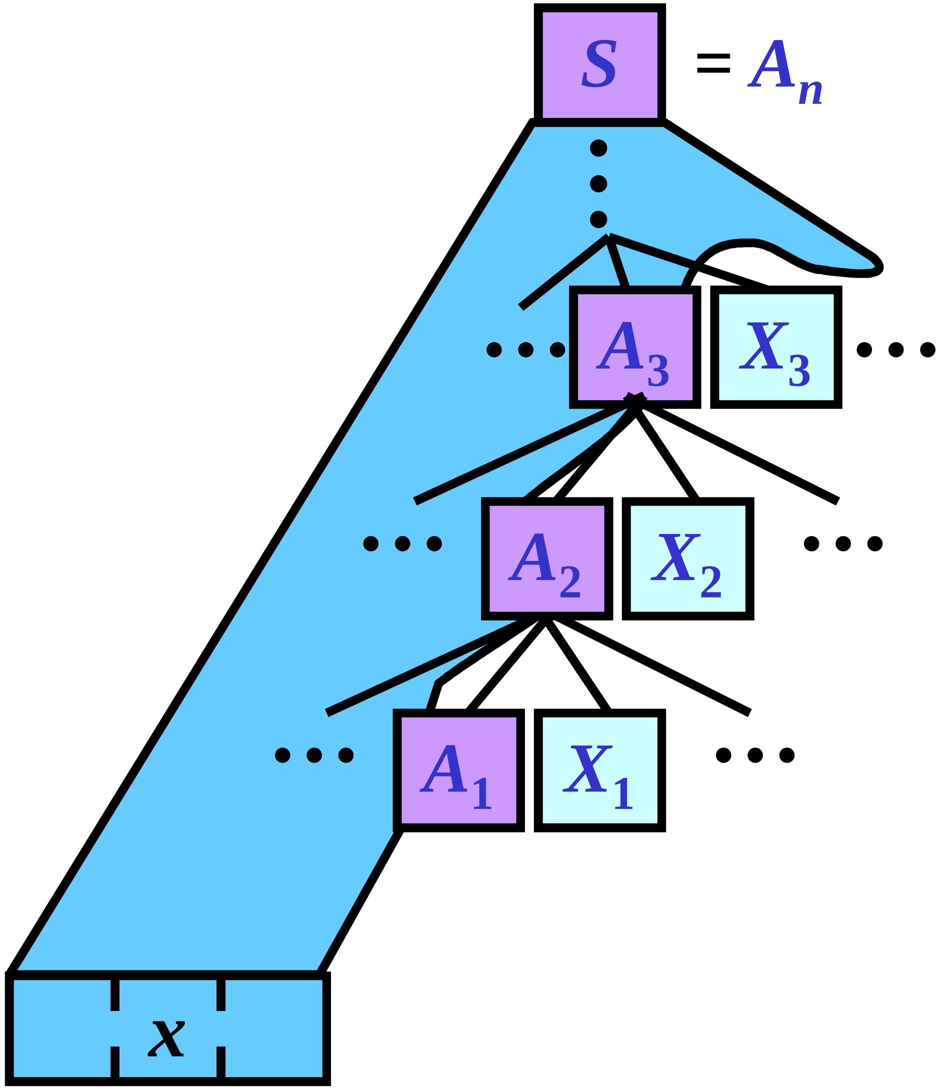


Panic-Mode Recovery: Illustration 1/2

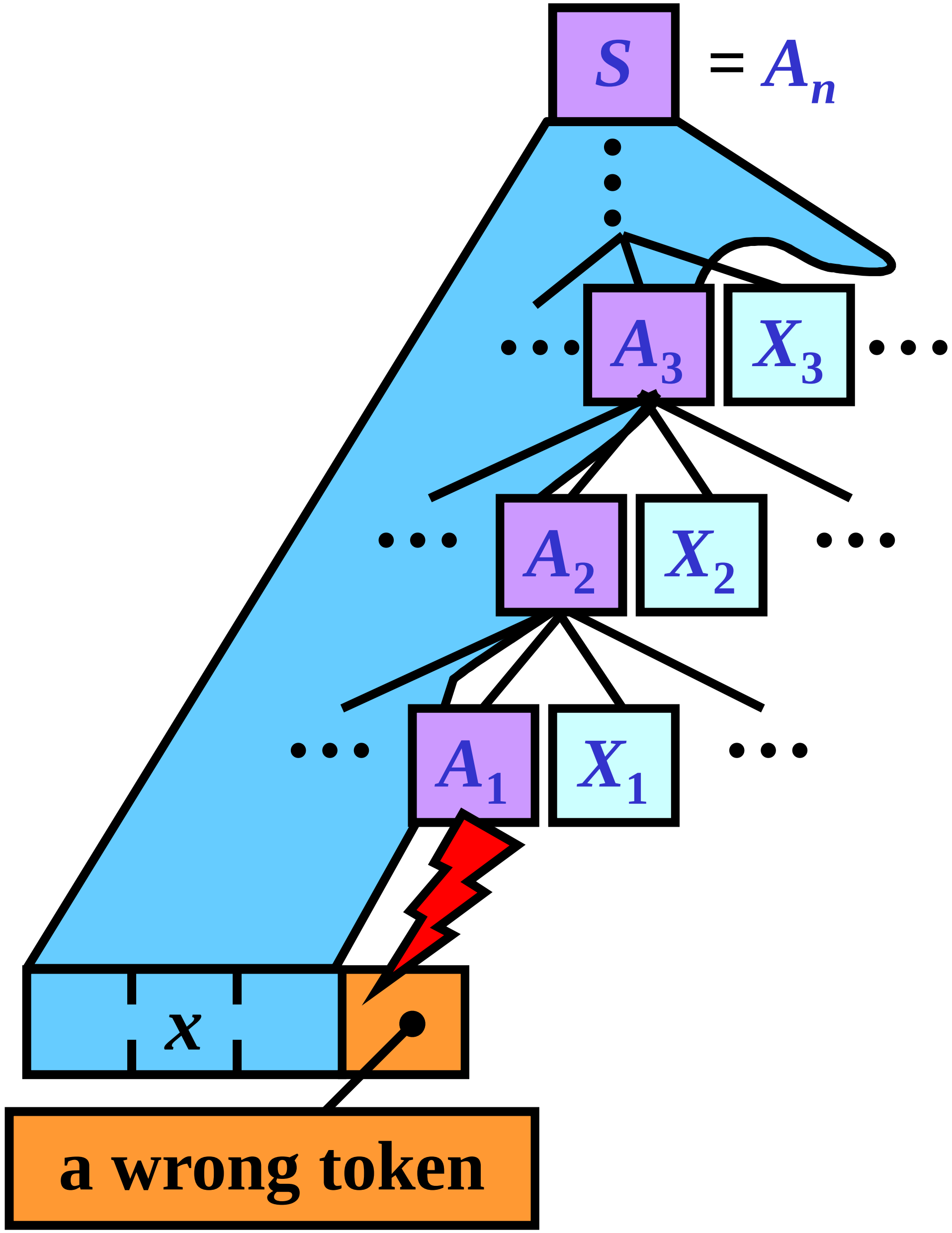
Let $a \in \textbf{\textit{Follow}}(A_1)$.
Then, continue from X_1



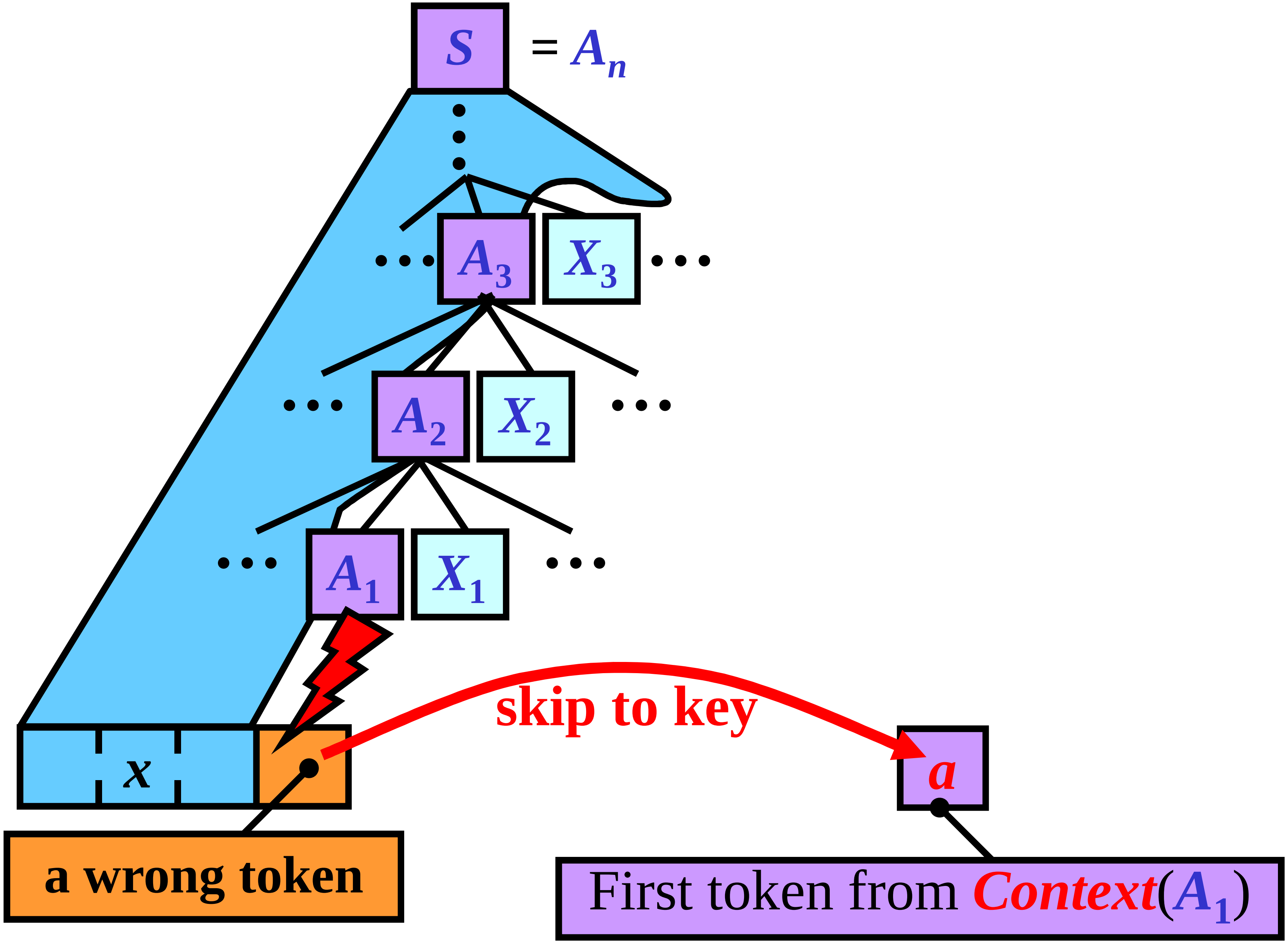
Panic-Mode Recovery: Illustration 2/2



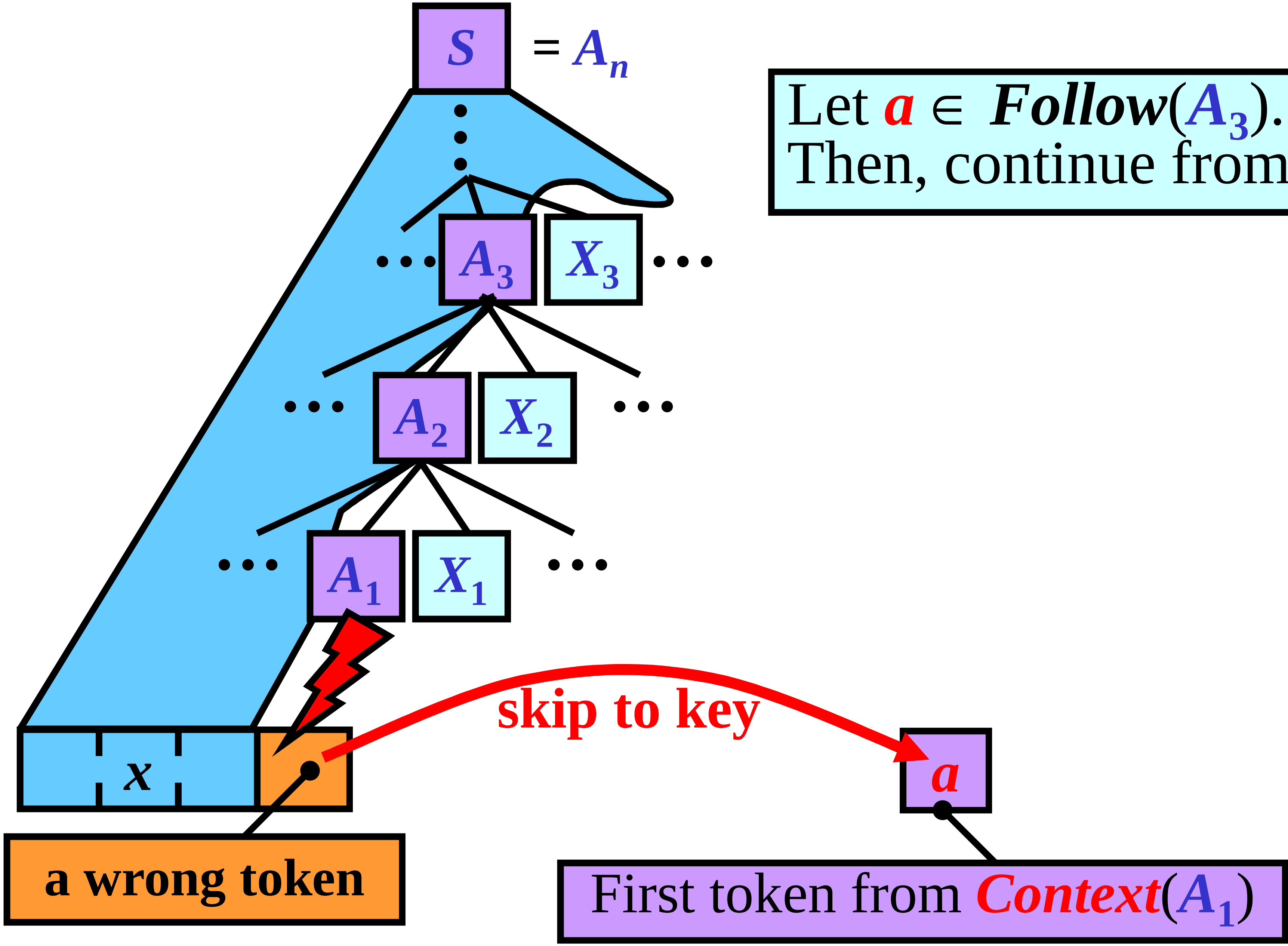
Panic-Mode Recovery: Illustration 2/2



Panic-Mode Recovery: Illustration 2/2

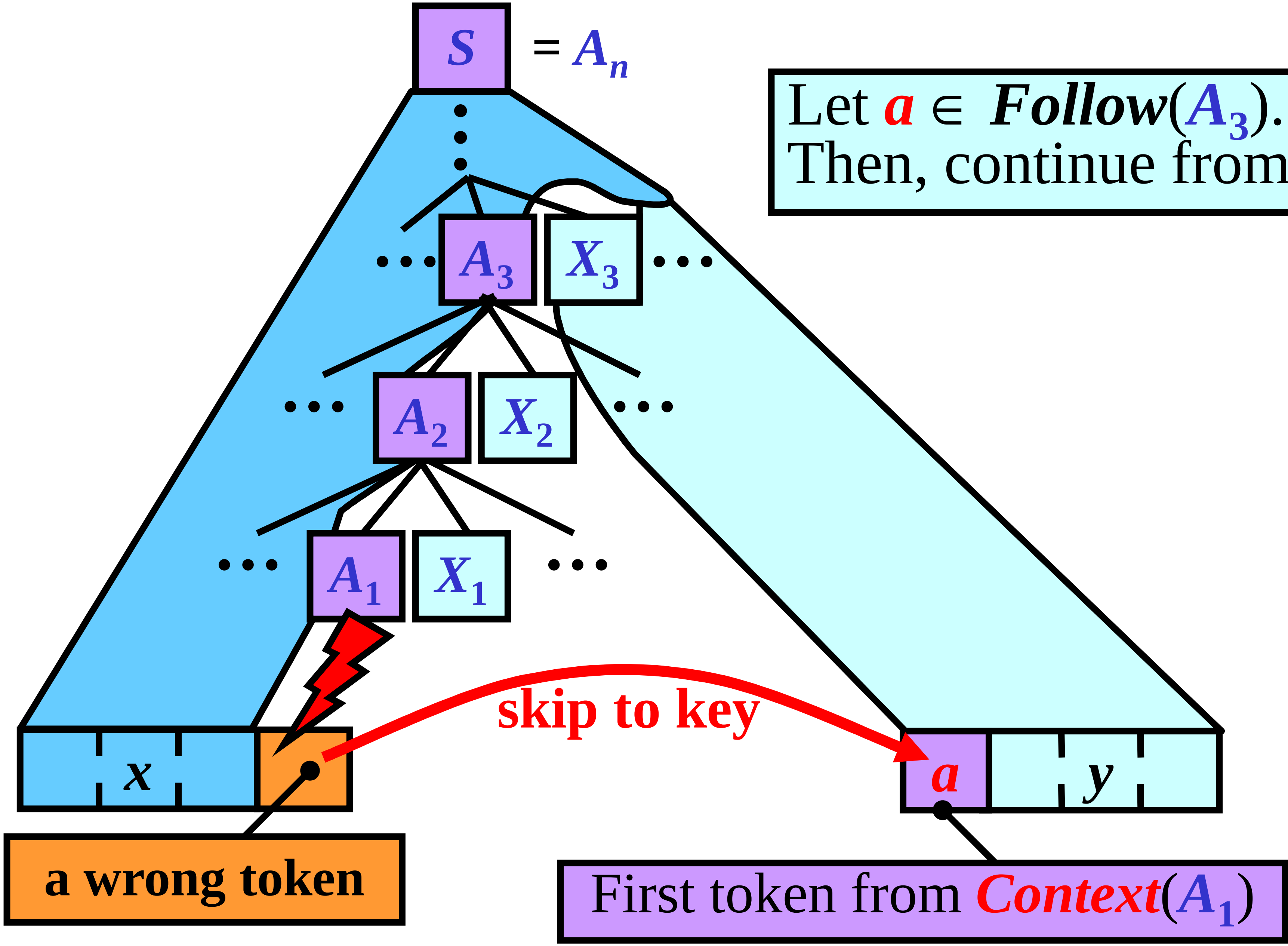


Panic-Mode Recovery: Illustration 2/2



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

Panic-Mode Recovery: Illustration 2/2



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

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

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

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

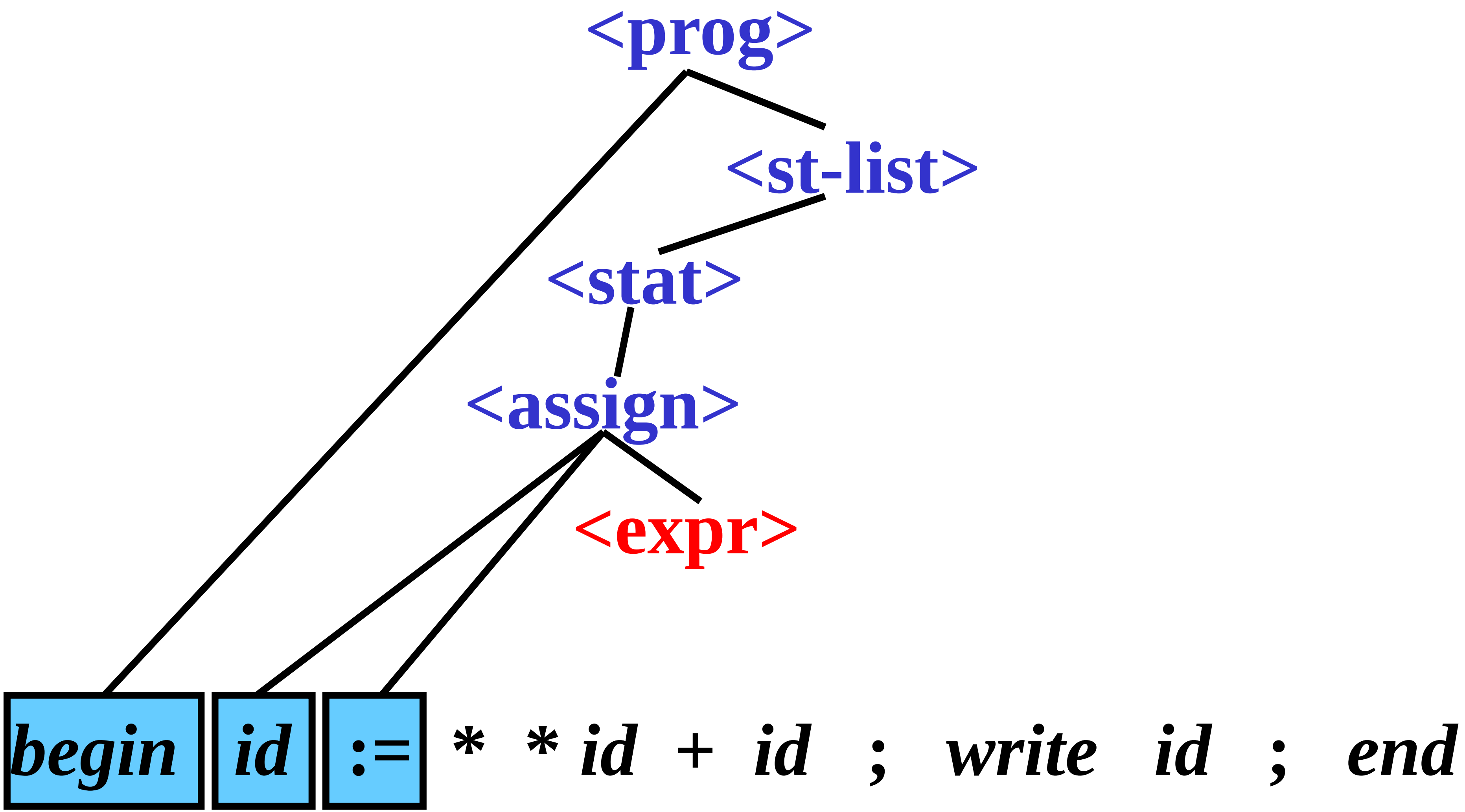
- **Method:**
- Let ***A*** be pushdown top & no rule is applicable:
- **repeat**

  a := GetNextToken;

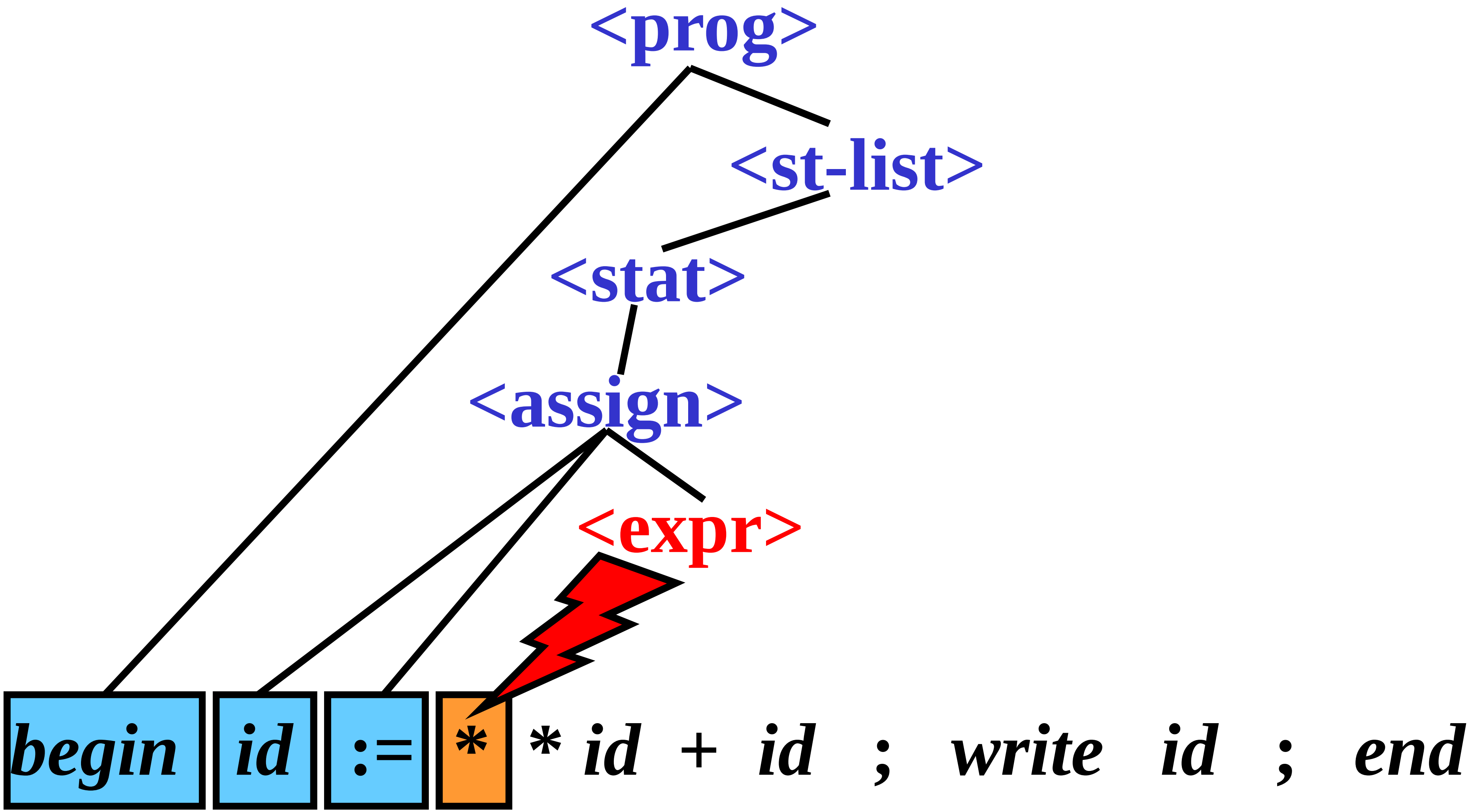
  {These tokens are skipped}

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

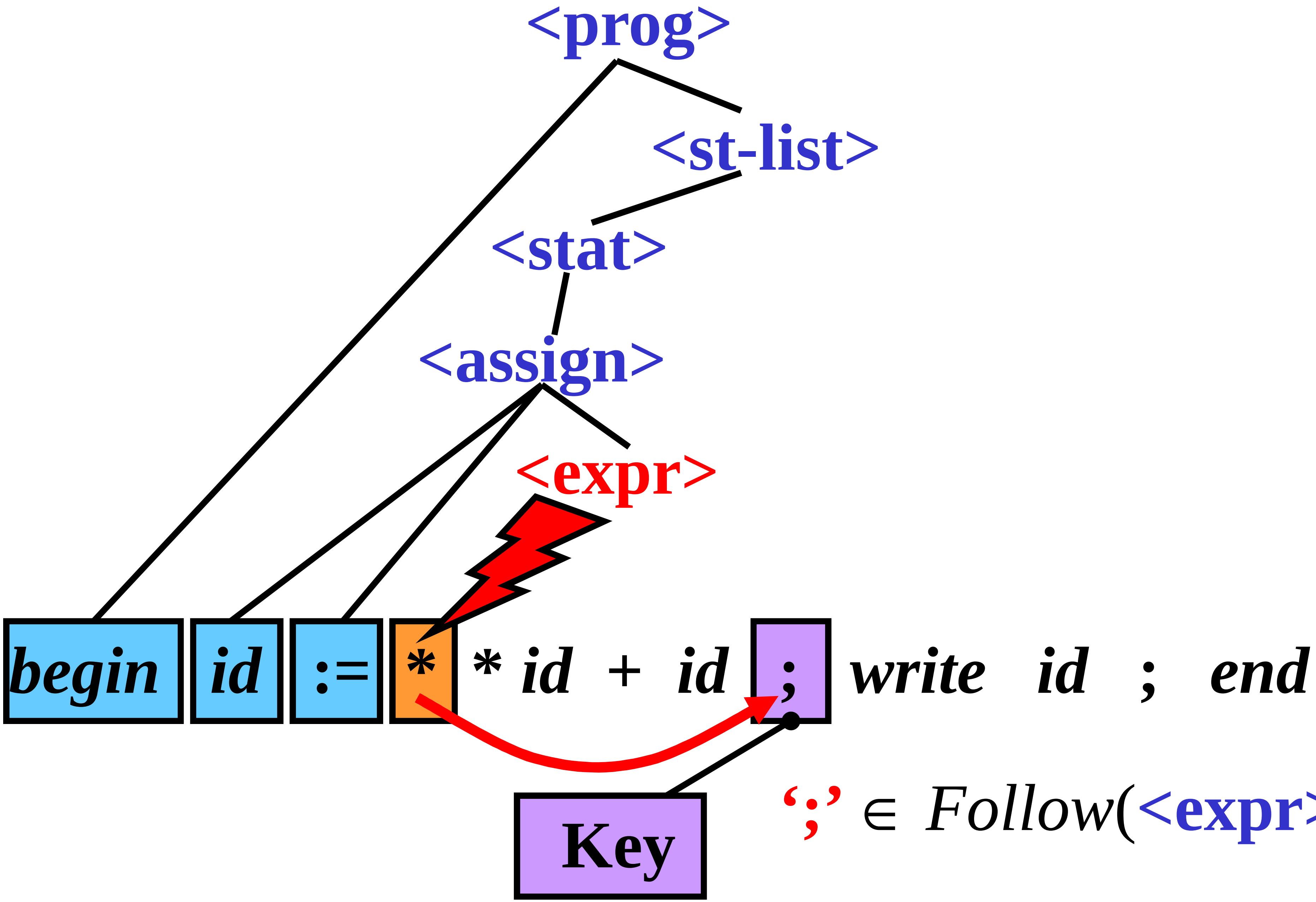
Variant I: Example



Variant I: Example

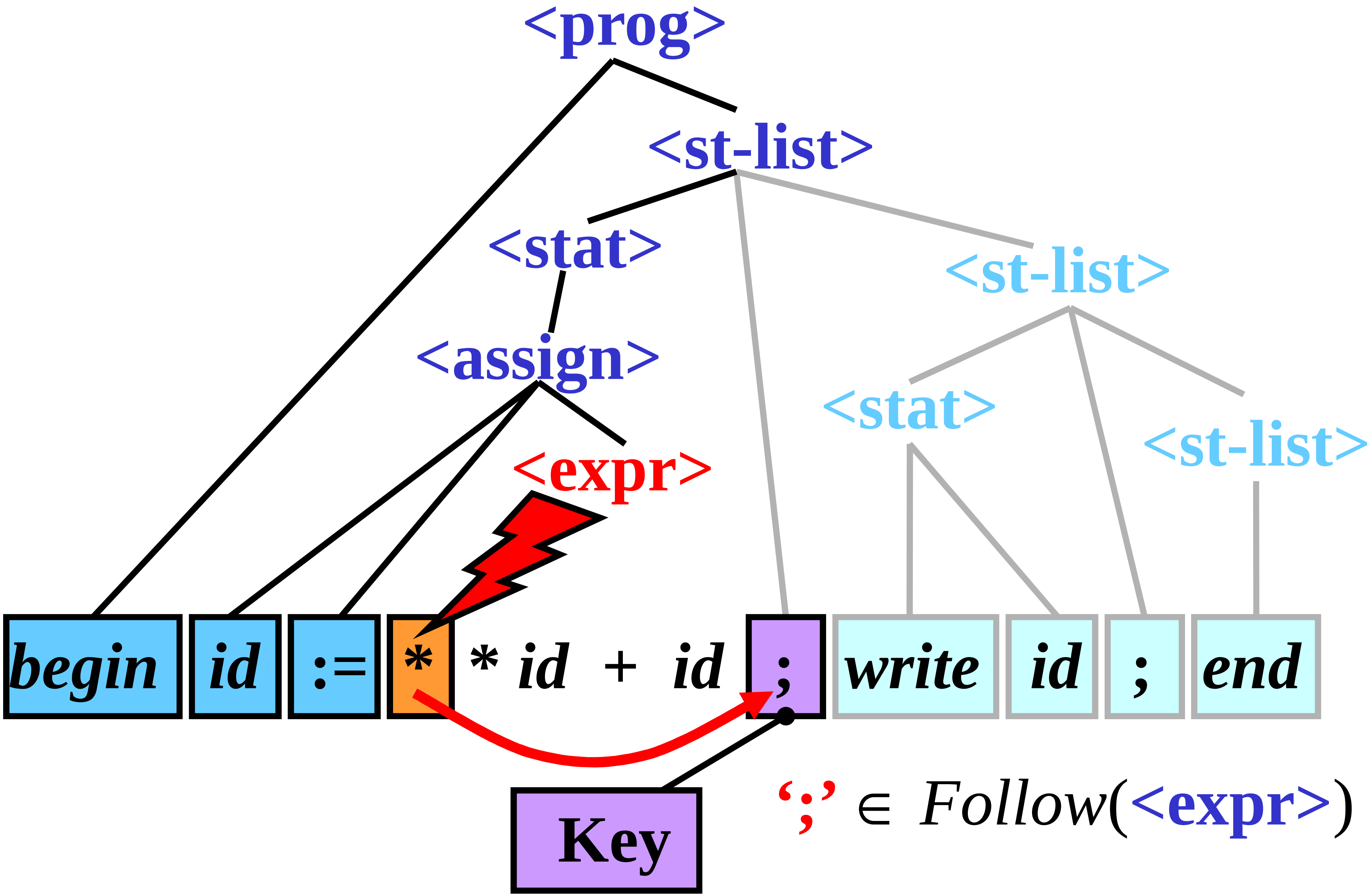


Variant I: Example



';' $\in$ *Follow*(**<expr>**)

Variant I: Example



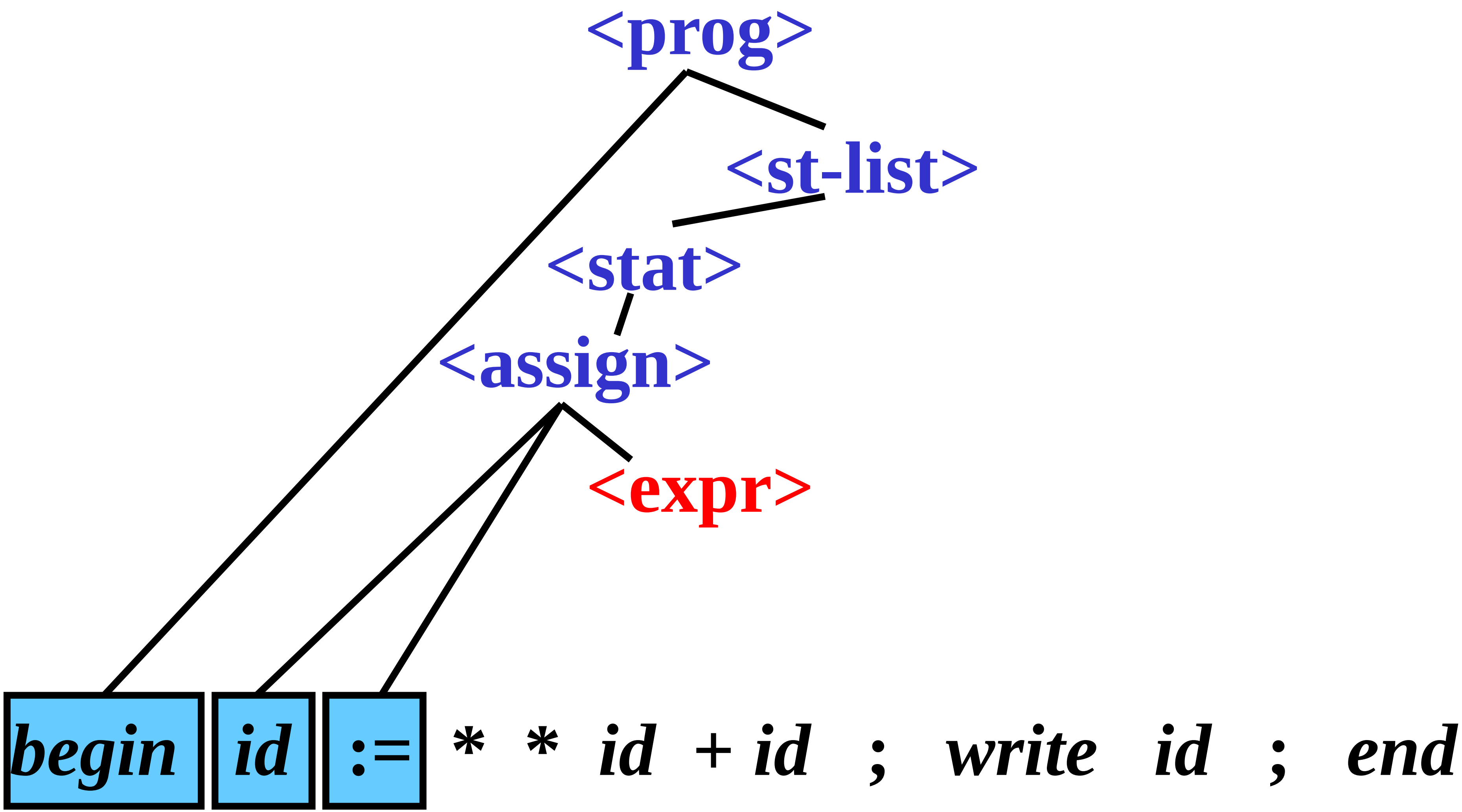
Context(*X*) for Predictive Parser: Variant II

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

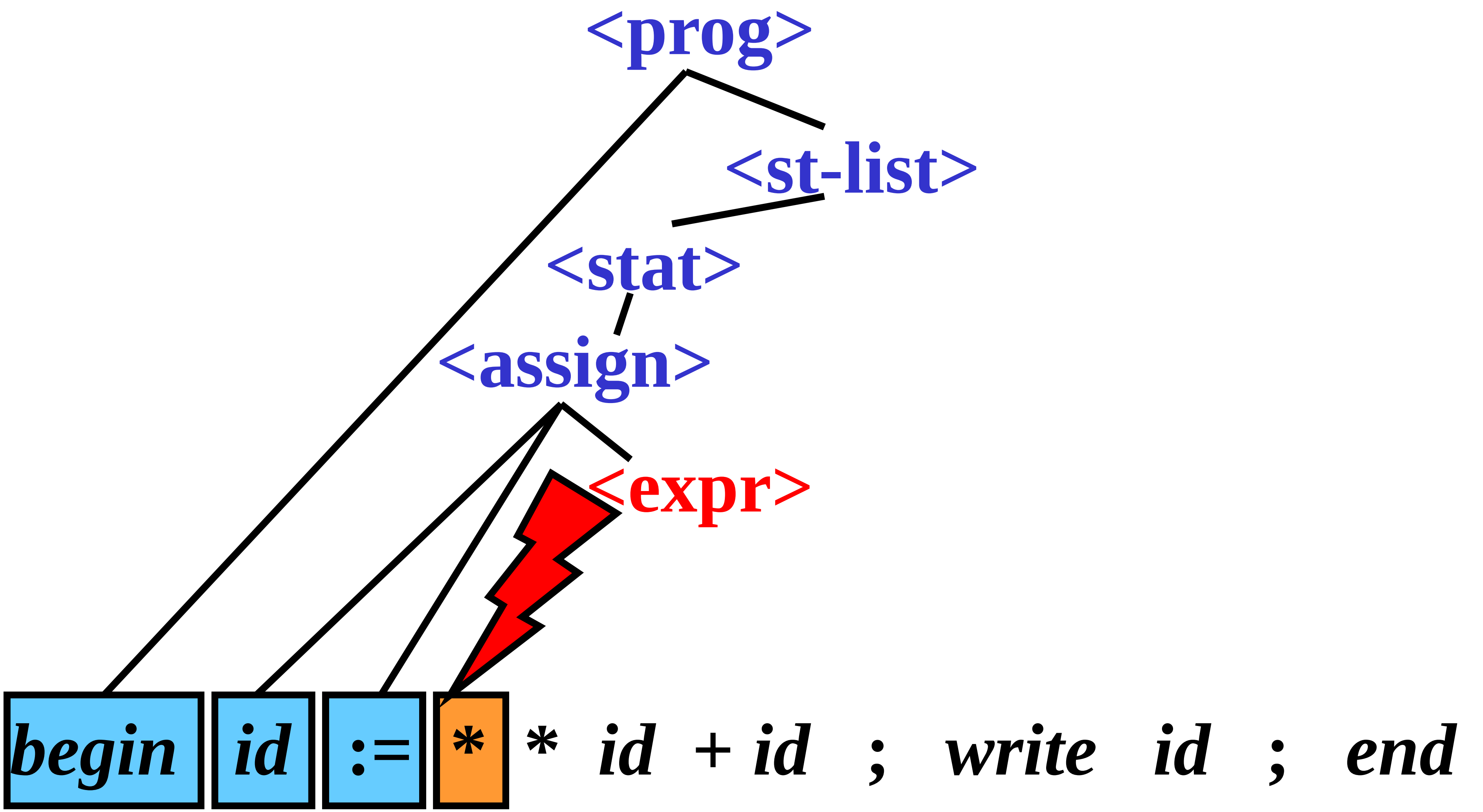
$Context(A) = First(A) \cup Follow(A)$ for every $A \in N$

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

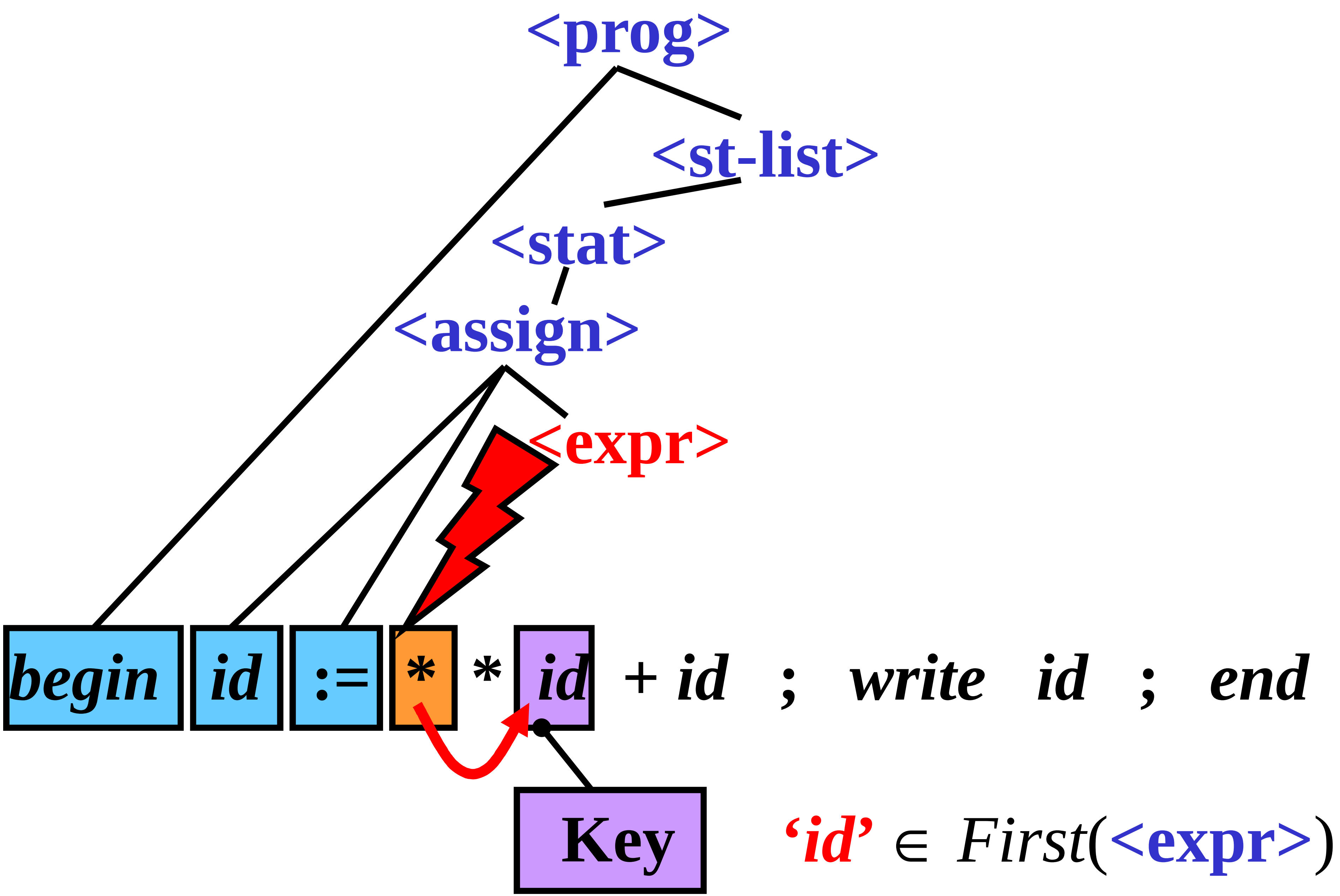
Variant II: Example



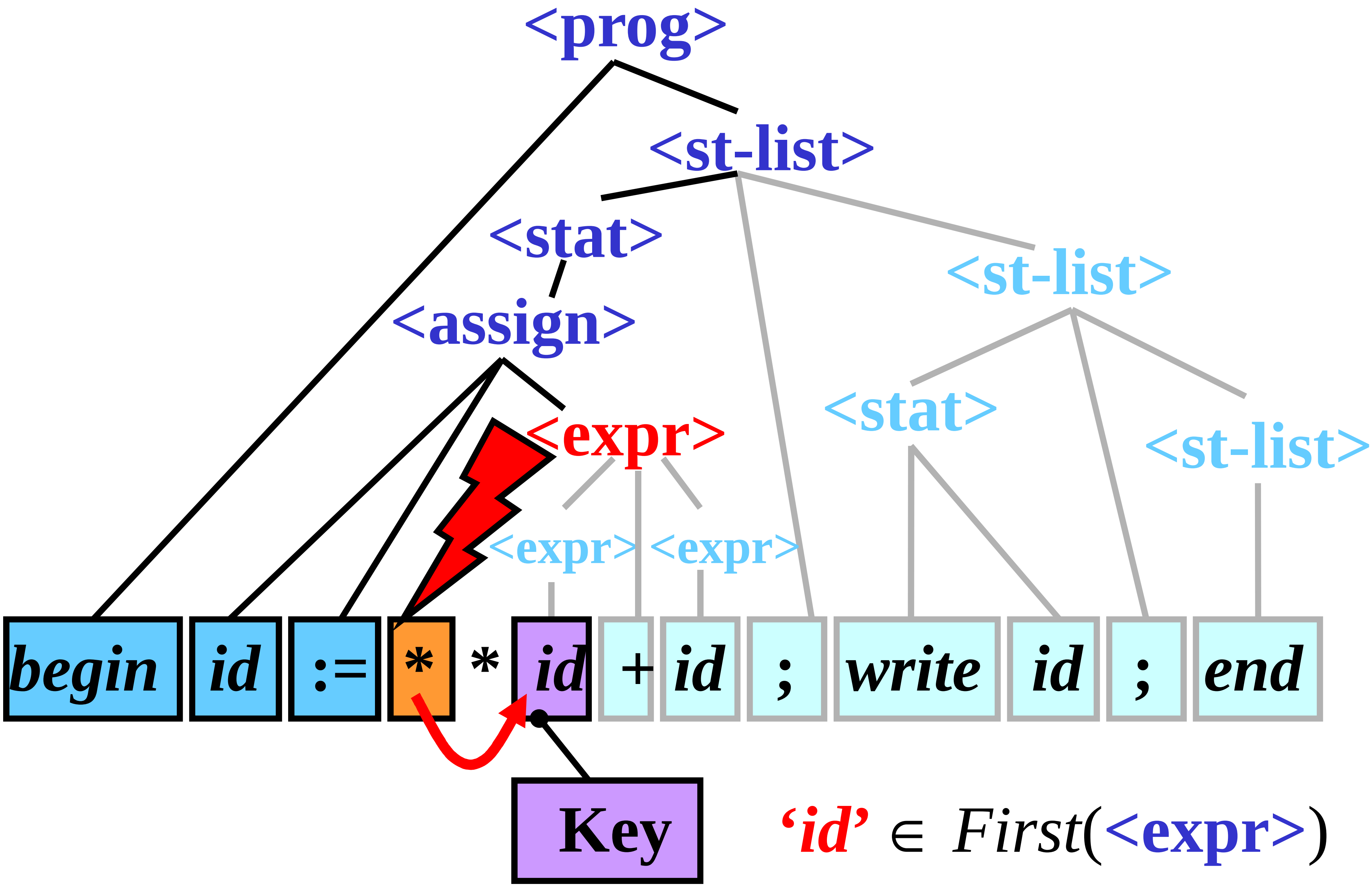
Variant II: Example



Variant II: Example



Variant II: Example



'id' $\in$ *First*(**<expr>**)