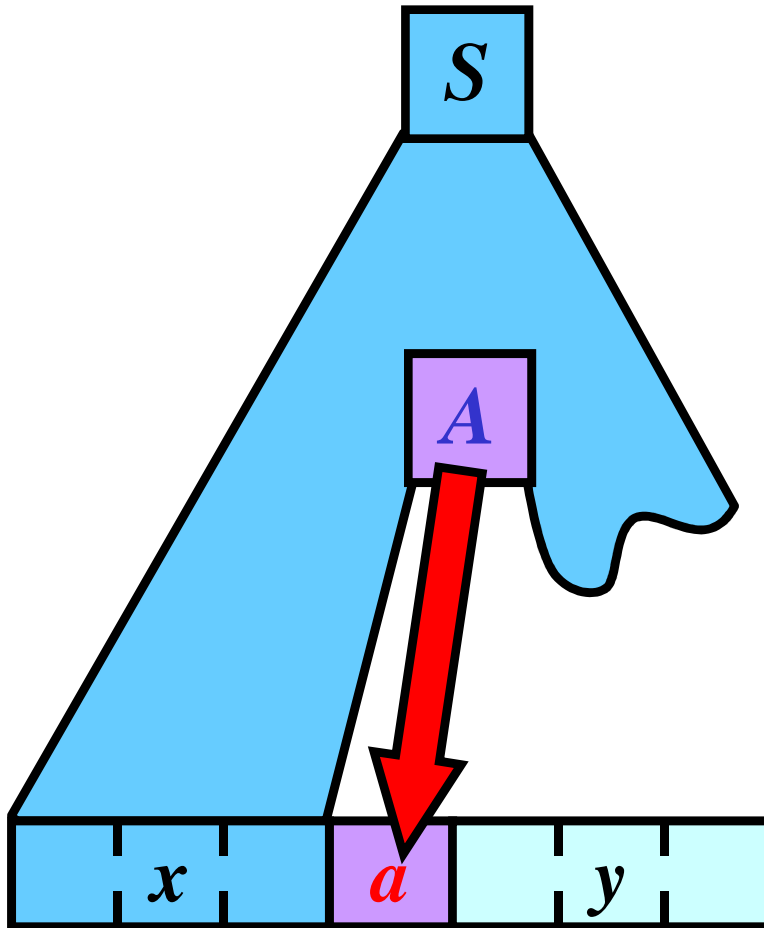


Part VII.

Top-Down Parsing

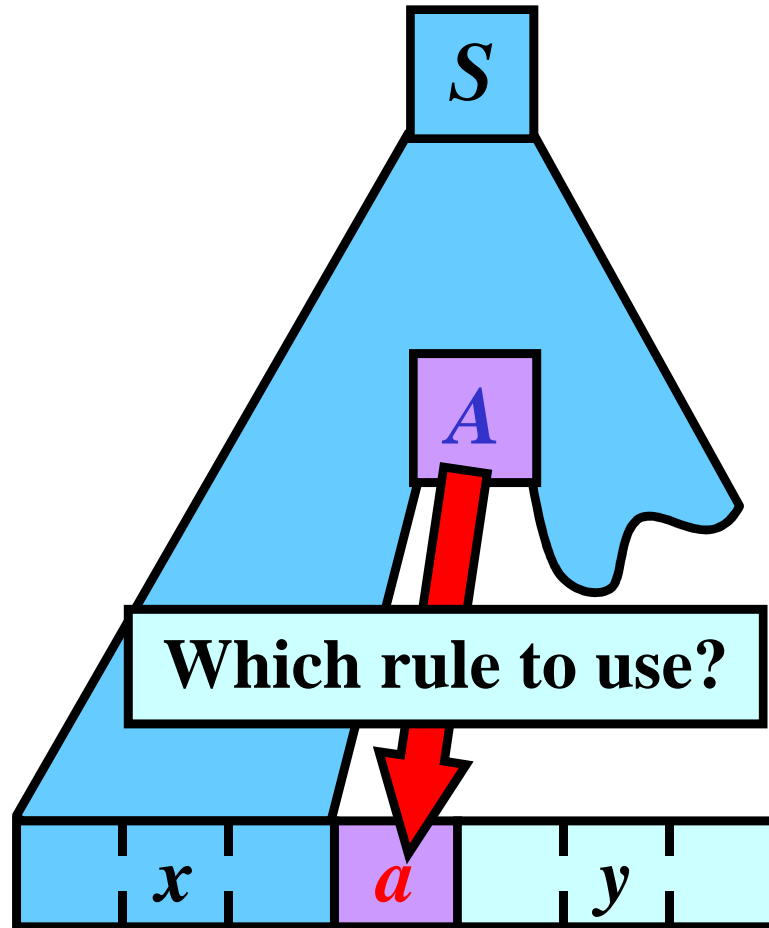
Top-Down Parsing: Introduction

Problem:



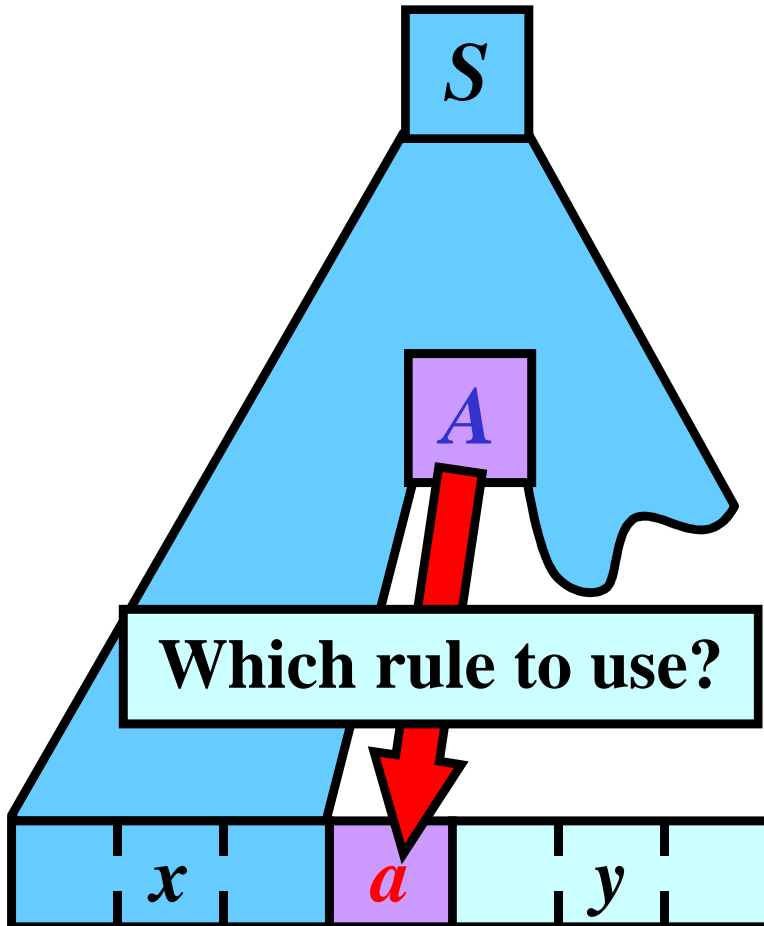
Top-Down Parsing: Introduction

Problem:



Top-Down Parsing: Introduction

Problem:



Basic idea:

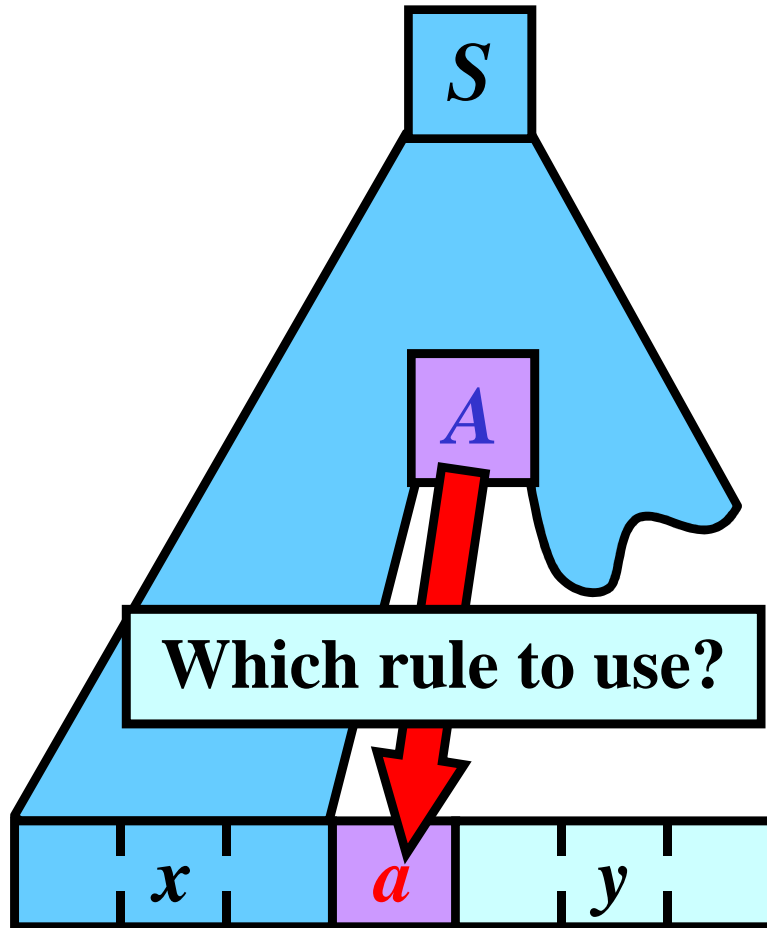
Table:

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

Use rule $r: A \rightarrow x$

Top-Down Parsing: Introduction

Problem:



Basic idea:

Table:

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

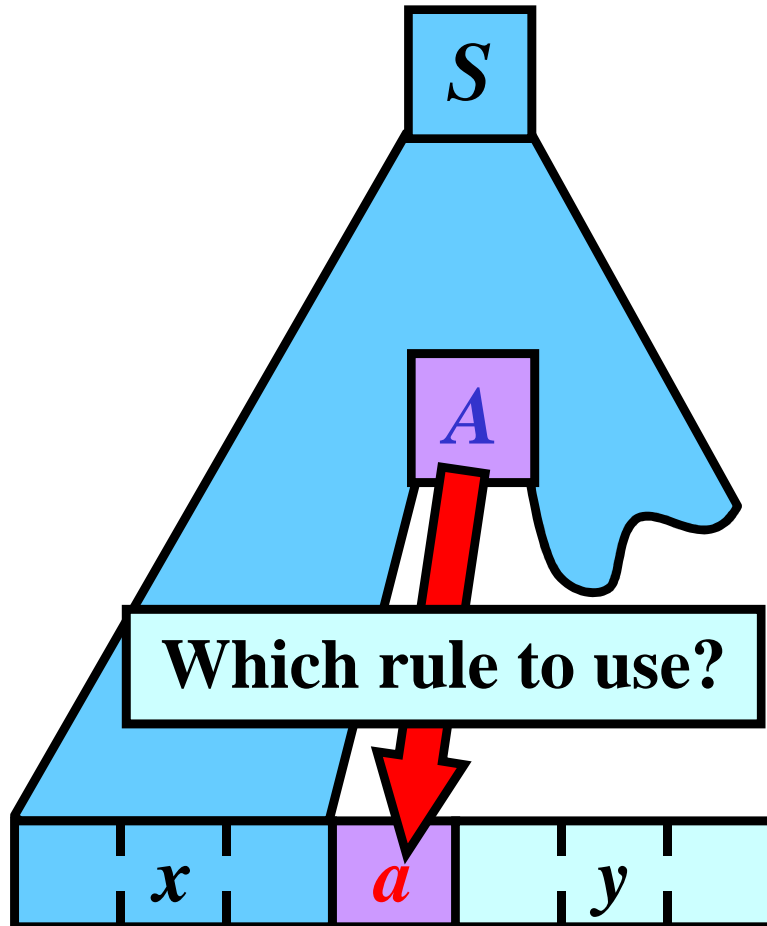


Use rule $r: A \rightarrow x$

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

Top-Down Parsing: Introduction

Problem:



Basic idea:

Table:

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

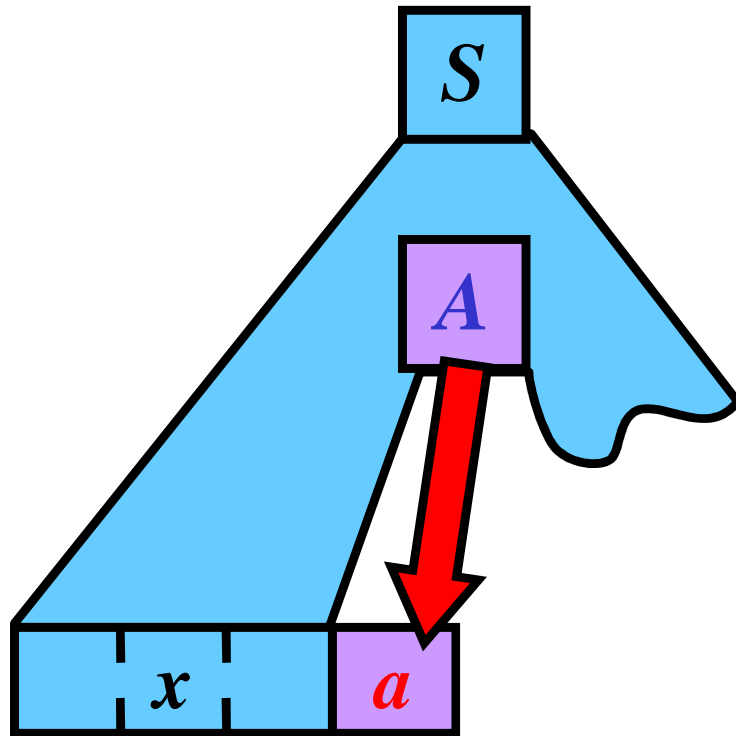


Use rule $r: A \rightarrow x$

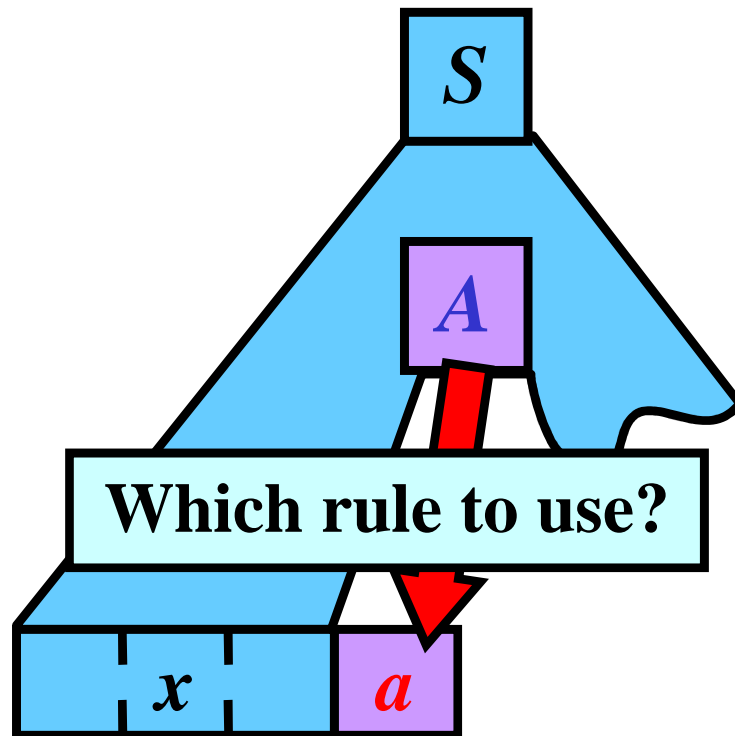
Question: Could you construct this table for **any** CFG?

Answer: **NO**

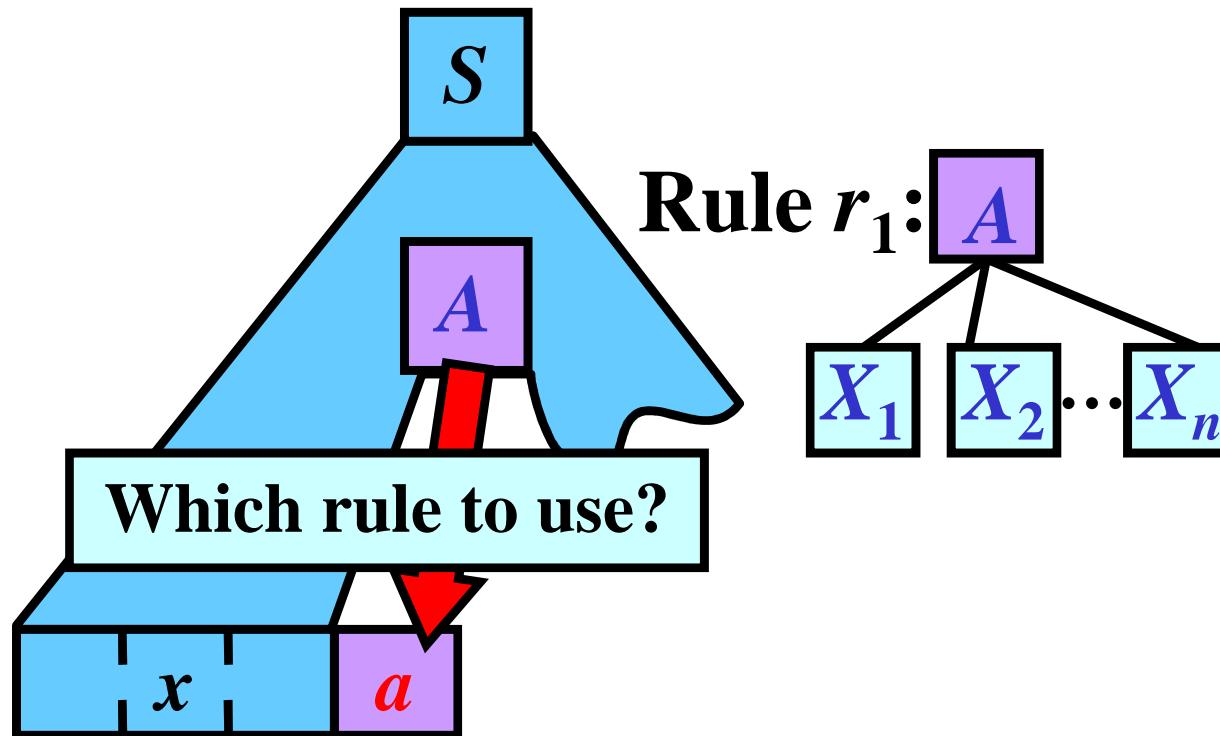
A Table-Based Selection of a Rule



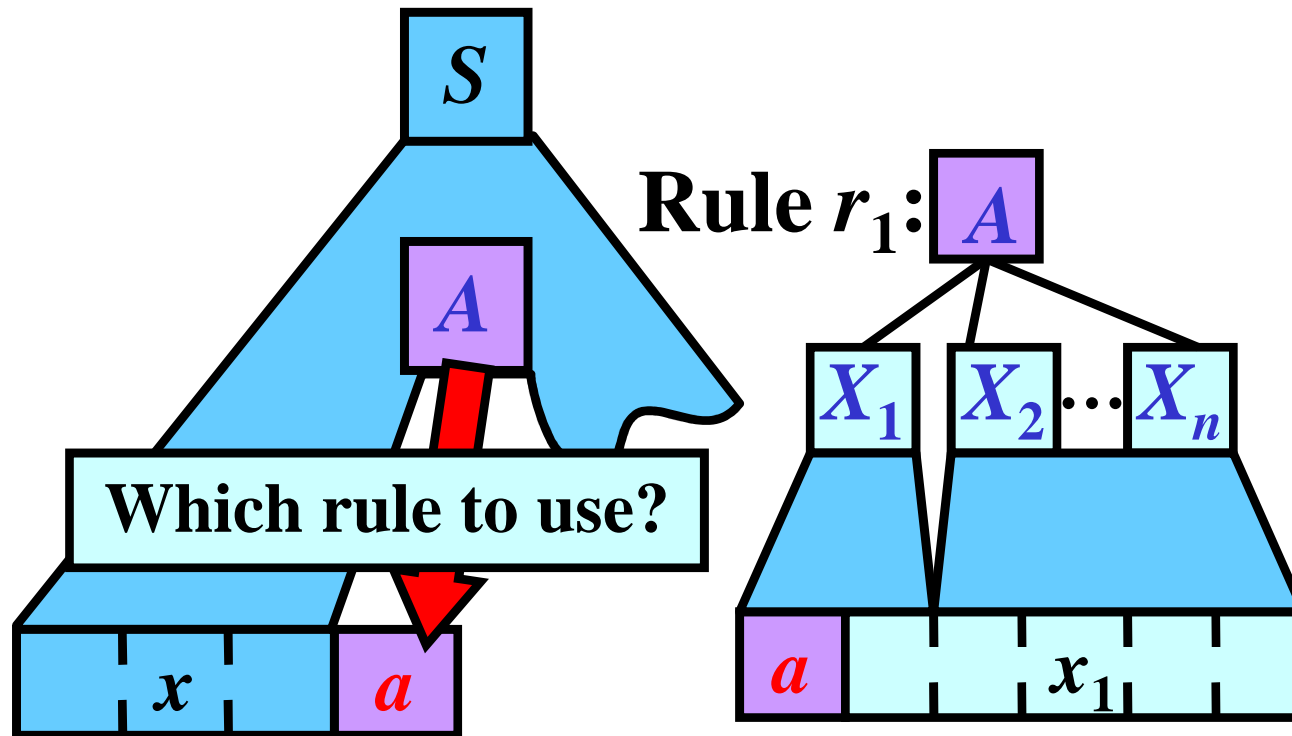
A Table-Based Selection of a Rule



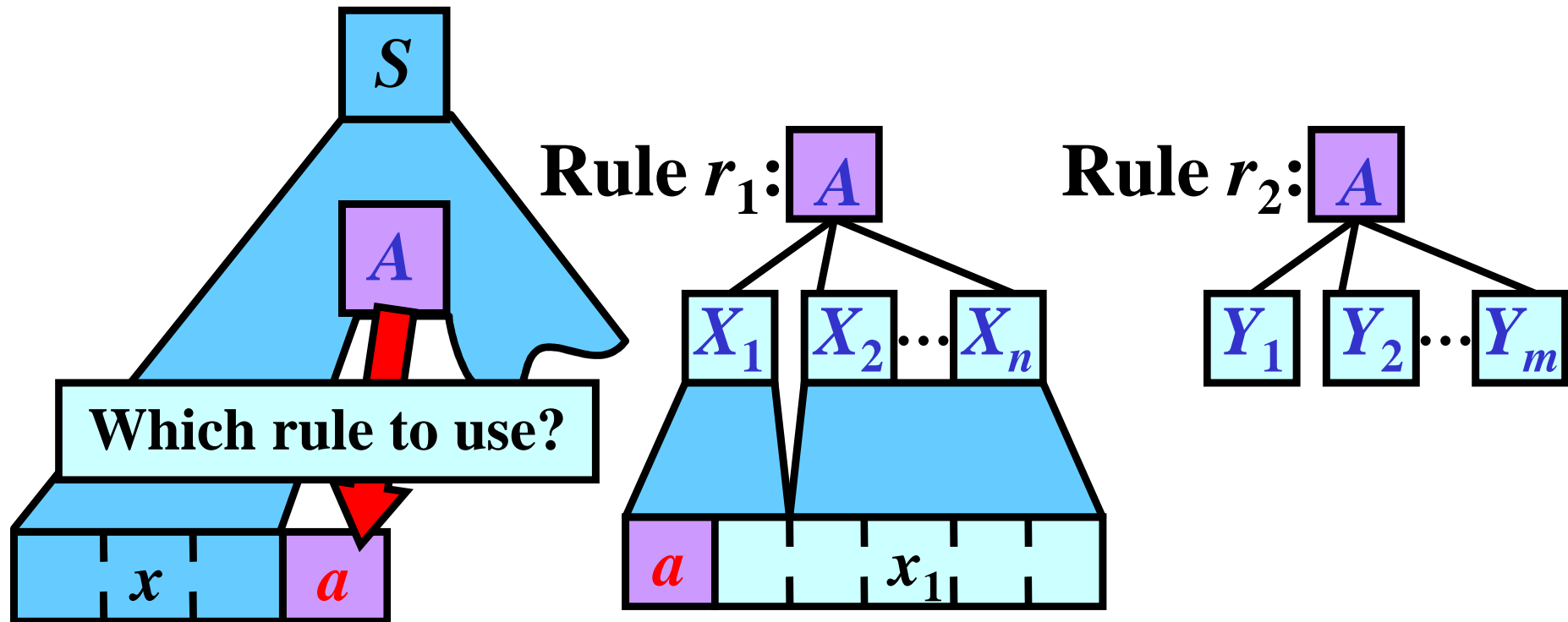
A Table-Based Selection of a Rule



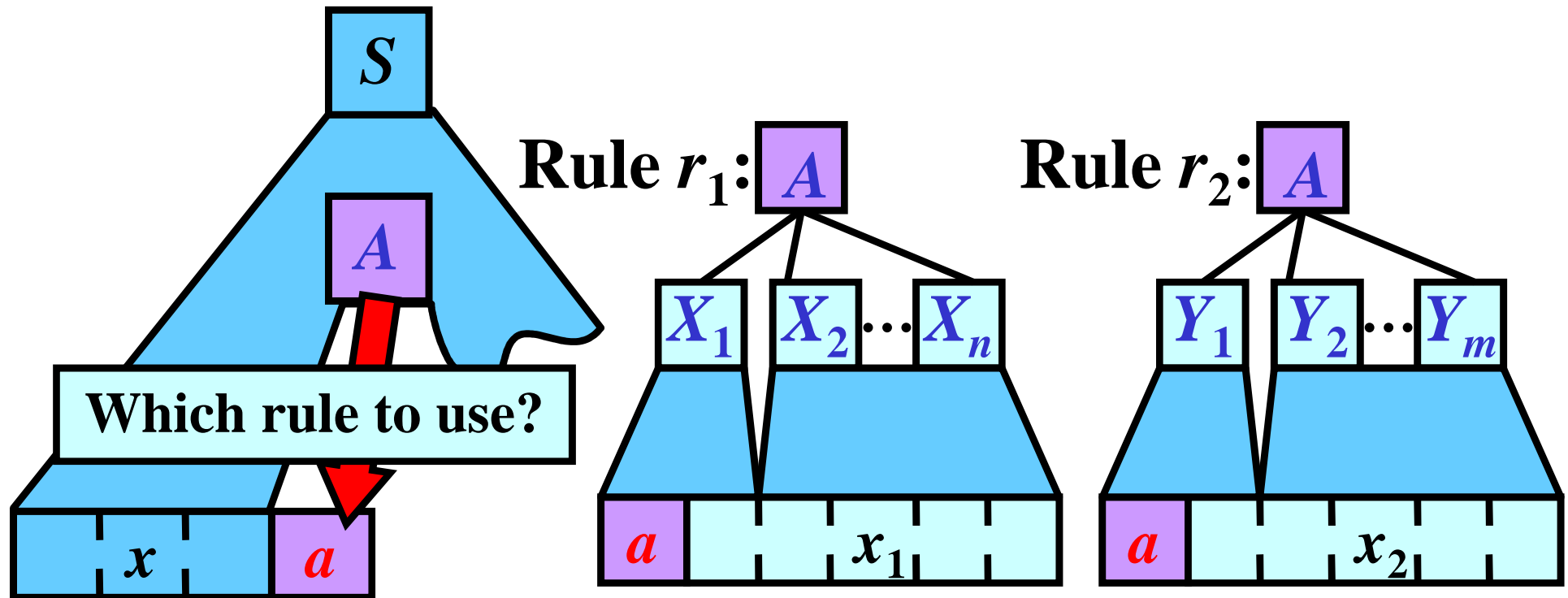
A Table-Based Selection of a Rule



A Table-Based Selection of a Rule



A Table-Based Selection of a Rule



A Table-Based Selection of a Rule

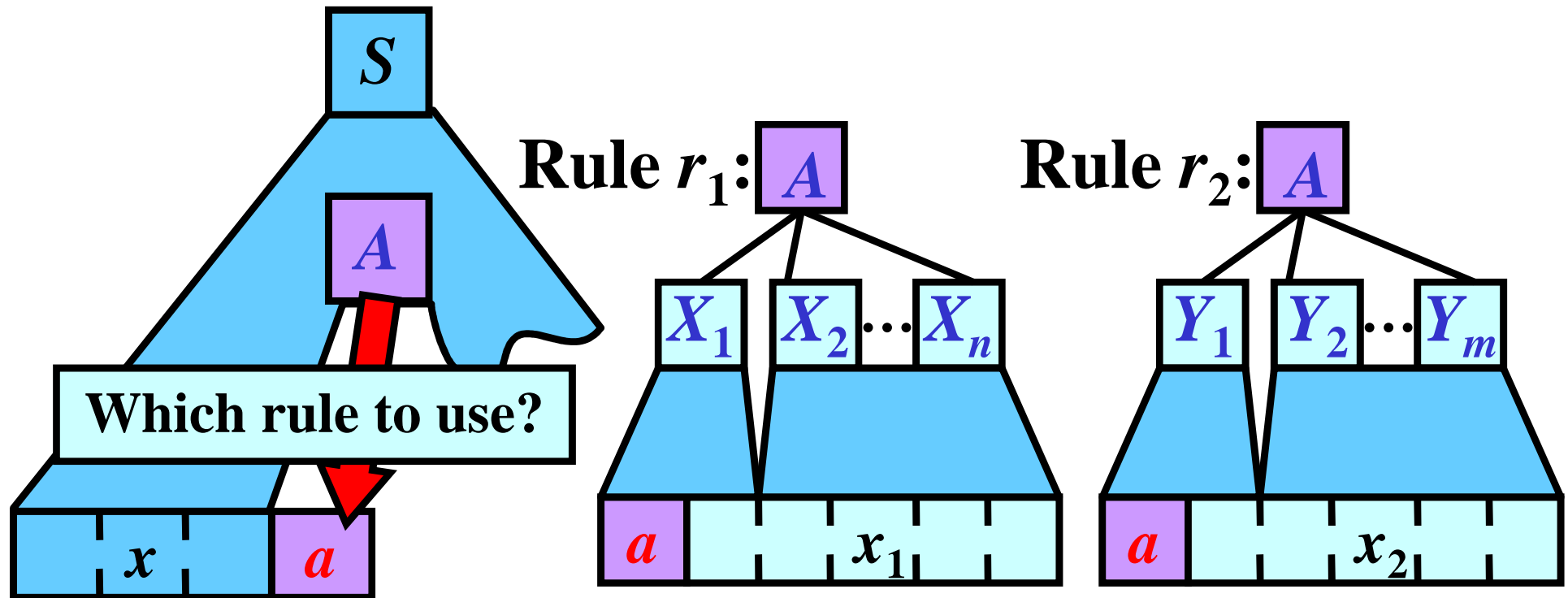


Table:

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

A Table-Based Selection of a Rule

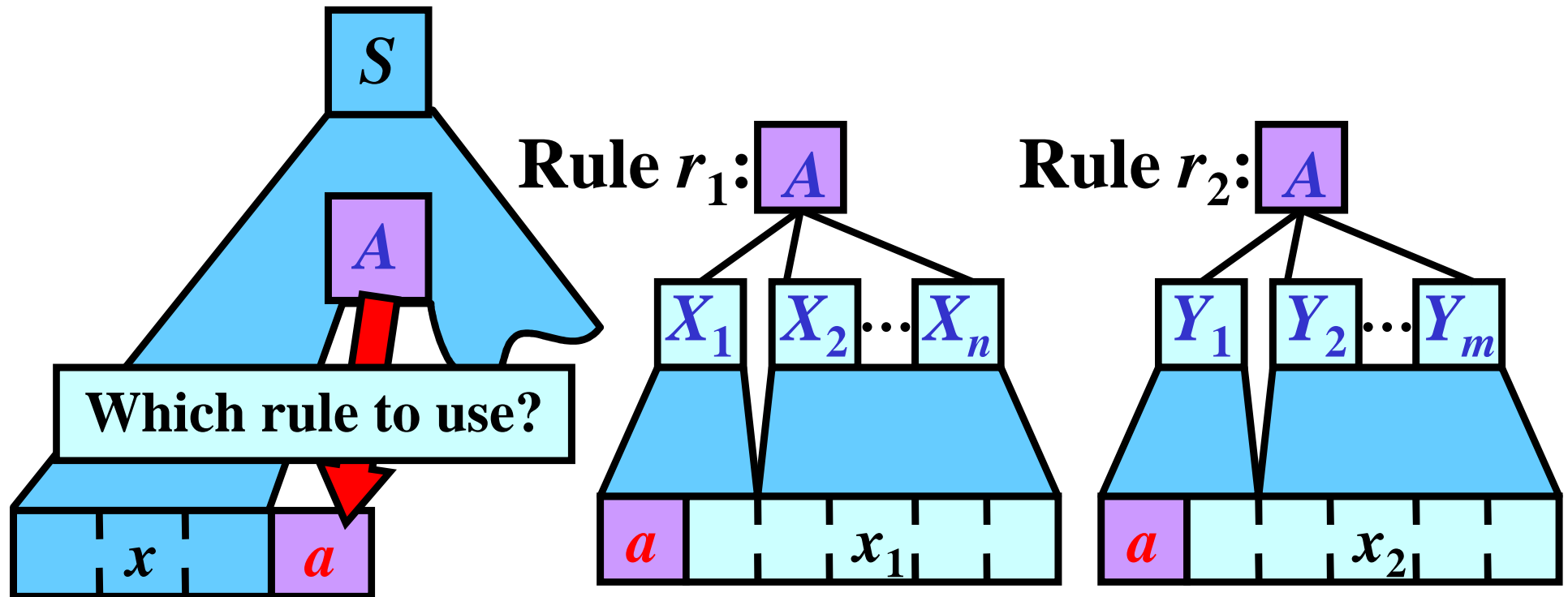


Table:

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

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

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

Set *First*

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

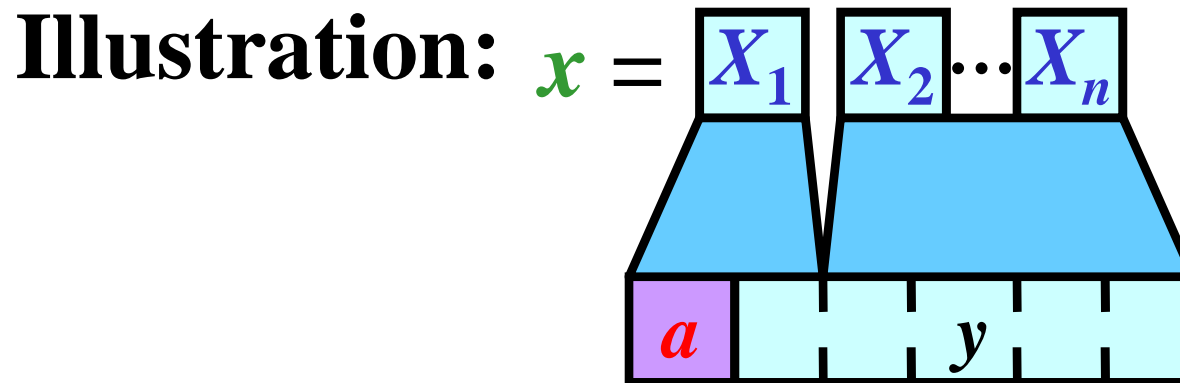
Definition: Let $G = (N, T, P, S)$ be a CFG. For every $x \in (N \cup T)^*$, we define the set *First*(x) as $First(x) = \{a: a \in T, x \Rightarrow^* ay; y \in (N \cup T)^*\}$.

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

Set *First*

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

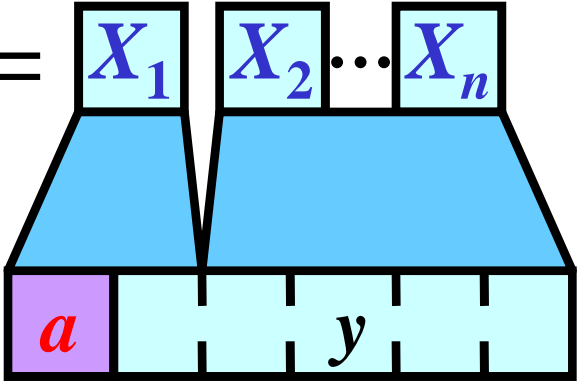
Definition: Let $G = (N, T, P, S)$ be a CFG. For every $x \in (N \cup T)^*$, we define the set *First*(x) as $First(x) = \{a: a \in T, x \Rightarrow^* ay; y \in (N \cup T)^*\}$.



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

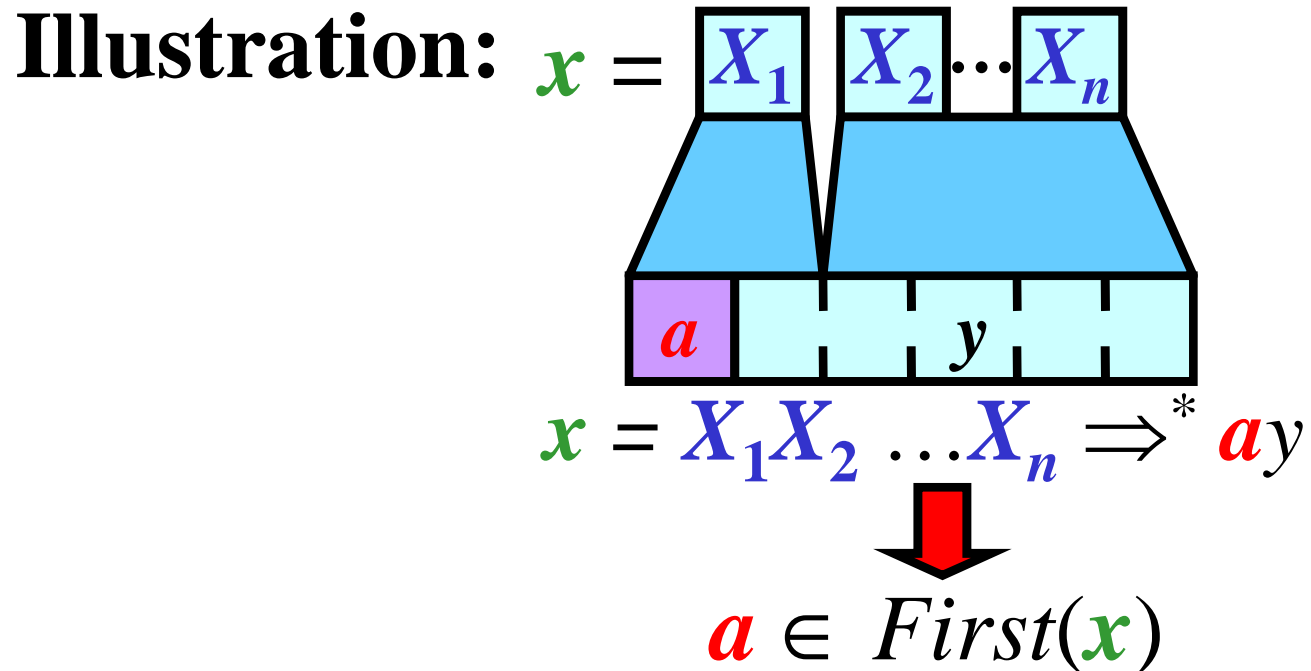
Illustration: $x =$ 

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

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



LL Grammars without ϵ -rules

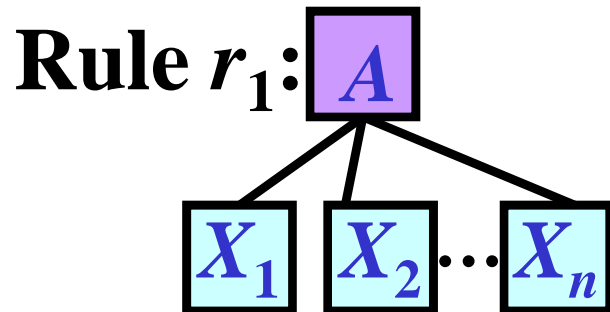
Definition: Let $G = (N, T, P, S)$ be a CFG without ϵ -rules. G is an *LL grammar* if for every $a \in T$ and every $A \in N$ there is **no more than one** rule $A \rightarrow X_1X_2...X_n \in P$ such that $a \in First(X_1X_2...X_n)$

Illustration:

LL Grammars without ϵ -rules

Definition: Let $G = (N, T, P, S)$ be a CFG 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 $a \in First(X_1X_2...X_n)$

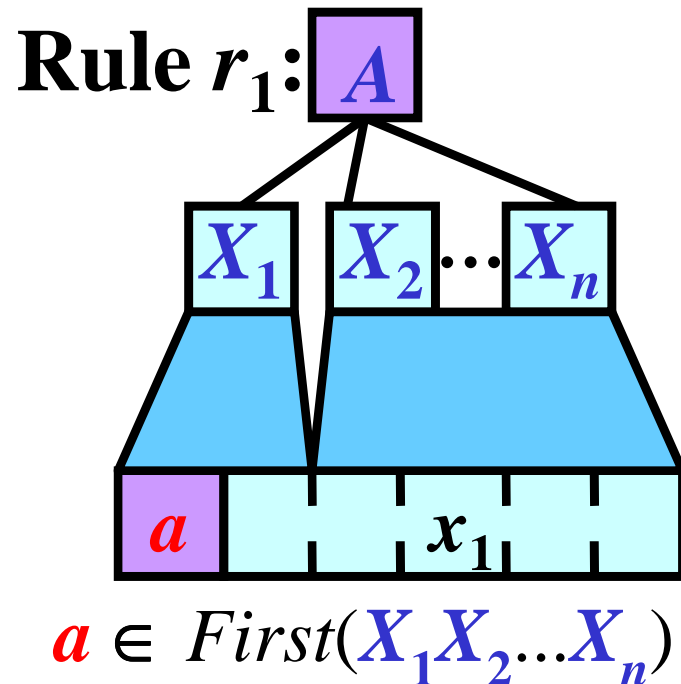
Illustration:



LL Grammars without ϵ -rules

Definition: Let $G = (N, T, P, S)$ be a CFG 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 $a \in First(X_1X_2...X_n)$

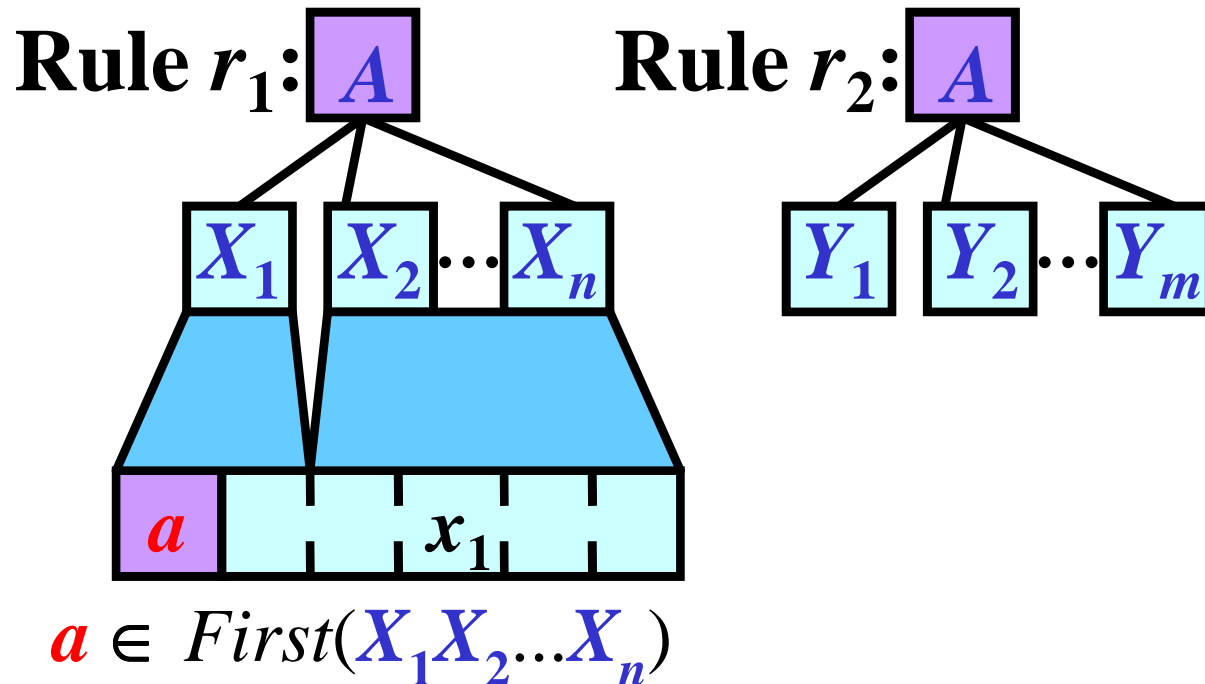
Illustration:



LL Grammars without ϵ -rules

Definition: Let $G = (N, T, P, S)$ be a CFG 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 $a \in First(X_1X_2...X_n)$

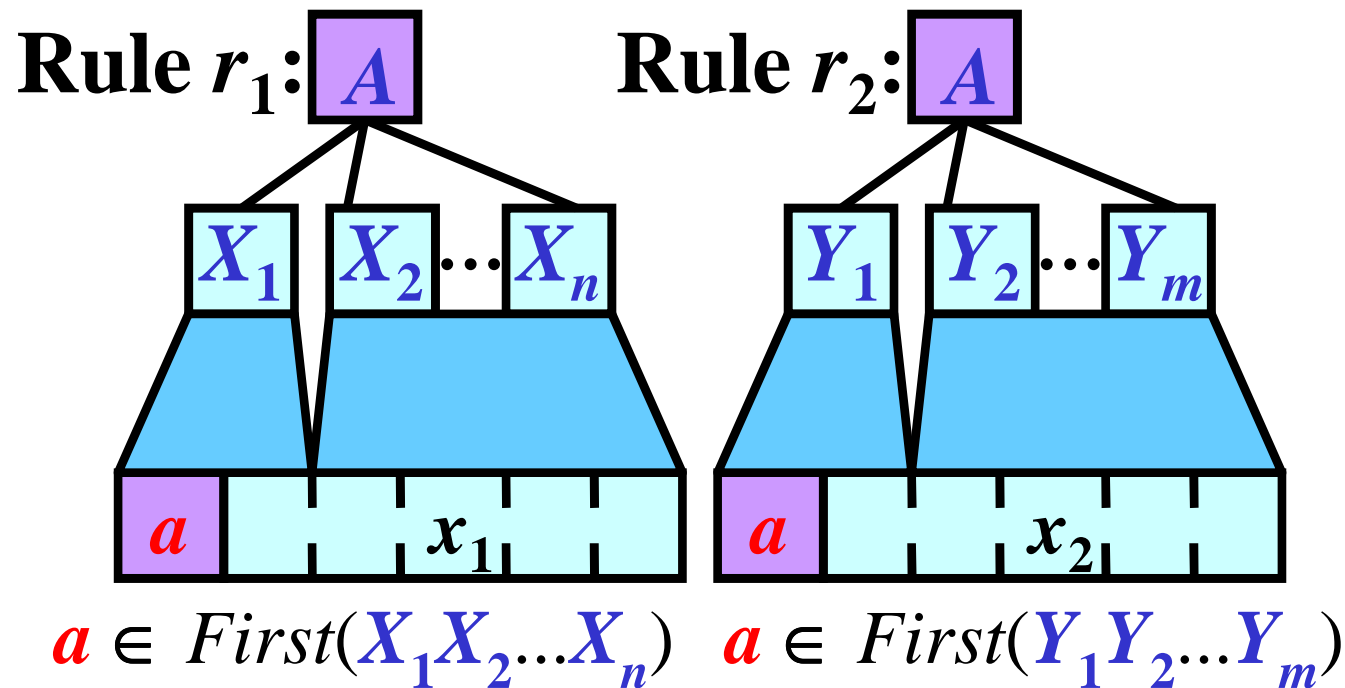
Illustration:



LL Grammars without ϵ -rules

Definition: Let $G = (N, T, P, S)$ be a CFG 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 $a \in First(X_1X_2...X_n)$

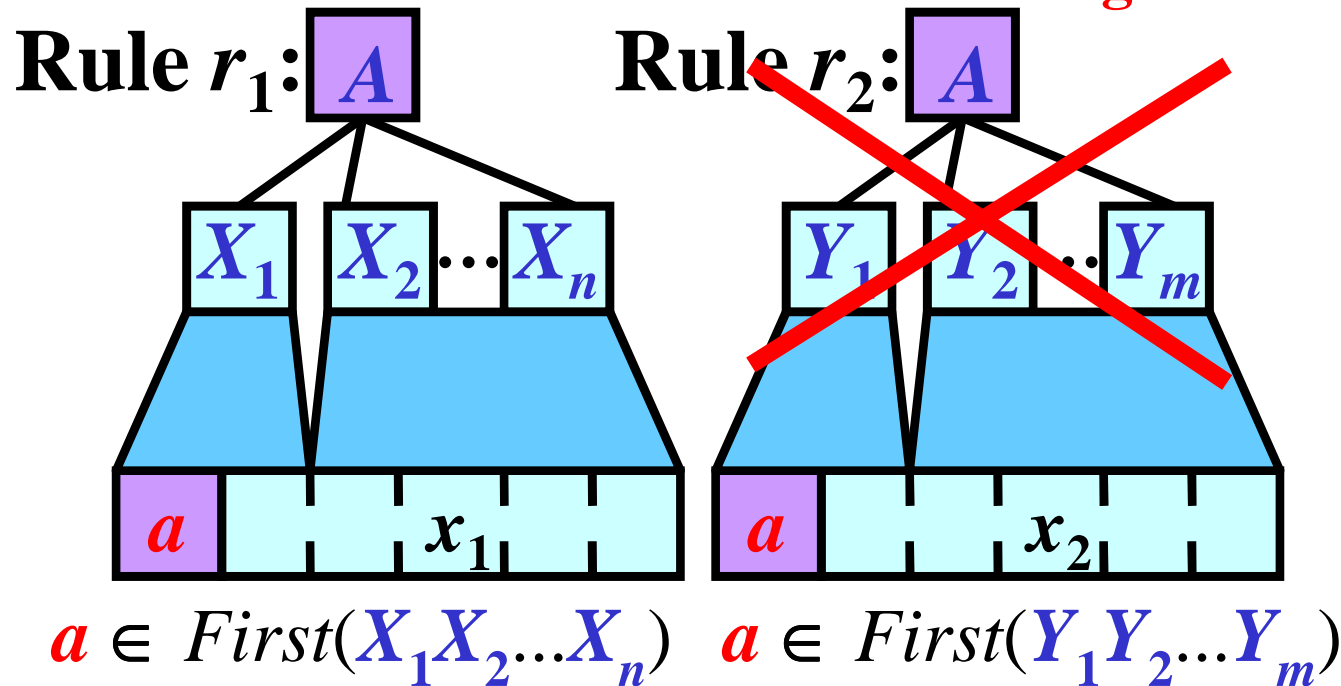
Illustration:



LL Grammars without ϵ -rules

Definition: Let $G = (N, T, P, S)$ be a CFG 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 $a \in First(X_1X_2...X_n)$

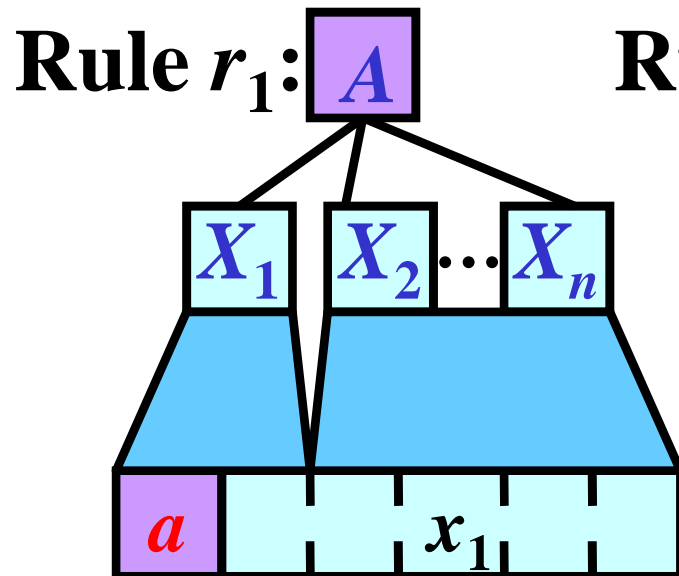
Illustration: **Ruled out in an LL grammar**



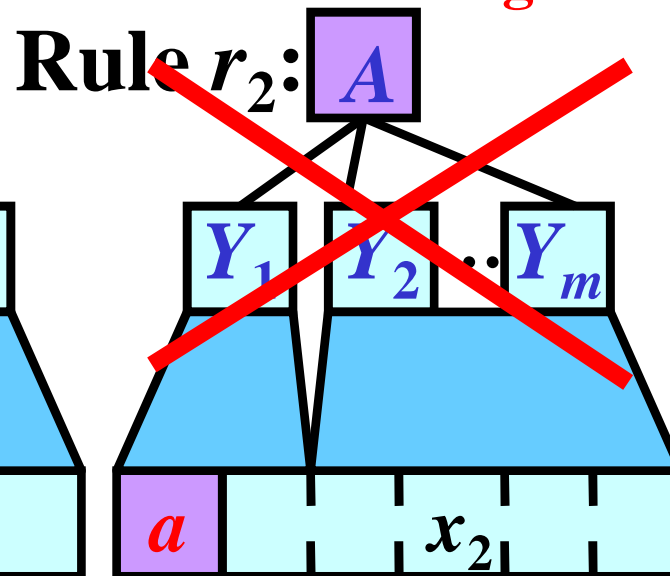
LL Grammars without ϵ -rules

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: **Ruled out in an LL grammar** **Table:**



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



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

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

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

Simple *Programming Language* (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
```

∈ SPL

Algorithm: *First*(X)

- **Input:** $G = (N, T, P, S)$ without ϵ -rules
 - **Output:** $First(X)$ for every $X \in N \cup T$
-
- **Method:**
 - for each $a \in T$: $First(a) := \{a\}$
 - Apply the following rule until no *First* set can be changed:
 - if $A \rightarrow X_1 X_2 \dots X_n \in P$, 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 $A \rightarrow X_1 X_2 \dots X_n \in P$, then add $First(X_1)$ to $First(A)$
-

Illustration:

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

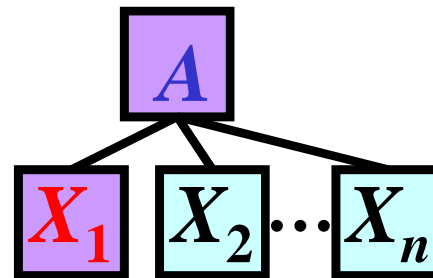
Algorithm: *First*(X)

- **Input:** $G = (N, T, P, S)$ without ϵ -rules
 - **Output:** $First(X)$ for every $X \in N \cup T$
-
- **Method:**
 - for each $a \in T$: $First(a) := \{a\}$
 - Apply the following rule until no *First* set can be changed:
 - if $A \rightarrow X_1 X_2 \dots X_n \in P$, then add $First(X_1)$ to $First(A)$
-

Illustration:

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

2)

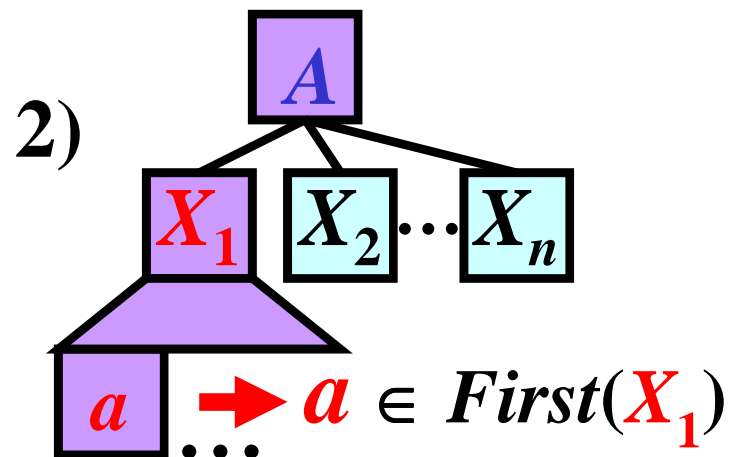


Algorithm: *First*(X)

- **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 $A \rightarrow X_1 X_2 \dots X_n \in P$, then add $First(X_1)$ to $First(A)$
-

Illustration:

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

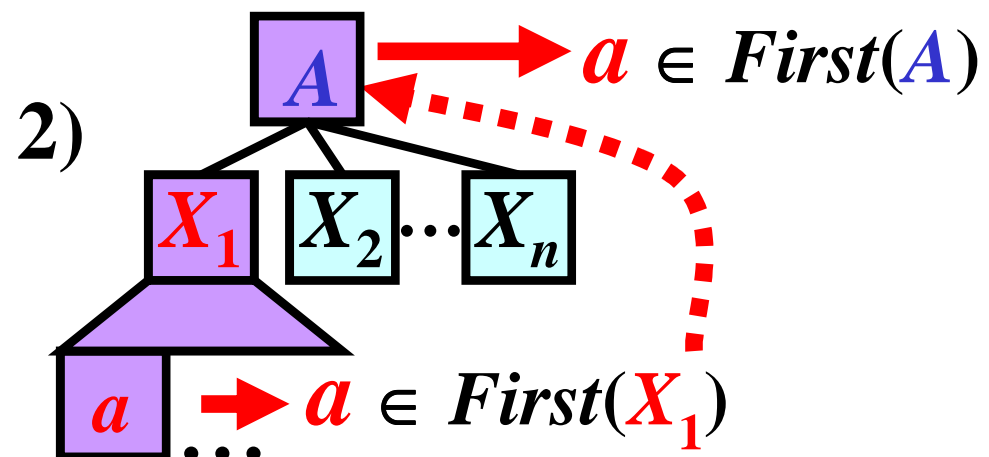


Algorithm: *First*(X)

- **Input:** $G = (N, T, P, S)$ without ϵ -rules
 - **Output:** $First(X)$ for every $X \in N \cup T$
-
- **Method:**
 - for each $a \in T$: $First(a) := \{a\}$
 - Apply the following rule until no *First* set can be changed:
 - if $A \rightarrow X_1 X_2 \dots X_n \in P$, then add $First(X_1)$ to $First(A)$
-

Illustration:

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



First(X) for SPL: Example

$First(\underline{\text{begin}}) := \{\underline{\text{begin}}\}$	$First(\underline{\text{id}}) := \{\underline{\text{id}}\}$	$First(\underline{,}) := \{\underline{,}\}$
$First(\underline{\text{end}}) := \{\underline{\text{end}}\}$	$First(\underline{\text{int}}) := \{\underline{\text{int}}\}$	$First(\underline{(}) := \{\underline{(}\}$
$First(\underline{\text{read}}) := \{\underline{\text{read}}\}$	$First(\underline{:=}) := \{\underline{:=}\}$	$First(\underline{)}) := \{\underline{)}\}$
$First(\underline{\text{write}}) := \{\underline{\text{write}}\}$	$First(\underline{\text{add}}) := \{\underline{\text{add}}\}$	$First(\underline{;}) := \{\underline{;}\}$

First(X) for SPL: Example

$First(\underline{\text{begin}}) := \{\underline{\text{begin}}\}$	$First(\underline{\text{id}}) := \{\underline{\text{id}}\}$	$First(\underline{,}) := \{\underline{,}\}$
$First(\underline{\text{end}}) := \{\underline{\text{end}}\}$	$First(\underline{\text{int}}) := \{\underline{\text{int}}\}$	$First(\underline{(}) := \{\underline{(}\}$
$First(\underline{\text{read}}) := \{\underline{\text{read}}\}$	$First(\underline{:=}) := \{\underline{:=}\}$	$First(\underline{)}) := \{\underline{)}\}$
$First(\underline{\text{write}}) := \{\underline{\text{write}}\}$	$First(\underline{\text{add}}) := \{\underline{\text{add}}\}$	$First(\underline{;}) := \{\underline{;}\}$

$\langle \text{item} \rangle \rightarrow \underline{\text{id}} \in P:$	add $First(\underline{\text{id}})$	to $First(\langle \text{item} \rangle)$
$\langle \text{item} \rangle \rightarrow \underline{\text{int}} \in P:$	add $First(\underline{\text{int}})$	to $First(\langle \text{item} \rangle)$

Summary: $First(\langle \text{item} \rangle) = \{\underline{\text{id}}, \underline{\text{int}}\}$

First(X) for SPL: Example

$First(\underline{\text{begin}}) := \{\underline{\text{begin}}\}$	$First(\underline{\text{id}}) := \{\underline{\text{id}}\}$	$First(\underline{,}) := \{\underline{,}\}$
$First(\underline{\text{end}}) := \{\underline{\text{end}}\}$	$First(\underline{\text{int}}) := \{\underline{\text{int}}\}$	$First(\underline{(}) := \{\underline{(}\}$
$First(\underline{\text{read}}) := \{\underline{\text{read}}\}$	$First(\underline{:=}) := \{\underline{:=}\}$	$First(\underline{)}) := \{\underline{)}\}$
$First(\underline{\text{write}}) := \{\underline{\text{write}}\}$	$First(\underline{\text{add}}) := \{\underline{\text{add}}\}$	$First(\underline{;}) := \{\underline{;}\}$

$\langle \text{item} \rangle \rightarrow \underline{\text{id}} \in P:$	add $First(\underline{\text{id}})$	to $First(\langle \text{item} \rangle)$
$\langle \text{item} \rangle \rightarrow \underline{\text{int}} \in P:$	add $First(\underline{\text{int}})$	to $First(\langle \text{item} \rangle)$
Summary: $First(\langle \text{item} \rangle) = \{\underline{\text{id}}, \underline{\text{int}}\}$		

$\langle \text{it-list} \rangle \rightarrow \underline{,} \in P:$	add $First(\underline{,})$	to $First(\langle \text{it-list} \rangle)$
$\langle \text{it-list} \rangle \rightarrow \underline{,} \dots \in P:$	add $First(\underline{,})$	to $First(\langle \text{it-list} \rangle)$
Summary: $First(\langle \text{it-list} \rangle) = \{\underline{,}, \underline{,}\}$		

First(X) for SPL: Example

$First(\underline{\text{begin}}) := \{\underline{\text{begin}}\}$	$First(\underline{\text{id}}) := \{\underline{\text{id}}\}$	$First(\underline{\text{,}}) := \{\underline{\text{,}}\}$
$First(\underline{\text{end}}) := \{\underline{\text{end}}\}$	$First(\underline{\text{int}}) := \{\underline{\text{int}}\}$	$First(\underline{\text{(}}) := \{\underline{\text{(}}\}$
$First(\underline{\text{read}}) := \{\underline{\text{read}}\}$	$First(\underline{\text{:=}}) := \{\underline{\text{:=}}\}$	$First(\underline{\text{)}}) := \{\underline{\text{)}}\}$
$First(\underline{\text{write}}) := \{\underline{\text{write}}\}$	$First(\underline{\text{add}}) := \{\underline{\text{add}}\}$	$First(\underline{\text{;}}) := \{\underline{\text{;}}\}$

$\langle \text{item} \rangle \rightarrow \underline{\text{id}} \in P:$	add $First(\underline{\text{id}})$	to $First(\langle \text{item} \rangle)$
$\langle \text{item} \rangle \rightarrow \underline{\text{int}} \in P:$	add $First(\underline{\text{int}})$	to $First(\langle \text{item} \rangle)$
Summary: $First(\langle \text{item} \rangle) = \{\underline{\text{id}}, \underline{\text{int}}\}$		

$\langle \text{it-list} \rangle \rightarrow \underline{\text{,}} \in P:$	add $First(\underline{\text{,}})$	to $First(\langle \text{it-list} \rangle)$
$\langle \text{it-list} \rangle \rightarrow \underline{\text{,}} \dots \in P:$	add $First(\underline{\text{,}})$	to $First(\langle \text{it-list} \rangle)$
Summary: $First(\langle \text{it-list} \rangle) = \{\underline{\text{,}}, \underline{\text{,}}\}$		

$\langle \text{stat} \rangle \rightarrow \underline{\text{id}} \dots \in P:$	add $First(\underline{\text{id}})$	to $First(\langle \text{stat} \rangle)$
$\langle \text{stat} \rangle \rightarrow \underline{\text{write}} \dots \in P:$	add $First(\underline{\text{write}})$	to $First(\langle \text{stat} \rangle)$
$\langle \text{stat} \rangle \rightarrow \underline{\text{read}} \dots \in P:$	add $First(\underline{\text{read}})$	to $First(\langle \text{stat} \rangle)$
Summary: $First(\langle \text{stat} \rangle) = \{\underline{\text{id}}, \underline{\text{write}}, \underline{\text{read}}\}$		

First(X) for SPL: Example

$First(\underline{\text{begin}}) := \{\underline{\text{begin}}\}$	$First(\underline{\text{id}}) := \{\underline{\text{id}}\}$	$First(\underline{\text{,}}) := \{\underline{\text{,}}\}$
$First(\underline{\text{end}}) := \{\underline{\text{end}}\}$	$First(\underline{\text{int}}) := \{\underline{\text{int}}\}$	$First(\underline{\text{(}}) := \{\underline{\text{(}}\}$
$First(\underline{\text{read}}) := \{\underline{\text{read}}\}$	$First(\underline{\text{:=}}) := \{\underline{\text{:=}}\}$	$First(\underline{\text{)}}) := \{\underline{\text{)}}\}$
$First(\underline{\text{write}}) := \{\underline{\text{write}}\}$	$First(\underline{\text{add}}) := \{\underline{\text{add}}\}$	$First(\underline{\text{;}}) := \{\underline{\text{;}}\}$

$\langle \text{item} \rangle \rightarrow \underline{\text{id}} \in P:$	add $First(\underline{\text{id}})$	to $First(\langle \text{item} \rangle)$
$\langle \text{item} \rangle \rightarrow \underline{\text{int}} \in P:$	add $First(\underline{\text{int}})$	to $First(\langle \text{item} \rangle)$
Summary: $First(\langle \text{item} \rangle) = \{\underline{\text{id}}, \underline{\text{int}}\}$		

$\langle \text{it-list} \rangle \rightarrow \underline{\text{)}} \in P:$	add $First(\underline{\text{)}})$	to $First(\langle \text{it-list} \rangle)$
$\langle \text{it-list} \rangle \rightarrow \underline{\text{,}} \dots \in P:$	add $First(\underline{\text{,}})$	to $First(\langle \text{it-list} \rangle)$
Summary: $First(\langle \text{it-list} \rangle) = \{\underline{\text{)}} , \underline{\text{,}}\}$		

$\langle \text{stat} \rangle \rightarrow \underline{\text{id}} \dots \in P:$	add $First(\underline{\text{id}})$	to $First(\langle \text{stat} \rangle)$
$\langle \text{stat} \rangle \rightarrow \underline{\text{write}} \dots \in P:$	add $First(\underline{\text{write}})$	to $First(\langle \text{stat} \rangle)$
$\langle \text{stat} \rangle \rightarrow \underline{\text{read}} \dots \in P:$	add $First(\underline{\text{read}})$	to $First(\langle \text{stat} \rangle)$
Summary: $First(\langle \text{stat} \rangle) = \{\underline{\text{id}}, \underline{\text{write}}, \underline{\text{read}}\}$		

$\langle \text{st-list} \rangle \rightarrow \underline{\text{end}} \in P:$	add $First(\underline{\text{end}})$	to $First(\langle \text{st-list} \rangle)$
$\langle \text{st-list} \rangle \rightarrow \langle \text{stat} \rangle \dots \in P:$	add $First(\langle \text{stat} \rangle)$	to $First(\langle \text{st-list} \rangle)$
Summary: $First(\langle \text{st-list} \rangle) = \{\underline{\text{id}}, \underline{\text{write}}, \underline{\text{read}}, \underline{\text{end}}\}$		

First(X) for SPL: Example

$First(\underline{\text{begin}}) := \{\underline{\text{begin}}\}$	$First(\underline{\text{id}}) := \{\underline{\text{id}}\}$	$First(\underline{,}) := \{\underline{,}\}$
$First(\underline{\text{end}}) := \{\underline{\text{end}}\}$	$First(\underline{\text{int}}) := \{\underline{\text{int}}\}$	$First(\underline{(}) := \{\underline{(}\}$
$First(\underline{\text{read}}) := \{\underline{\text{read}}\}$	$First(\underline{:=}) := \{\underline{:=}\}$	$First(\underline{)}) := \{\underline{)}\}$
$First(\underline{\text{write}}) := \{\underline{\text{write}}\}$	$First(\underline{\text{add}}) := \{\underline{\text{add}}\}$	$First(\underline{;}) := \{\underline{;}\}$

$\langle \text{item} \rangle \rightarrow \underline{\text{id}} \in P:$	$\text{add } First(\underline{\text{id}})$	$\text{to } First(\langle \text{item} \rangle)$
$\langle \text{item} \rangle \rightarrow \underline{\text{int}} \in P:$	$\text{add } First(\underline{\text{int}})$	$\text{to } First(\langle \text{item} \rangle)$
Summary: $First(\langle \text{item} \rangle) = \{\underline{\text{id}}, \underline{\text{int}}\}$		

$\langle \text{it-list} \rangle \rightarrow \underline{)} \in P:$	$\text{add } First(\underline{)})$	$\text{to } First(\langle \text{it-list} \rangle)$
$\langle \text{it-list} \rangle \rightarrow \underline{,} \dots \in P:$	$\text{add } First(\underline{,})$	$\text{to } First(\langle \text{it-list} \rangle)$
Summary: $First(\langle \text{it-list} \rangle) = \{\underline{)}, \underline{,}\}$		

$\langle \text{stat} \rangle \rightarrow \underline{\text{id}} \dots \in P:$	$\text{add } First(\underline{\text{id}})$	$\text{to } First(\langle \text{stat} \rangle)$
$\langle \text{stat} \rangle \rightarrow \underline{\text{write}} \dots \in P:$	$\text{add } First(\underline{\text{write}})$	$\text{to } First(\langle \text{stat} \rangle)$
$\langle \text{stat} \rangle \rightarrow \underline{\text{read}} \dots \in P:$	$\text{add } First(\underline{\text{read}})$	$\text{to } First(\langle \text{stat} \rangle)$
Summary: $First(\langle \text{stat} \rangle) = \{\underline{\text{id}}, \underline{\text{write}}, \underline{\text{read}}\}$		

$\langle \text{st-list} \rangle \rightarrow \underline{\text{end}} \in P:$	$\text{add } First(\underline{\text{end}})$	$\text{to } First(\langle \text{st-list} \rangle)$
$\langle \text{st-list} \rangle \rightarrow \langle \text{stat} \rangle \dots \in P:$	$\text{add } First(\langle \text{stat} \rangle)$	$\text{to } First(\langle \text{st-list} \rangle)$
Summary: $First(\langle \text{st-list} \rangle) = \{\underline{\text{id}}, \underline{\text{write}}, \underline{\text{read}}, \underline{\text{end}}\}$		

$\langle \text{prog} \rangle \rightarrow \underline{\text{begin}} \dots \in P:$	$\text{add } First(\underline{\text{begin}})$	$\text{to } First(\langle \text{prog} \rangle)$
Summary: $First(\langle \text{prog} \rangle) = \{\underline{\text{begin}}\}$		

Construction of LL Table

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

Construction of LL Table

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

$\alpha(A, a) = A \rightarrow X_1 X_2 \dots X_n \in P$
 if $a \in \text{First}(X_1)$; otherwise,
 $\alpha(A, a)$ is blank \Rightarrow **ERROR**

Construction of LL Table

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

$\alpha(A, a) = A \rightarrow X_1 X_2 \dots X_n \in P$
 if $a \in \text{First}(X_1)$; otherwise,
 $\alpha(A, a)$ is blank \Rightarrow **ERROR**

Task: LL table for SPL

	id	int	:=	...
$\langle \text{prog} \rangle$				
$\langle \text{st-list} \rangle$				
$\langle \text{stat} \rangle$				
$\langle \text{it-list} \rangle$				
$\langle \text{item} \rangle$				

Rule r : $A \rightarrow X_1 X_2 \dots X_n$	$\text{First}(X_1)$
1: $\langle \text{prog} \rangle \rightarrow \text{begin} \dots$	{ <u>begin</u> }
2: $\langle \text{st-list} \rangle \rightarrow \langle \text{stat} \rangle \dots$	{ <u>id</u> , <u>write</u> , <u>read</u> }
3: $\langle \text{st-list} \rangle \rightarrow \text{end}$	{ <u>end</u> }
4: $\langle \text{stat} \rangle \rightarrow \text{read} \dots$	{ <u>read</u> }
5: $\langle \text{stat} \rangle \rightarrow \text{write} \dots$	{ <u>write</u> }
6: $\langle \text{stat} \rangle \rightarrow \text{id} \dots$	{ <u>id</u> }
7: $\langle \text{it-list} \rangle \rightarrow , \dots$	{ <u>,</u> }
8: $\langle \text{it-list} \rangle \rightarrow)$	{ <u>)</u> }
9: $\langle \text{item} \rangle \rightarrow \text{int}$	{ <u>int</u> }
10: $\langle \text{item} \rangle \rightarrow \text{id}$	{ <u>id</u> }

Construction of LL Table

α	...	<i>a</i>	...
...			
<i>A</i>		$\alpha(A, a)$	
...			

$\alpha(A, a) = A \rightarrow X_1 X_2 \dots X_n \in P$
 if $a \in \text{First}(X_1)$; otherwise,
 $\alpha(A, a)$ is blank \Rightarrow **ERROR**

Task: LL table for SPL

	<i>id</i>	<i>int</i>	<i>:=</i>	...
<i><prog></i>		$\text{id} \in \text{First}(\text{<stat>})$		
<i><st-list></i>				
<i><stat></i>				
<i><it-list></i>				
<i><item></i>				

Rule $r: A \rightarrow X_1 X_2 \dots X_n$	$\text{First}(X_1)$
1: <i><prog></i> \rightarrow <i>begin</i> ...	{ <u><i>begin</i></u> }
2: <i><st-list></i> \rightarrow <i><stat></i> ...	{ <u><i>id</i></u> , <u><i>write</i></u> , <u><i>read</i></u> }
3: <i><st-list></i> \rightarrow <i>end</i>	{ <u><i>end</i></u> }
4: <i><stat></i> \rightarrow <i>read</i> ...	{ <u><i>read</i></u> }
5: <i><stat></i> \rightarrow <i>write</i> ...	{ <u><i>write</i></u> }
6: <i><stat></i> \rightarrow <i>id</i> ...	{ <u><i>id</i></u> }
7: <i><it-list></i> \rightarrow , ...	{ <u><i>,</i></u> }
8: <i><it-list></i> \rightarrow)	{ <u><i>)</i></u> }
9: <i><item></i> \rightarrow <i>int</i>	{ <u><i>int</i></u> }
10: <i><item></i> \rightarrow <i>id</i>	{ <u><i>id</i></u> }

Construction of LL Table

α	...	<i>a</i>	...
...			
<i>A</i>		$\alpha(A, a)$	
...			

$\alpha(A, a) = A \rightarrow X_1 X_2 \dots X_n \in P$
 if $a \in \text{First}(X_1)$; otherwise,
 $\alpha(A, a)$ is blank \Rightarrow **ERROR**

Task: LL table for SPL

	<i>id</i>	<i>int</i>	<i>:=</i>	...
<i><prog></i>				
<i><st-list></i>	2	$\text{id} \in \text{First}(\text{<stat>})$		
<i><stat></i>	6	$\text{id} \in \text{First}(\text{id})$		
<i><it-list></i>				
<i><item></i>				

Rule r : $A \rightarrow X_1 X_2 \dots X_n$	$\text{First}(X_1)$
1: <i><prog></i> \rightarrow <i>begin</i> ...	{ <u>begin</u> }
2: <i><st-list></i> \rightarrow <i><stat></i> ...	{ <u>id</u> , <u>write</u> , <u>read</u> }
3: <i><st-list></i> \rightarrow <i>end</i>	{ <u>end</u> }
4: <i><stat></i> \rightarrow <i>read</i> ...	{ <u>read</u> }
5: <i><stat></i> \rightarrow <i>write</i> ...	{ <u>write</u> }
6: <i><stat></i> \rightarrow <i>id</i> ...	{ <u>id</u> }
7: <i><it-list></i> \rightarrow , ...	{ <u>,</u> }
8: <i><it-list></i> \rightarrow)	{ <u>)</u> }
9: <i><item></i> \rightarrow <i>int</i>	{ <u>int</u> }
10: <i><item></i> \rightarrow <i>id</i>	{ <u>id</u> }

Construction of LL Table

α	...	<i>a</i>	...
...			
<i>A</i>		$\alpha(A, a)$	
...			

$\alpha(A, a) = A \rightarrow X_1 X_2 \dots X_n \in P$
 if $a \in \text{First}(X_1)$; otherwise,
 $\alpha(A, a)$ is blank \Rightarrow **ERROR**

Task: LL table for SPL

	<i>id</i>	<i>int</i>	<i>:=</i>	...
<i><prog></i>		<i>id</i> \in		
<i><st-list></i>	2 \leftarrow	$\text{First}(\text{<stat>})$		
<i><stat></i>	6 \leftarrow	<i>id</i> $\in \text{First}(\text{<id>})$		
<i><it-list></i>				
<i><item></i>	10 \leftarrow	<i>id</i> $\in \text{First}(\text{<id>})$		

Rule r : $A \rightarrow X_1 X_2 \dots X_n$	$\text{First}(X_1)$
1: <i><prog></i> \rightarrow <i>begin</i> ...	{ <u><i>begin</i></u> }
2: <i><st-list></i> \rightarrow <i><stat></i> ...	{ <u><i>id</i></u> , <u><i>write</i></u> , <u><i>read</i></u> }
3: <i><st-list></i> \rightarrow <i>end</i>	{ <u><i>end</i></u> }
4: <i><stat></i> \rightarrow <i>read</i> ...	{ <u><i>read</i></u> }
5: <i><stat></i> \rightarrow <i>write</i> ...	{ <u><i>write</i></u> }
6: <i><stat></i> \rightarrow <i>id</i> ...	{ <u><i>id</i></u> }
7: <i><it-list></i> \rightarrow , ...	{ <u><i>,</i></u> }
8: <i><it-list></i> \rightarrow)	{ <u><i>)</i></u> }
9: <i><item></i> \rightarrow <i>int</i>	{ <u><i>int</i></u> }
10: <i><item></i> \rightarrow <i>id</i>	{ <u><i>id</i></u> }

Construction of LL Table

α	...	<i>a</i>	...
...			
<i>A</i>		$\alpha(A, a)$	
...			

$\alpha(A, a) = A \rightarrow X_1 X_2 \dots X_n \in P$
 if $a \in \text{First}(X_1)$; otherwise,
 $\alpha(A, a)$ is blank \Rightarrow **ERROR**

Task: LL table for SPL

	<i>id</i>	<i>int</i>	<i>:=</i>	...
<i><prog></i>		<i>id</i> \in		
<i><st-list></i>	2 \leftarrow	$\text{First}(\text{<stat>})$		
<i><stat></i>	6 \leftarrow	<i>id</i> $\in \text{First}(\text{<id>})$		
<i><it-list></i>				
<i><item></i>	10 \leftarrow	<i>id</i> $\in \text{First}(\text{<id>})$		

Construct the rest
analogically.

Rule $r: A \rightarrow X_1 X_2 \dots X_n$	$\text{First}(X_1)$
1: <i><prog></i> \rightarrow <i>begin</i> ...	{ <u>begin</u> }
2: <i><st-list></i> \rightarrow <i><stat></i> ...	{ <u>id</u> , <u>write</u> , <u>read</u> }
3: <i><st-list></i> \rightarrow <i>end</i>	{ <u>end</u> }
4: <i><stat></i> \rightarrow <i>read</i> ...	{ <u>read</u> }
5: <i><stat></i> \rightarrow <i>write</i> ...	{ <u>write</u> }
6: <i><stat></i> \rightarrow <i>id</i> ...	{ <u>id</u> }
7: <i><it-list></i> \rightarrow , ...	{,}
8: <i><it-list></i> \rightarrow)	{ <u>)</u> }
9: <i><item></i> \rightarrow <i>int</i>	{ <u>int</u> }
10: <i><item></i> \rightarrow <i>id</i>	{ <u>id</u> }

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

<prog>

**Lexical
Analyzer**

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

<prog>

**Lexical
Analyzer**

begin

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

<prog>

**Lexical
Analyzer**

begin

Parsing Based on LL Table: Example

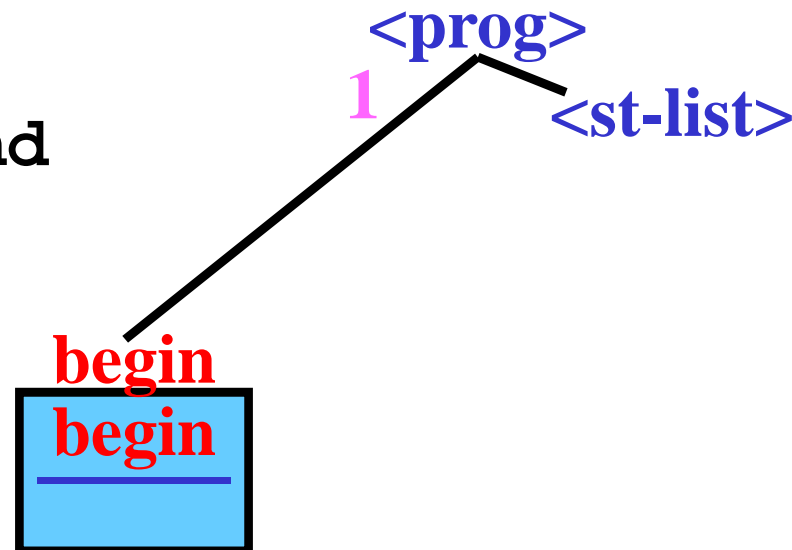
1: <prog> → begin <st-list> 6: <stat> → id := add (...
 2: <st-list> → <stat> ; <st-list> 7: <it-list> → , <item> <it-list>
 3: <st-list> → end 8: <it-list> →)
 4: <stat> → read id 9: <item> → int
 5: <stat> → write <item> 10: <item> → id

	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

**Lexical
Analyzer**



Parsing Based on LL Table: Example

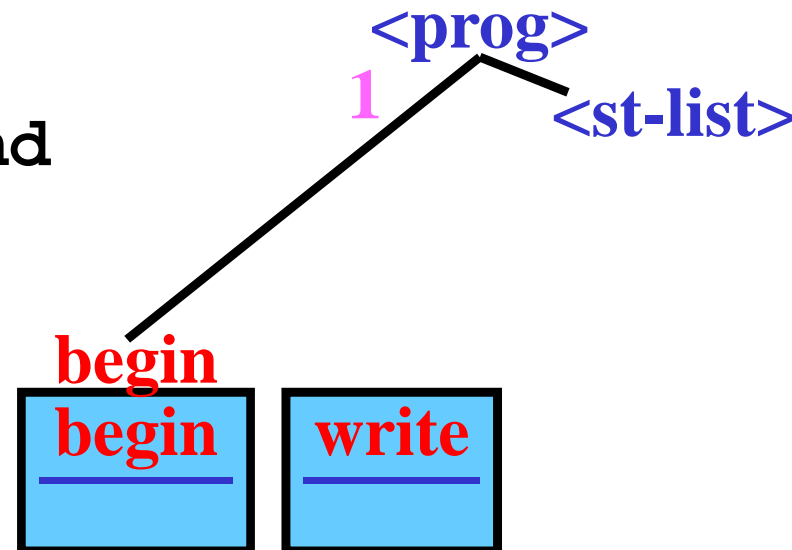
1: <prog> → begin <st-list> 6: <stat> → id := add (...
 2: <st-list> → <stat> ; <st-list> 7: <it-list> → , <item> <it-list>
 3: <st-list> → end 8: <it-list> →)
 4: <stat> → read id 9: <item> → int
 5: <stat> → write <item> 10: <item> → id

	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

**Lexical
Analyzer**



Parsing Based on LL Table: Example

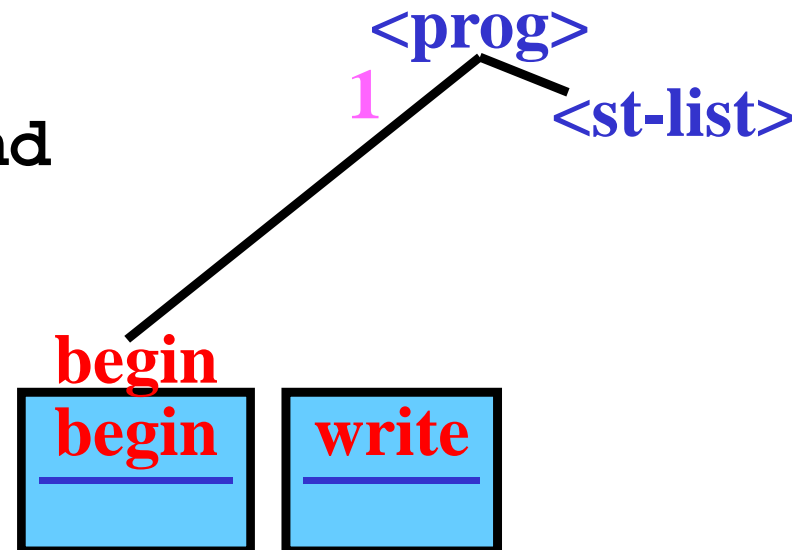
1: <prog> → begin <st-list> 6: <stat> → id := add (...
 2: <st-list> → <stat> ; <st-list> 7: <it-list> → , <item> <it-list>
 3: <st-list> → end 8: <it-list> →)
 4: <stat> → read id 9: <item> → int
 5: <stat> → write <item> 10: <item> → id

	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

**Lexical
Analyzer**



Parsing Based on LL Table: Example

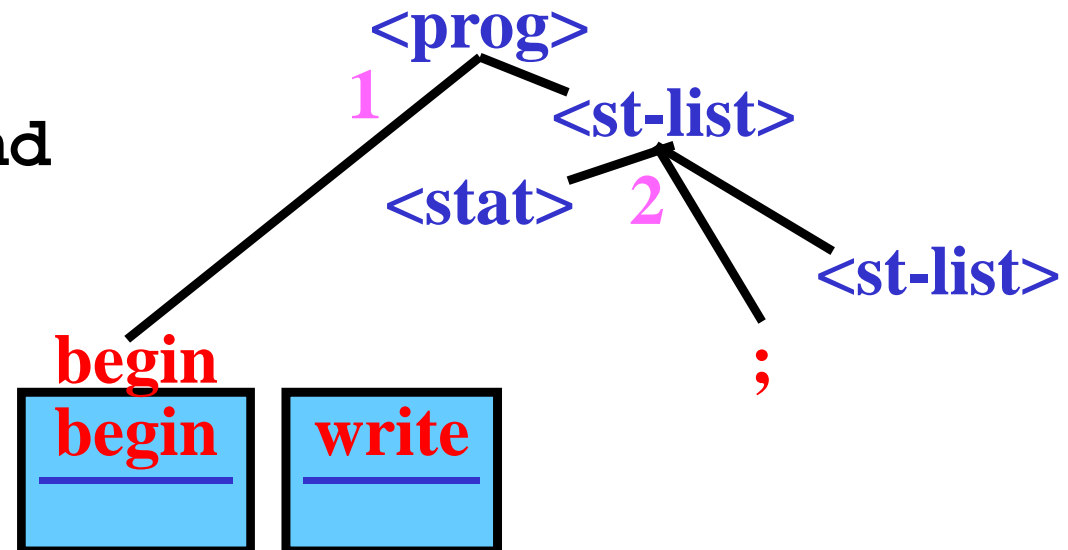
1: <prog> → begin <st-list> 6: <stat> → id := add (...
 2: <st-list> → <stat> ; <st-list> 7: <it-list> → , <item> <it-list>
 3: <st-list> → end 8: <it-list> →)
 4: <stat> → read id 9: <item> → int
 5: <stat> → write <item> 10: <item> → id

	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

**Lexical
Analyzer**



Parsing Based on LL Table: Example

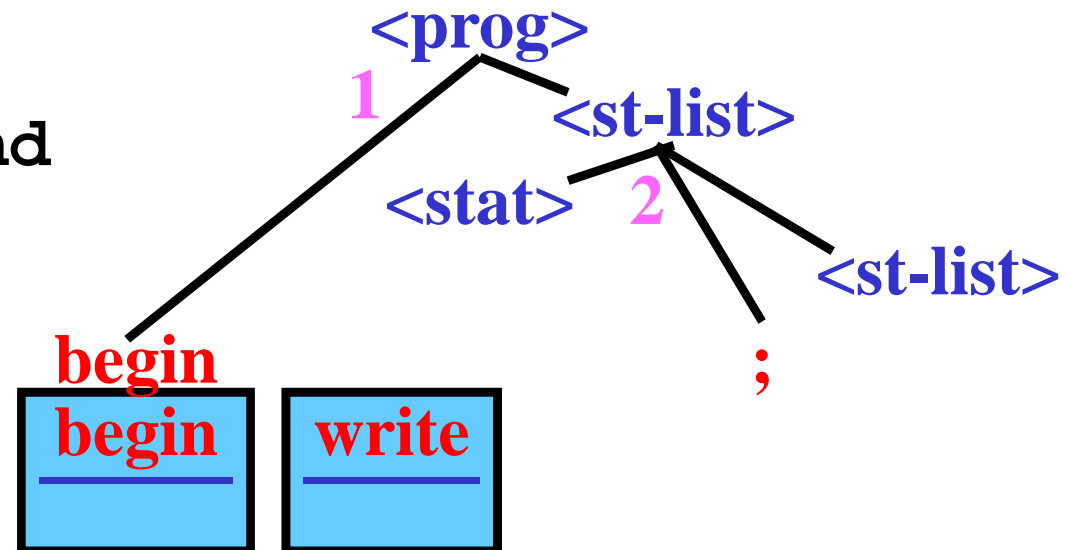
1: <prog> → begin <st-list> 6: <stat> → id := add (...
 2: <st-list> → <stat> , <st-list> 7: <it-list> → , <item> <it-list>
 3: <st-list> → end 8: <it-list> →)
 4: <stat> → read id 9: <item> → int
 5: <stat> → write <item> 10: <item> → id

	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

**Lexical
Analyzer**



Parsing Based on LL Table: Example

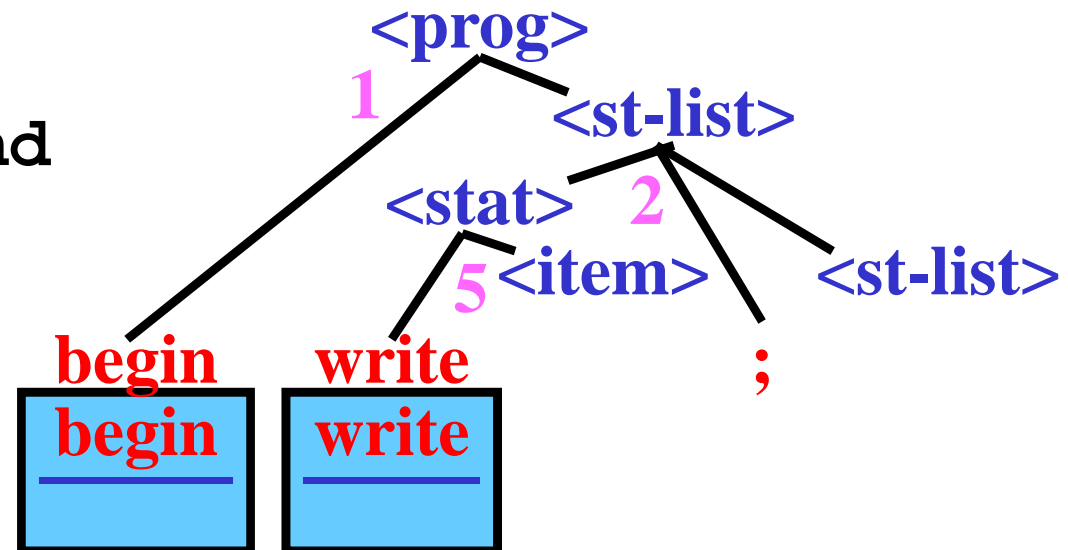
1: <prog> → begin <st-list> 6: <stat> → id := add (...
 2: <st-list> → <stat> ; <st-list> 7: <it-list> → , <item> <it-list>
 3: <st-list> → end 8: <it-list> →)
 4: <stat> → read id 9: <item> → int
 5: <stat> → write <item> 10: <item> → id

	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

**Lexical
Analyzer**



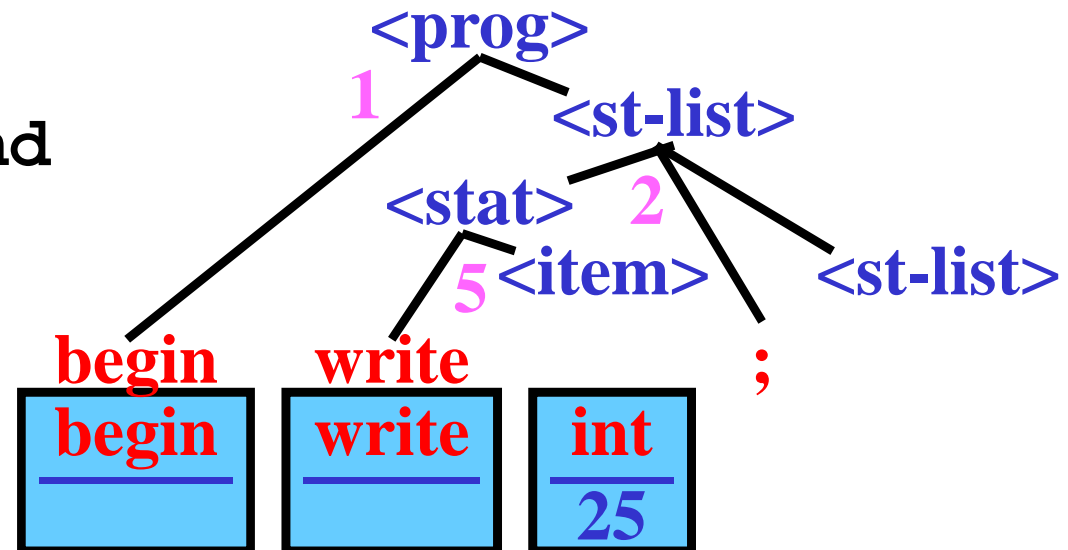
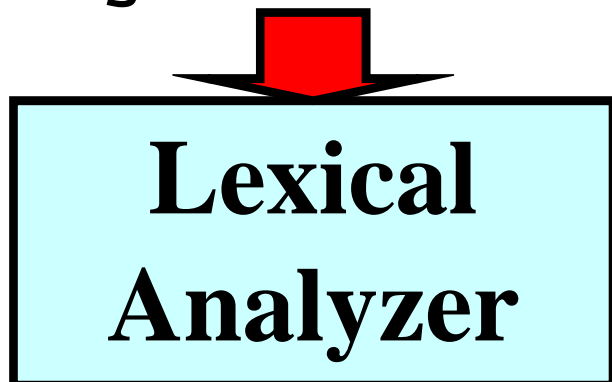
Parsing Based on LL Table: Example

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



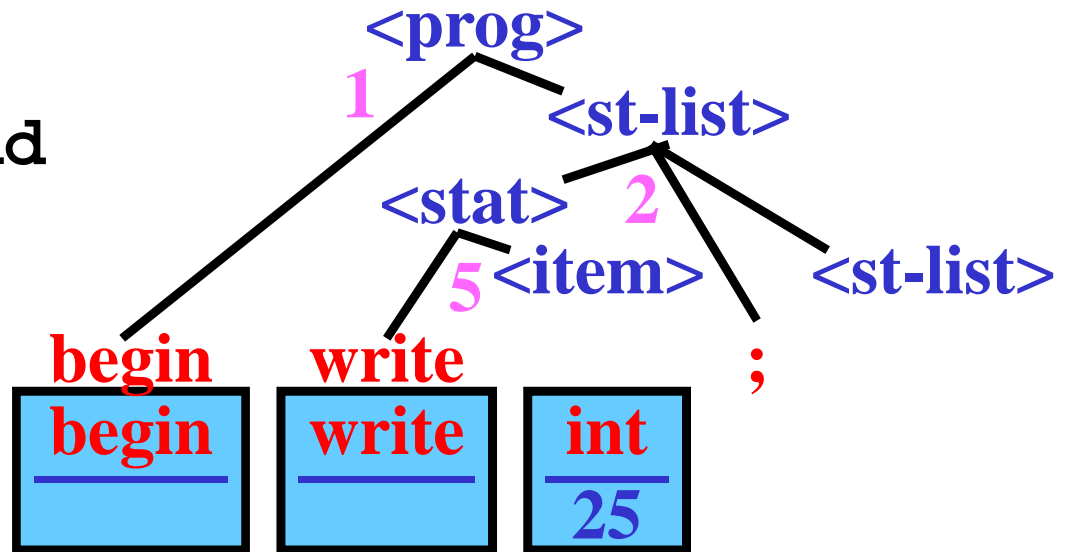
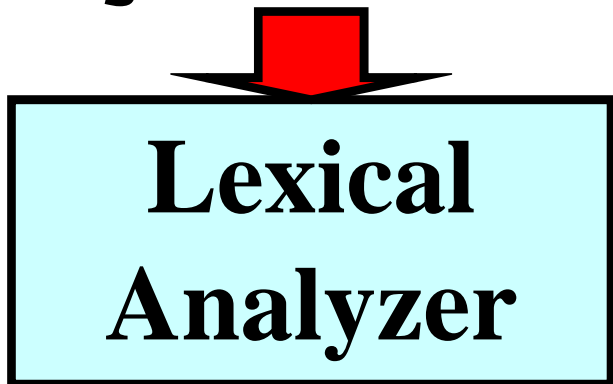
Parsing Based on LL Table: Example

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

	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

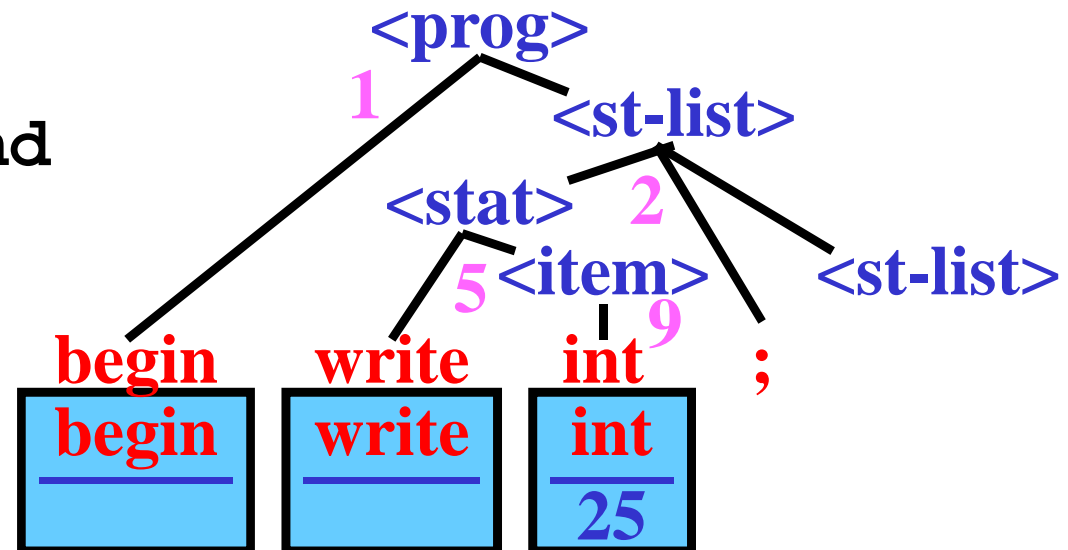
1: <prog> → begin <st-list> 6: <stat> → id := add (...
 2: <st-list> → <stat> ; <st-list> 7: <it-list> → , <item> <it-list>
 3: <st-list> → end 8: <it-list> →)
 4: <stat> → read id 9: <item> → int
 5: <stat> → write <item> 10: <item> → id

	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

**Lexical
Analyzer**



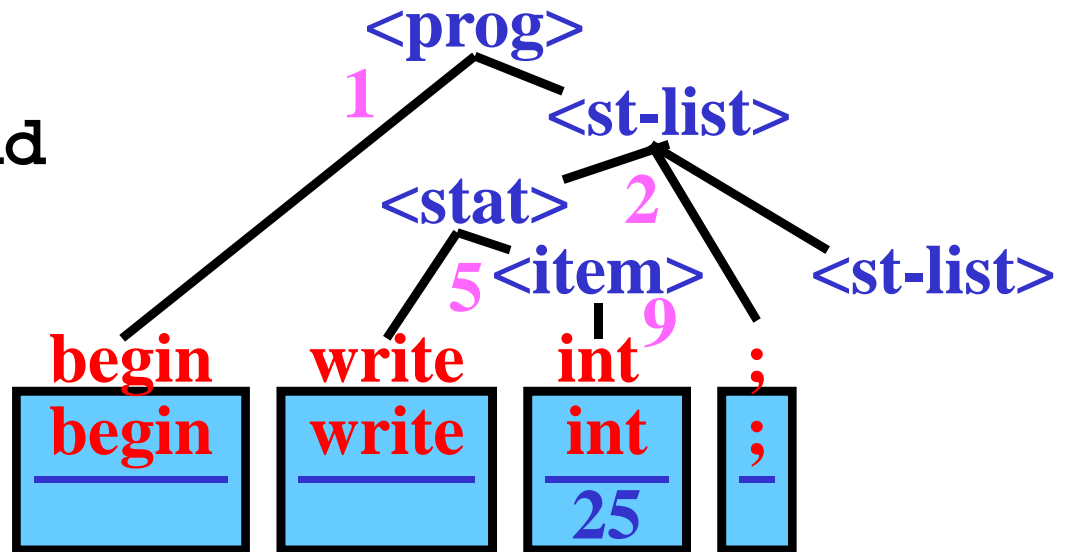
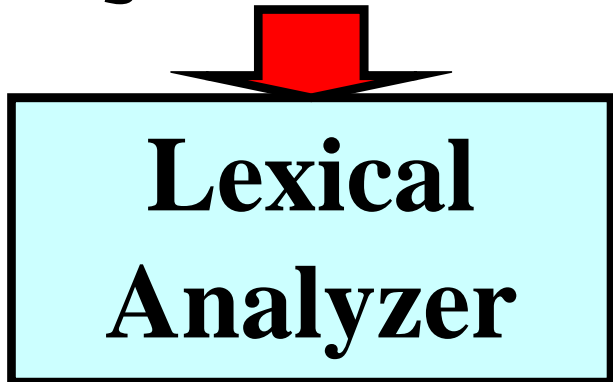
Parsing Based on LL Table: Example

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

	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

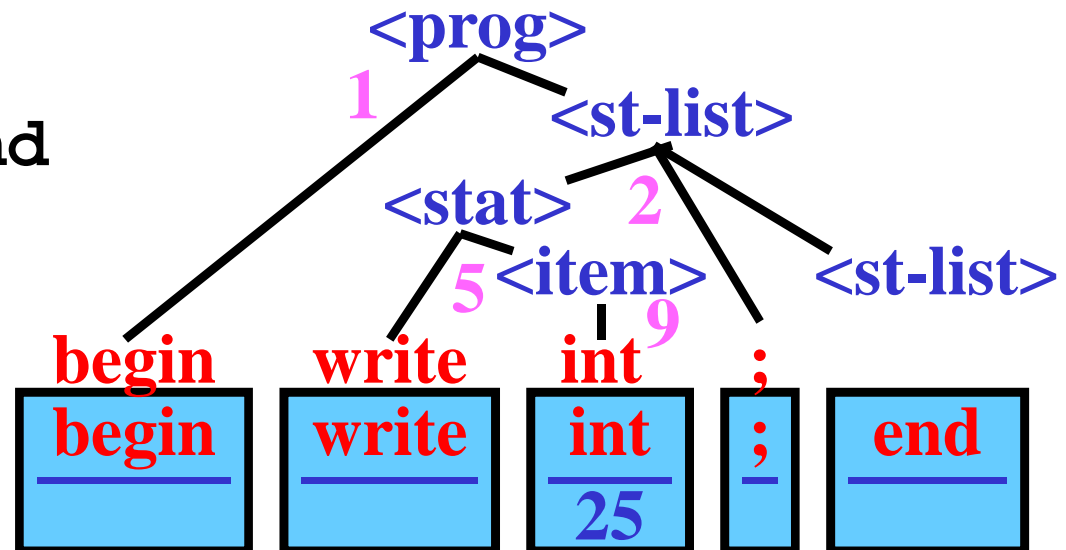
1: <prog>	→ <u>begin</u> <st-list>	6: <stat>	→ <u>id</u> <u>:=</u> <u>add</u> (...
2: <st-list>	→ <stat> <u>:</u> <st-list>	7: <it-list>	→ <u>,</u> <item> <it-list>
3: <st-list>	→ <u>end</u>	8: <it-list>	→ <u>)</u>
4: <stat>	→ <u>read</u> <u>id</u>	9: <item>	→ <u>int</u>
5: <stat>	→ <u>write</u> <item>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

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

Source program:

begin write 25; end

**Lexical
Analyzer**



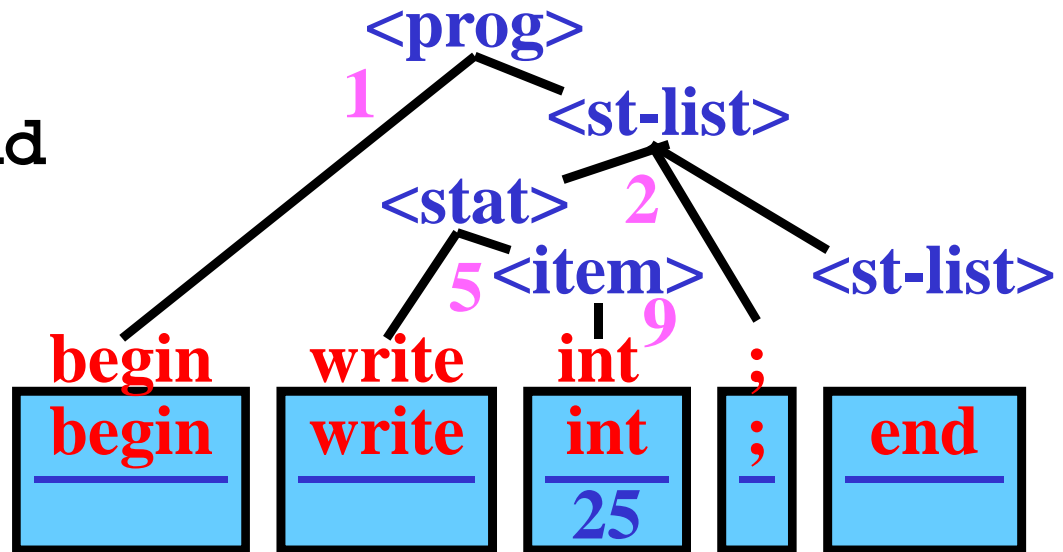
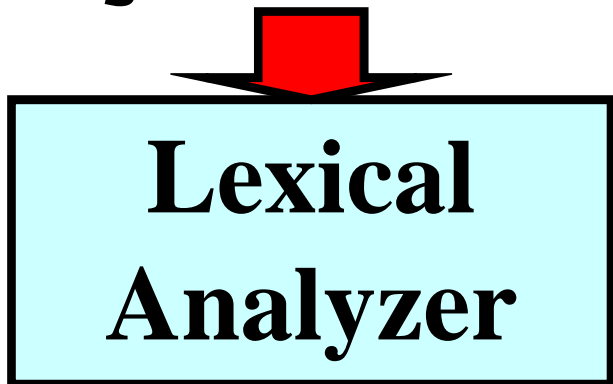
Parsing Based on LL Table: Example

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

	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
```



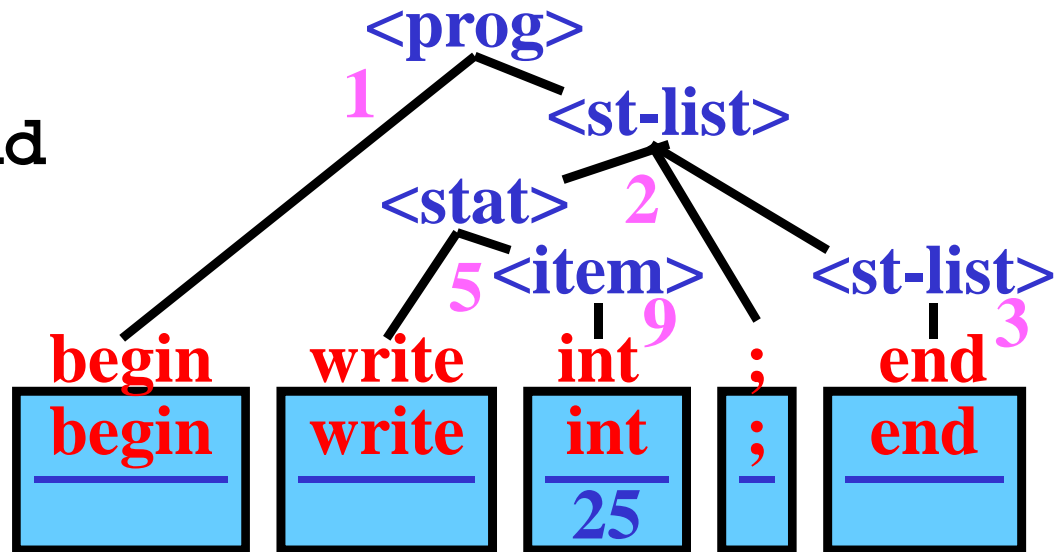
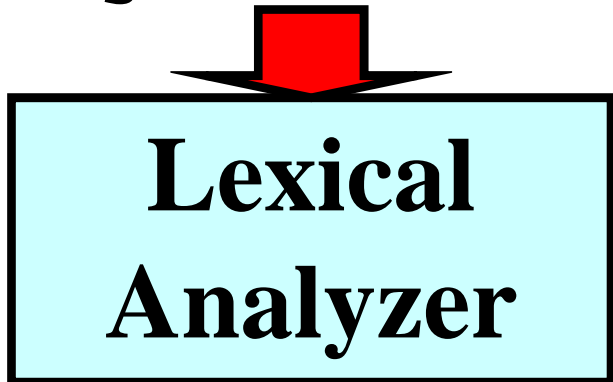
Parsing Based on LL Table: Example

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

	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:

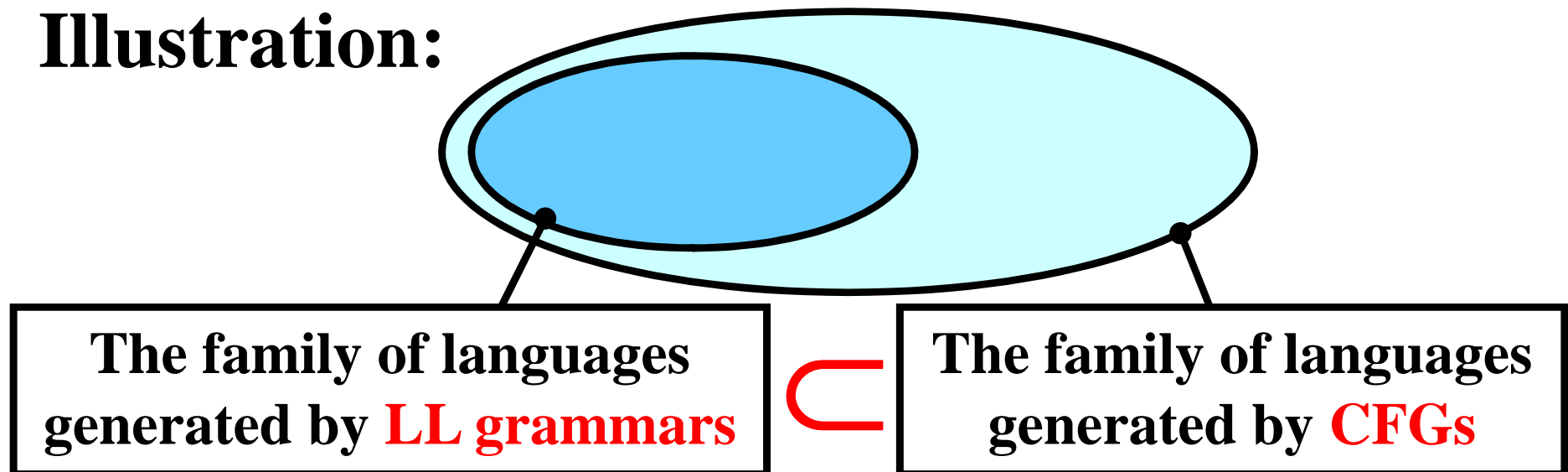
```
begin write 25; end
```



LL Grammars: Useful Transformations

Generally: CFG are stronger than LL grammars

Illustration:



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

Basic conversions:

1) Factorization

2) Left recursion replacement

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

Factorization

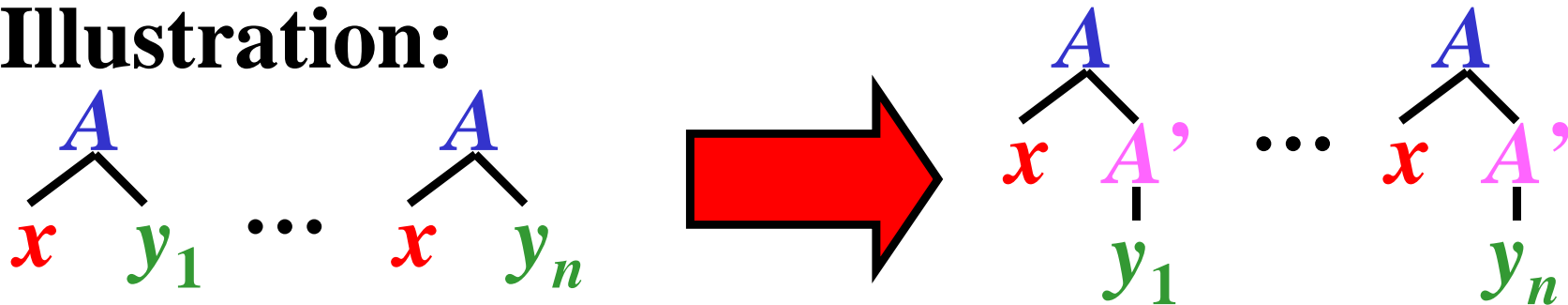
Idea: Replace rules of the form

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

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

where A' is a new nonterminal

Illustration:



Factorization

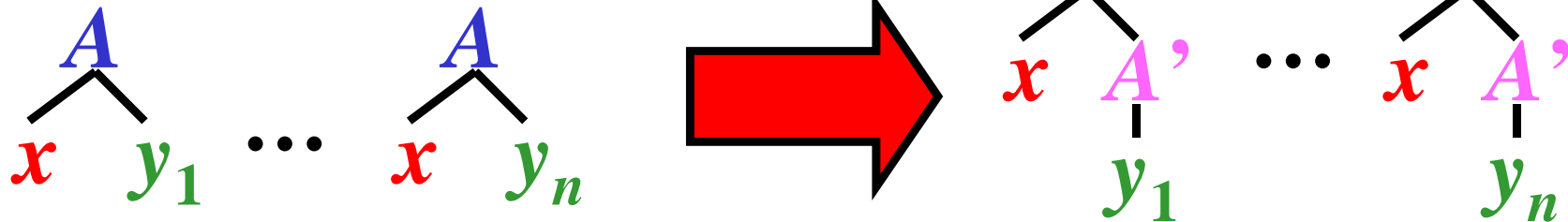
Idea: Replace rules of the form

$A \rightarrow \mathbf{x}y_1, A \rightarrow \mathbf{x}y_2, \dots, A \rightarrow \mathbf{x}y_n$ with

$A \rightarrow \mathbf{x}A', A' \rightarrow y_1, A' \rightarrow y_2, \dots, A' \rightarrow y_n,$

where A' is a new nonterminal

Illustration:



Example:

$\langle \text{stat} \rangle \rightarrow \underline{\text{write}} \underline{\text{id}}$

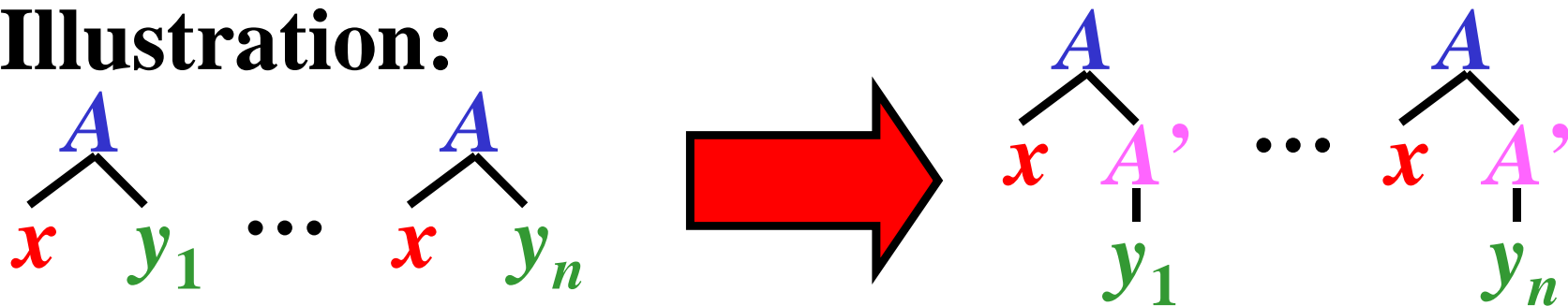
$\langle \text{stat} \rangle \rightarrow \underline{\text{write}} \underline{\text{int}}$

Factorization

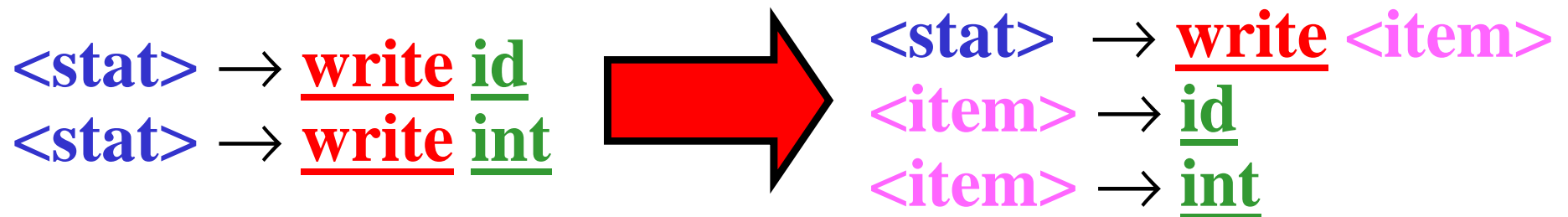
Idea: Replace rules of the form

$A \rightarrow xy_1, A \rightarrow xy_2, \dots, A \rightarrow xy_n$ with
 $A \rightarrow xA', A' \rightarrow y_1, A' \rightarrow y_2, \dots, A' \rightarrow y_n$,
 where A' is a new nonterminal

Illustration:



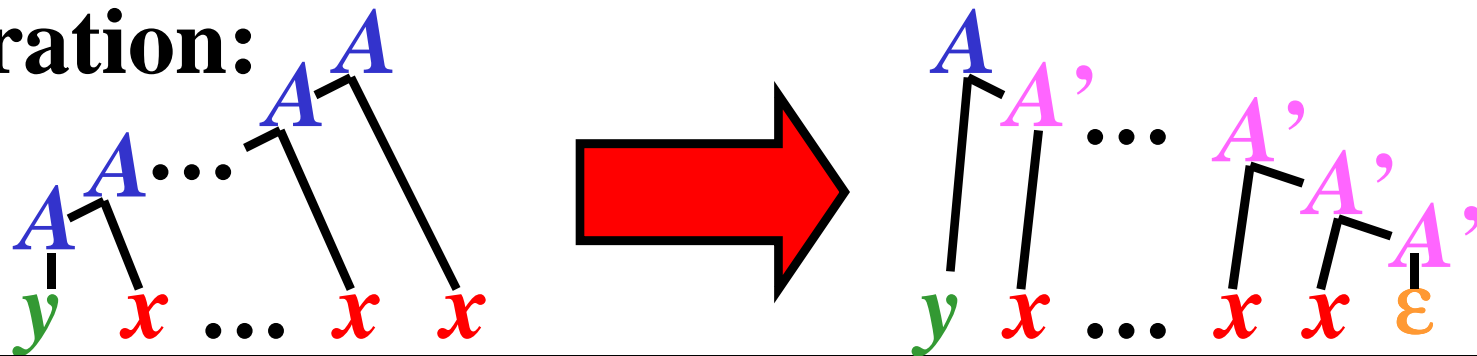
Example:



Left Recursion Replacement

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

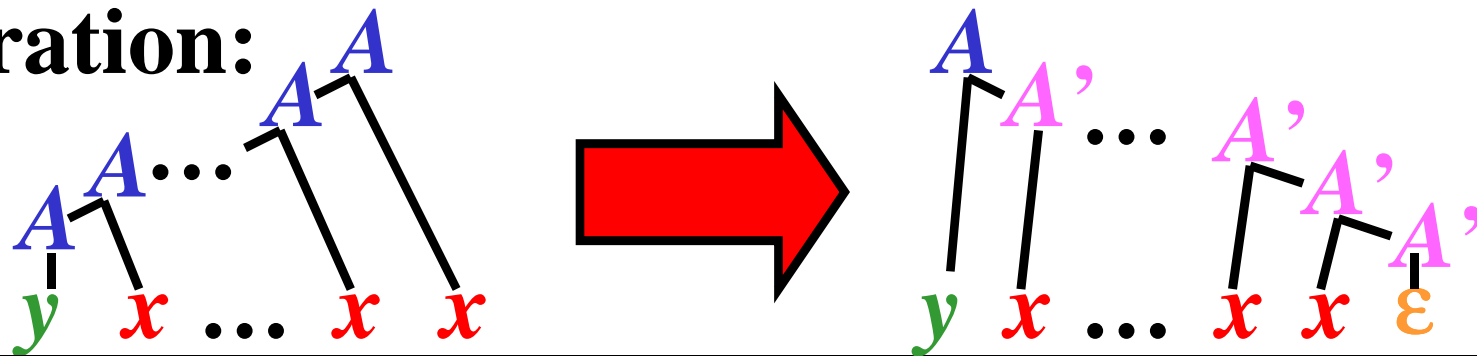
Illustration:



Left Recursion Replacement

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

Illustration:



Example:

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F$

$T \rightarrow F$

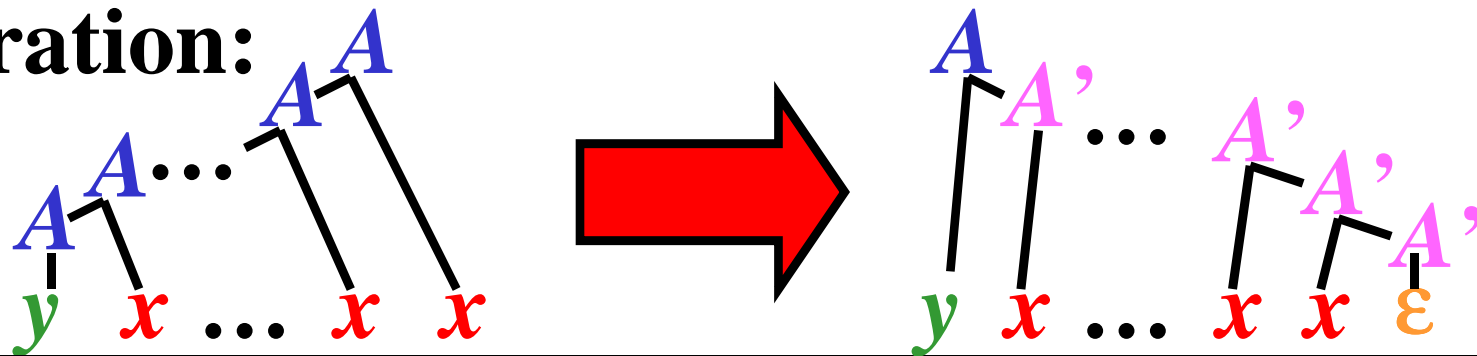
$F \rightarrow (E)$

$F \rightarrow i$

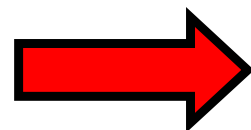
Left Recursion Replacement

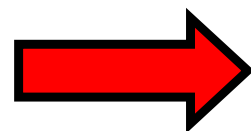
Idea: Replace rules of the form $A \rightarrow Ax$, $A \rightarrow y$ with $A \rightarrow yA'$, $A' \rightarrow xA'$, $A' \rightarrow \epsilon$, where A' is a new nonterminal.

Illustration:



Example:

$$\left. \begin{array}{l} E \rightarrow E+T \\ E \rightarrow T \\ T \rightarrow T*F \\ T \rightarrow F \end{array} \right\}$$


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


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

$$F \rightarrow (E)$$

$$F \rightarrow (E)$$

$$F \rightarrow i$$

$$F \rightarrow i$$

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:

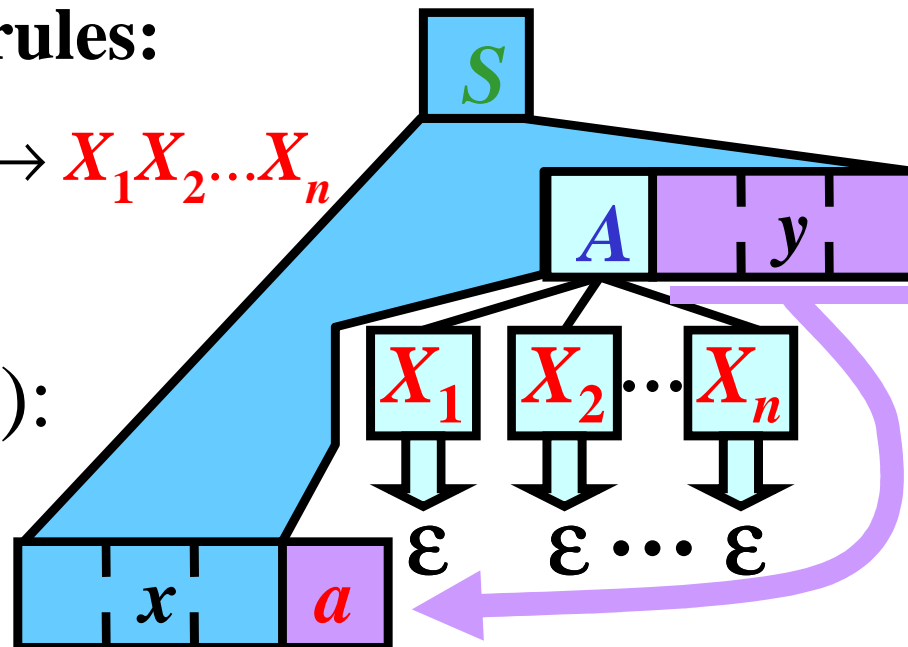
Assume that every input string of tokens ends with \$.

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

Main problem with ϵ -rules:

Rule $r: A \rightarrow X_1 X_2 \dots X_n$

Maybe: $a \notin \text{First}(A)$:



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

Grammar for Arithmetical Expressions

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

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

 $\}$

Example:

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

Set *Empty*

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

Definition: Let $G = (N, T, P, S)$ be a CFG.

Empty(\mathbf{x}) = $\{\varepsilon\}$ if $\mathbf{x} \Rightarrow^* \varepsilon$; otherwise,

Empty(\mathbf{x}) = \emptyset , where $x \in (N \cup T)^*$.

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

Set *Empty*

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

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

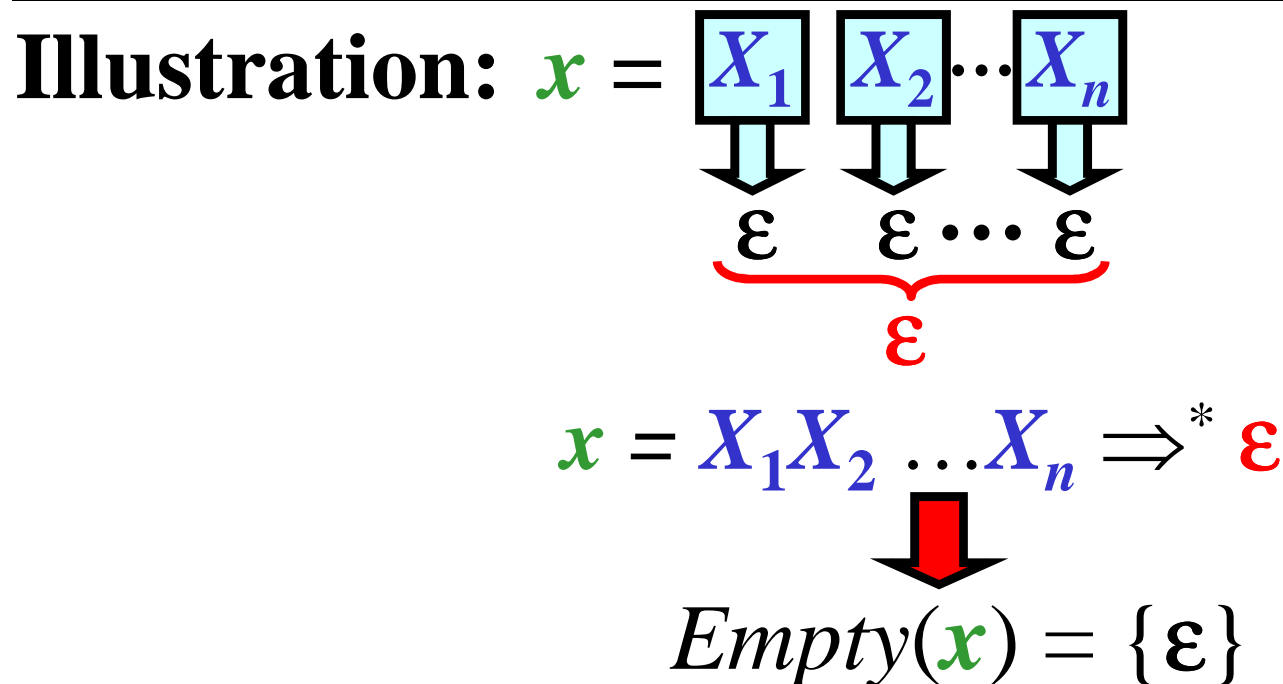
Illustration: $\mathbf{x} = \boxed{X_1} \boxed{X_2} \cdots \boxed{X_n}$

$\varepsilon \quad \varepsilon \cdots \varepsilon$
 $\underbrace{\hspace{10em}}_{\varepsilon}$

Set *Empty*

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

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



Algorithm: *Empty*(X)

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

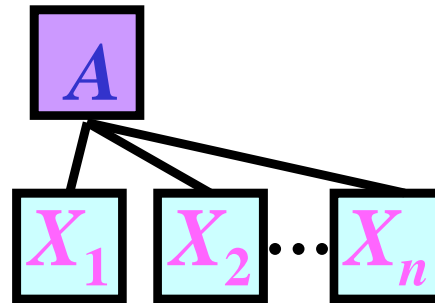
- **Method:**
- for each $a \in T$: $Empty(a) := \emptyset$
- for each $A \in N$:
 - if $A \rightarrow \varepsilon \in P$ then $Empty(A) := \{\varepsilon\}$
 - else $Empty(A) := \emptyset$
- Apply the following rule until no *Empty* set can be changed:
 - if $A \rightarrow X_1 X_2 \dots X_n \in P$ and $Empty(X_i) = \{\varepsilon\}$ for all $i = 1, \dots, n$ then $Empty(A) = \{\varepsilon\}$

Previous Algorithm: Illustration

- 1) for each $a \in T$: $Empty(a) := \emptyset$ because $a \not\Rightarrow^* \varepsilon$
 - 2) for each $r: A \rightarrow \varepsilon \in P$: $Empty(A) := \{\varepsilon\}$ because $A \Rightarrow^1 \varepsilon [r]$
-
- 3) Apply the following rules until no *Empty* set can be changed:

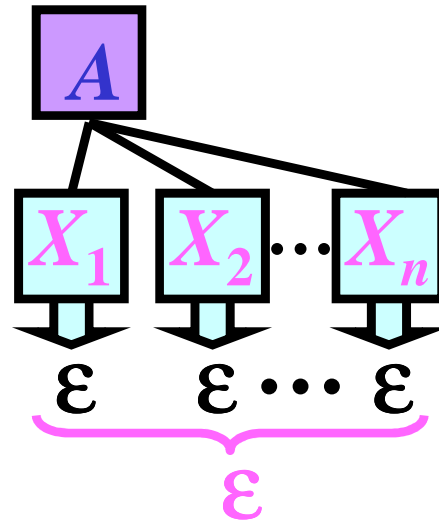
Previous Algorithm: Illustration

- 1) for each $a \in T$: $Empty(a) := \emptyset$ because $a \not\Rightarrow^* \varepsilon$
 - 2) for each $r: A \rightarrow \varepsilon \in P$: $Empty(A) := \{\varepsilon\}$ because $A \Rightarrow^1 \varepsilon [r]$
-
- 3) Apply the following rules until no *Empty* set can be changed:
- if $A \rightarrow X_1 X_2 \dots X_n \in P$ and $Empty(X_i) = \{\varepsilon\}$
for all $i = 1, \dots, n$ then $Empty(A) = \{\varepsilon\}$



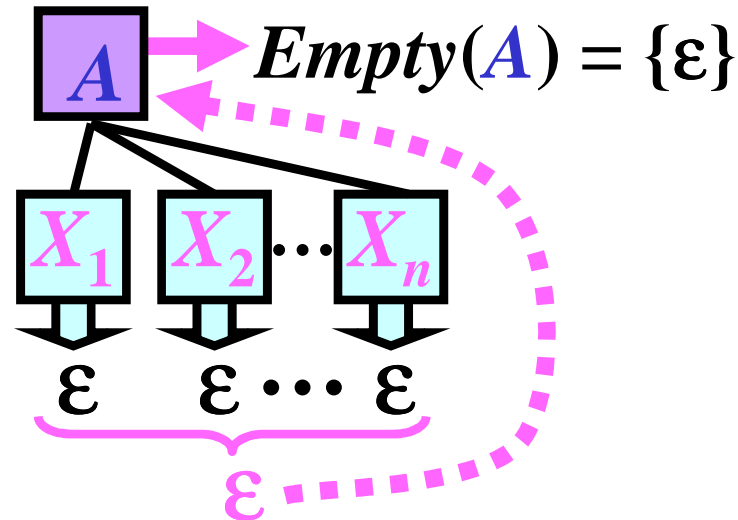
Previous Algorithm: Illustration

- 1) for each $a \in T$: $Empty(a) := \emptyset$ because $a \not\Rightarrow^* \varepsilon$
 - 2) for each $r: A \rightarrow \varepsilon \in P$: $Empty(A) := \{\varepsilon\}$ because $A \Rightarrow^1 \varepsilon [r]$
-
- 3) Apply the following rules until no *Empty* set can be changed:
- if $A \rightarrow X_1 X_2 \dots X_n \in P$ and $Empty(X_i) = \{\varepsilon\}$
for all $i = 1, \dots, n$ then $Empty(A) = \{\varepsilon\}$



Previous Algorithm: Illustration

- 1) for each $a \in T$: $Empty(a) := \emptyset$ because $a \not\Rightarrow^* \varepsilon$
 - 2) for each $r: A \rightarrow \varepsilon \in P$: $Empty(A) := \{\varepsilon\}$ because $A \Rightarrow^1 \varepsilon [r]$
-
- 3) Apply the following rules until no *Empty* set can be changed:
- if $A \rightarrow X_1 X_2 \dots X_n \in P$ and $Empty(X_i) = \{\varepsilon\}$ for all $i = 1, \dots, n$ then $Empty(A) = \{\varepsilon\}$



Empty(X) for G_{expr3} : Example

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

Initialization:

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

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

Algorithm: *First*(X)

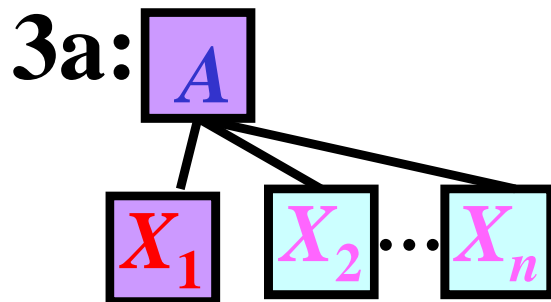
- **Input:** $G = (N, T, P, S)$
 - **Output:** $First(X)$ for every $X \in N \cup T$
-
- **Method:**
 - for each $a \in T$: $First(a) := \{a\}$
 - for each $A \in N$: $First(A) := \emptyset$
 - Apply the following rule until no *First* set can be changed:
 - if $A \rightarrow X_1 X_2 \dots X_{k-1} X_k \dots X_n \in P$ then
 - add all symbols from $First(X_1)$ to $First(A)$
 - if $Empty(X_i) = \{\epsilon\}$ for all $i = 1, \dots, k-1$, where $k \leq n$ then add all symbols from $First(X_k)$ to $First(A)$

Previous Algorithm: Illustration

- 1) for each $a \in T$: $First(a) := \{a\}$ because $a \Rightarrow^0 a$
 - 2) for each $A \in N$: $First(A) := \emptyset$ (initialization)
-
- 3) Apply the following rules until no *First* set or *Empty* set can be changed:
- if $A \rightarrow X_1 X_2 \dots X_{k-1} X_k \dots X_n \in P$ then

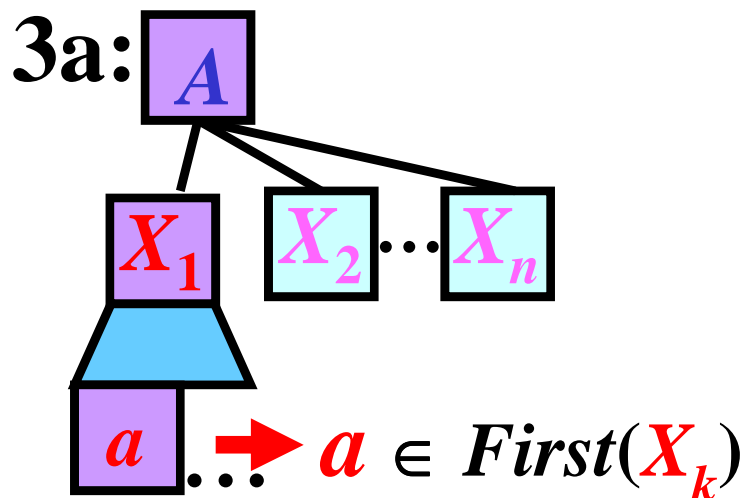
Previous Algorithm: Illustration

- 1) for each $a \in T$: $First(a) := \{a\}$ because $a \Rightarrow^0 a$
 - 2) for each $A \in N$: $First(A) := \emptyset$ (initialization)
-
- 3) Apply the following rules until no *First* set or *Empty* set can be changed:
- if $A \rightarrow X_1 X_2 \dots X_{k-1} X_k \dots X_n \in P$ then
 - 3a) add all symbols from $First(X_1)$ to $First(A)$



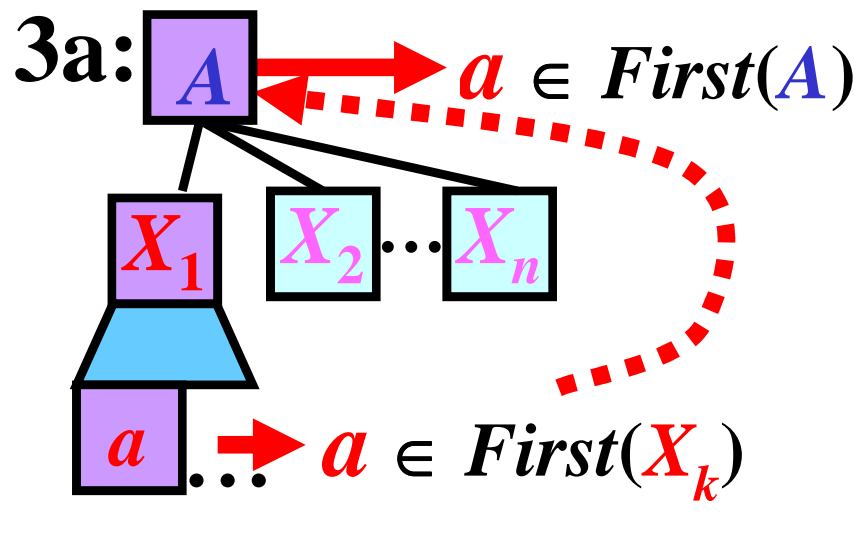
Previous Algorithm: Illustration

- 1) for each $a \in T$: $First(a) := \{a\}$ because $a \Rightarrow^0 a$
 - 2) for each $A \in N$: $First(A) := \emptyset$ (initialization)
-
- 3) Apply the following rules until no *First* set or *Empty* set can be changed:
- if $A \rightarrow X_1 X_2 \dots X_{k-1} X_k \dots X_n \in P$ then
 - 3a) add all symbols from $First(X_1)$ to $First(A)$



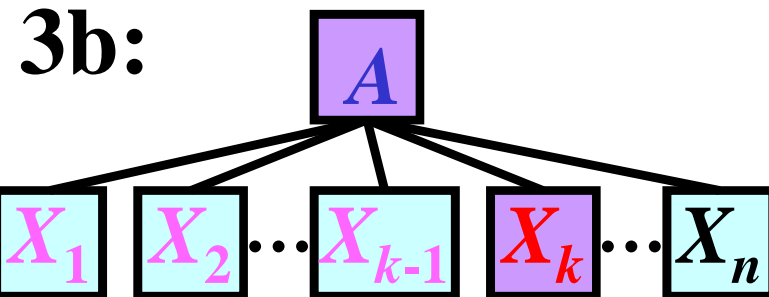
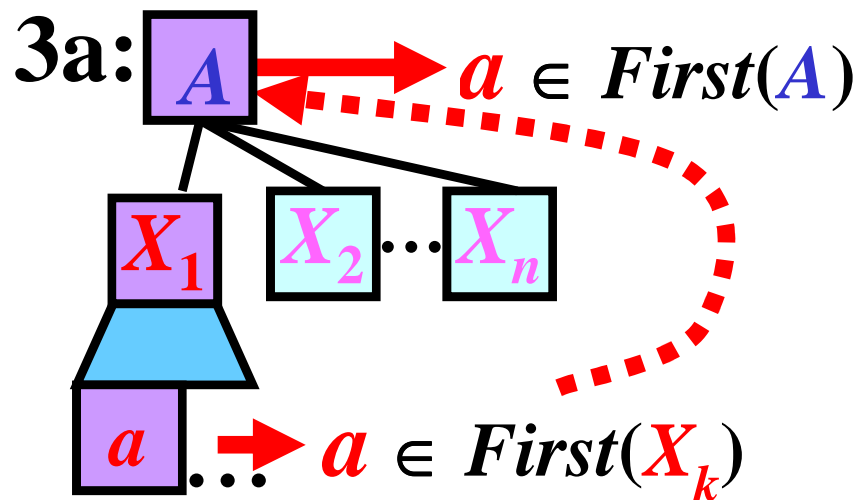
Previous Algorithm: Illustration

- 1) for each $a \in T$: $First(a) := \{a\}$ because $a \Rightarrow^0 a$
 - 2) for each $A \in N$: $First(A) := \emptyset$ (initialization)
-
- 3) Apply the following rules until no *First* set or *Empty* set can be changed:
- if $A \rightarrow X_1 X_2 \dots X_{k-1} X_k \dots X_n \in P$ then
 - 3a) add all symbols from $First(X_1)$ to $First(A)$



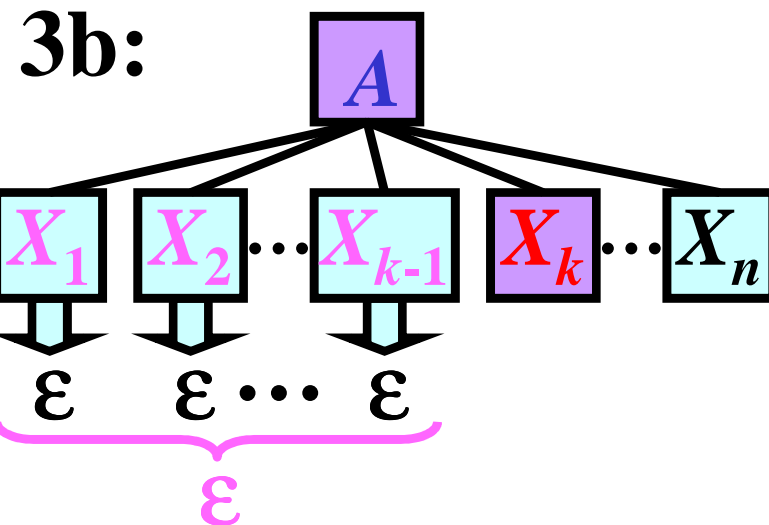
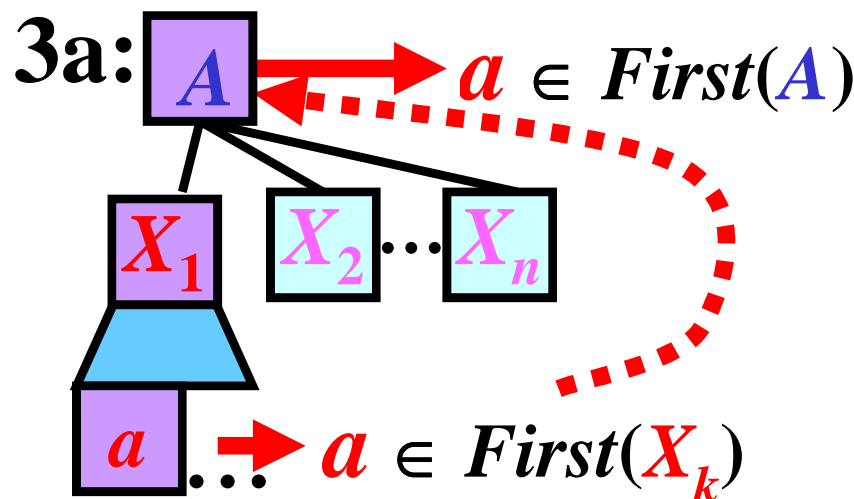
Previous Algorithm: Illustration

- 1) for each $a \in T$: $First(a) := \{a\}$ because $a \Rightarrow^0 a$
 - 2) for each $A \in N$: $First(A) := \emptyset$ (initialization)
-
- 3) Apply the following rules until no *First* set or *Empty* set can be changed:
 - if $A \rightarrow X_1 X_2 \dots X_{k-1} X_k \dots X_n \in P$ then
 - 3a) add all symbols from $First(X_1)$ to $First(A)$
 - 3b) if $Empty(X_i) = \{\epsilon\}$ for all $i = 1, \dots, k-1$, where $k < n$ then add all symbols from $First(X_k)$ to $First(A)$:



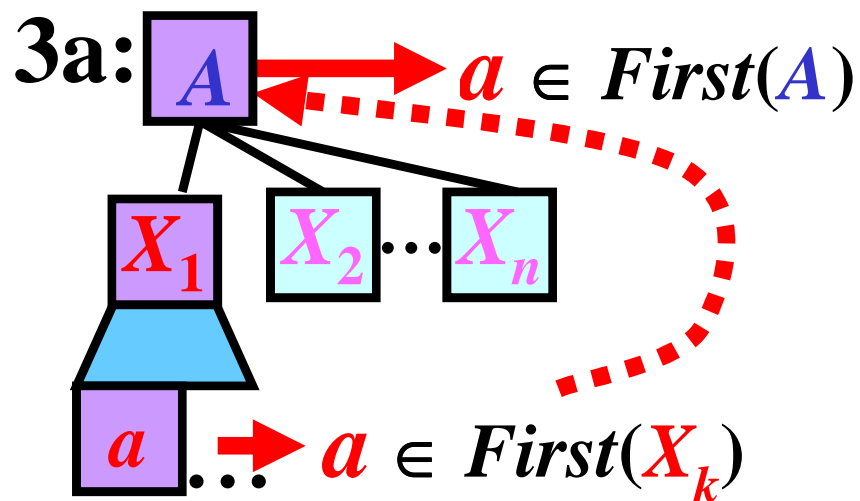
Previous Algorithm: Illustration

- 1) for each $a \in T$: $First(a) := \{a\}$ because $a \Rightarrow^0 a$
 - 2) for each $A \in N$: $First(A) := \emptyset$ (initialization)
-
- 3) Apply the following rules until no *First* set or *Empty* set can be changed:
- if $A \rightarrow X_1 X_2 \dots X_{k-1} X_k \dots X_n \in P$ then
 - 3a) add all symbols from $First(X_1)$ to $First(A)$
 - 3b) if $Empty(X_i) = \{\epsilon\}$ for all $i = 1, \dots, k-1$, where $k < n$
 then add all symbols from $First(X_k)$ to $First(A)$:

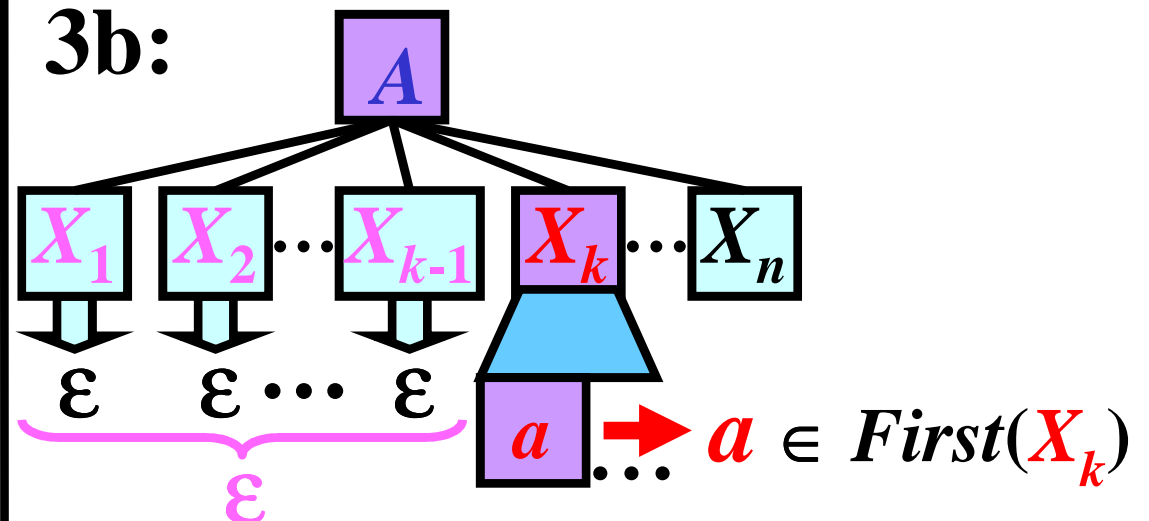


Previous Algorithm: Illustration

- 1) for each $a \in T$: $First(a) := \{a\}$ because $a \Rightarrow^0 a$
 - 2) for each $A \in N$: $First(A) := \emptyset$ (initialization)
-
- 3) Apply the following rules until no *First* set or *Empty* set can be changed:
- if $A \rightarrow X_1 X_2 \dots X_{k-1} X_k \dots X_n \in P$ then
 - 3a) add all symbols from $First(X_1)$ to $First(A)$
 - 3b) if $Empty(X_i) = \{\epsilon\}$ for all $i = 1, \dots, k-1$, where $k < n$
 then add all symbols from $First(X_k)$ to $First(A)$:

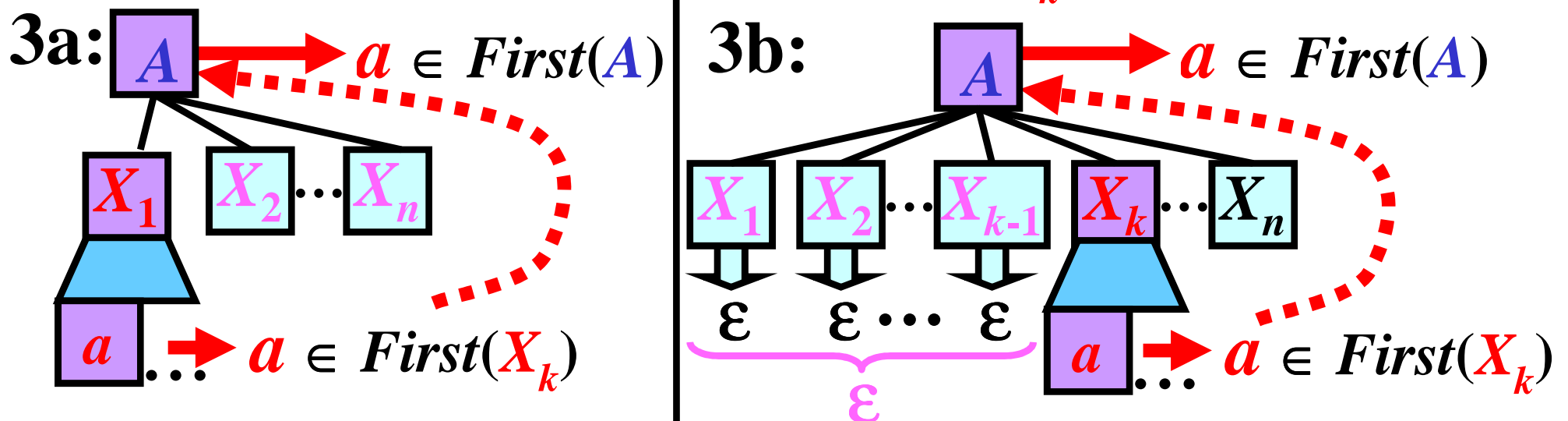


3b:



Previous Algorithm: Illustration

- 1) for each $a \in T$: $First(a) := \{a\}$ because $a \Rightarrow^0 a$
 - 2) for each $A \in N$: $First(A) := \emptyset$ (initialization)
-
- 3) Apply the following rules until no *First* set or *Empty* set can be changed:
- if $A \rightarrow X_1 X_2 \dots X_{k-1} X_k \dots X_n \in P$ then
 - 3a) add all symbols from $First(X_1)$ to $First(A)$
 - 3b) if $Empty(X_i) = \{\epsilon\}$ for all $i = 1, \dots, k-1$, where $k < n$
 then add all symbols from $First(X_k)$ to $First(A)$:



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

Initialization:

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

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

$First(X)$ 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: $First(T') = \{*\}$

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

$First(X)$ 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:

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

$F \rightarrow i \in P:$ **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: $First(E') = \{+\}$

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

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

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

$E \rightarrow TE' \in P:$ **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

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

Set <i>Empty</i> for all $X \in N \cup T$:	$Empty(\mathbf{i}) := \emptyset$	$Empty(\mathbf{E}) := \emptyset$
	$Empty(+) := \emptyset$	$Empty(\mathbf{E}') := \{\varepsilon\}$
	$Empty(*) := \emptyset$	$Empty(\mathbf{T}) := \emptyset$
	$Empty(() := \emptyset$	$Empty(\mathbf{T}') := \{\varepsilon\}$
	$Empty()) := \emptyset$	$Empty(\mathbf{F}) := \emptyset$

Set <i>First</i> for all $X \in N \cup T$:	$First(\mathbf{i}) := \{\mathbf{i}\}$	$First(\mathbf{E}) := \{\mathbf{i}, (\}$
	$First(+) := \{+\}$	$First(\mathbf{E}') := \{+\}$
	$First(*) := \{*\}$	$First(\mathbf{T}) := \{\mathbf{i}, (\}$
	$First(() := \{(\}$	$First(\mathbf{T}') := \{*\}$
	$First()) := \{) \}$	$First(\mathbf{F}) := \{\mathbf{i}, (\}$

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

Algorithm: $First(X_1X_2\dots X_n)$

- **Input:** $G = (N, T, P, S)$; $First(X)$ & $Empty(X)$ for every $X \in N \cup T$; $x = X_1X_2\dots X_n$, where $x \in (N \cup T)^+$
 - **Output:** $First(X_1X_2\dots X_n)$
-
- **Method:**
 - $First(X_1X_2\dots X_n) := First(X_1)$
 - Apply the following rule until nothing can be added to $First(X_1X_2\dots X_{k-1}X_k\dots X_n)$:
 - if $Empty(X_i) = \{\epsilon\}$ for all $i = 1, \dots, k-1$, where $k \leq n$
 then add all symbols from $First(X_k)$ to $First(X_1X_2\dots X_n)$
-

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

Illustration:

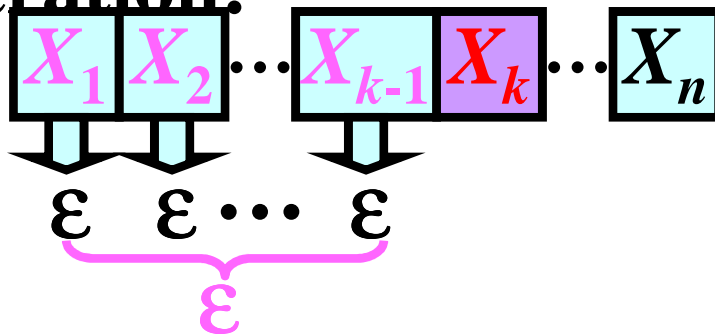


Algorithm: $First(X_1X_2\dots X_n)$

- **Input:** $G = (N, T, P, S)$; $First(X)$ & $Empty(X)$ for every $X \in N \cup T$; $x = X_1X_2\dots X_n$, where $x \in (N \cup T)^+$
 - **Output:** $First(X_1X_2\dots X_n)$
-
- **Method:**
 - $First(X_1X_2\dots X_n) := First(X_1)$
 - Apply the following rule until nothing can be added to $First(X_1X_2\dots X_{k-1}X_k\dots X_n)$:
 - if $Empty(X_i) = \{\epsilon\}$ for all $i = 1, \dots, k-1$, where $k \leq n$
 - then add all symbols from $First(X_k)$ to $First(X_1X_2\dots X_n)$
-

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

Illustration:

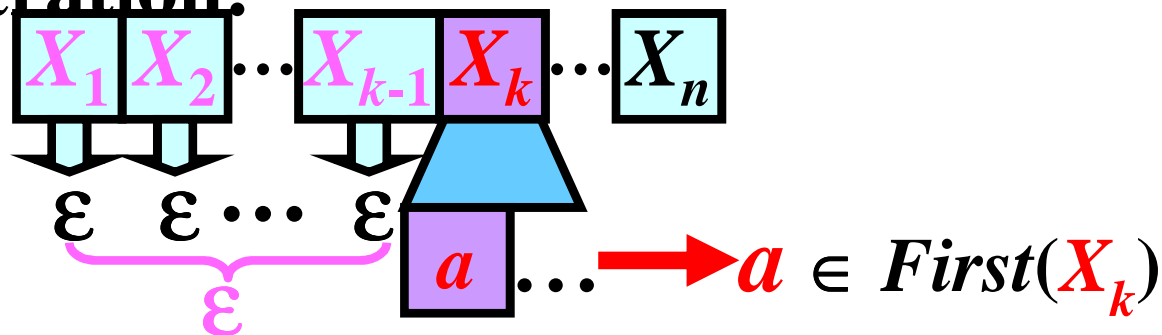


Algorithm: $First(X_1X_2\dots X_n)$

- **Input:** $G = (N, T, P, S)$; $First(X)$ & $Empty(X)$ for every $X \in N \cup T$; $x = X_1X_2\dots X_n$, where $x \in (N \cup T)^+$
 - **Output:** $First(X_1X_2\dots X_n)$
-
- **Method:**
 - $First(X_1X_2\dots X_n) := First(X_1)$
 - Apply the following rule until nothing can be added to $First(X_1X_2\dots X_{k-1}X_k\dots X_n)$:
 - if $Empty(X_i) = \{\epsilon\}$ for all $i = 1, \dots, k-1$, where $k \leq n$
 - then add all symbols from $First(X_k)$ to $First(X_1X_2\dots X_n)$
-

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

Illustration:

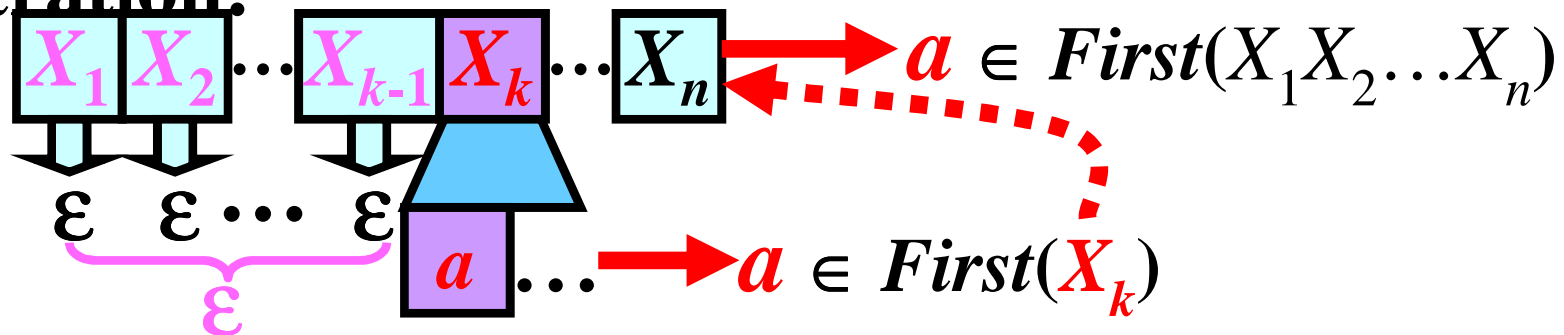


Algorithm: $First(X_1X_2\dots X_n)$

- **Input:** $G = (N, T, P, S)$; $First(X)$ & $Empty(X)$ for every $X \in N \cup T$; $x = X_1X_2\dots X_n$, where $x \in (N \cup T)^+$
 - **Output:** $First(X_1X_2\dots X_n)$
-
- **Method:**
 - $First(X_1X_2\dots X_n) := First(X_1)$
 - Apply the following rule until nothing can be added to $First(X_1X_2\dots X_{k-1}X_k\dots X_n)$:
 - if $Empty(X_i) = \{\epsilon\}$ for all $i = 1, \dots, k-1$, where $k \leq n$
 - then add all symbols from $First(X_k)$ to $First(X_1X_2\dots X_n)$
-

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

Illustration:



$First(X_1X_2\dots X_n)$: Example

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

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

Task: $First(\underline{E}'\underline{T}'\underline{F}\underline{E}\underline{T})$

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

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

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

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

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

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

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

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

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

Illustration: X_1 X_2 \dots X_n

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:**

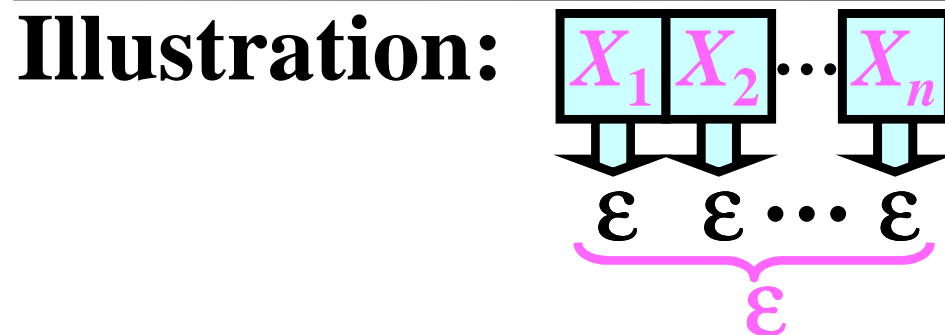
- **if** $Empty(X_i) = \{\varepsilon\}$ for all $i = 1, \dots, n$ **then**

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

else

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

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



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:**

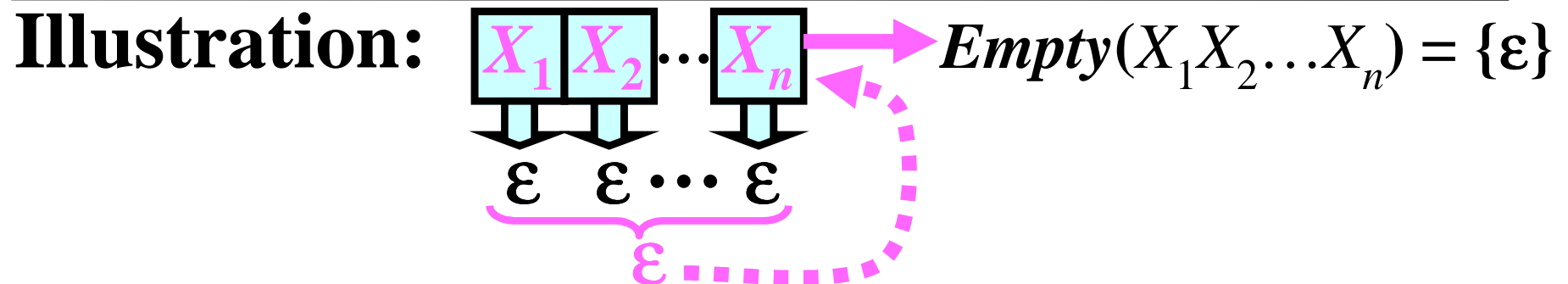
- **if** $Empty(X_i) = \{\varepsilon\}$ for all $i = 1, \dots, n$ **then**

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

else

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

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



$Empty(X_1X_2\dots X_n)$: Example

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

Set <i>Empty</i> for all $X \in N$:	$Empty(\mathbf{E})$	$:= \emptyset$
	$Empty(\mathbf{E'})$	$:= \{\varepsilon\}$
	$Empty(\mathbf{T})$	$:= \emptyset$
	$Empty(\mathbf{T'})$	$:= \{\varepsilon\}$
	$Empty(\mathbf{F})$	$:= \emptyset$

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

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

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

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

Illustration:

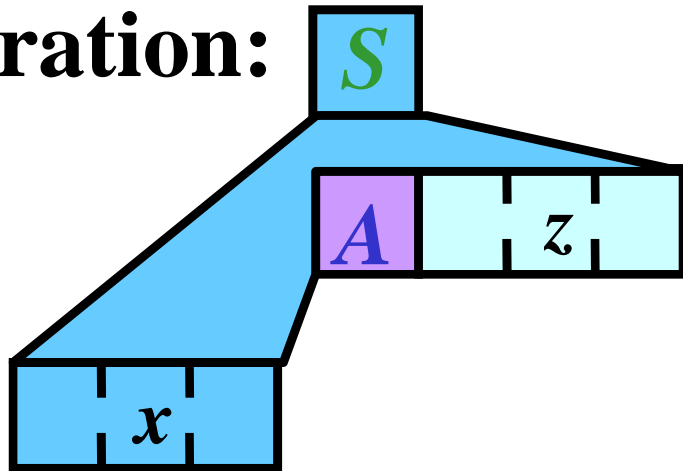
Set *Follow*

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

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

Illustration:



$$S \Rightarrow^* xAz$$

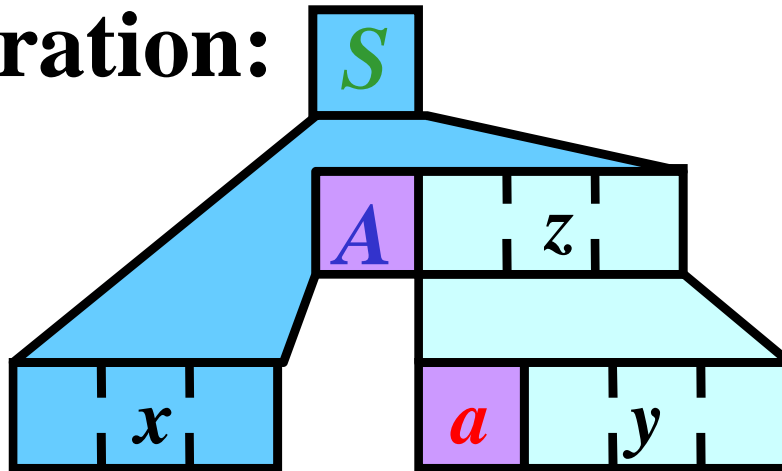
Set *Follow*

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

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

Illustration:



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

$$\downarrow$$

$$a \in \text{Follow}(A)$$

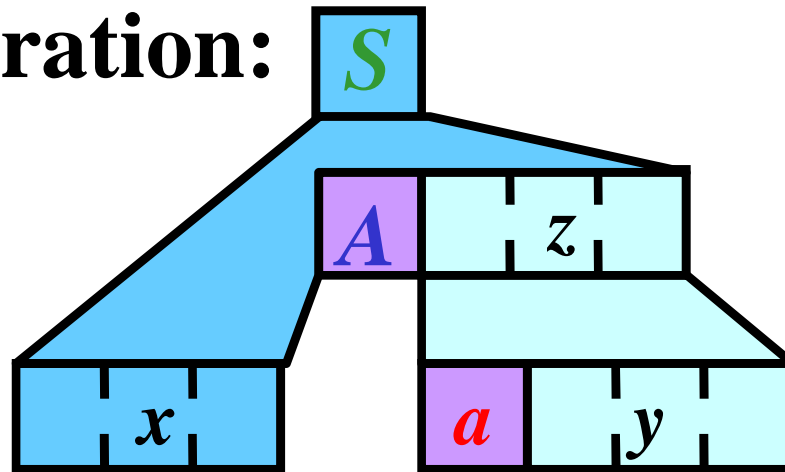
Set *Follow*

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

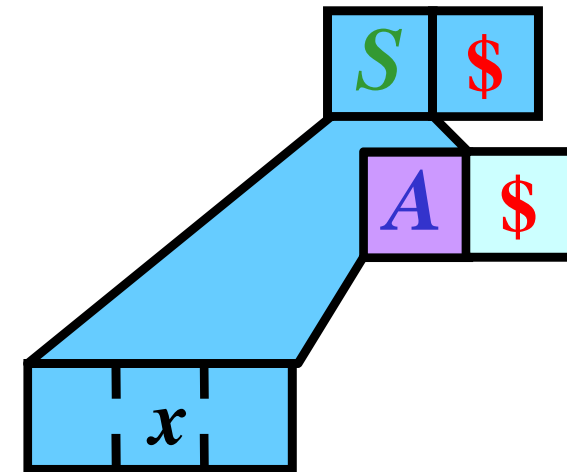
$$\text{Follow}(A) = \{a: a \in T, S \Rightarrow^* xAy, x, y \in (N \cup T)^*\} \\ \cup \{\$: S \Rightarrow^* xA, x \in (N \cup T)^*\}$$

Illustration:



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

\downarrow
 $a \in \text{Follow}(A)$



$S \Rightarrow^* xA\$$

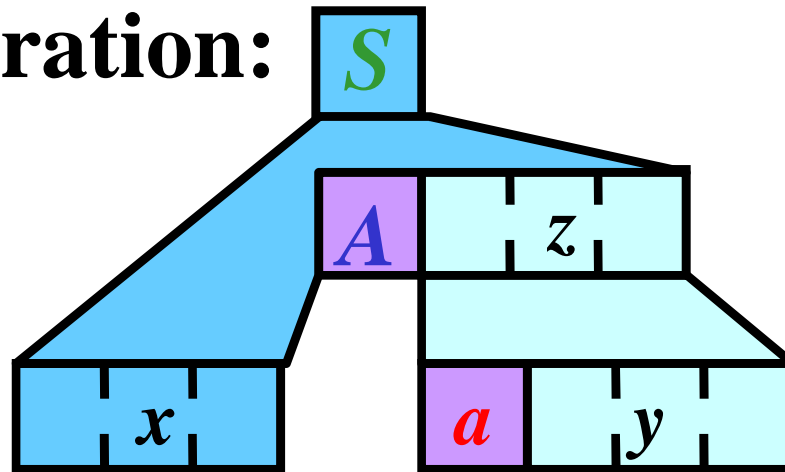
Set *Follow*

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

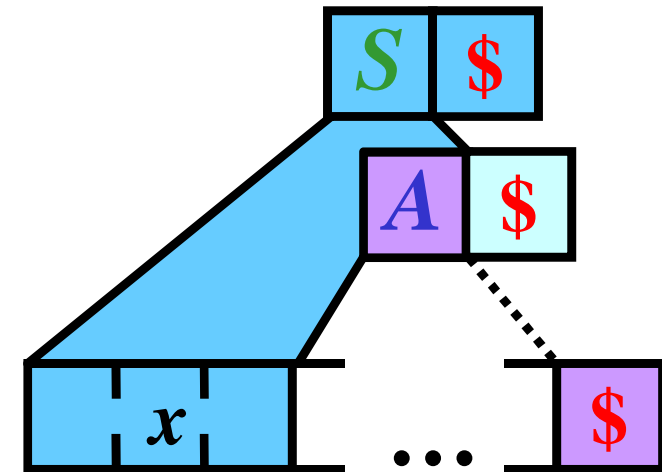
$$\text{Follow}(A) = \{a: a \in T, S \Rightarrow^* xAy, x, y \in (N \cup T)^*\} \\ \cup \{\$: S \Rightarrow^* xA, x \in (N \cup T)^*\}$$

Illustration:



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

\downarrow
 $a \in \text{Follow}(A)$



$S \Rightarrow^* xA$

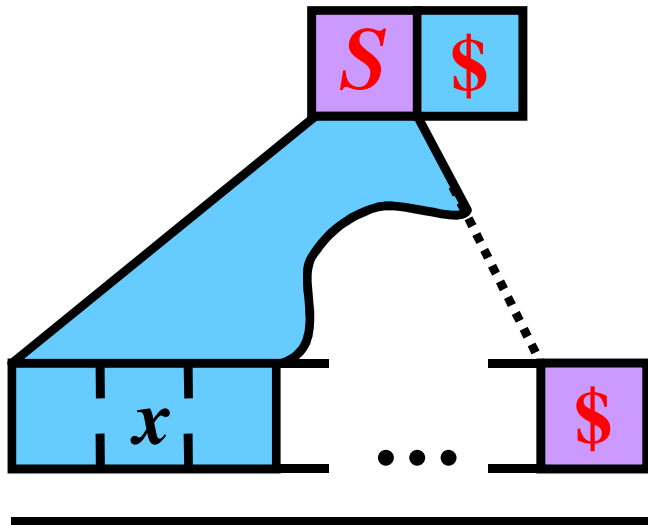
\downarrow
 $\$ \in \text{Follow}(A)$

Algorithm: *Follow*(*A*)

- **Input:** $G = (N, T, P, \mathbf{S})$;
 - **Output:** *Follow*(*A*) for every $A \in N$
-
- **Method:**
 - $\text{Follow}(\mathbf{S}) := \{\mathbf{\$}\}$;
 - Apply the following rules until no *Follow* set can be changed:
 - if $\mathbf{A} \rightarrow x\mathbf{B}y \in P$ then
 - if $y \neq \varepsilon$ then
 - add all symbols from $\text{First}(y)$ to $\text{Follow}(\mathbf{B})$;
 - if $\text{Empty}(y) = \{\varepsilon\}$ then
 - add all symbols from $\text{Follow}(\mathbf{A})$ to $\text{Follow}(\mathbf{B})$;

Previous Algorithm: Illustration

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

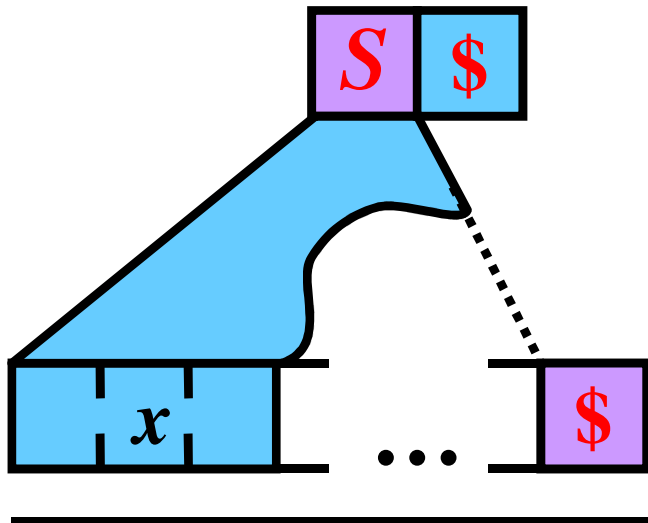


2) Apply the following rules until no *Follow* set can be changed:

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

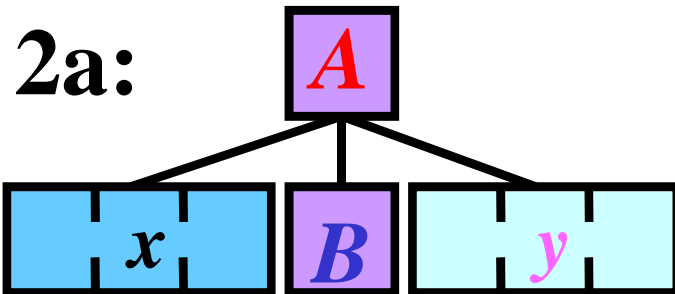
Previous Algorithm: Illustration

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



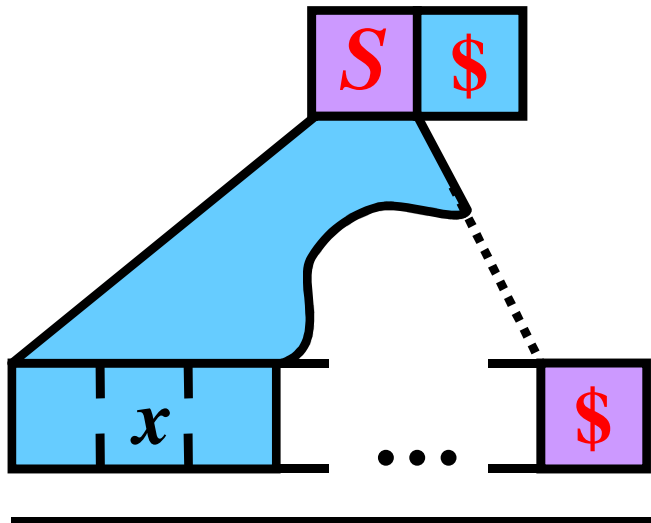
2) Apply the following rules until no *Follow* set can be changed:

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



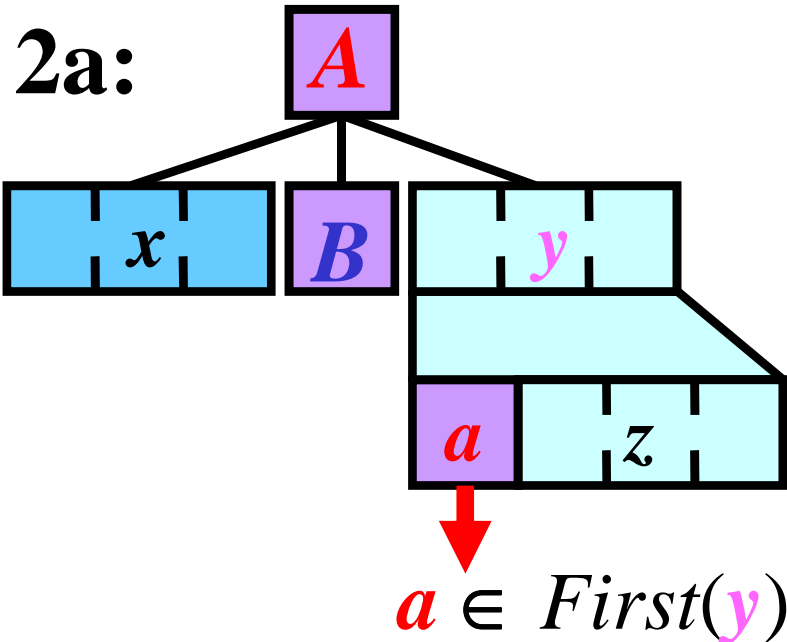
Previous Algorithm: Illustration

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



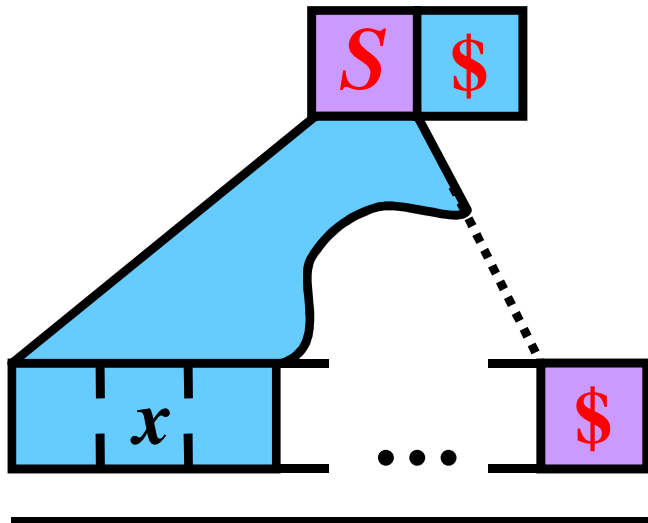
2) Apply the following rules until no *Follow* set can be changed:

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



Previous Algorithm: Illustration

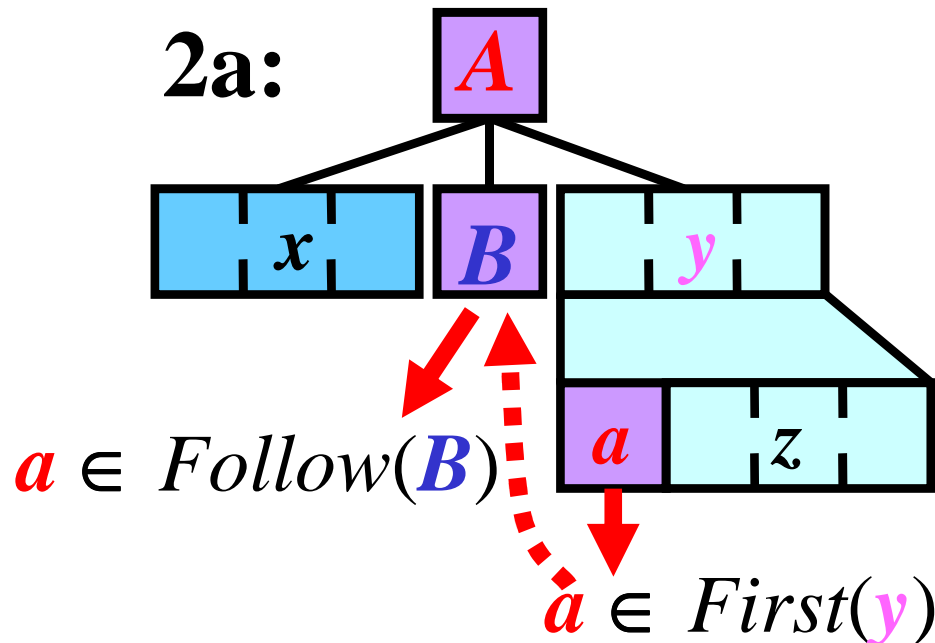
1) $Follow(\mathbf{S}) := \{\mathbf{\$}\}$



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

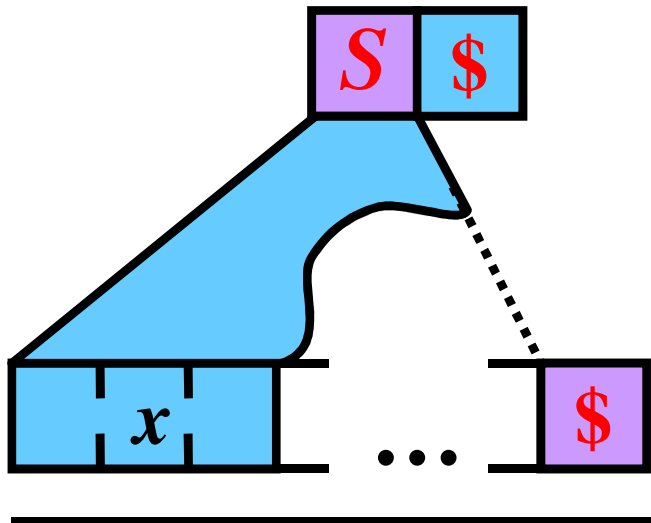
- if $\mathbf{A} \rightarrow \mathbf{xBy} \in P$ then
 - 2a) if $\mathbf{y} \neq \epsilon$ then add all symbols from $First(\mathbf{y})$ to $Follow(\mathbf{B})$

2a:



Previous Algorithm: Illustration

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



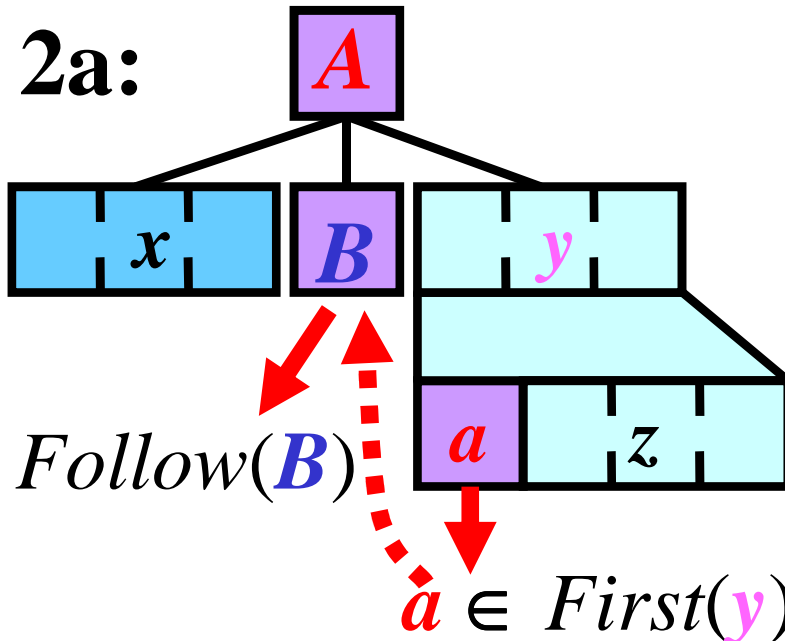
2) Apply the following rules until no *Follow* set can be changed:

- if $\mathbf{A} \rightarrow x\mathbf{B}y \in P$ then

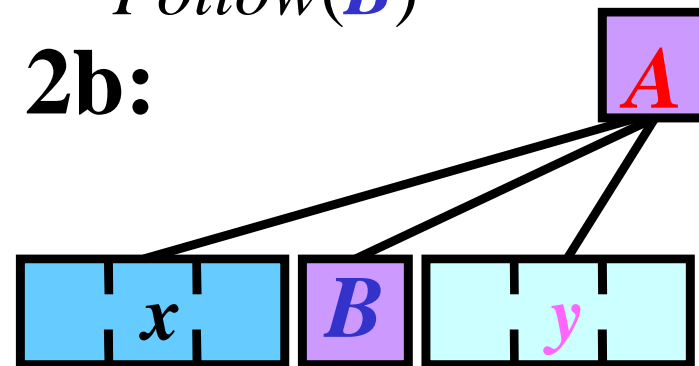
2a) if $y \neq \epsilon$ then add all symbols from $First(y)$ to $Follow(\mathbf{B})$

2b) if $Empty(y) = \{\epsilon\}$ then add all symbols from $Follow(\mathbf{A})$ to $Follow(\mathbf{B})$

2a:

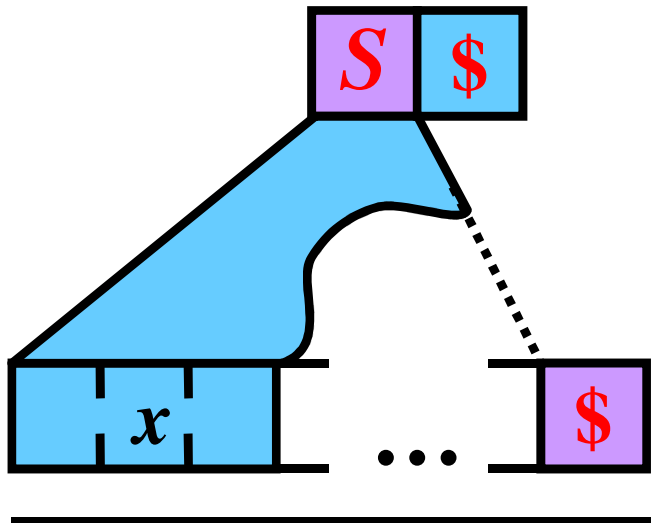


2b:



Previous Algorithm: Illustration

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



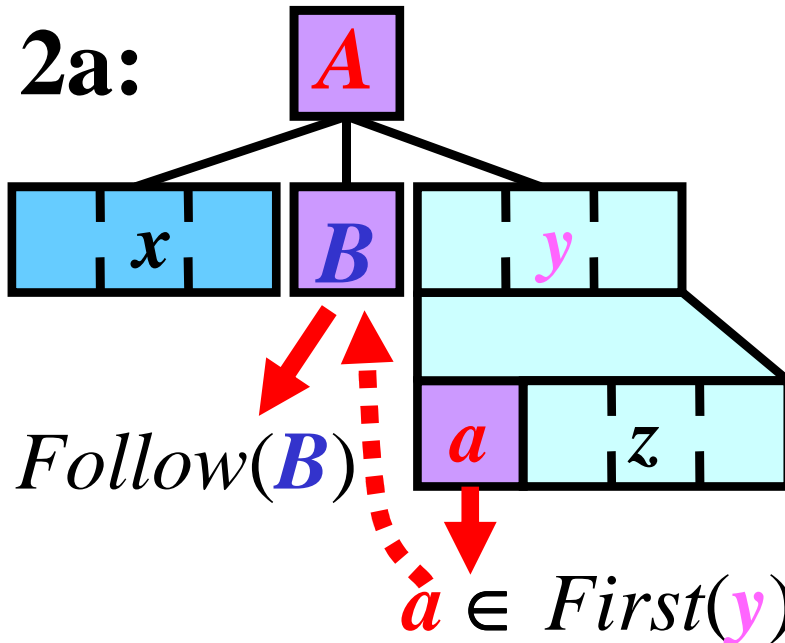
2) Apply the following rules until no *Follow* set can be changed:

- if $\mathbf{A} \rightarrow x\mathbf{B}y \in P$ then

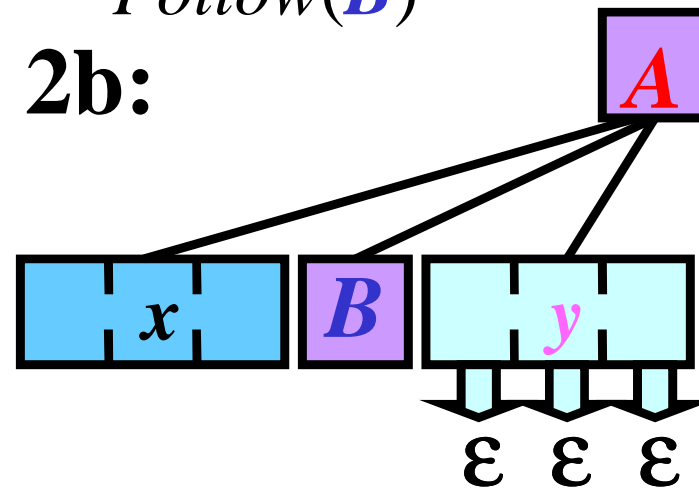
2a) if $y \neq \epsilon$ then add all symbols from $First(y)$ to $Follow(\mathbf{B})$

2b) if $Empty(y) = \{\epsilon\}$ then add all symbols from $Follow(\mathbf{A})$ to $Follow(\mathbf{B})$

2a:

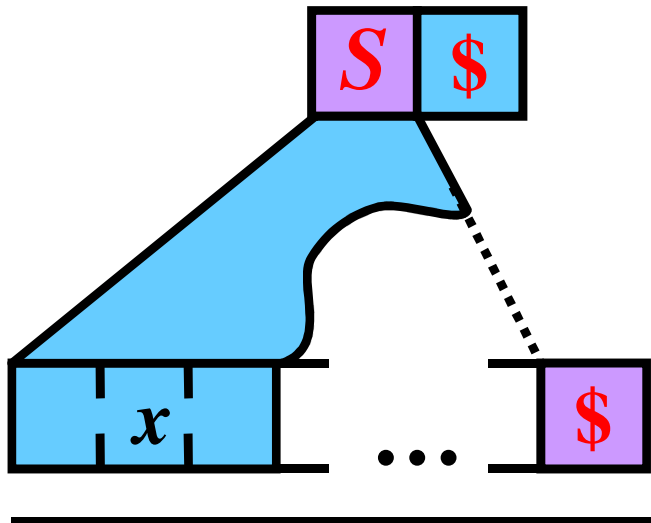


2b:



Previous Algorithm: Illustration

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



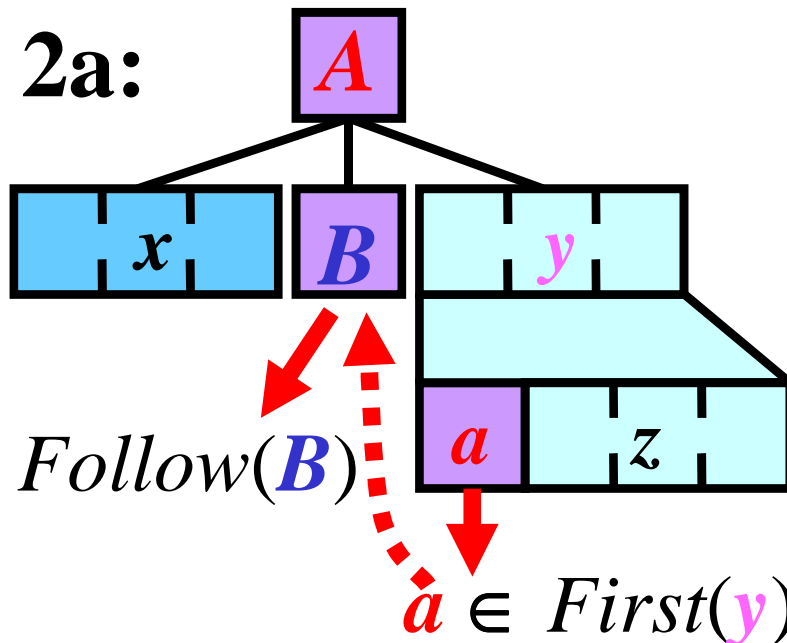
2) Apply the following rules until no *Follow* set can be changed:

- if $\mathbf{A} \rightarrow x\mathbf{B}y \in P$ then

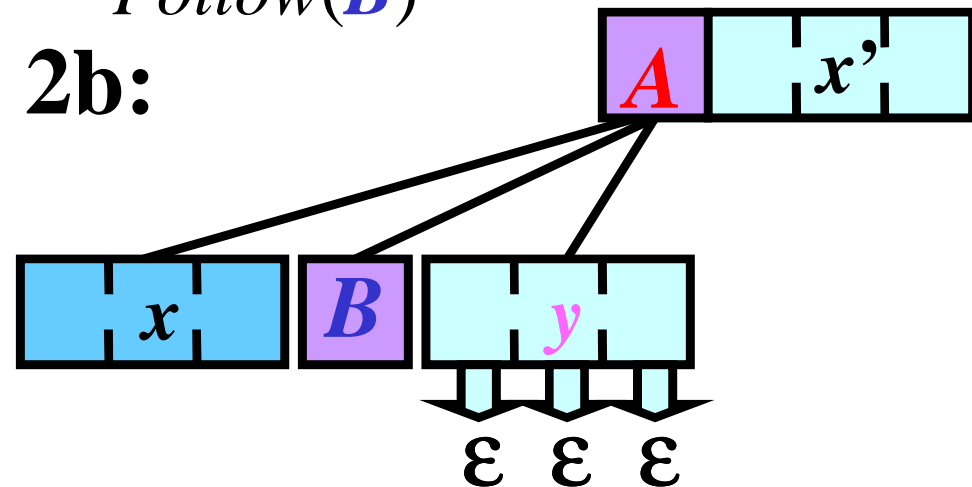
2a) if $y \neq \epsilon$ then add all symbols from $First(y)$ to $Follow(\mathbf{B})$

2b) if $Empty(y) = \{\epsilon\}$ then add all symbols from $Follow(\mathbf{A})$ to $Follow(\mathbf{B})$

2a:

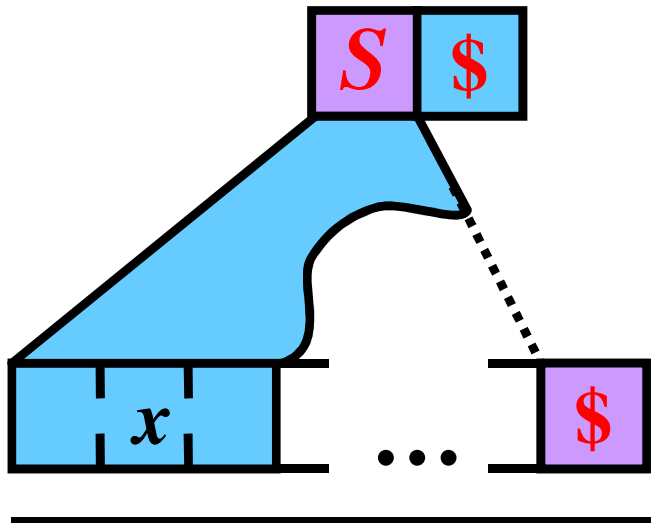


2b:



Previous Algorithm: Illustration

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



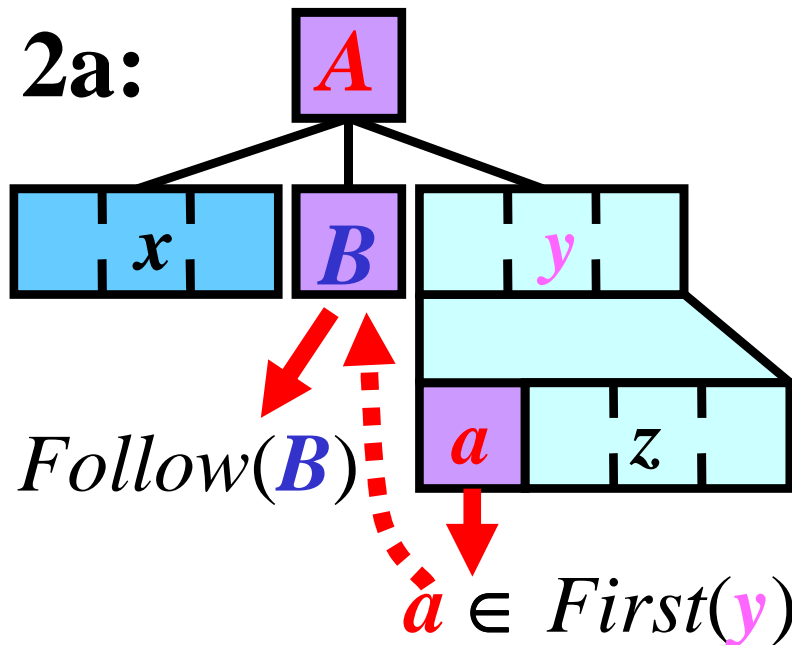
2) Apply the following rules until no *Follow* set can be changed:

- if $\mathbf{A} \rightarrow x\mathbf{B}y \in P$ then

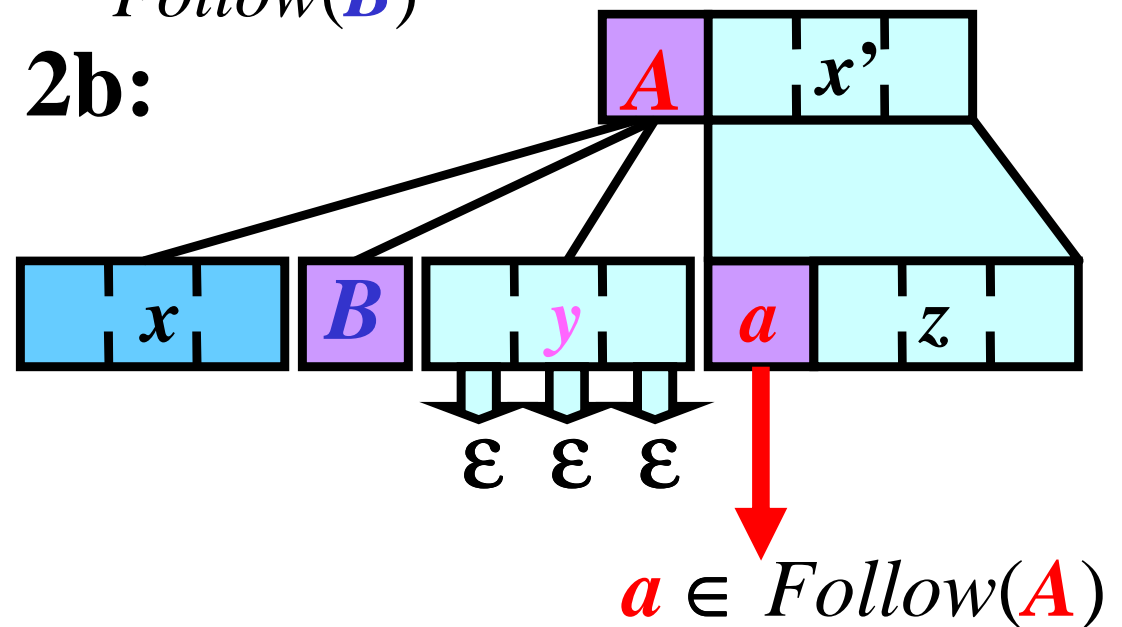
2a) if $y \neq \epsilon$ then add all symbols from $First(y)$ to $Follow(\mathbf{B})$

2b) if $Empty(y) = \{\epsilon\}$ then add all symbols from $Follow(\mathbf{A})$ to $Follow(\mathbf{B})$

2a:

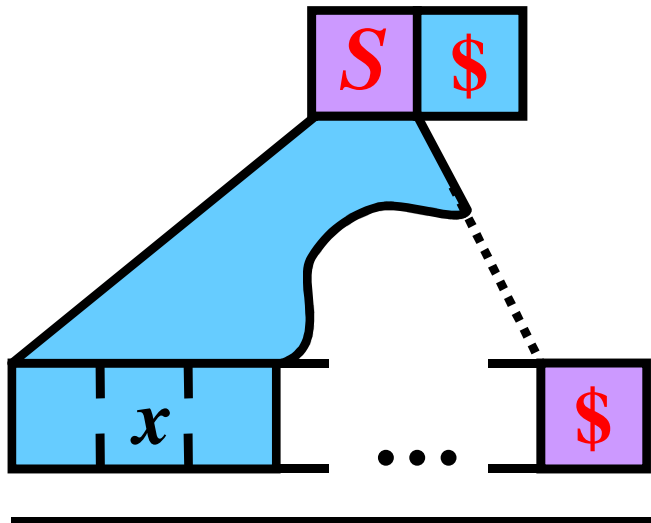


2b:



Previous Algorithm: Illustration

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



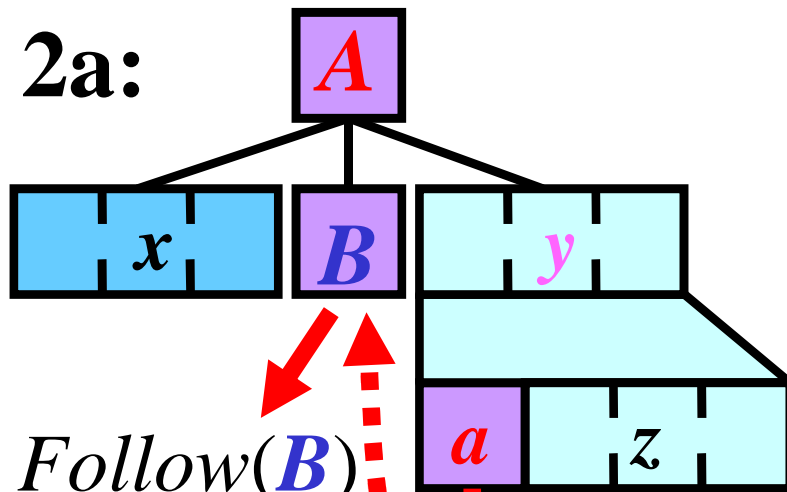
2) Apply the following rules until no *Follow* set can be changed:

- if $\mathbf{A} \rightarrow \mathbf{xBy} \in P$ then

2a) if $\mathbf{y} \neq \epsilon$ then add all symbols from $First(\mathbf{y})$ to $Follow(\mathbf{B})$

2b) if $Empty(\mathbf{y}) = \{\epsilon\}$ then add all symbols from $Follow(\mathbf{A})$ to $Follow(\mathbf{B})$

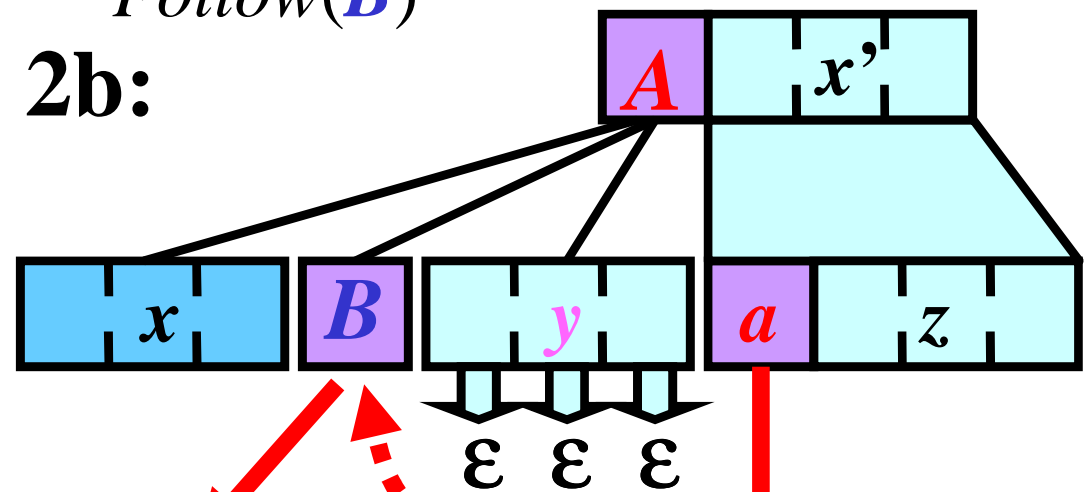
2a:



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

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

2b:



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

$a \in Follow(\mathbf{A})$

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

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

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


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

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

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

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

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

1) $\mathbf{F} \rightarrow (\mathbf{E}) \in P$:

 $\neq \epsilon$

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

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

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

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

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

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

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

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


 $\neq \varepsilon$

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

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

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

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

$$1) \text{ **F** } \rightarrow (\text{ **E** }) \in P: \quad \text{add First}(\text{ **E** }) = \{ \} \quad \text{to Follow}(\text{ **E** })$$

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

$$2) \textcolor{red}{E} \rightarrow T\textcolor{blue}{E}' \textcolor{green}{\sqcup} \in P: \\ \textcolor{green}{\varepsilon}: \textit{Empty}(\textcolor{green}{\varepsilon}) = \{\varepsilon\}$$

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

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

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

$$1) \text{ **F** } \rightarrow (\text{ **E** }) \in P: \quad \text{add First}(\text{ **E** }) = \{ \} \quad \text{to Follow}(\text{ **E** })$$

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

$$2) \text{ } \mathbf{E} \rightarrow T\mathbf{E}' \quad \mathbf{E}' \in P: \quad \mathbf{add} \text{ } Follow(\mathbf{E}) = \{\$, \text{)}\} \text{ to } Follow(\mathbf{E}') \\ \quad \quad \quad \mathbf{\epsilon}: Empty(\mathbf{\epsilon}) = \{\epsilon\}$$

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

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

$$\mathbf{0}) \textit{Follow}(\textcolor{blue}{E}) := \{\textcolor{red}{\$}\}$$
$$1) \text{ **F** } \rightarrow (\underbrace{\text{**E**}}_{\neq \varepsilon}) \in P: \quad \text{add } First() = \{\} \quad \text{to } Follow(\text{**E**})$$

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

$$2) \text{ } \mathbf{E} \rightarrow T\mathbf{E}' \quad \mathbf{E}' \in P: \quad \mathbf{add} \text{ } Follow(\mathbf{E}) = \{\$, \text{)}\} \text{ to } Follow(\mathbf{E}') \\ \quad \quad \quad \mathbf{\epsilon}: \text{ } Empty(\mathbf{\epsilon}) = \{\epsilon\}$$
$$E \rightarrow TE' \in P:$$

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

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

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

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

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

$$2) \text{ } \mathbf{E} \rightarrow T\mathbf{E}' \quad \mathbf{E}' \in P: \quad \mathbf{add} \text{ } Follow(\mathbf{E}) = \{\$, \text{)}\} \text{ to } Follow(\mathbf{E}') \\ \quad \quad \quad \mathbf{\epsilon}: Empty(\mathbf{\epsilon}) = \{\epsilon\}$$
$$\mathbf{\textcolor{red}{E}} \rightarrow \mathbf{\textcolor{blue}{T}} \mathbf{\textcolor{green}{E'}} \in P: \quad \mathbf{add} \, First(\mathbf{\textcolor{green}{E'}}) = \{+\} \quad \mathbf{to} \, Follow(\mathbf{\textcolor{blue}{T}})$$

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

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

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

$$1) \text{ **F** } \rightarrow (\text{ **E** }) \in P: \quad \text{add First}(\text{ **E** }) = \{ \} \quad \text{to Follow}(\text{ **E** })$$

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

$$2) \text{ } \mathbf{E} \rightarrow T\mathbf{E}' \quad \mathbf{E}' \in P: \quad \mathbf{add} \text{ } Follow(\mathbf{E}) = \{\$, \text{)}\} \text{ to } Follow(\mathbf{E}') \\ \quad \quad \quad \mathbf{\epsilon}: \text{ } Empty(\mathbf{\epsilon}) = \{\epsilon\}$$
$$\mathbf{E} \rightarrow \mathbf{T}\mathbf{E}' \in P: \quad \mathbf{add} \, First(\mathbf{E}') = \{+\} \quad \mathbf{to} \, Follow(\mathbf{T})$$
$$\begin{array}{l} \textcolor{red}{E} \rightarrow \textcolor{blue}{T}\textcolor{green}{E}' \in P: \\ \textit{Empty}(\textcolor{green}{E}') = \{\varepsilon\} \end{array}$$

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

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

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

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

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

$$2) \text{ } \mathbf{E} \rightarrow T\mathbf{E}' \quad \mathbf{E}' \in P: \quad \mathbf{add} \text{ } Follow(\mathbf{E}) = \{\$, \text{)}\} \text{ to } Follow(\mathbf{E}') \\ \quad \quad \quad \mathbf{\epsilon}: Empty(\mathbf{\epsilon}) = \{\epsilon\}$$
$$\mathbf{\textcolor{red}{E}} \rightarrow \mathbf{\textcolor{blue}{T}} \mathbf{\textcolor{green}{E'}} \in P: \quad \mathbf{add} \, First(\mathbf{\textcolor{green}{E'}}) = \{+\} \quad \mathbf{to} \, Follow(\mathbf{\textcolor{blue}{T}})$$
$$\begin{aligned} \mathbf{E} \rightarrow \mathbf{T} \mathbf{E}' \in P: \quad & \mathbf{add} \text{ Follow}(\mathbf{E}) = \{\$, \text{)}\} \mathbf{to} \text{ Follow}(\mathbf{T}) \\ & \text{Empty}(\mathbf{E}') = \{\varepsilon\} \end{aligned}$$

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

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

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

$$1) \text{ **F** } \rightarrow (\text{ **E** }) \in P: \quad \text{add First}(\text{ **E** }) = \{ \} \quad \text{to Follow}(\text{ **E** })$$

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

$$2) \text{ } \mathbf{E} \rightarrow T\mathbf{E}' \quad \mathbf{E}' \in P: \quad \mathbf{add} \text{ } Follow(\mathbf{E}) = \{\$, \text{)}\} \text{ to } Follow(\mathbf{E}') \\ \quad \quad \quad \mathbf{\epsilon}: \text{ } Empty(\mathbf{\epsilon}) = \{\epsilon\}$$
$$\mathbf{E} \rightarrow \mathbf{T}\mathbf{E}' \in P: \quad \mathbf{add} \, First(\mathbf{E}') = \{+\} \quad \mathbf{to} \, Follow(\mathbf{T})$$
$$\begin{aligned} \mathbf{E} \rightarrow \mathbf{T}\mathbf{E}' \in P: & \quad \mathbf{add} \text{ Follow}(\mathbf{E}) = \{\$, \text{)}\} \text{ to } \text{Follow}(\mathbf{T}) \\ & \quad \text{Empty}(\mathbf{E}') = \{\varepsilon\} \end{aligned}$$

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

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

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

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

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

3) $\mathbf{E}' \rightarrow +\mathbf{T}\mathbf{E}' \in P$: **add** $Follow(\mathbf{E}') = \{\$,)\}$ **to** $Follow(\mathbf{E}')$
 $\quad \quad \quad \epsilon: Empty(\epsilon) = \{\epsilon\}$

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

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

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

Summary: Nothing is changed

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

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

3) $\mathbf{E}' \rightarrow +\mathbf{T}\mathbf{E}' \in P$: **add** $Follow(\mathbf{E}') = \{\$,)\}$ **to** $Follow(\mathbf{E}')$
 $\quad \quad \quad \epsilon: Empty(\epsilon) = \{\epsilon\}$

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

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

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

Summary: Nothing is changed

4) $\mathbf{T} \rightarrow \mathbf{F}\mathbf{T}' \in P$: **add** $Follow(\mathbf{T}) = \{+, \$,)\}$ **to** $Follow(\mathbf{T}')$
 $\quad \quad \quad \epsilon: Empty(\epsilon) = \{\epsilon\}$

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

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

$Empty(\mathbf{T}') = \{\epsilon\}$

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

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

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

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

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

5) $\mathbf{T}' \rightarrow * \mathbf{F} \mathbf{T}' \in P$: **add** $Follow(\mathbf{T}') = \{+, \$,)\}$ **to** $Follow(\mathbf{T}')$
 $\epsilon: Empty(\epsilon) = \{\epsilon\}$

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

$\mathbf{T}' \rightarrow * \mathbf{F} \mathbf{T}' \neq \epsilon \in P$: **add** $Follow(\mathbf{T}') = \{+, \$,)\}$ **to** $Follow(\mathbf{F})$
 $Empty(\mathbf{T}') = \{\epsilon\}$

End: Nothing is changed.

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

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

5) $\mathbf{T}' \rightarrow * \mathbf{F} \mathbf{T}' \in P$: **add** $Follow(\mathbf{T}') = \{+, \$,)\}$ **to** $Follow(\mathbf{T}')$
 $\epsilon: Empty(\epsilon) = \{\epsilon\}$

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

$\mathbf{T}' \rightarrow * \mathbf{F} \mathbf{T}' \neq \epsilon \in P$: **add** $Follow(\mathbf{T}') = \{+, \$,)\}$ **to** $Follow(\mathbf{F})$
 $Empty(\mathbf{T}') = \{\epsilon\}$

End: Nothing is changed.

Summary:

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

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

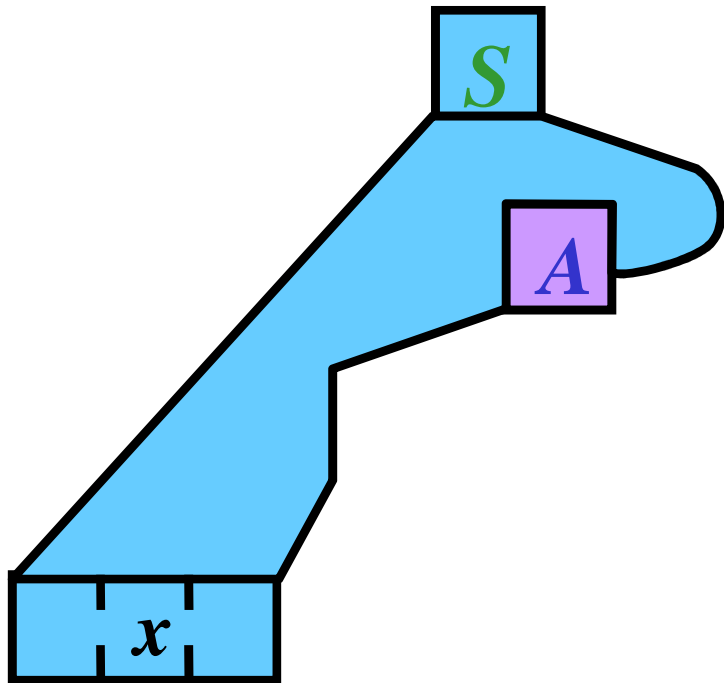
Set *Predict*($A \rightarrow X_1X_2\dots X_n$): Illustration

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

⋮

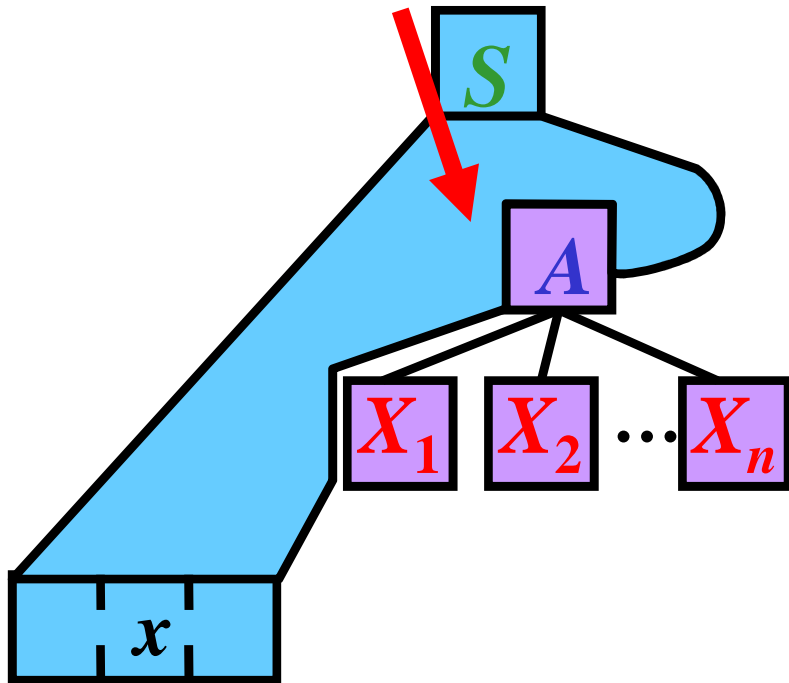
Set *Predict*($A \rightarrow X_1X_2\dots X_n$): Illustration

$\text{Empty}(\mathbf{X_1X_2\dots X_n}) = \emptyset$ vs. $\text{Empty}(\mathbf{X_1X_2\dots X_n}) = \{\epsilon\}$



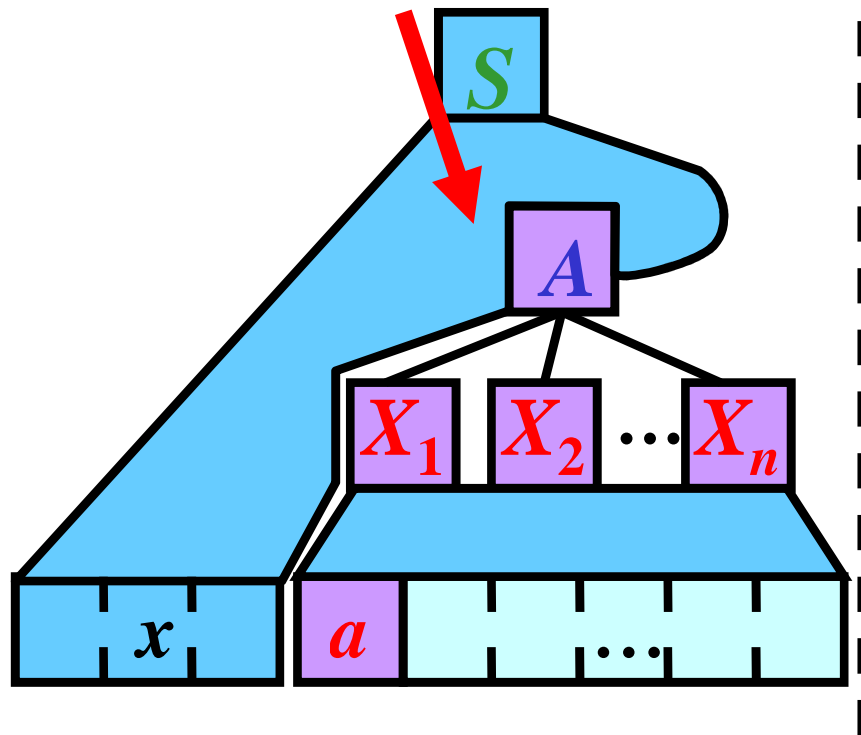
Set $Predict(A \rightarrow X_1X_2...X_n)$: Illustration

$\underbrace{Empty(\mathbf{X_1X_2...X_n}) = \emptyset}_{\text{red brace}} \text{ vs. } Empty(\mathbf{X_1X_2...X_n}) = \{\epsilon\}$



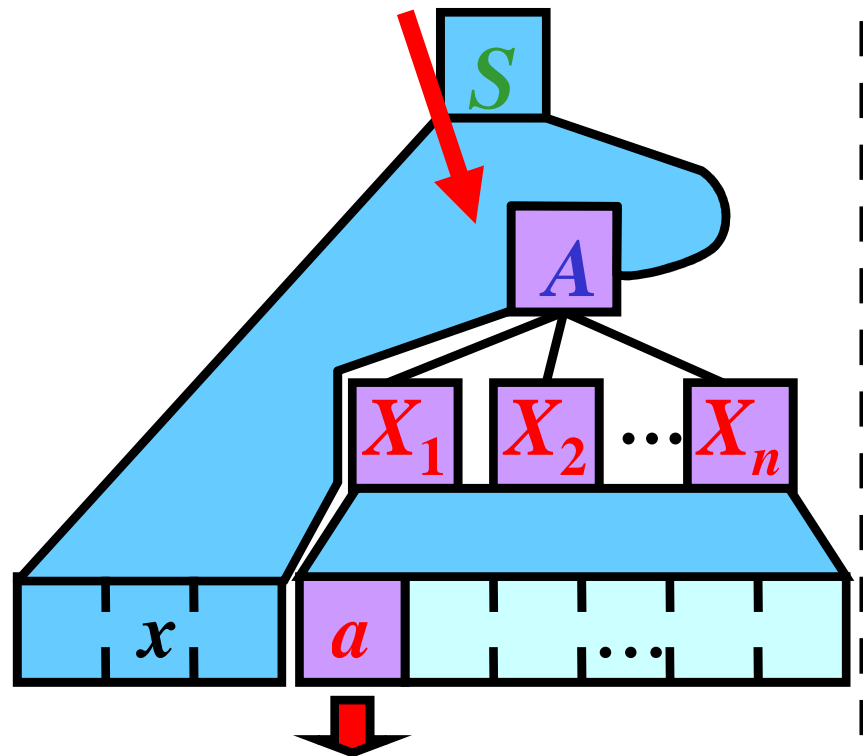
Set $Predict(A \rightarrow X_1X_2...X_n)$: Illustration

$\underbrace{Empty(\mathbf{X_1X_2...X_n}) = \emptyset}_{\text{red bracket}} \text{ vs. } Empty(\mathbf{X_1X_2...X_n}) = \{\epsilon\}$



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

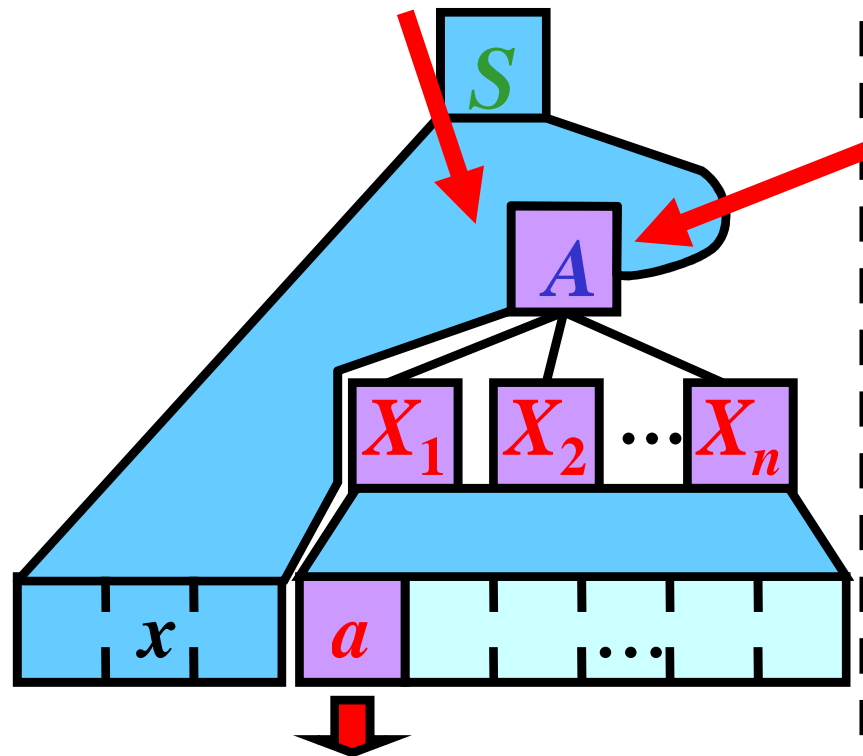
$\underbrace{Empty(\mathbf{X_1X_2...X_n}) = \emptyset}_{\text{red brace}} \text{ vs. } Empty(\mathbf{X_1X_2...X_n}) = \{\epsilon\}$



$\mathbf{a} \in First(\mathbf{X_1X_2...X_n})$

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

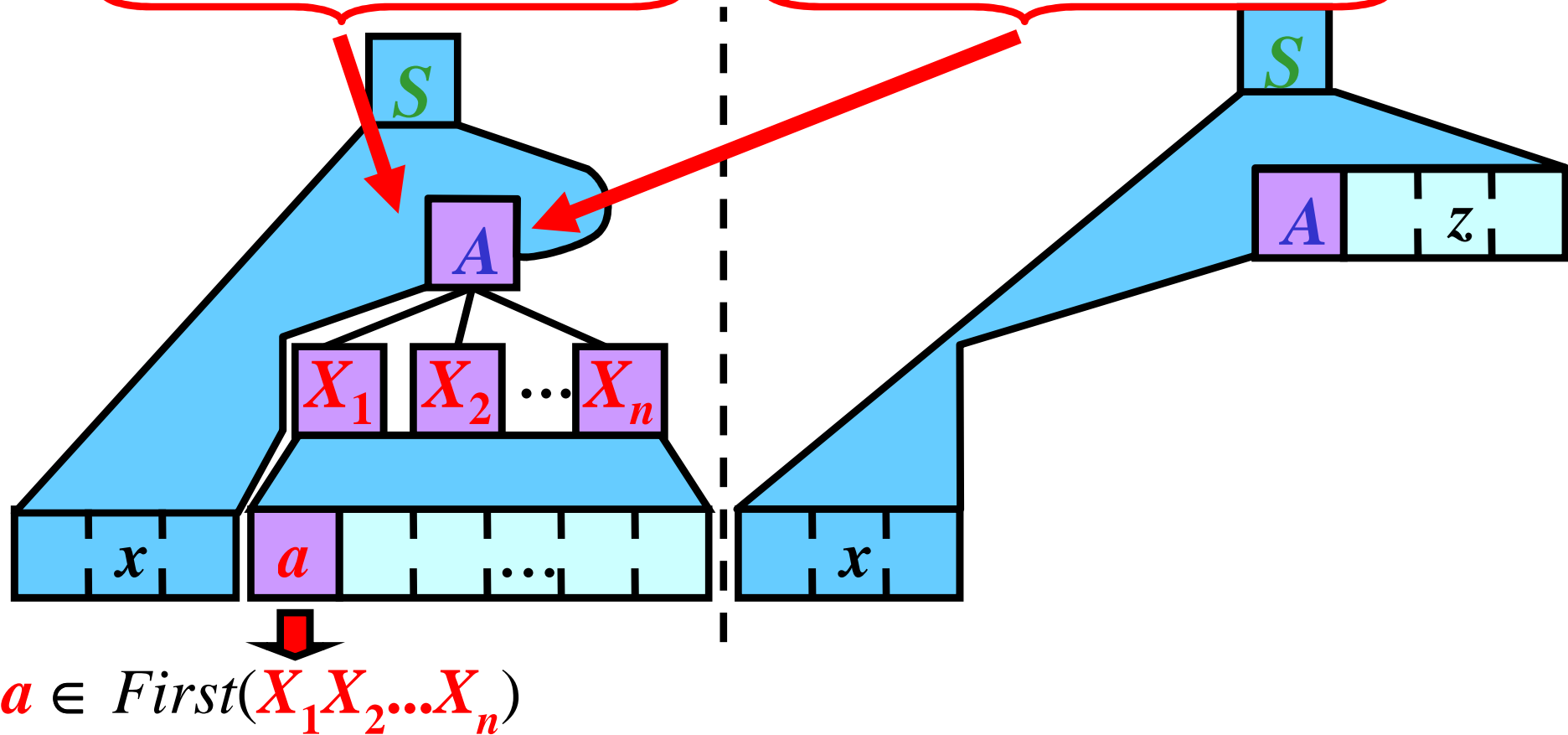
$\underbrace{Empty(X_1X_2...X_n) = \emptyset}_{\text{Left side}} \text{ vs. } \underbrace{Empty(X_1X_2...X_n) = \{\epsilon\}}_{\text{Right side}}$



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

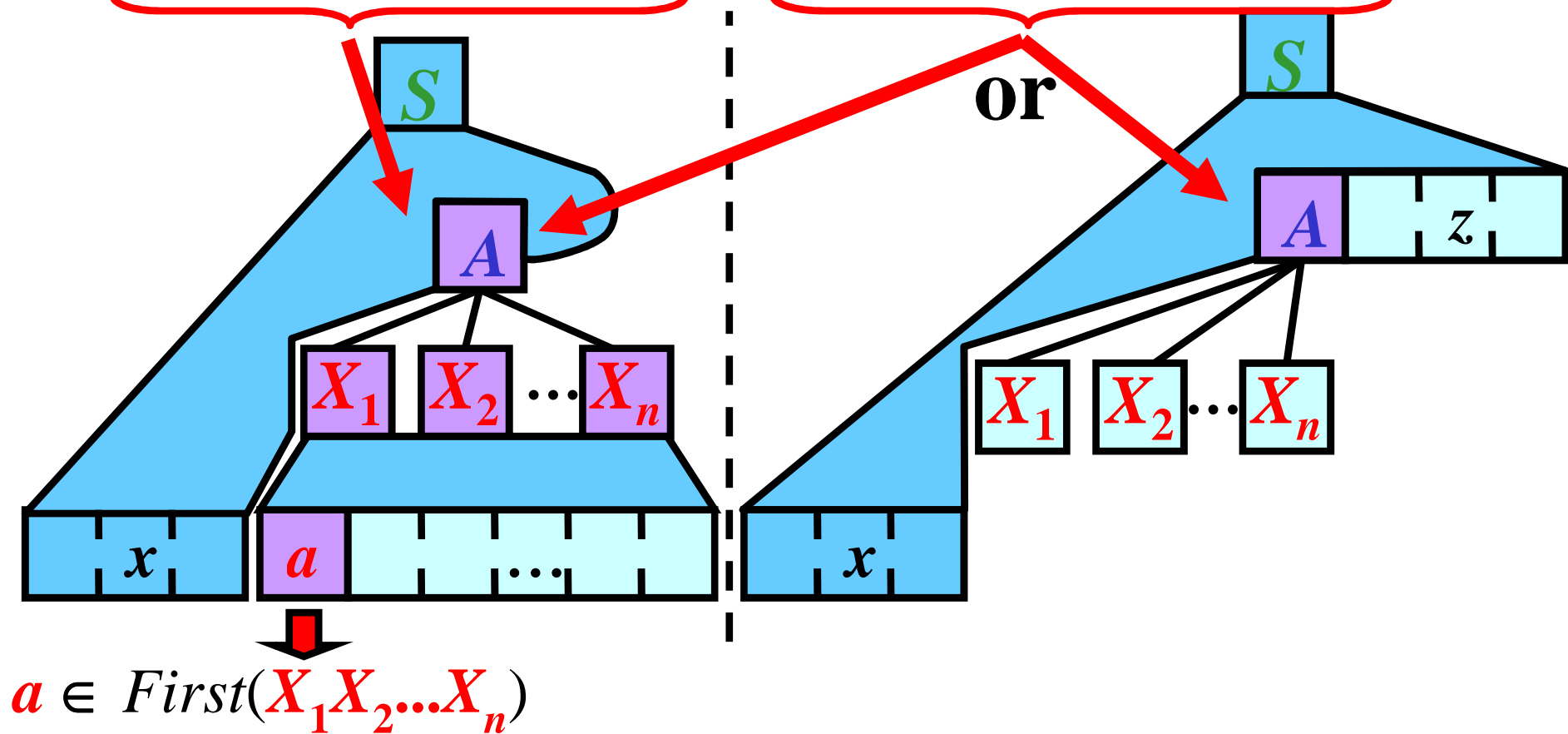
Set $Predict(A \rightarrow X_1X_2...X_n)$: Illustration

$\underbrace{Empty(X_1X_2...X_n) = \emptyset}_{\text{Left}} \text{ vs. } \underbrace{Empty(X_1X_2...X_n) = \{\epsilon\}}_{\text{Right}}$



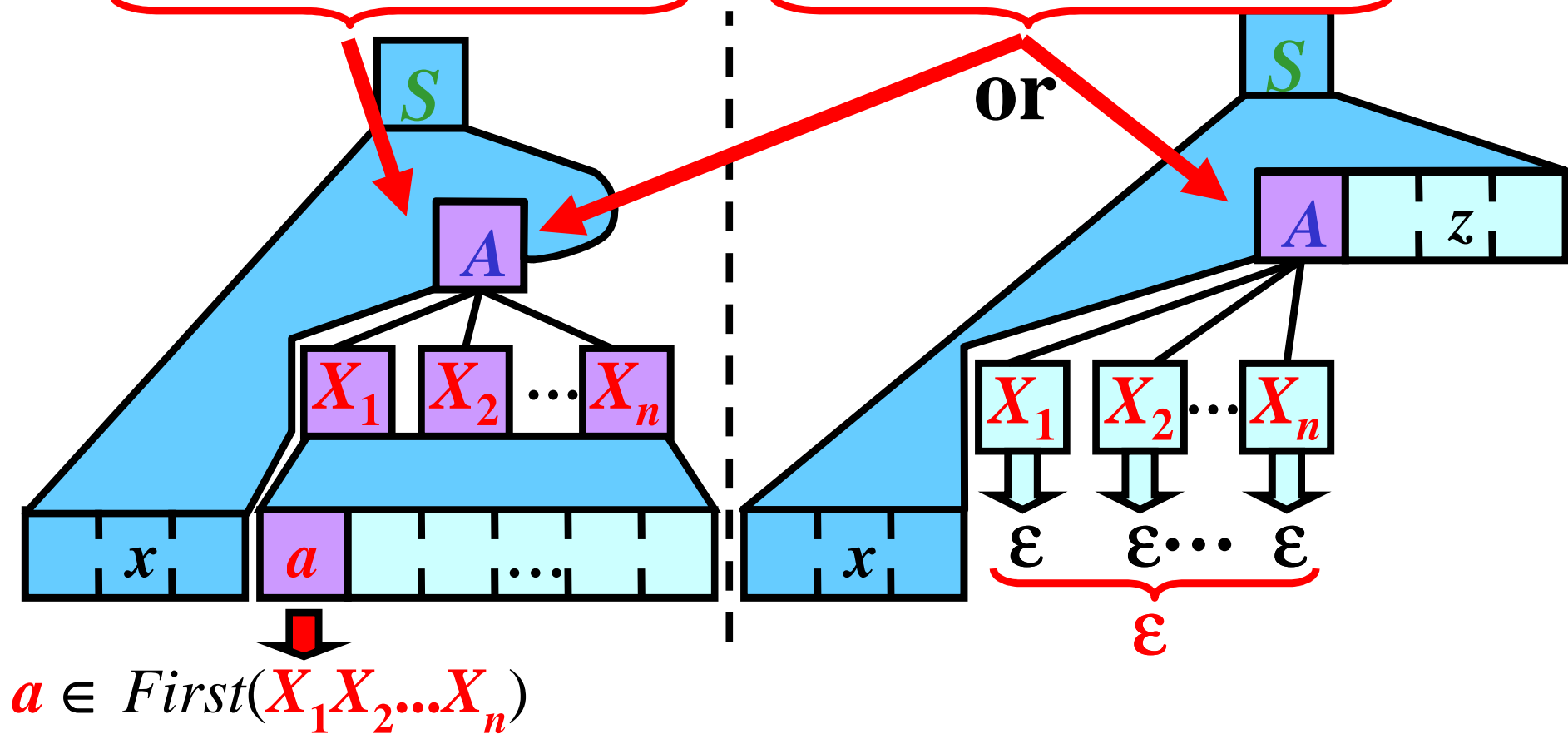
Set $Predict(A \rightarrow X_1X_2...X_n)$: Illustration

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



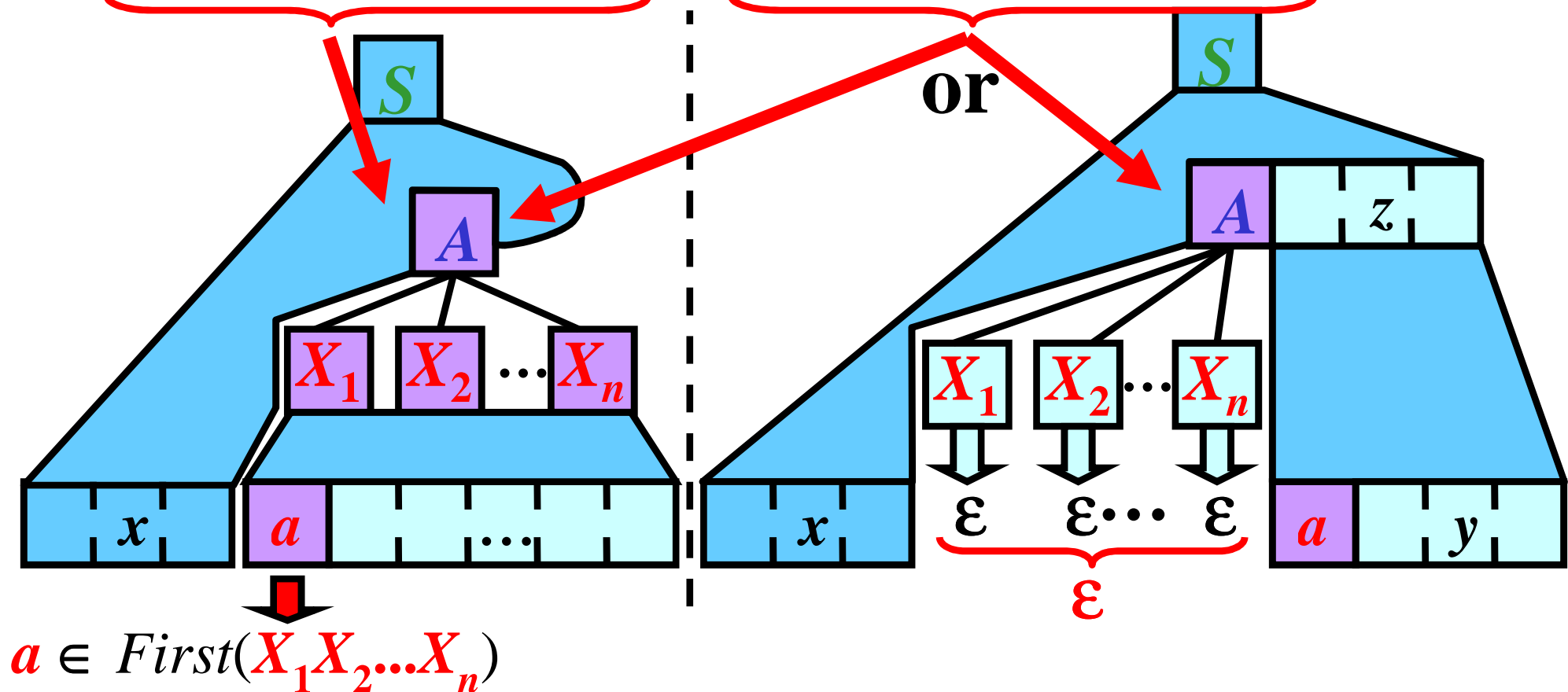
Set $Predict(A \rightarrow X_1X_2...X_n)$: Illustration

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

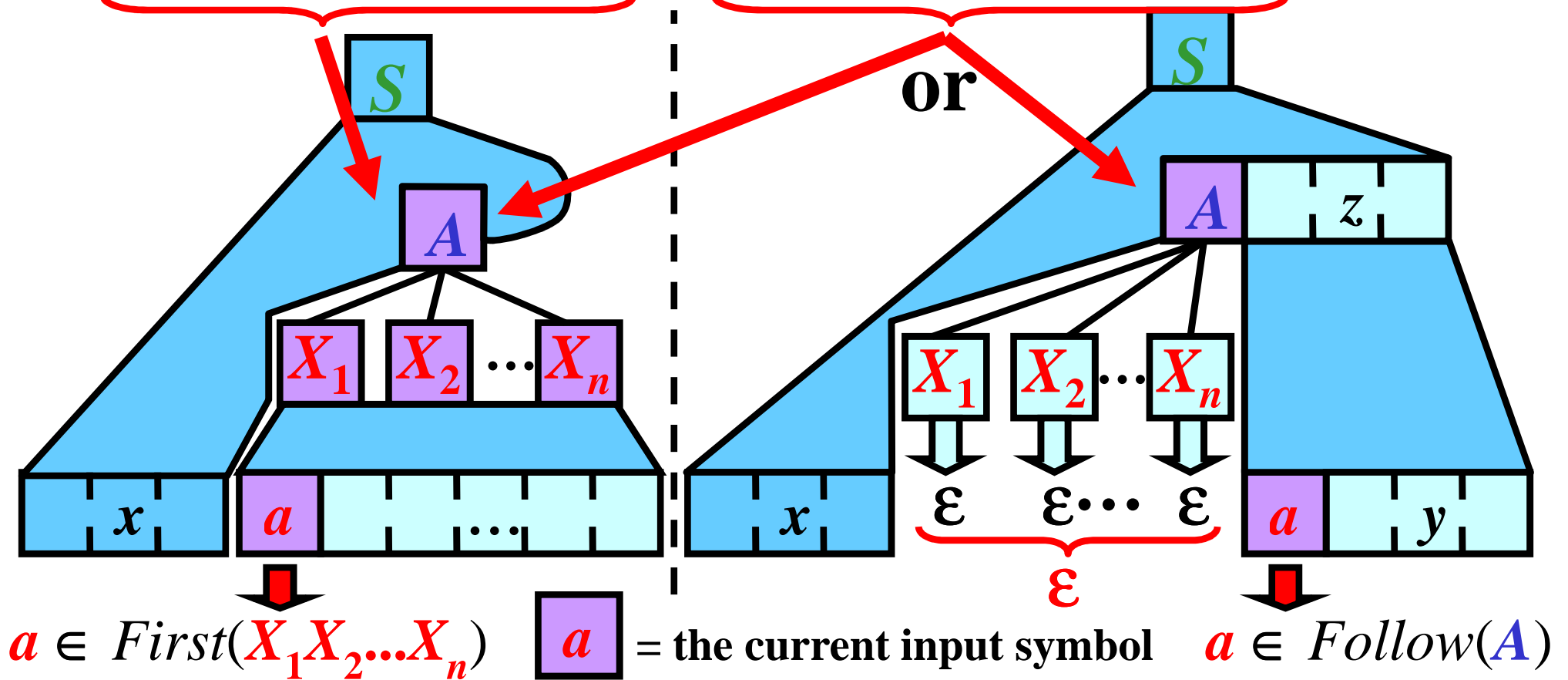


Set $Predict(A \rightarrow X_1X_2...X_n)$: Illustration

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

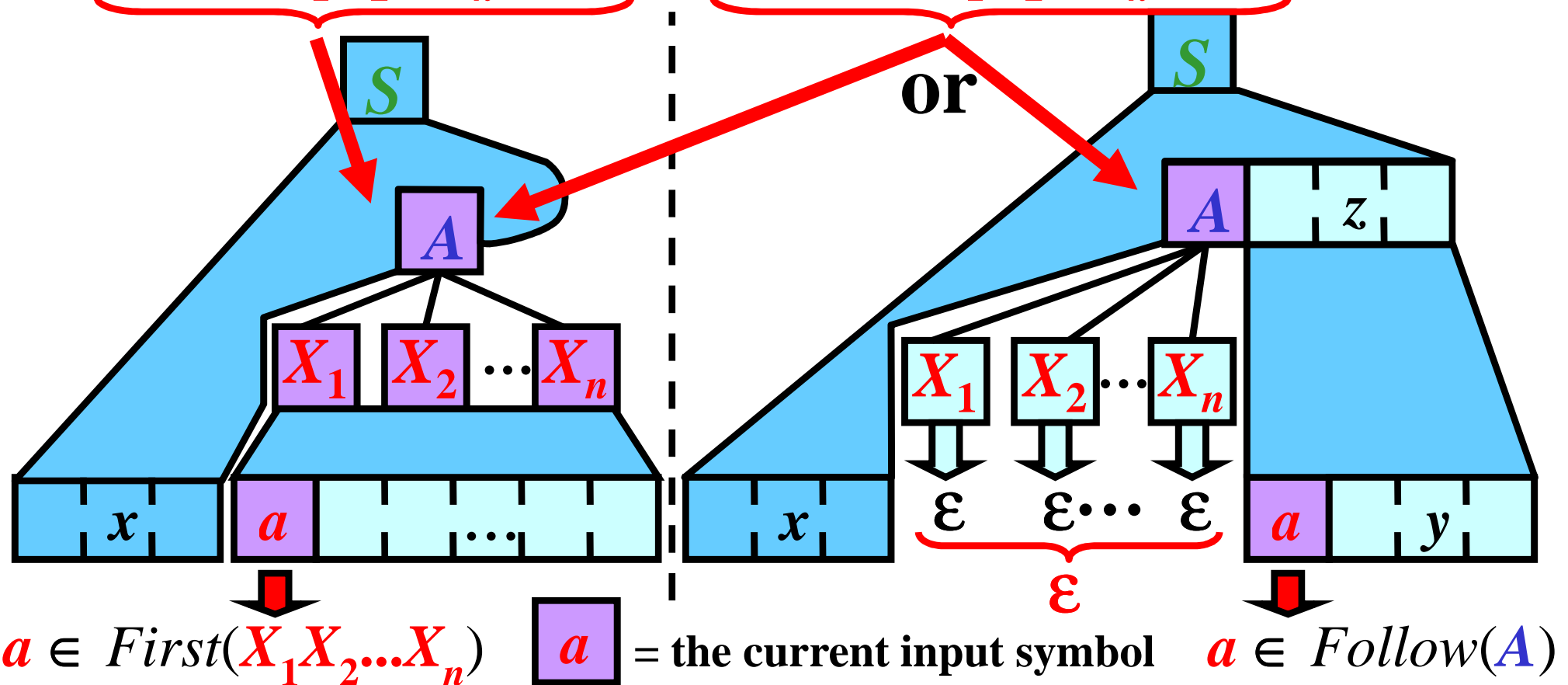


Set $Predict(A \rightarrow X_1X_2...X_n)$: Illustration

$$Empty(\mathbf{X}_1\mathbf{X}_2\cdots\mathbf{X}_n) = \emptyset \text{ vs. } Empty(\mathbf{X}_1\mathbf{X}_2\cdots\mathbf{X}_n) = \{\varepsilon\}$$


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

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



Summary: if $Empty(X_1X_2...X_n) = \{\epsilon\}$ then

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

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

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

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

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

1: $\mathbf{E} \rightarrow \mathbf{T}\mathbf{E}'$

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

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

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

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

1: $\mathbf{E} \rightarrow \mathbf{T}\mathbf{E}'$

$\text{Empty}(\mathbf{T}\mathbf{E}') = \emptyset$ because $\text{Empty}(\mathbf{T}) = \emptyset$

$\text{Predict}(\mathbf{1}) := \text{First}(\mathbf{T}\mathbf{E}') = \text{First}(\mathbf{T}) = \{\mathbf{i}, (\}$

2: $\mathbf{E}' \rightarrow +\mathbf{T}\mathbf{E}'$

$\text{Empty}(+\mathbf{T}\mathbf{E}') = \emptyset$ because $\text{Empty}(+) = \emptyset$

$\text{Predict}(\mathbf{2}) := \text{First}(+\mathbf{T}\mathbf{E}') = \text{First}(+) = \{+\}$

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

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

1: $\mathbf{E} \rightarrow \mathbf{T}\mathbf{E}'$

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

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

2: $\mathbf{E}' \rightarrow +\mathbf{T}\mathbf{E}'$

$Empty(+\mathbf{T}\mathbf{E}') = \emptyset$ because $Empty(+) = \emptyset$

$Predict(\mathbf{2}) := First(+\mathbf{T}\mathbf{E}') = First(+) = \{+\}$

3: $\mathbf{E}' \rightarrow \epsilon$

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

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

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

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

1: $\mathbf{E} \rightarrow \mathbf{TE}'$

$Empty(\mathbf{TE}') = \emptyset$ because $Empty(\mathbf{T}) = \emptyset$

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

2: $\mathbf{E}' \rightarrow +\mathbf{TE}'$

$Empty(+\mathbf{TE}') = \emptyset$ because $Empty(+) = \emptyset$

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

3: $\mathbf{E}' \rightarrow \epsilon$

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

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

4: $\mathbf{T} \rightarrow \mathbf{FT}'$

$Empty(\mathbf{FT}') = \emptyset$ because $Empty(\mathbf{F}) = \emptyset$

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

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

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

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

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

5: $\mathbf{T}' \rightarrow * \mathbf{FT}'$

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

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

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

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

5: $\mathbf{T}' \rightarrow * \mathbf{FT}'$

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

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

6: $\mathbf{T}' \rightarrow \epsilon$

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

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

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

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

5: $\mathbf{T}' \rightarrow * \mathbf{F} \mathbf{T}'$

$Empty(* \mathbf{F} \mathbf{T}') = \emptyset$ because $Empty(*) = \emptyset$

$Predict(5) := First(* \mathbf{F} \mathbf{T}') = First(*) = \{*\}$

6: $\mathbf{T}' \rightarrow \epsilon$

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

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

7: $\mathbf{F} \rightarrow (\mathbf{E})$

$Empty((\mathbf{E})) = \emptyset$ because $Empty(()) = \emptyset$

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

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

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

5: $\mathbf{T}' \rightarrow * \mathbf{F} \mathbf{T}'$

$Empty(* \mathbf{F} \mathbf{T}') = \emptyset$ because $Empty(*) = \emptyset$

$Predict(5) := First(* \mathbf{F} \mathbf{T}') = First(*) = \{*\}$

6: $\mathbf{T}' \rightarrow \epsilon$

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

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

7: $\mathbf{F} \rightarrow (\mathbf{E})$

$Empty((\mathbf{E})) = \emptyset$ because $Empty(() = \emptyset$

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

8: $\mathbf{F} \rightarrow \mathbf{i}$

$Empty(\mathbf{i}) = \emptyset$

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

Construction of LL Table

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

Construction of LL Table

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

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

Construction of LL Table

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

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

Task: LL table for G_{expr1}

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

Rule r	$\text{Predict}(r)$
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		$\alpha(A, a)$	
...			

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

Task: LL table for G_{expr1}

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

$i \in \text{Predict}(1)$

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

Construction of LL Table

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

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

Task: LL table for G_{expr1}

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

Rule r	$\text{Predict}(r)$
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		$\alpha(A, a)$	
...			

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

Task: LL table for G_{expr1}

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

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

Construction of LL Table

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

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

Task: LL table for G_{expr1}

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

Construct the rest analogically.

Rule r	$\text{Predict}(r)$
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: $F \rightarrow i$	$\{i\}$

Parsing Based on LL Table: Example

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

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

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

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

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

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

E

i * *i* \$

Parsing Based on LL Table: Example

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

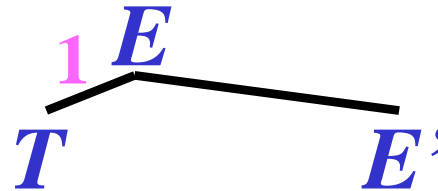
1: $E \rightarrow TE'$ 5: $T' \rightarrow *FT'$

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

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

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

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



$i * i$ \$

Parsing Based on LL Table: Example

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

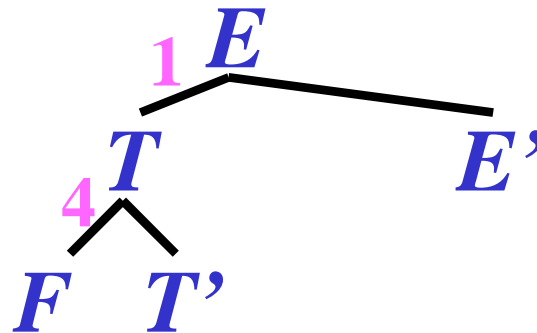
1: $E \rightarrow TE'$ 5: $T' \rightarrow *FT'$

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

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

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

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



i * *i* \$

Parsing Based on LL Table: Example

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

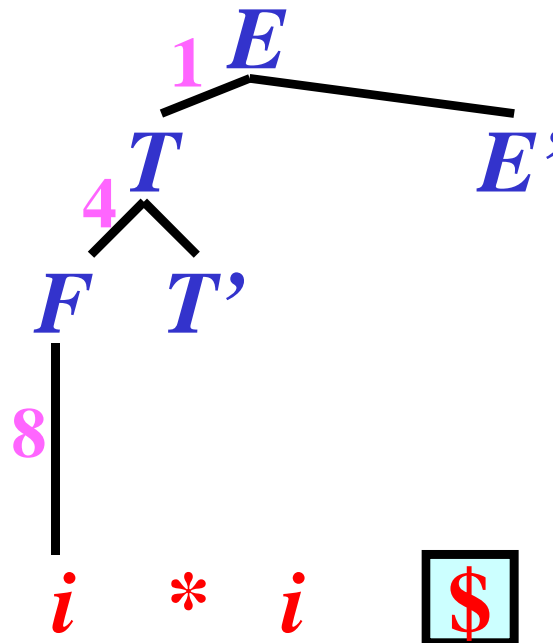
1: $E \rightarrow TE'$ 5: $T' \rightarrow *FT'$

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

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

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

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



Parsing Based on LL Table: Example

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

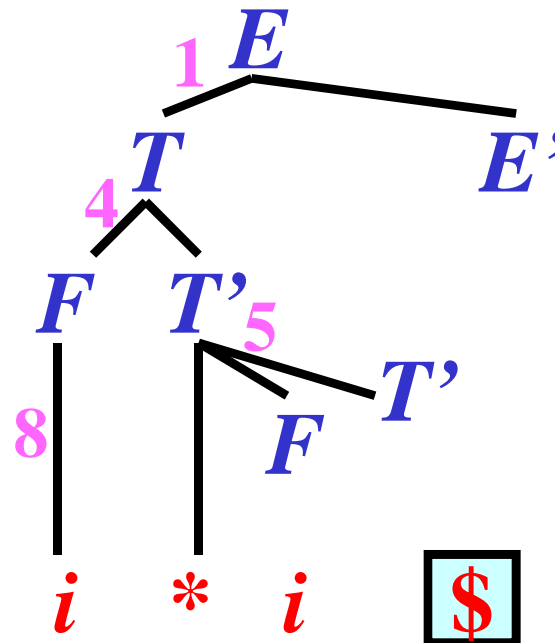
1: $E \rightarrow TE'$ 5: $T' \rightarrow *FT'$

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

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

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

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



Parsing Based on LL Table: Example

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

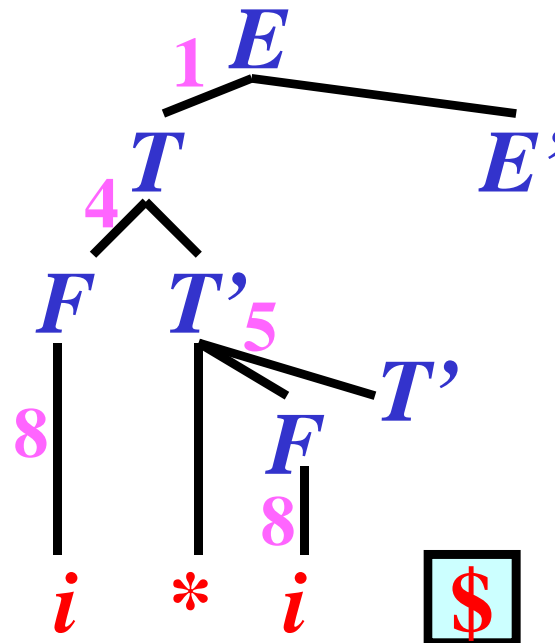
1: $E \rightarrow TE'$ 5: $T' \rightarrow *FT'$

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

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

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

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



Parsing Based on LL Table: Example

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

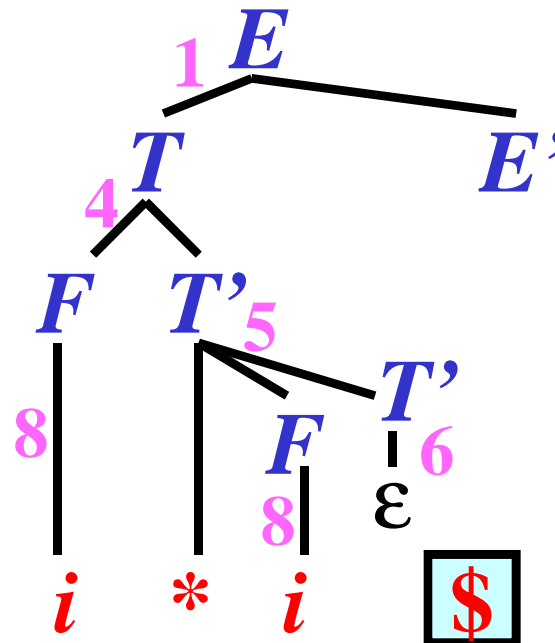
1: $E \rightarrow TE'$ 5: $T' \rightarrow *FT'$

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

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

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

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



Parsing Based on LL Table: Example

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

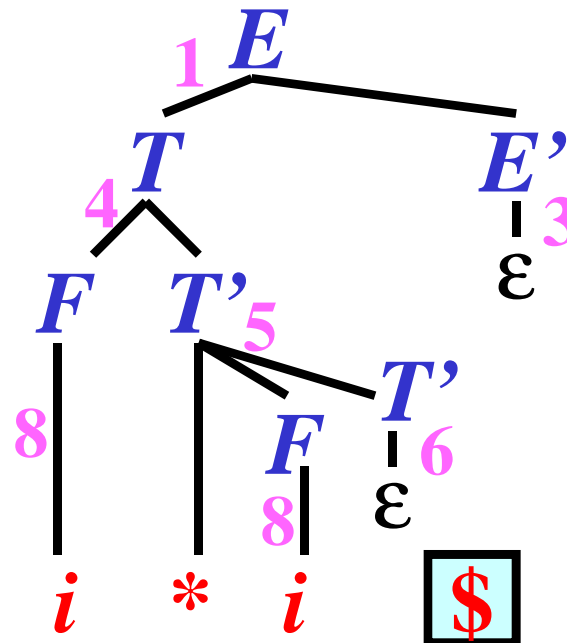
1: $E \rightarrow TE'$ 5: $T' \rightarrow *FT'$

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

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

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

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



LL Grammars with ε -rules: Definition

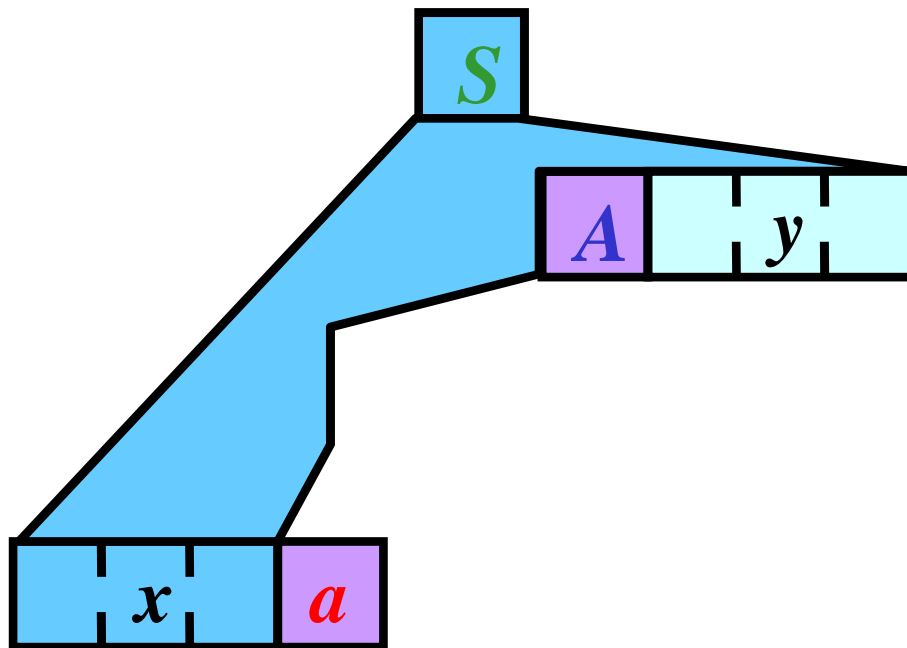
Definition: Let $G = (N, T, P, S)$ be a CFG. G is an *LL grammar* if for every $a \in T$ and every $A \in N$ there is **no more than one** A -rule $A \rightarrow X_1X_2...X_n \in P$ such that $a \in \text{Predict}(A \rightarrow X_1X_2...X_n)$

Illustration:

LL Grammars with ϵ -rules: Definition

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

Illustration:



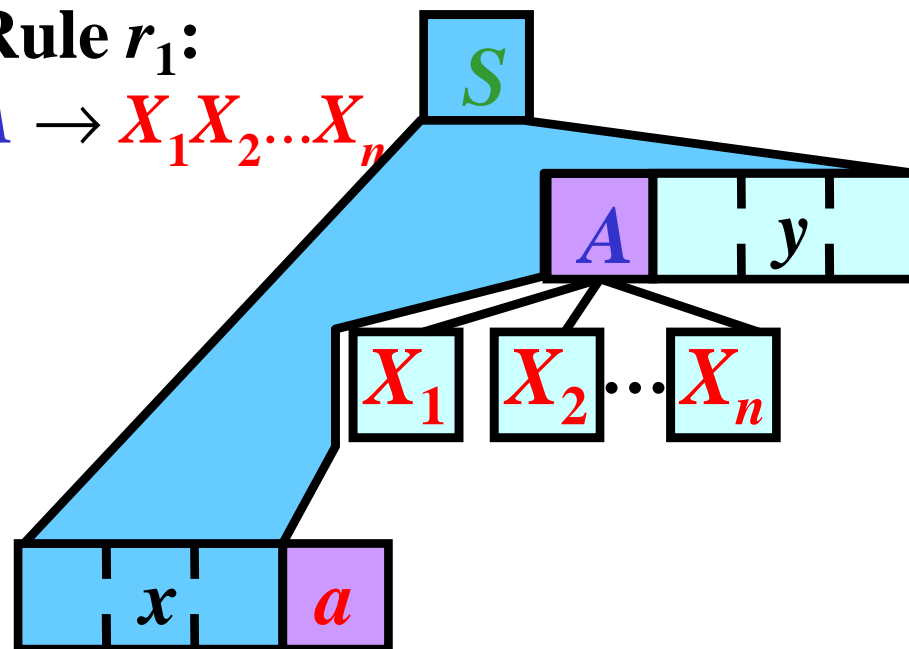
LL Grammars with ϵ -rules: Definition

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 $a \in \text{Predict}(A \rightarrow X_1X_2\dots X_n)$

Illustration:

Rule r_1 :

$A \rightarrow X_1X_2\dots X_n$



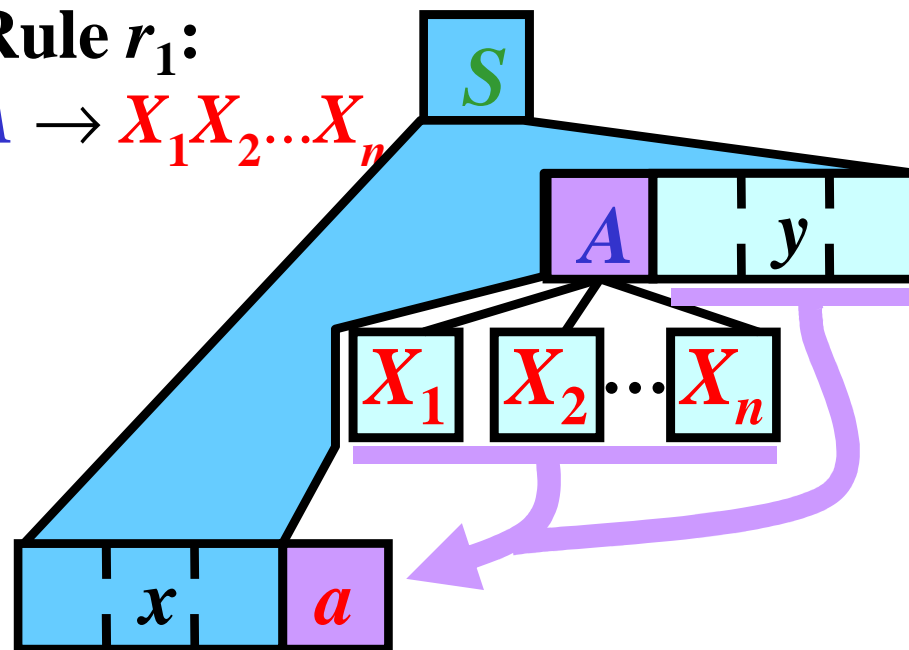
LL Grammars with ϵ -rules: Definition

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

Illustration:

Rule r_1 :

$A \rightarrow X_1X_2\dots X_n$



$a \in \text{Predict}(A \rightarrow X_1X_2\dots X_n)$

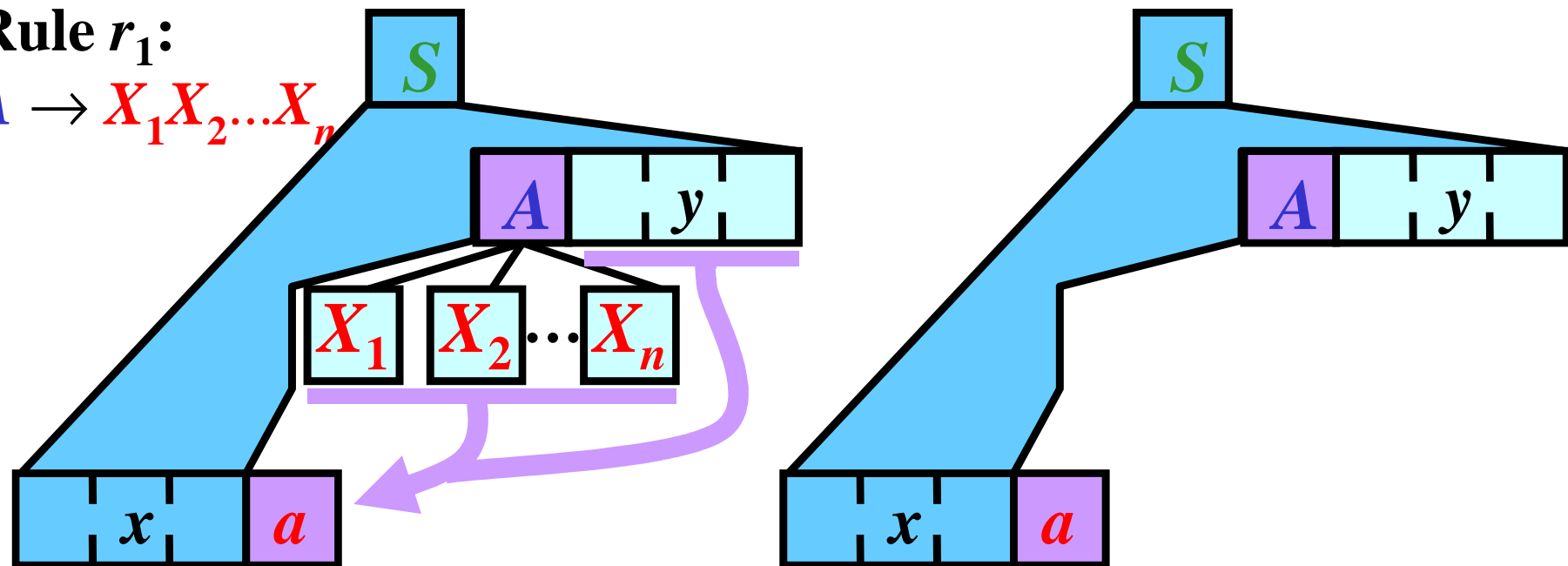
LL Grammars with ϵ -rules: Definition

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

Illustration:

Rule r_1 :

$A \rightarrow X_1X_2...X_n$



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

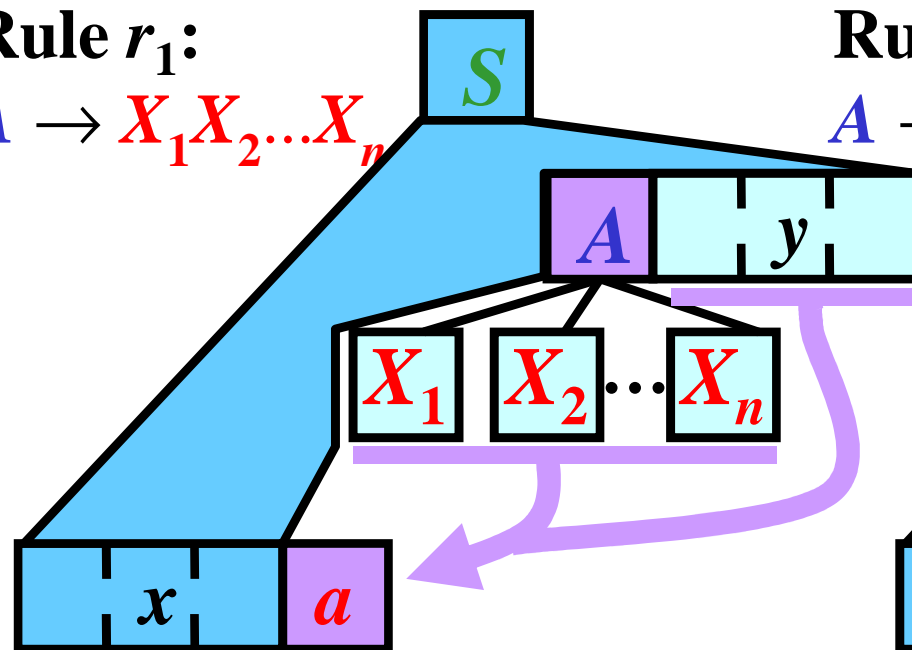
LL Grammars with ϵ -rules: Definition

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

Illustration:

Rule r_1 :

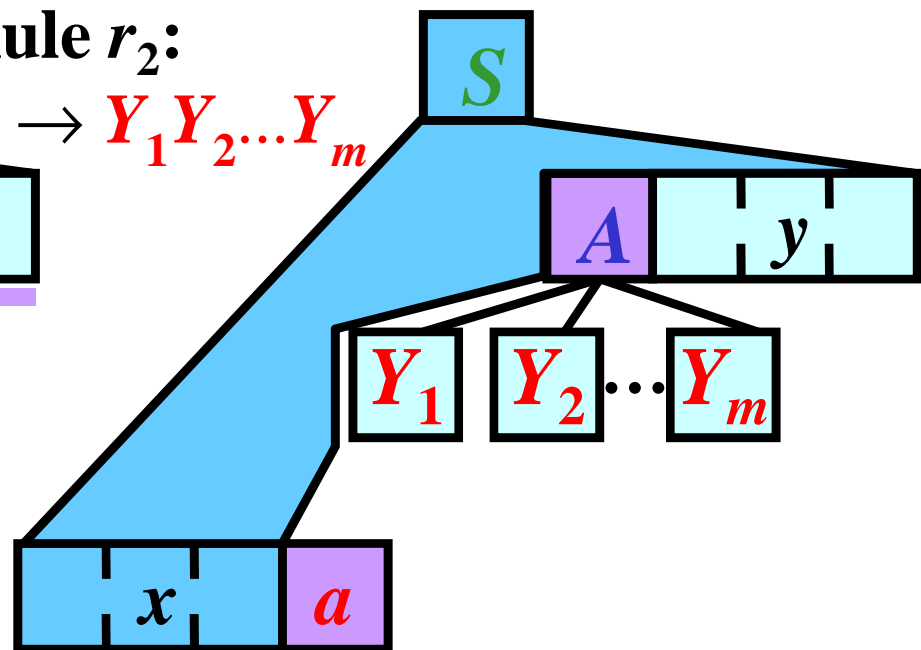
$A \rightarrow X_1X_2\dots X_n$



$a \in \text{Predict}(A \rightarrow X_1X_2\dots X_n)$

Rule r_2 :

$A \rightarrow Y_1Y_2\dots Y_m$



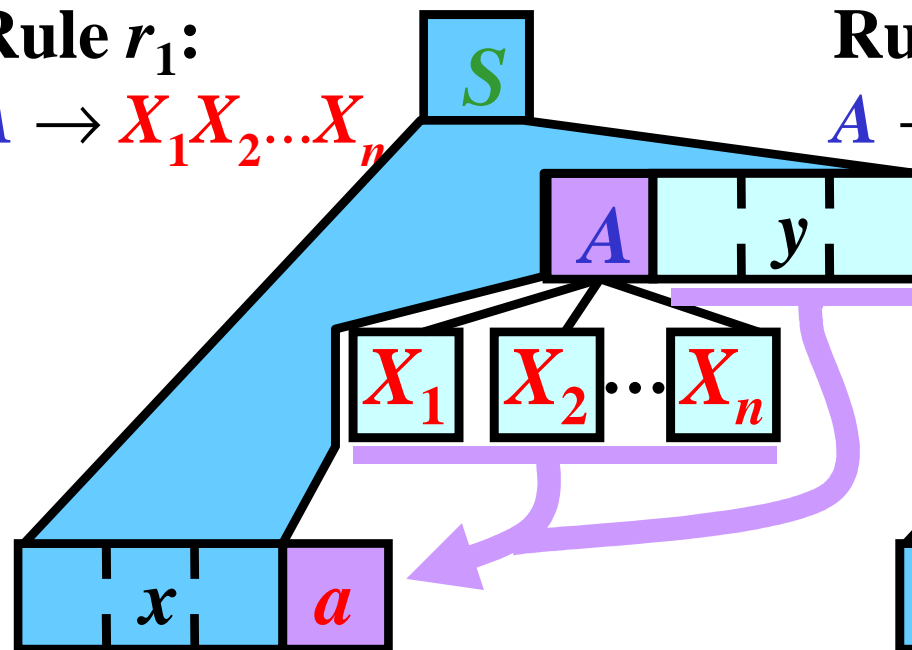
LL Grammars with ϵ -rules: Definition

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

Illustration:

Rule r_1 :

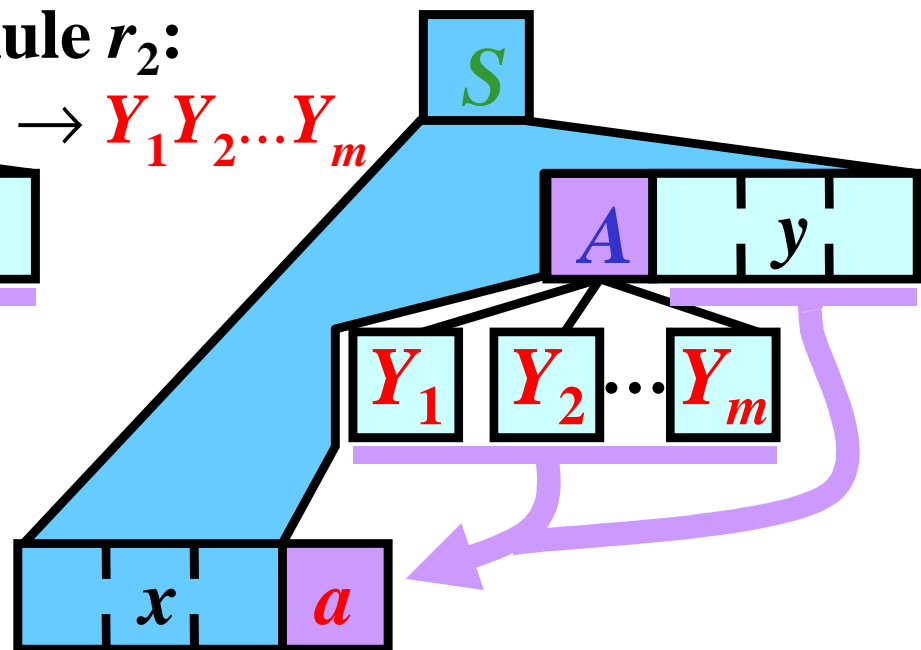
$A \rightarrow X_1X_2...X_n$



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

Rule r_2 :

$A \rightarrow Y_1Y_2...Y_m$



$a \in \text{Predict}(A \rightarrow Y_1Y_2...Y_m)$

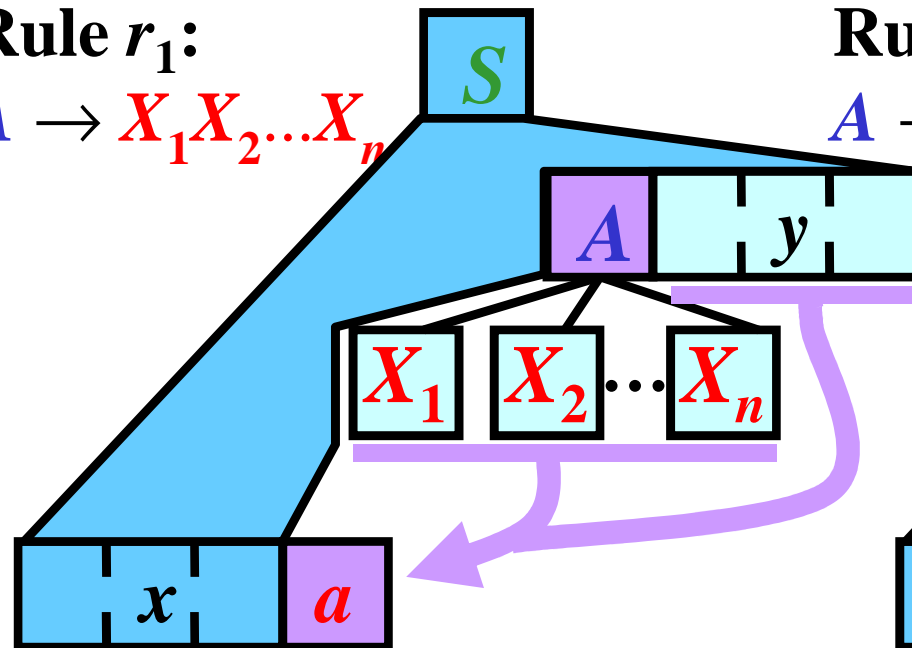
LL Grammars with ϵ -rules: Definition

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

Illustration:

Rule r_1 :

$A \rightarrow X_1X_2...X_n$

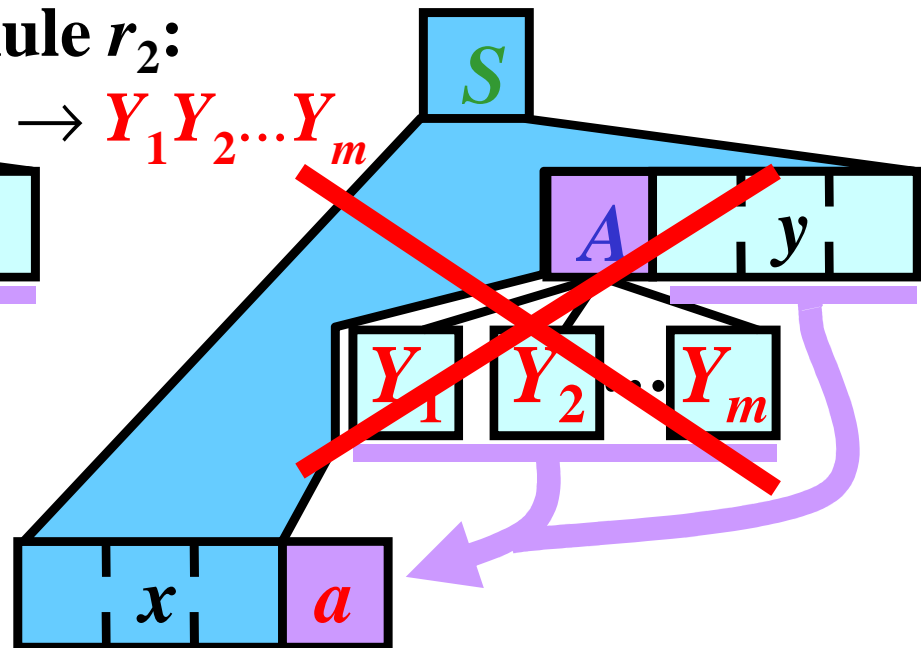


$a \in \text{Predict}(A \rightarrow X_1X_2...X_n)$

Ruled out in an LL grammar

Rule r_2 :

$A \rightarrow Y_1Y_2...Y_m$



$a \in \text{Predict}(A \rightarrow Y_1Y_2...Y_m)$

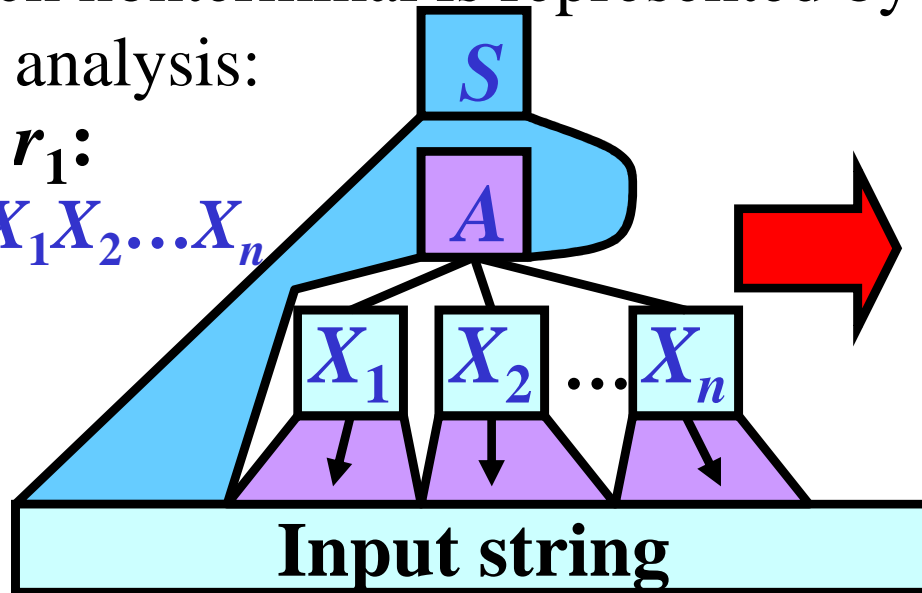
LL Analyzer Implementation

1) Recursive-Descent Parsing

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

Rule r_1 :

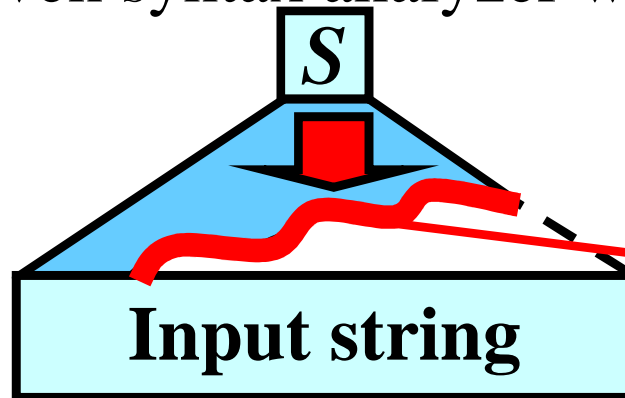
$A \rightarrow X_1 X_2 \dots X_n$



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

2) Predictive Parsing

- Table-driven syntax analyzer with pushdown



**These symbols are
in the pushdown.**

Recursive Descent: Example 1/4

```

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

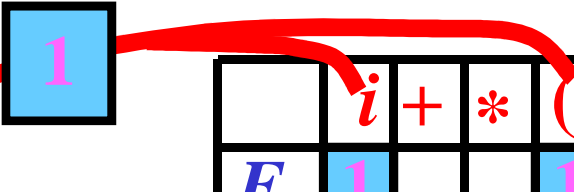
```

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

```

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

```



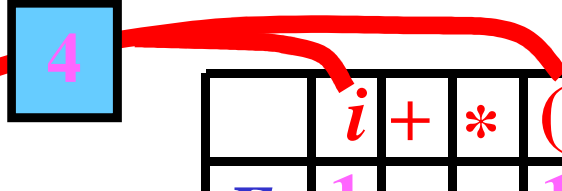
	<i>i</i>	+	*	()	\$
<i>E</i>	1			1		
<i>E'</i>		2			3	3
<i>T</i>	4			4		
<i>T'</i>		6	5		6	6
<i>F</i>	8			7		

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

```

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

```



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

Recursive Descent: Example 2/4

- For $E' \in N$: Rules 2: $E' \rightarrow +TE'$, 3: $E' \rightarrow \varepsilon$

```

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

```

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

Recursive Descent: Example 3/4

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

```

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

```

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

Recursive Descent: Example 4/4

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

```

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

```

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

Main body:

```

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

```

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

Start:

Input string:

$i * i \$$

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

Start: **GetNextToken;**
 Call E;

Input string:

i * i \$

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

Start: **GetNextToken;**

Call E;

Input string:

i * i \$

E:

For token = i :
Call T, Call E1

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

Start: **GetNextToken;**

Call E;

Input string:

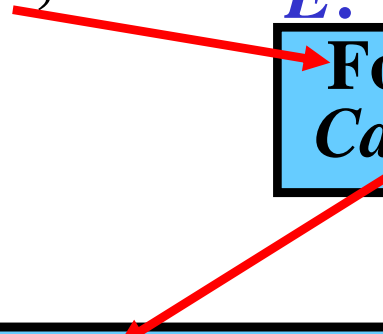
i * i \$

E:

For token = i :
Call T, Call E1

T:

For token = i :
Call F, Call T1



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

Start: **GetNextToken;**

Call E;

Input string:

i $*$ i $\$$

E:

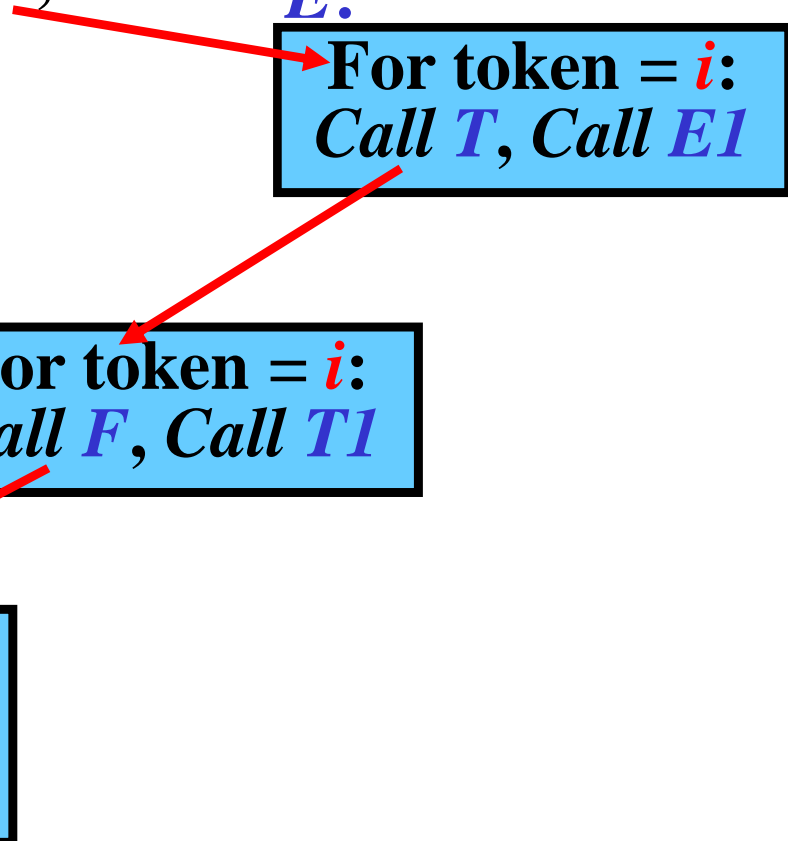
For token = i :
Call T, Call E1

T:

For token = i :
Call F, Call T1

F:

For token = i :
GetNextToken;
Return TRUE;



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

Start: **GetNextToken;**

Call E;

Input string:

i $*$ i $\$$

E:

For token = i :
Call T, Call E1

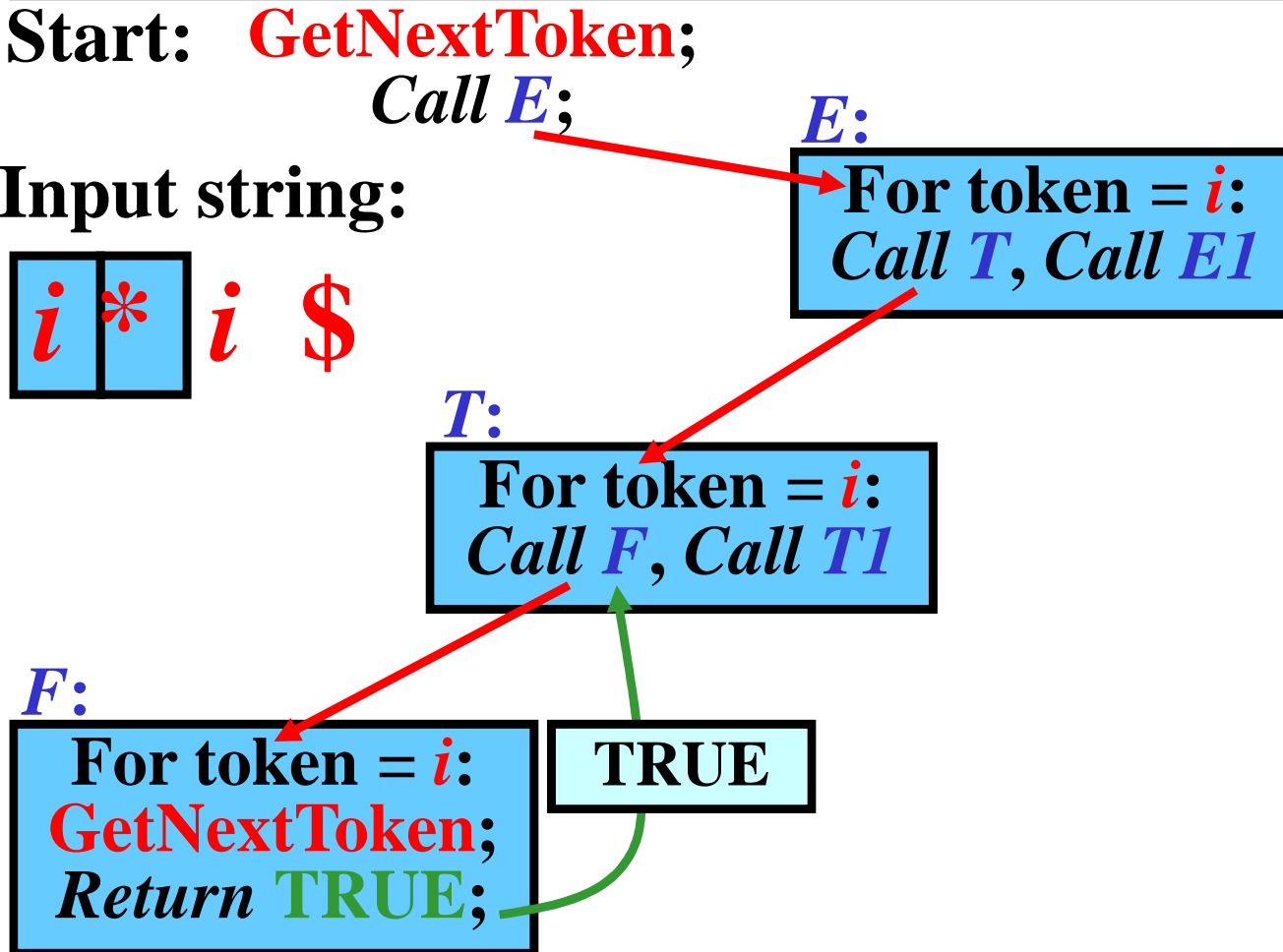
T:

For token = i :
Call F, Call T1

F:

For token = i :
GetNextToken;
Return TRUE;

TRUE



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

Start: **GetNextToken;**

Call E;

Input string:

$i * i \$$

E:

For token = i :
Call T, Call E1

T:

For token = i :
Call F, Call T1

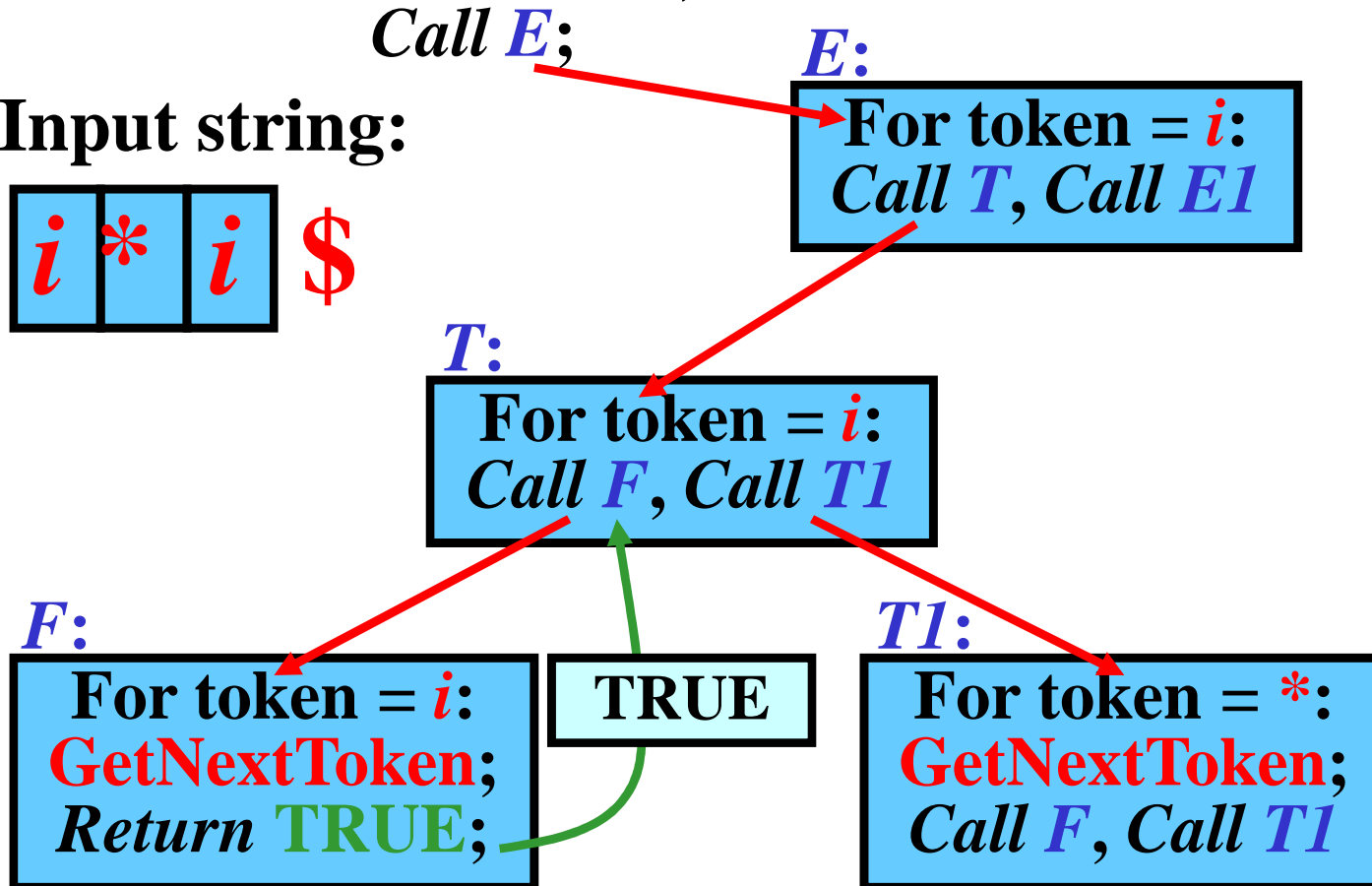
F:

For token = i :
GetNextToken;
Return TRUE;

TRUE

T1:

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

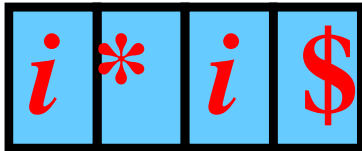


Recursive Descent: Illustration for $i*i\$$

Start: **GetNextToken;**

Call E ;

Input string:



E :

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

T :

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

F :

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

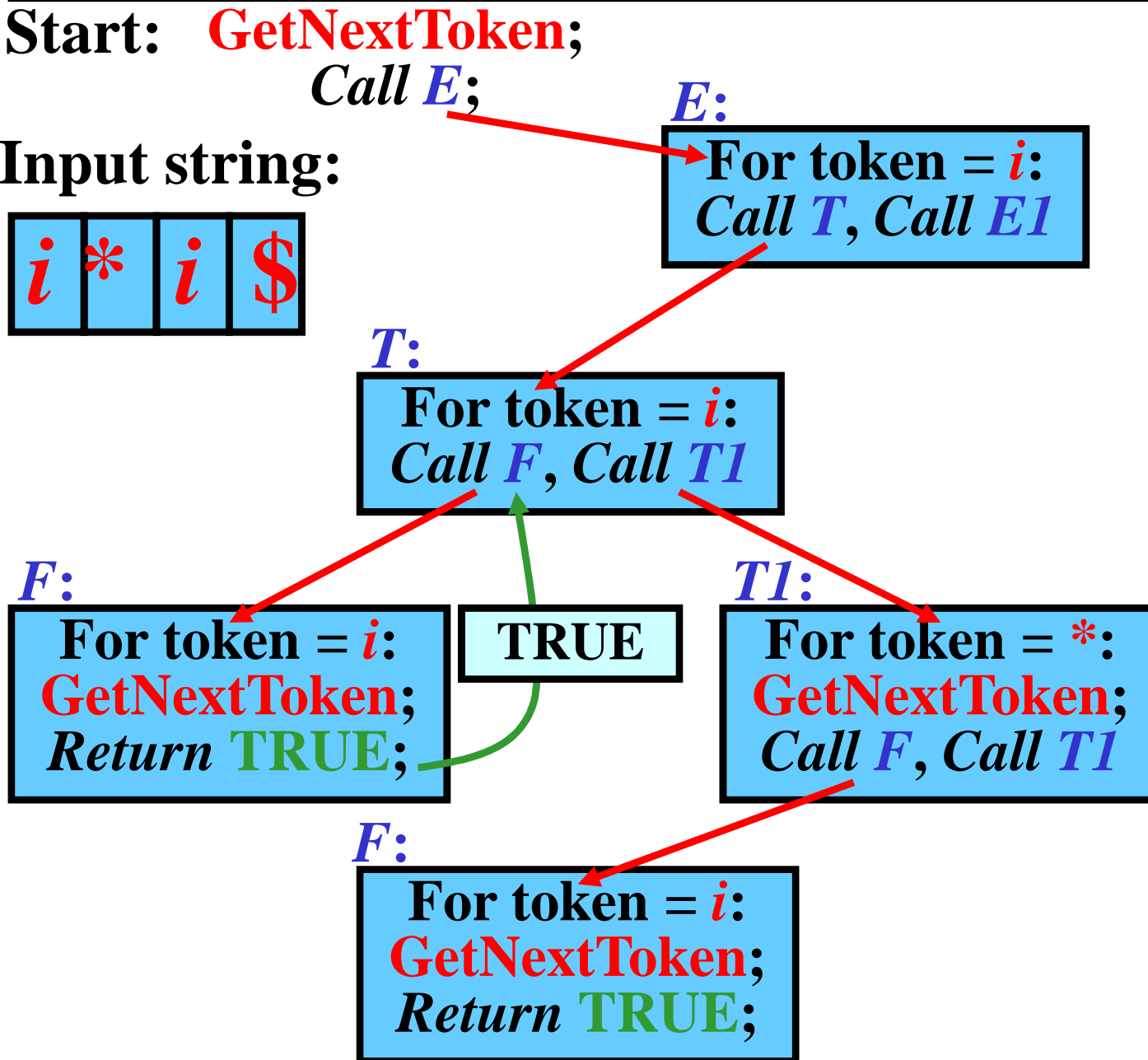
TRUE

$T1$:

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

F :

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

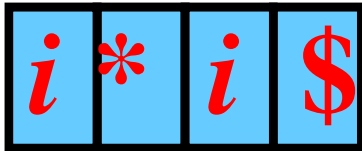


Recursive Descent: Illustration for $i*i\$$

Start: **GetNextToken;**

Call E;

Input string:



E:



T:



F:



TRUE

T1:



F:



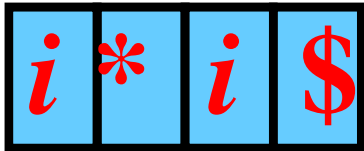
TRUE

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

Start: **GetNextToken;**

Call E ;

Input string:



E :

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

T :

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

F :

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

TRUE

$T1$:

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

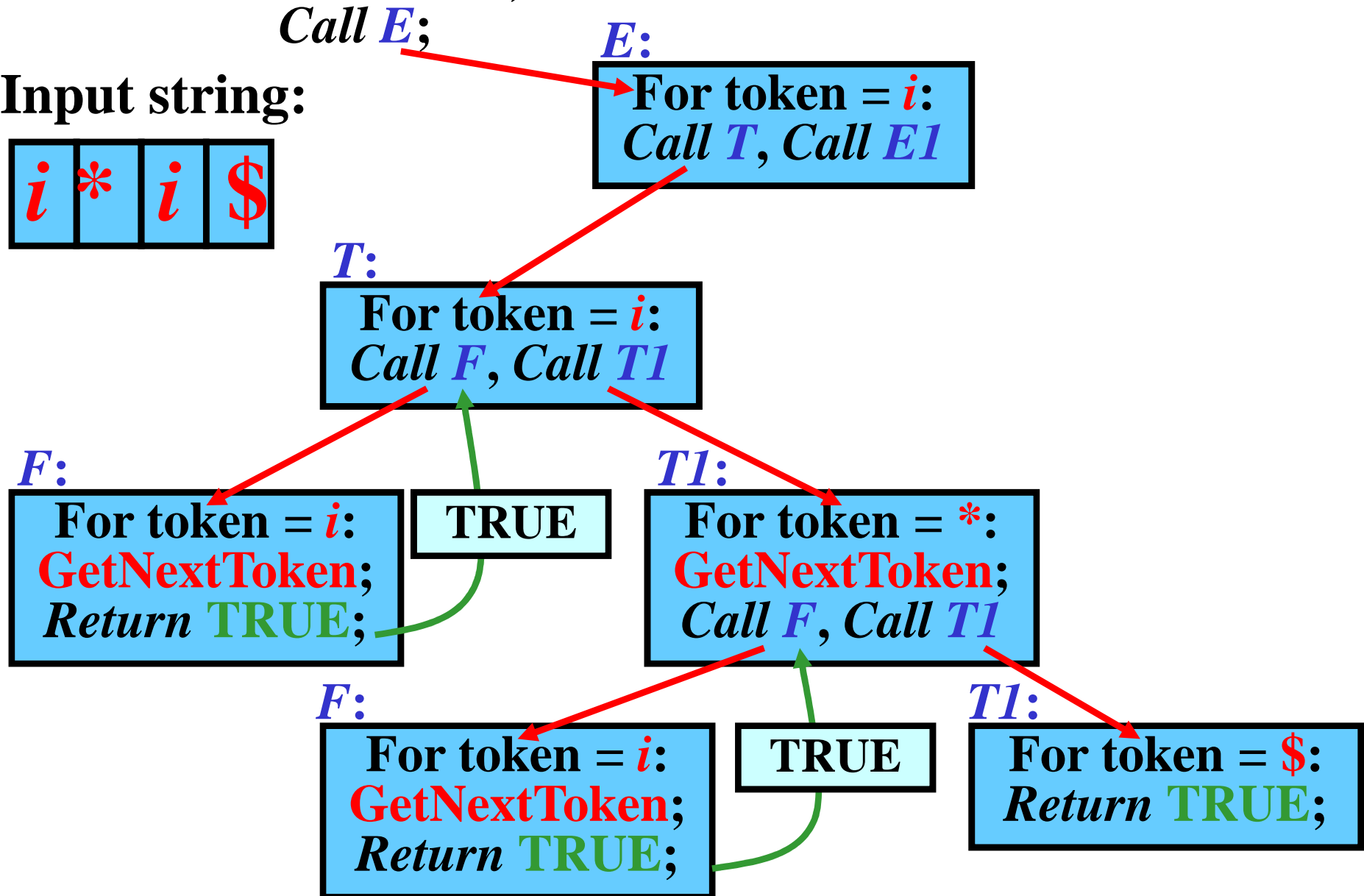
F :

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

TRUE

$T1$:

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

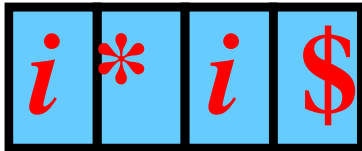


Recursive Descent: Illustration for $i*i\$$

Start: **GetNextToken;**

Call E ;

Input string:



E :



T :



F :



TRUE

$T1$:



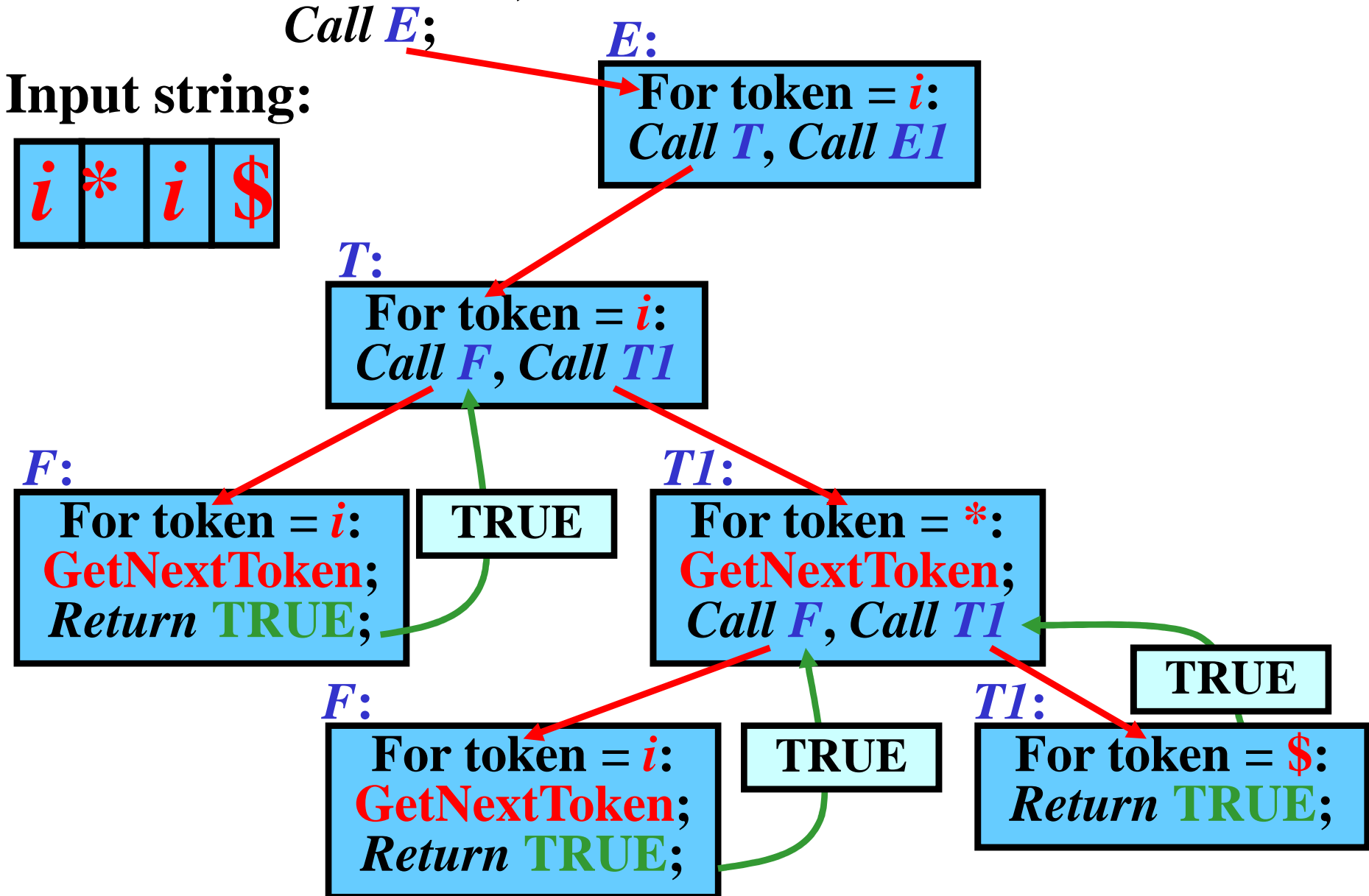
TRUE

F :



TRUE

$T1$:



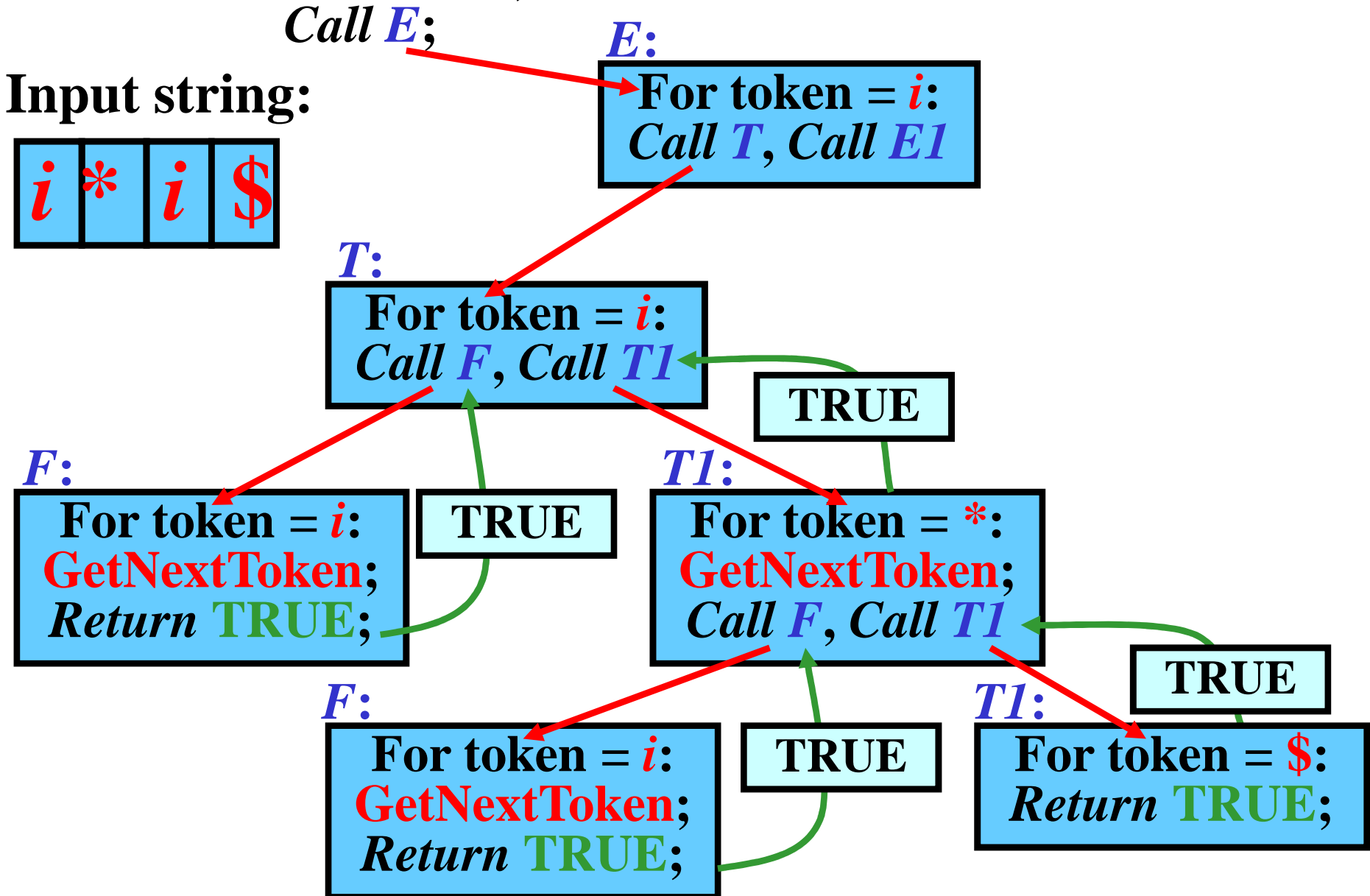
Recursive Descent: Illustration for $i*i\$$

Start: **GetNextToken;**

Call E;

Input string:

i	$*$	i	$\$$
-----	-----	-----	------

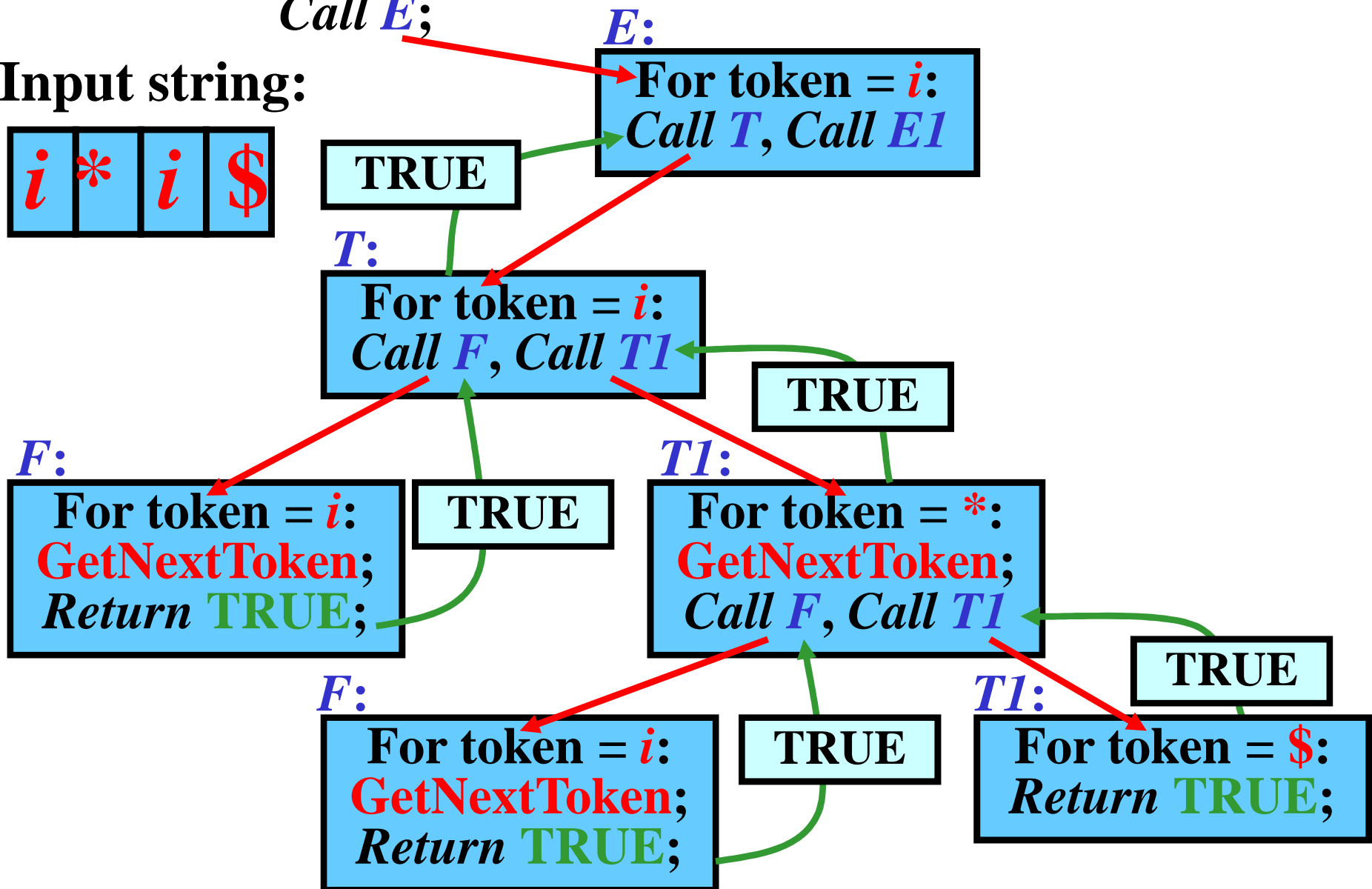
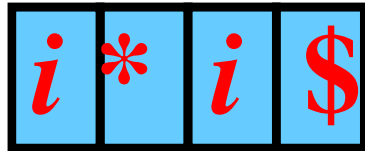


Recursive Descent: Illustration for $i*i\$$

Start: **GetNextToken;**

Call E;

Input string:

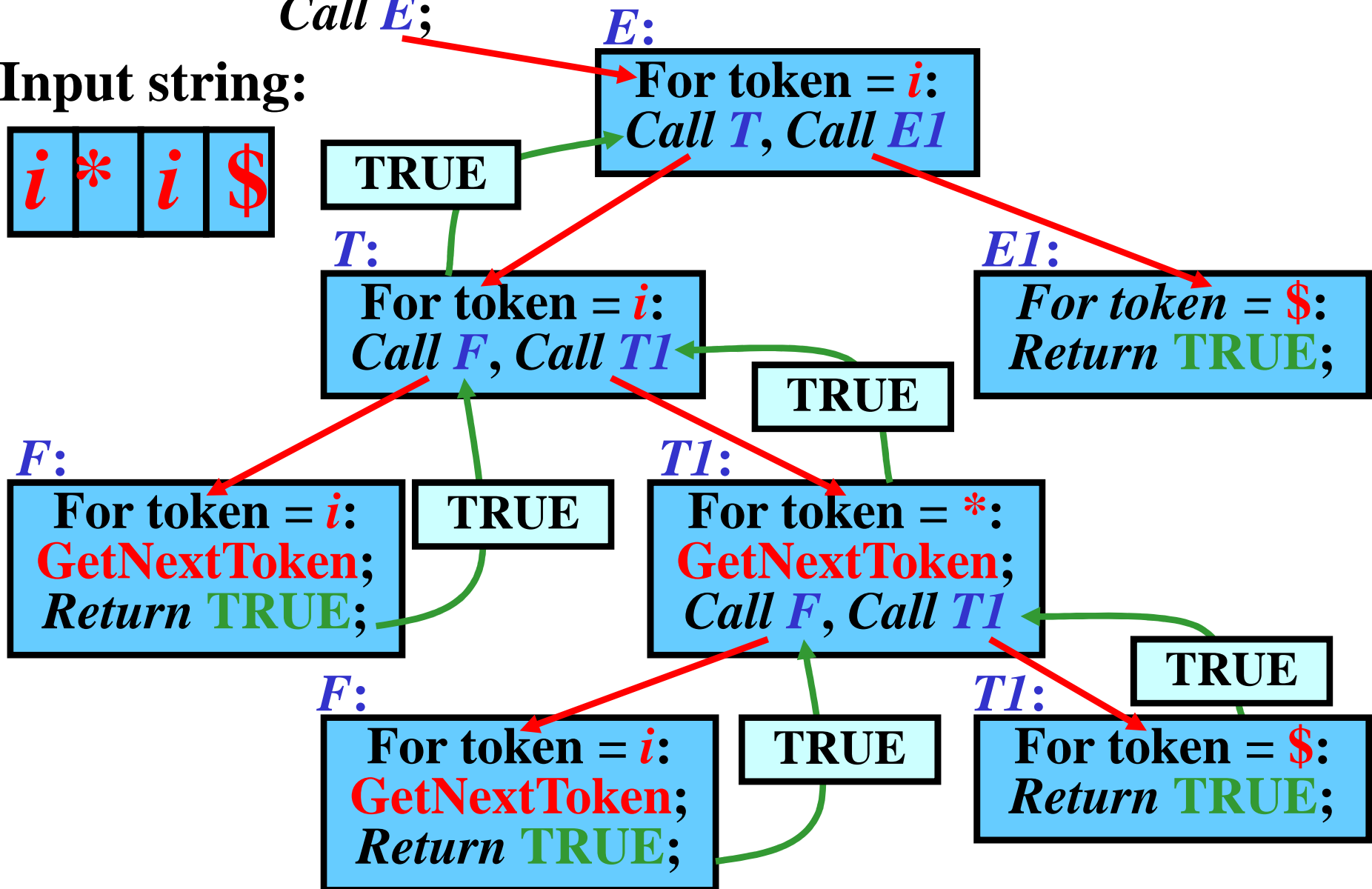
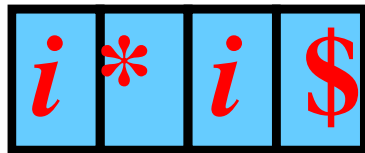


Recursive Descent: Illustration for $i*i\$$

Start: **GetNextToken;**

Call E;

Input string:

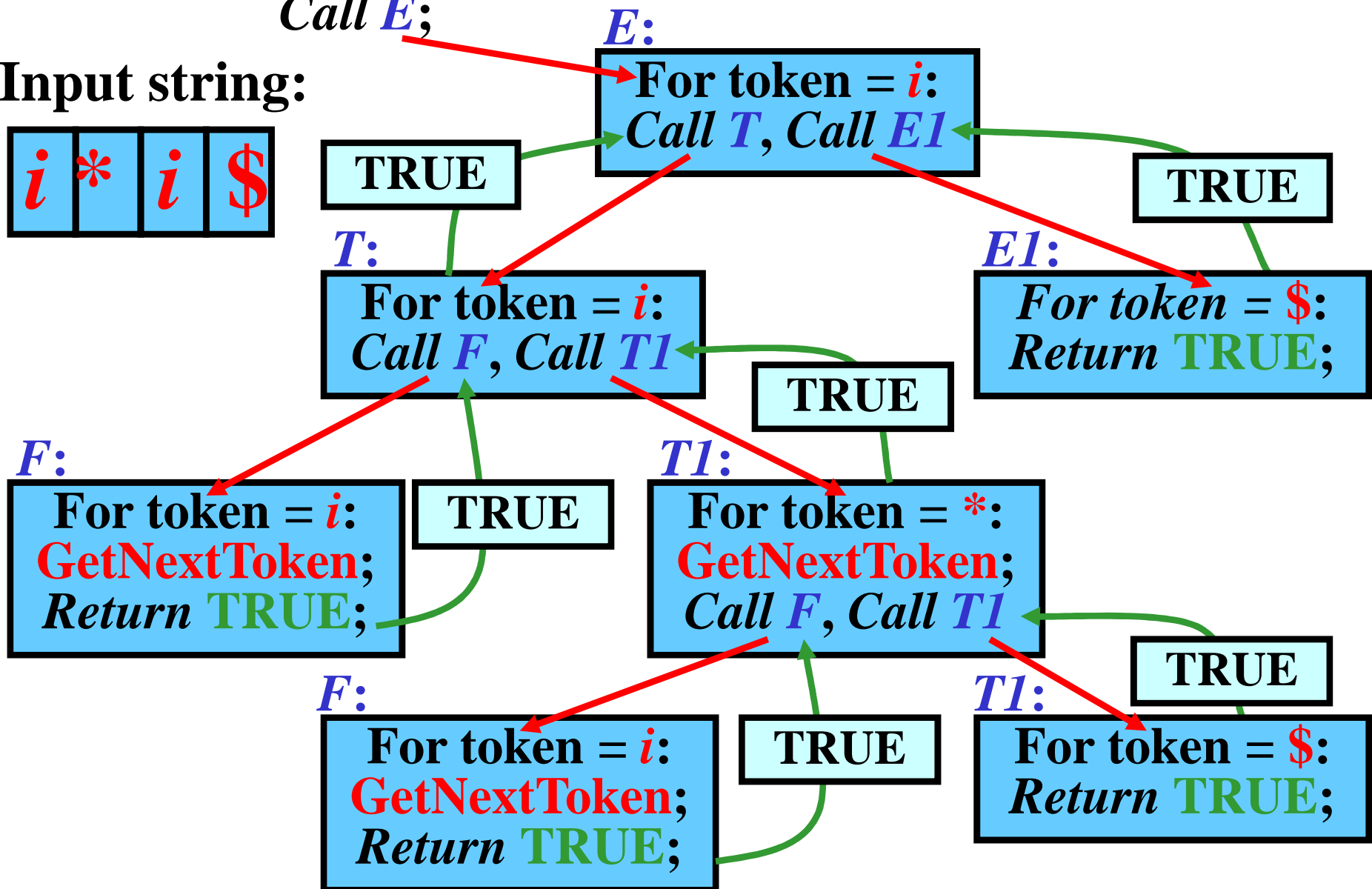
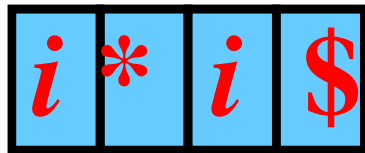


Recursive Descent: Illustration for $i*i\$$

Start: **GetNextToken;**

Call E;

Input string:

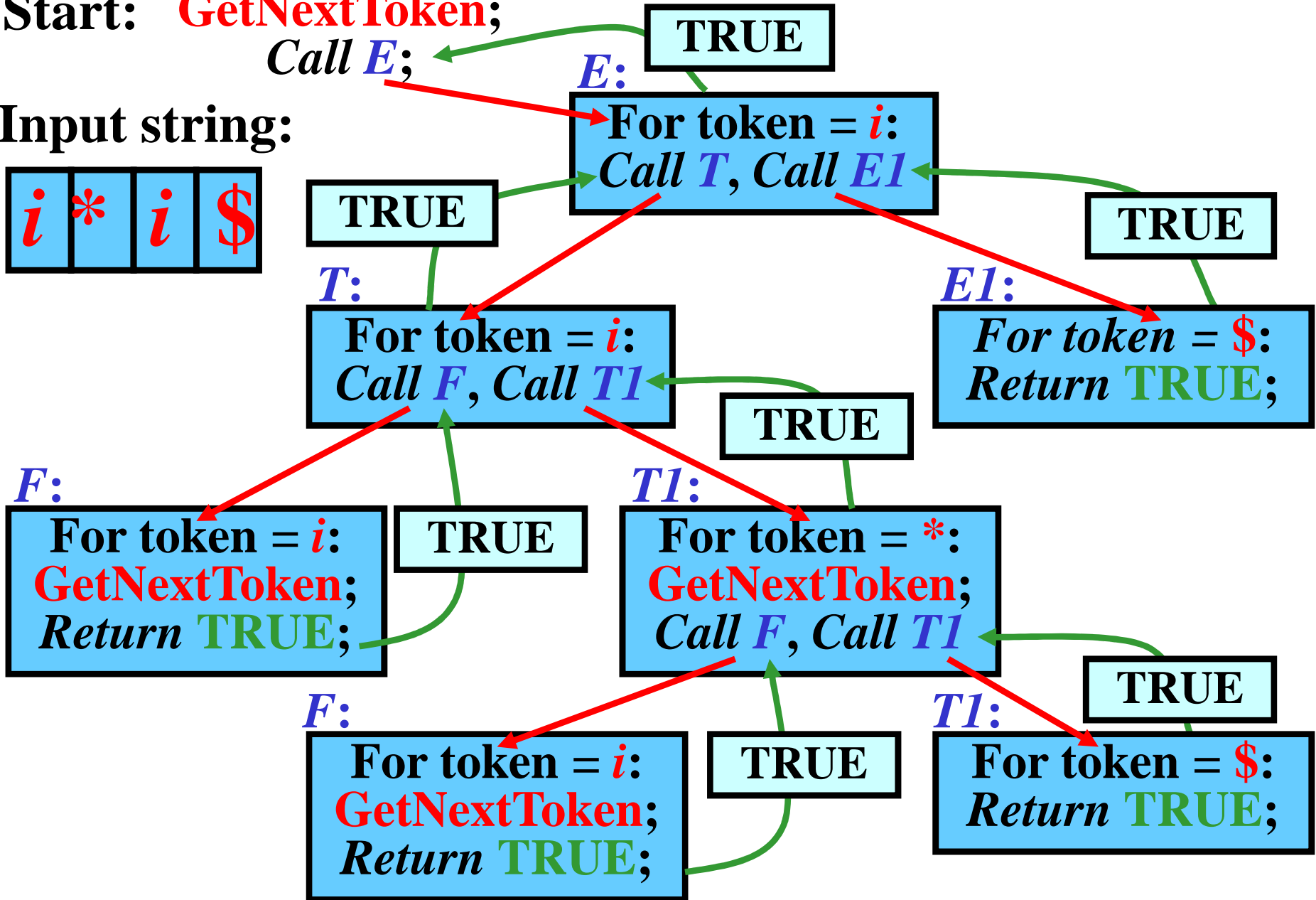


Recursive Descent: Illustration for $i*i$

Start: **GetNextToken;**

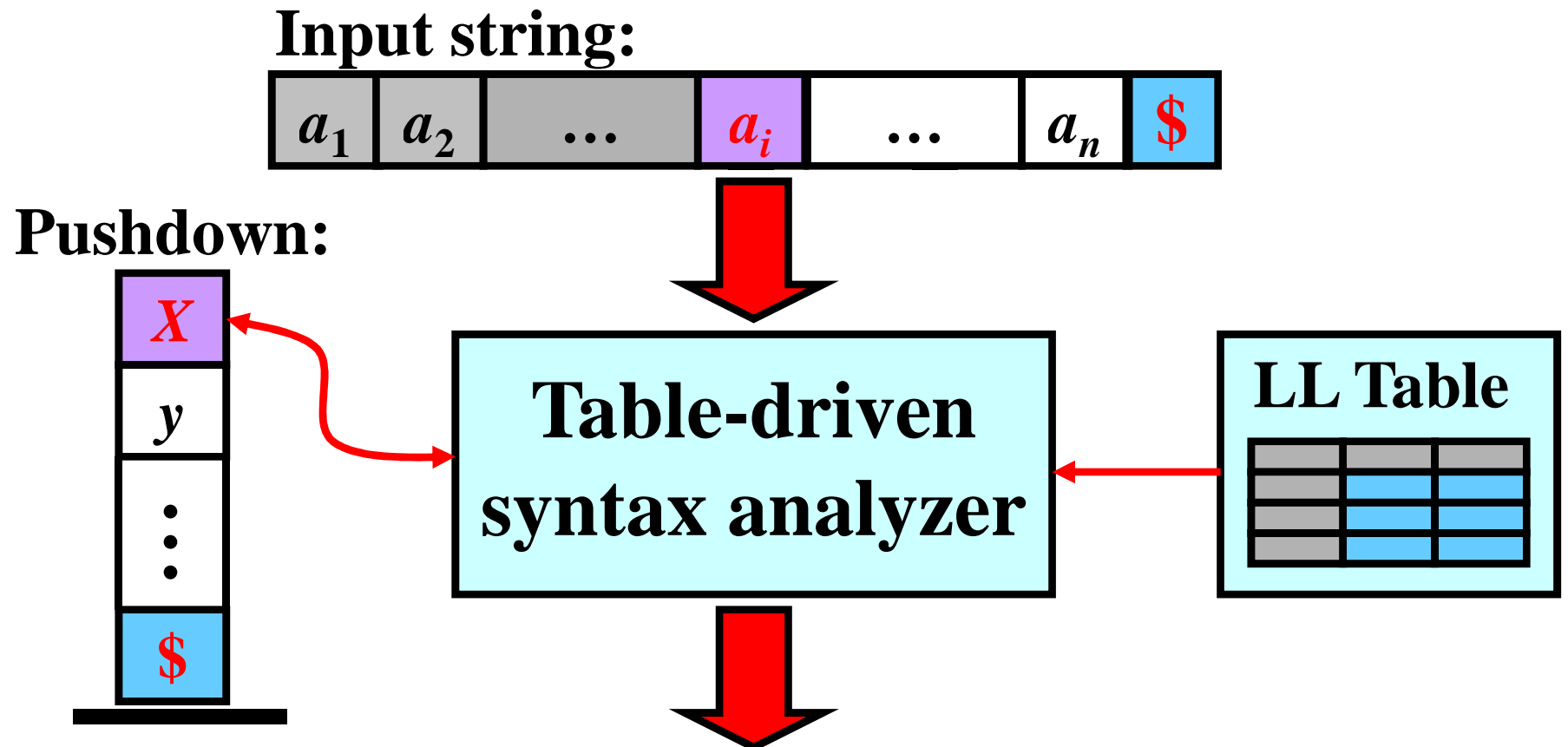
*Call **E**;*

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 ∈ T:** if **X = a** then pop(**X**) & read next **a** from
 input string
 else error;
 - **X ∈ N:** if **r: X → x** ∈ LL-table[**X, a**] then
 replace **X** with reversal(**x**) on the
 pushdown & write **r** to output
 else error;

end

Table-Driven Parsing: Example

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

Input string: $i * i \$$

Rules:

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

Pushdown	Input	Rule	Derivation

Table-Driven Parsing: Example

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

Input string: *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*

Pushdown	Input	Rule	Derivation
<i>\$E</i>	<i>i*i\$</i>	1: <i>E</i> → <i>TE'</i>	<i>E</i> ⇒ <i>TE'</i>
<i>\$E'T</i>	<i>i*i\$</i>	4: <i>T</i> → <i>FT'</i>	⇒ <i>FT'E'</i>

Table-Driven Parsing: Example

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

Input string: *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*

Pushdown	Input	Rule	Derivation
\$ <i>E</i>	<i>i</i> * <i>i</i> \$	1: <i>E</i> → <i>TE'</i>	<u><i>E</i></u> ⇒ <u><i>TE'</i></u>
\$ <i>E'</i> <i>T</i>	<i>i</i> * <i>i</i> \$	4: <i>T</i> → <i>FT'</i>	⇒ <u><i>FT'</i></u> <i>E'</i>
\$ <i>E'</i> <i>T'</i> <i>F</i>	<i>i</i> * <i>i</i> \$	8: <i>F</i> → <i>i</i>	⇒ <i>i</i> <u><i>T'</i></u> <i>E'</i>

Table-Driven Parsing: Example

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

Input string: *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*

Pushdown	Input	Rule	Derivation
<i>\$E</i>	<i>i*i\$</i>	1: <i>E</i> → <i>TE'</i>	<i>E</i> ⇒ <i>TE'</i>
<i>\$E'T</i>	<i>i*i\$</i>	4: <i>T</i> → <i>FT'</i>	⇒ <i>FT'E'</i>
<i>\$E'T'F</i>	<i>i*i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>iT'E'</i>
<i>\$E'T'i</i>	<i>i*i\$</i>		

Table-Driven Parsing: Example

	<i>i</i>	+	*	()	\$
<i>E</i>	1			1		
<i>E'</i>		2			3	3
<i>T</i>	4			4		
<i>T'</i>		6	5		6	6
<i>F</i>	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: $E \rightarrow TE'$	$\underline{E} \Rightarrow \underline{TE'}$
$\$E'T$	$i*i\$$	4: $T \rightarrow FT'$	$\Rightarrow \underline{FT'E'}$
$\$E'T'F$	$i*i\$$	8: $F \rightarrow i$	$\Rightarrow i\underline{T'E'}$
$\$E'T'i$	$i*i\$$		
$\$E'T'$	$*i\$$	5: $T' \rightarrow *FT'$	$\Rightarrow i*\underline{FT'E'}$

Table-Driven Parsing: Example

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

Input string: *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*

Pushdown	Input	Rule	Derivation
<i>\$E</i>	<i>i*i\$</i>	1: <i>E</i> → <i>TE'</i>	<i>E</i> ⇒ <i>TE'</i>
<i>\$E'T</i>	<i>i*i\$</i>	4: <i>T</i> → <i>FT'</i>	⇒ <i>FT'E'</i>
<i>\$E'T'F</i>	<i>i*i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>iT'E'</i>
<i>\$E'T'i</i>	<i>i*i\$</i>		
<i>\$E'T'</i>	<i>*i\$</i>	5: <i>T'</i> → * <i>FT'</i>	⇒ <i>i*FT'E'</i>
<i>\$E'T'F*</i>	<i>*i\$</i>		

Table-Driven Parsing: Example

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

Input string: *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*

Pushdown	Input	Rule	Derivation
\$ <i>E</i>	<i>i</i> * <i>i</i> \$	1: <i>E</i> → <i>TE'</i>	<u><i>E</i></u> ⇒ <u><i>TE'</i></u>
\$ <i>E'</i> <i>T</i>	<i>i</i> * <i>i</i> \$	4: <i>T</i> → <i>FT'</i>	⇒ <u><i>FT'</i></u> <i>E'</i>
\$ <i>E'</i> <i>T'</i> <i>F</i>	<i>i</i> * <i>i</i> \$	8: <i>F</i> → <i>i</i>	⇒ <i>i</i> <u><i>T'</i></u> <i>E'</i>
\$ <i>E'</i> <i>T'</i> <i>i</i>	<i>i</i> * <i>i</i> \$		
\$ <i>E'</i> <i>T'</i>	* <i>i</i> \$	5: <i>T'</i> → * <i>FT'</i>	⇒ <i>i</i> * <u><i>FT'</i></u> <i>E'</i>
\$ <i>E'</i> <i>T'</i> <i>F</i> *	* <i>i</i> \$		
\$ <i>E'</i> <i>T'</i> <i>F</i>	<i>i</i> \$	8: <i>F</i> → <i>i</i>	⇒ <i>i</i> * <i>i</i> <u><i>T'</i></u> <i>E'</i>

Table-Driven Parsing: Example

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

Input string: *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*

Pushdown	Input	Rule	Derivation
<i>\$E</i>	<i>i*i\$</i>	1: <i>E</i> → <i>TE'</i>	<i>E</i> ⇒ <i>TE'</i>
<i>\$E'T</i>	<i>i*i\$</i>	4: <i>T</i> → <i>FT'</i>	⇒ <i>FT'E'</i>
<i>\$E'T'F</i>	<i>i*i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>iT'E'</i>
<i>\$E'T'i</i>	<i>i*i\$</i>		
<i>\$E'T'</i>	<i>*i\$</i>	5: <i>T'</i> → * <i>FT'</i>	⇒ <i>i*FT'E'</i>
<i>\$E'T'F*</i>	<i>*i\$</i>		
<i>\$E'T'F</i>	<i>i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>i*iT'E'</i>
<i>\$E'T'i</i>	<i>i\$</i>		

Table-Driven Parsing: Example

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

Input string: *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*

Pushdown	Input	Rule	Derivation
<i>\$E</i>	<i>i*i\$</i>	1: <i>E</i> → <i>TE'</i>	<i>E</i> ⇒ <i>TE'</i>
<i>\$E'T</i>	<i>i*i\$</i>	4: <i>T</i> → <i>FT'</i>	⇒ <i>FT'E'</i>
<i>\$E'T'F</i>	<i>i*i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>iT'E'</i>
<i>\$E'T'i</i>	<i>i*i\$</i>		
<i>\$E'T'</i>	<i>*i\$</i>	5: <i>T'</i> → * <i>FT'</i>	⇒ <i>i*FT'E'</i>
<i>\$E'T'F*</i>	<i>*i\$</i>		
<i>\$E'T'F</i>	<i>i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>i*iT'E'</i>
<i>\$E'T'i</i>	<i>i\$</i>		
<i>\$E'T'</i>	<i>\$</i>	6: <i>T'</i> → ε	⇒ <i>i*iE'</i>

Table-Driven Parsing: Example

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

Input string: *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*

Pushdown	Input	Rule	Derivation
<i>\$E</i>	<i>i*i\$</i>	1: <i>E</i> → <i>TE'</i>	<i>E</i> ⇒ <i>TE'</i>
<i>\$E'T</i>	<i>i*i\$</i>	4: <i>T</i> → <i>FT'</i>	⇒ <i>FT'E'</i>
<i>\$E'T'F</i>	<i>i*i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>iT'E'</i>
<i>\$E'T'i</i>	<i>i*i\$</i>		
<i>\$E'T'</i>	<i>*i\$</i>	5: <i>T'</i> → * <i>FT'</i>	⇒ <i>i*FT'E'</i>
<i>\$E'T'F*</i>	<i>*i\$</i>		
<i>\$E'T'F</i>	<i>i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>i*iT'E'</i>
<i>\$E'T'i</i>	<i>i\$</i>		
<i>\$E'T'</i>	<i>\$</i>	6: <i>T'</i> → ε	⇒ <i>i*iE'</i>
<i>\$E'</i>	<i>\$</i>	3: <i>E'</i> → ε	⇒ <i>i*i</i>

Table-Driven Parsing: Example

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

Input string: *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*

Pushdown	Input	Rule	Derivation
<i>\$E</i>	<i>i*i\$</i>	1: <i>E</i> → <i>TE'</i>	<i>E</i> ⇒ <i>TE'</i>
<i>\$E'T</i>	<i>i*i\$</i>	4: <i>T</i> → <i>FT'</i>	⇒ <i>FT'E'</i>
<i>\$E'T'F</i>	<i>i*i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>iT'E'</i>
<i>\$E'T'i</i>	<i>i*i\$</i>		
<i>\$E'T'</i>	<i>*i\$</i>	5: <i>T'</i> → * <i>FT'</i>	⇒ <i>i*FT'E'</i>
<i>\$E'T'F*</i>	<i>*i\$</i>		
<i>\$E'T'F</i>	<i>i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>i*iT'E'</i>
<i>\$E'T'i</i>	<i>i\$</i>		
<i>\$E'T'</i>	<i>\$</i>	6: <i>T'</i> → ε	⇒ <i>i*iE'</i>
<i>\$E'</i>	<i>\$</i>	3: <i>E'</i> → ε	⇒ <i>i*i</i>
<i>\$</i>	<i>\$</i>		

Table-Driven Parsing: Example

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

Input string: *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*

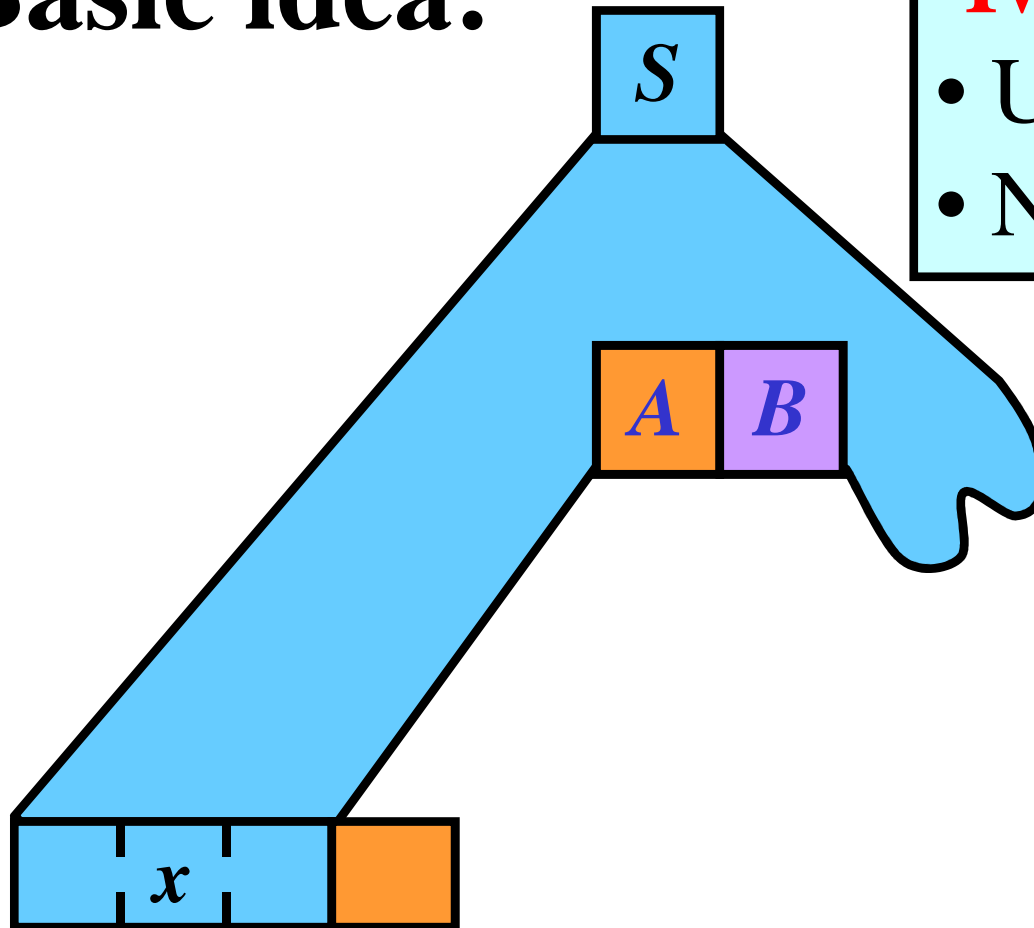
Pushdown	Input	Rule	Derivation
<i>\$E</i>	<i>i*i\$</i>	1: <i>E</i> → <i>TE'</i>	<i>E</i> ⇒ <i>TE'</i>
<i>\$E'T</i>	<i>i*i\$</i>	4: <i>T</i> → <i>FT'</i>	⇒ <i>FT'E'</i>
<i>\$E'T'F</i>	<i>i*i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>iT'E'</i>
<i>\$E'T'i</i>	<i>i*i\$</i>		
<i>\$E'T'</i>	<i>*i\$</i>	5: <i>T'</i> → * <i>FT'</i>	⇒ <i>i*FT'E'</i>
<i>\$E'T'F*</i>	<i>*i\$</i>		
<i>\$E'T'F</i>	<i>i\$</i>	8: <i>F</i> → <i>i</i>	⇒ <i>i*iT'E'</i>
<i>\$E'T'i</i>	<i>i\$</i>		
<i>\$E'T'</i>	<i>\$</i>	6: <i>T'</i> → ε	⇒ <i>i*iE'</i>
<i>\$E'</i>	<i>\$</i>	3: <i>E'</i> → ε	⇒ <i>i*i</i>
<i>\$</i>	<i>\$</i>		

Success

Left parse: 1485863

Handling Errors: Introduction

Basic idea:



Two kinds of errors:

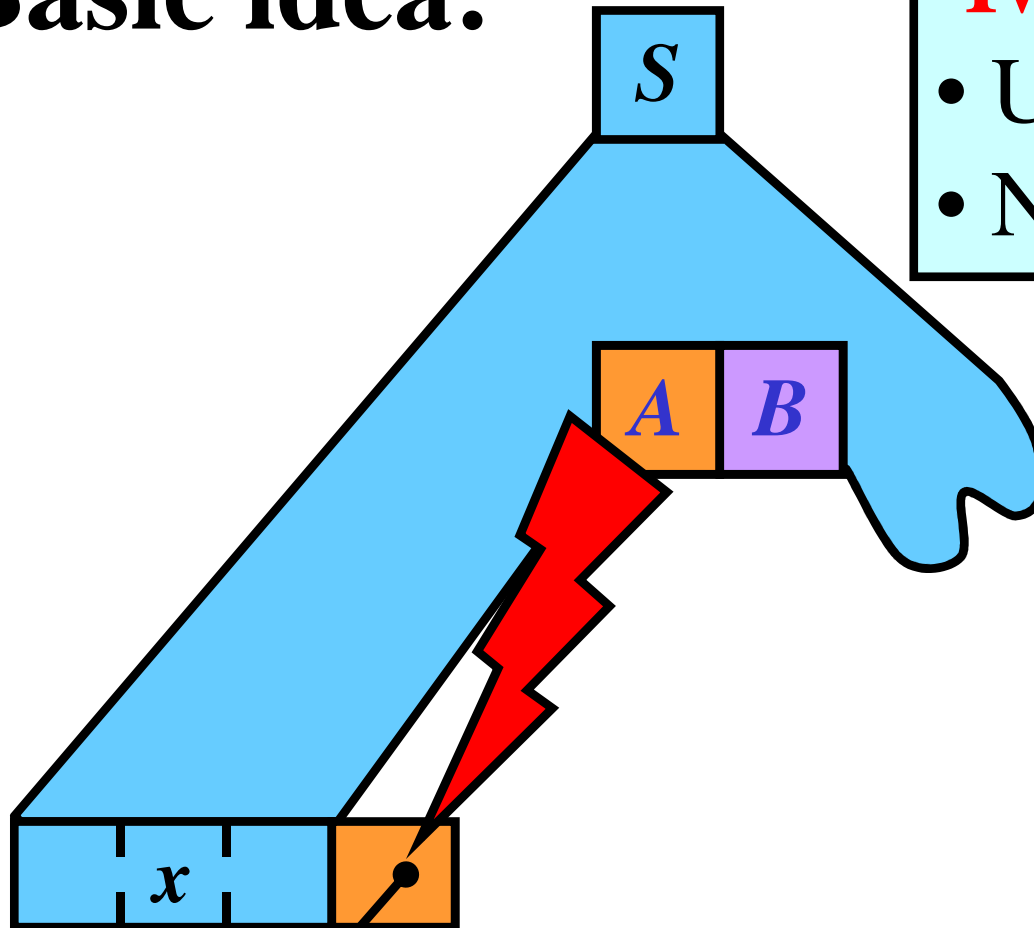
- Unexpected token
- No rule applicable

Handling Errors: Introduction

Basic idea:

Two kinds of errors:

- Unexpected token
- No rule applicable



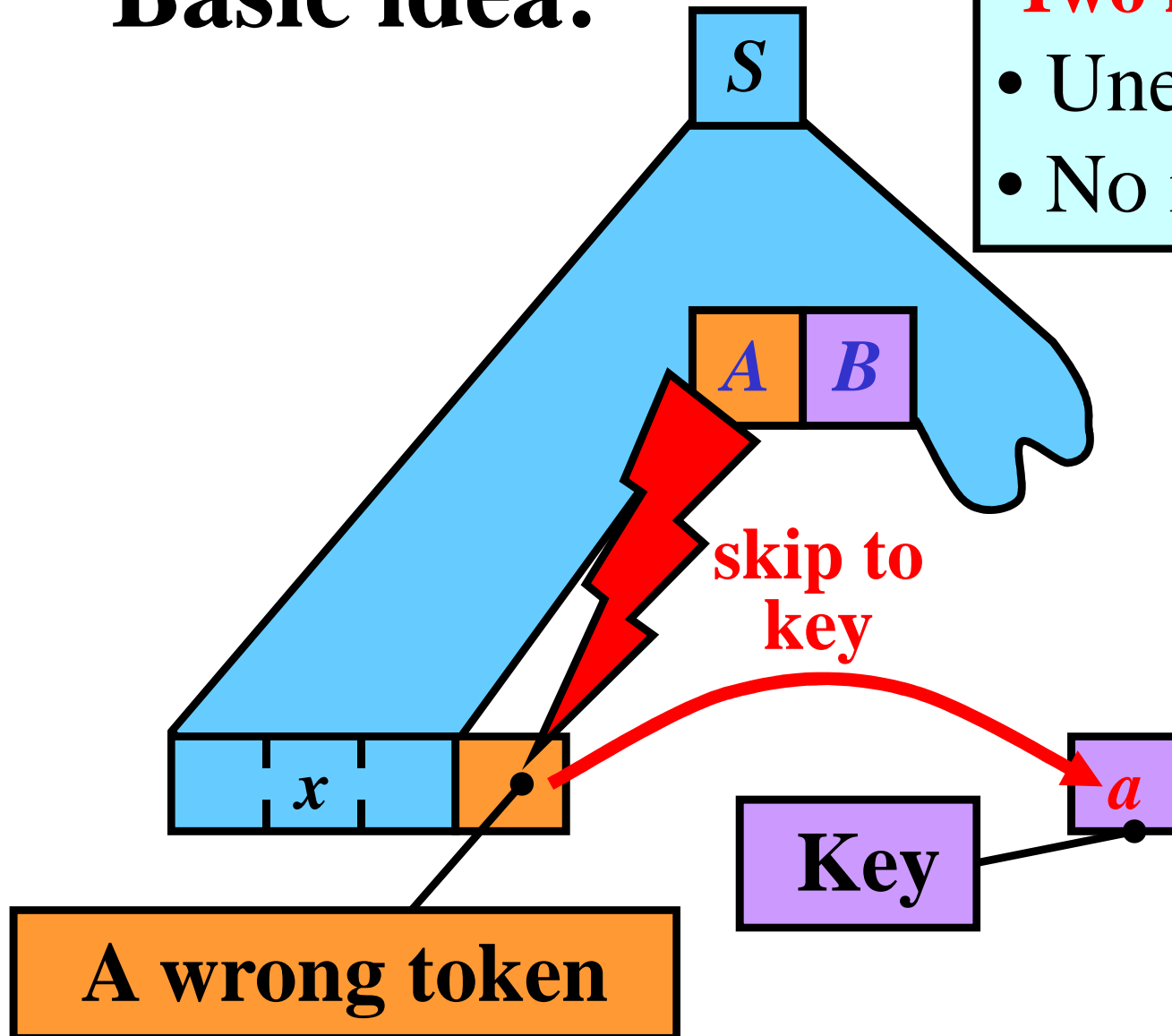
A wrong token

Handling Errors: Introduction

Basic idea:

Two kinds of errors:

- Unexpected token
- No rule applicable

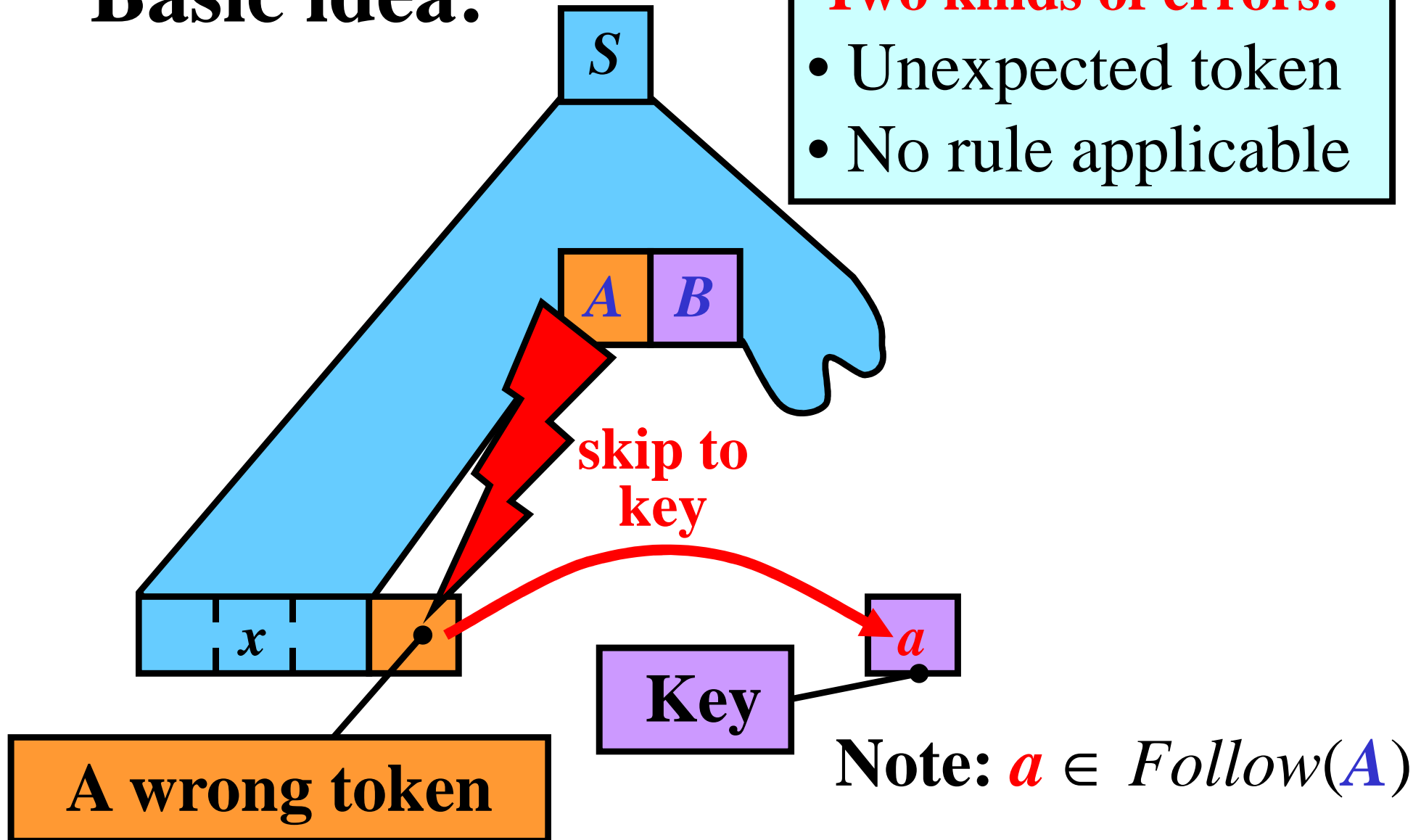


Handling Errors: Introduction

Basic idea:

Two kinds of errors:

- Unexpected token
- No rule applicable

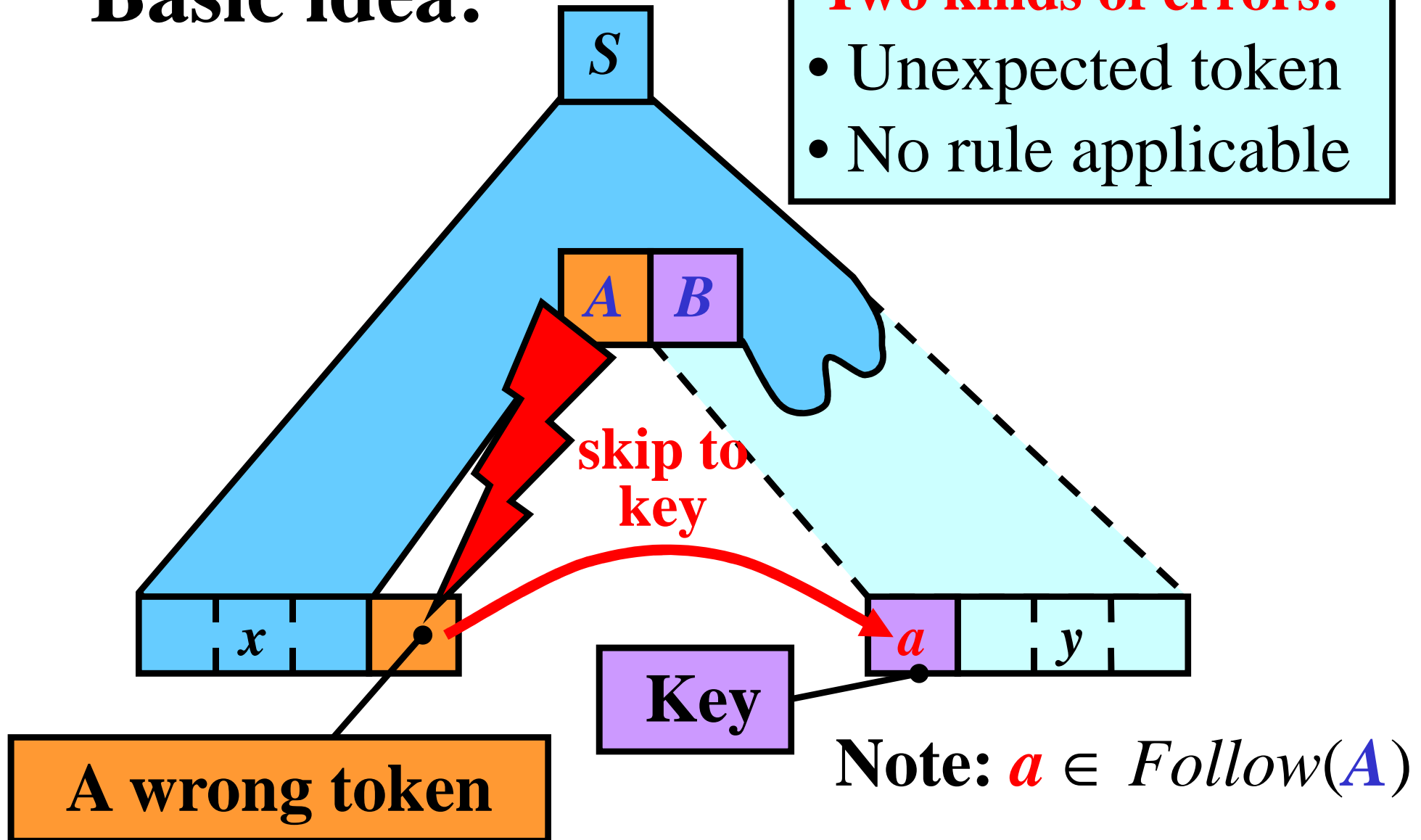


Handling Errors: Introduction

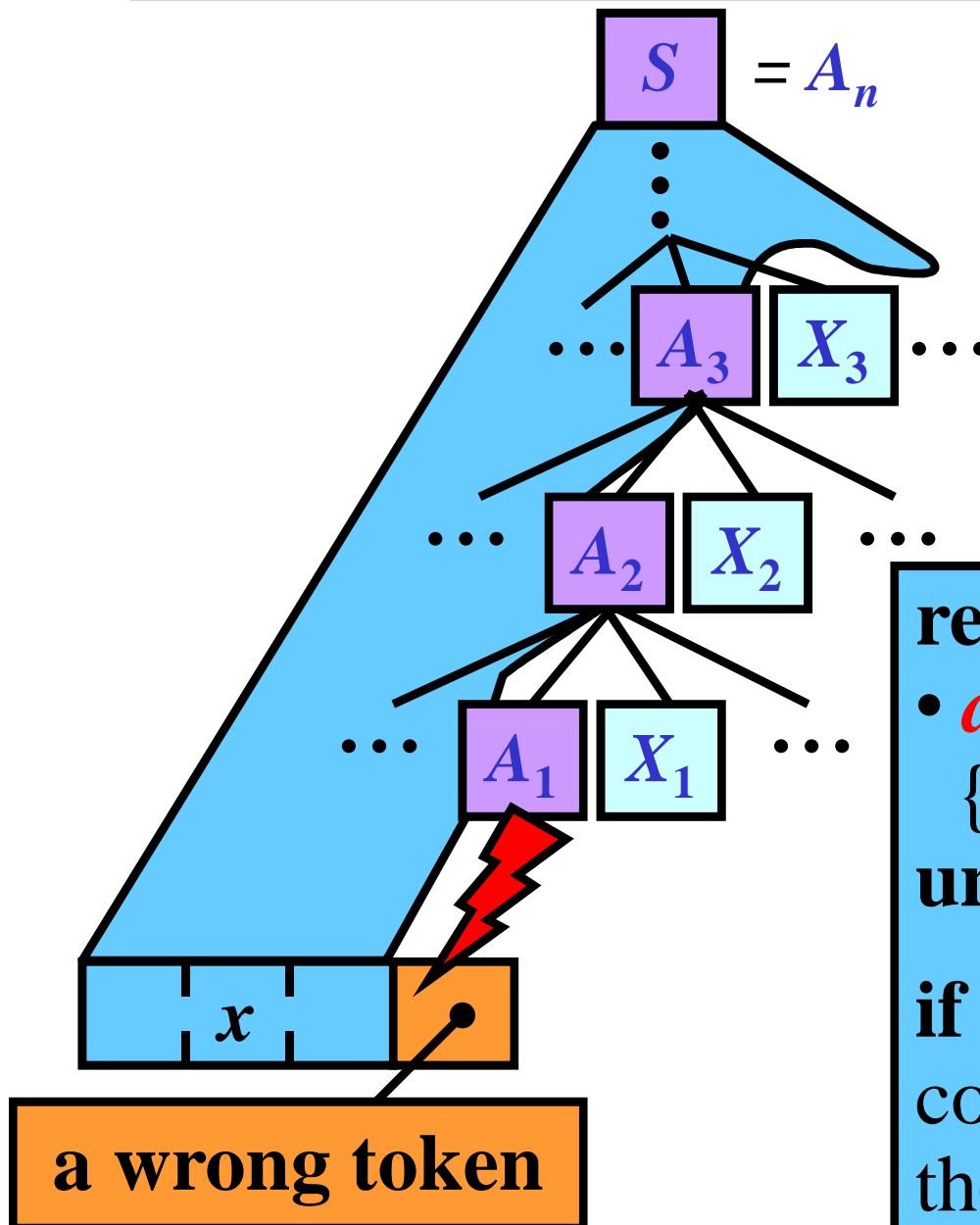
Basic idea:

Two kinds of errors:

- Unexpected token
- No rule applicable



Panic-Mode (Hartmann) Error Recovery



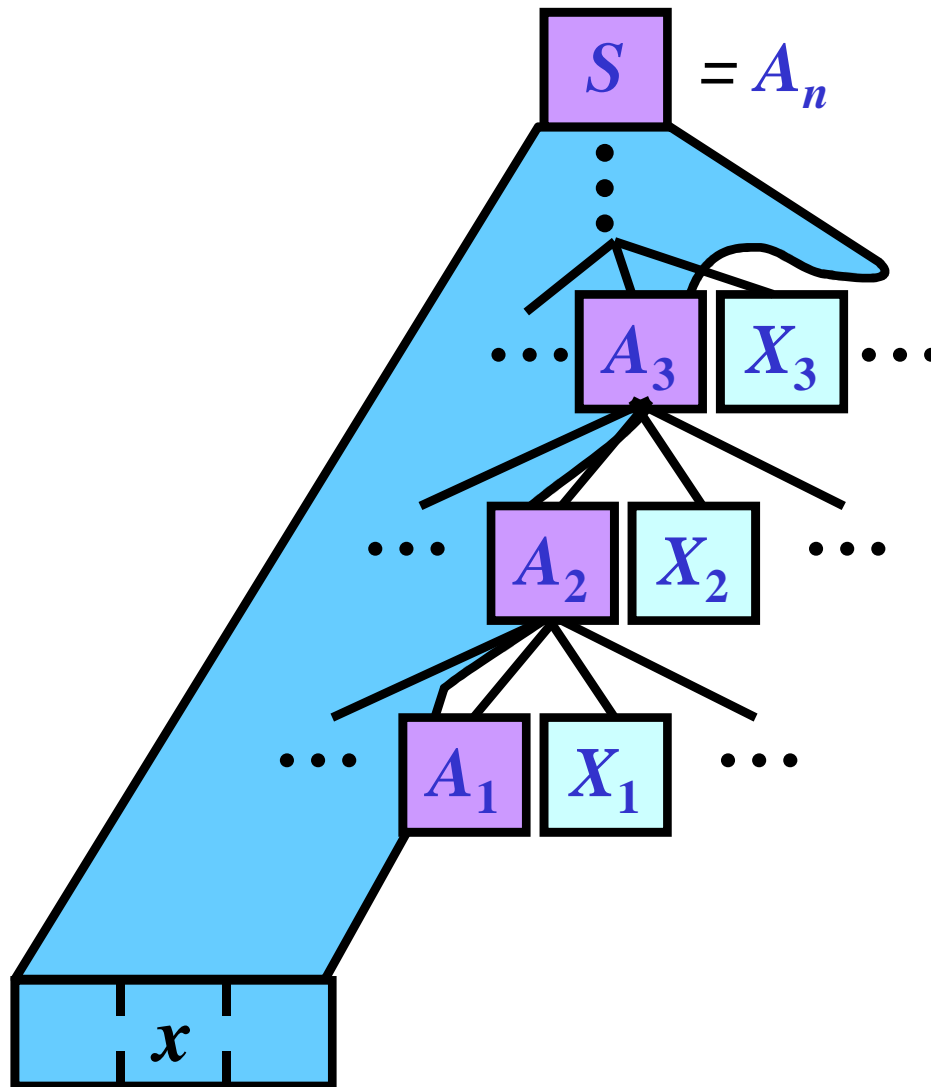
- Let **Context**(A_1) = $\text{Follow}(A_1) \cup \text{Follow}(A_2) \cup \dots \cup \text{Follow}(A_n)$

repeat

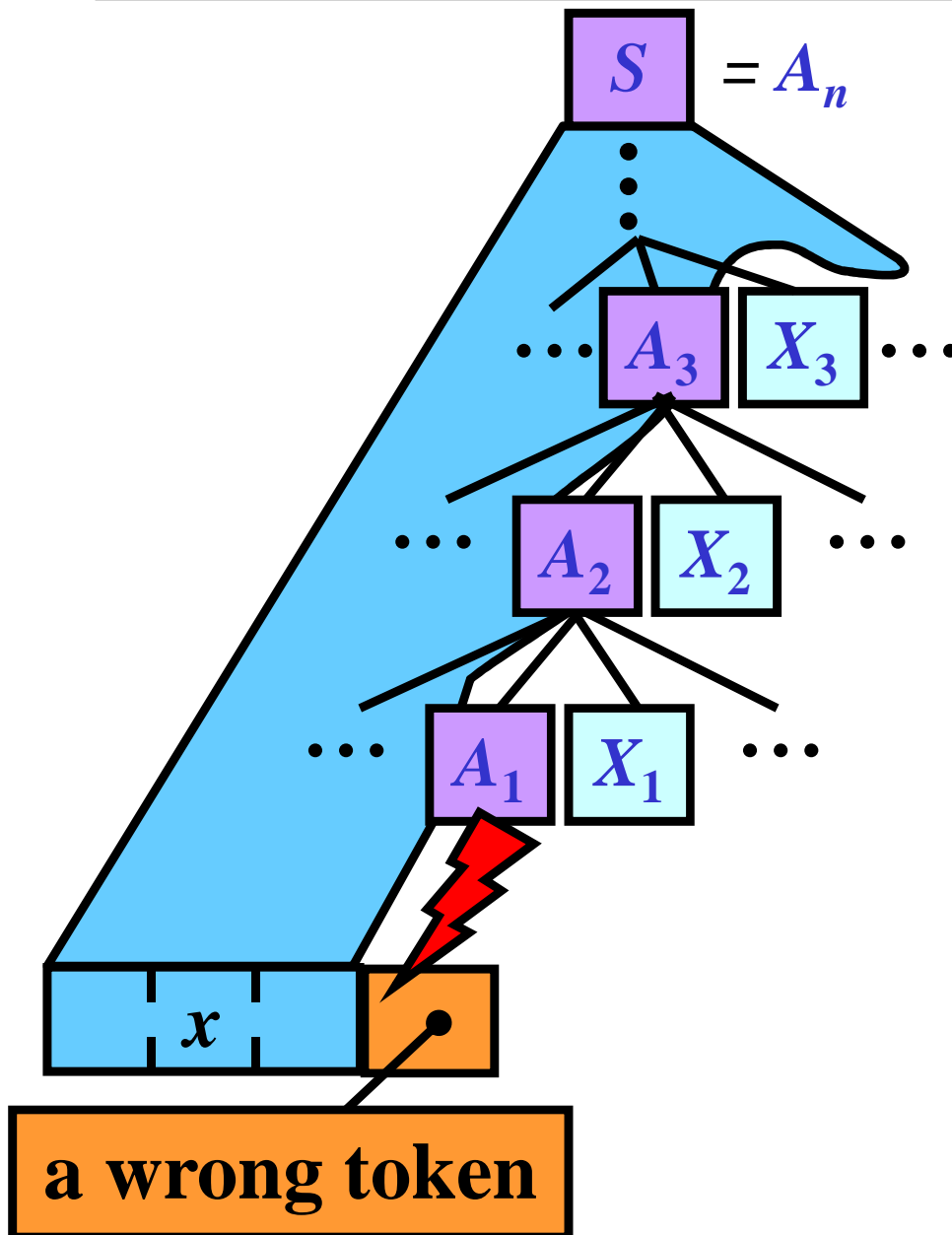
- $a := \text{GetNextToken};$
 {These tokens are skipped}
 until a in **Context**(A_1)

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

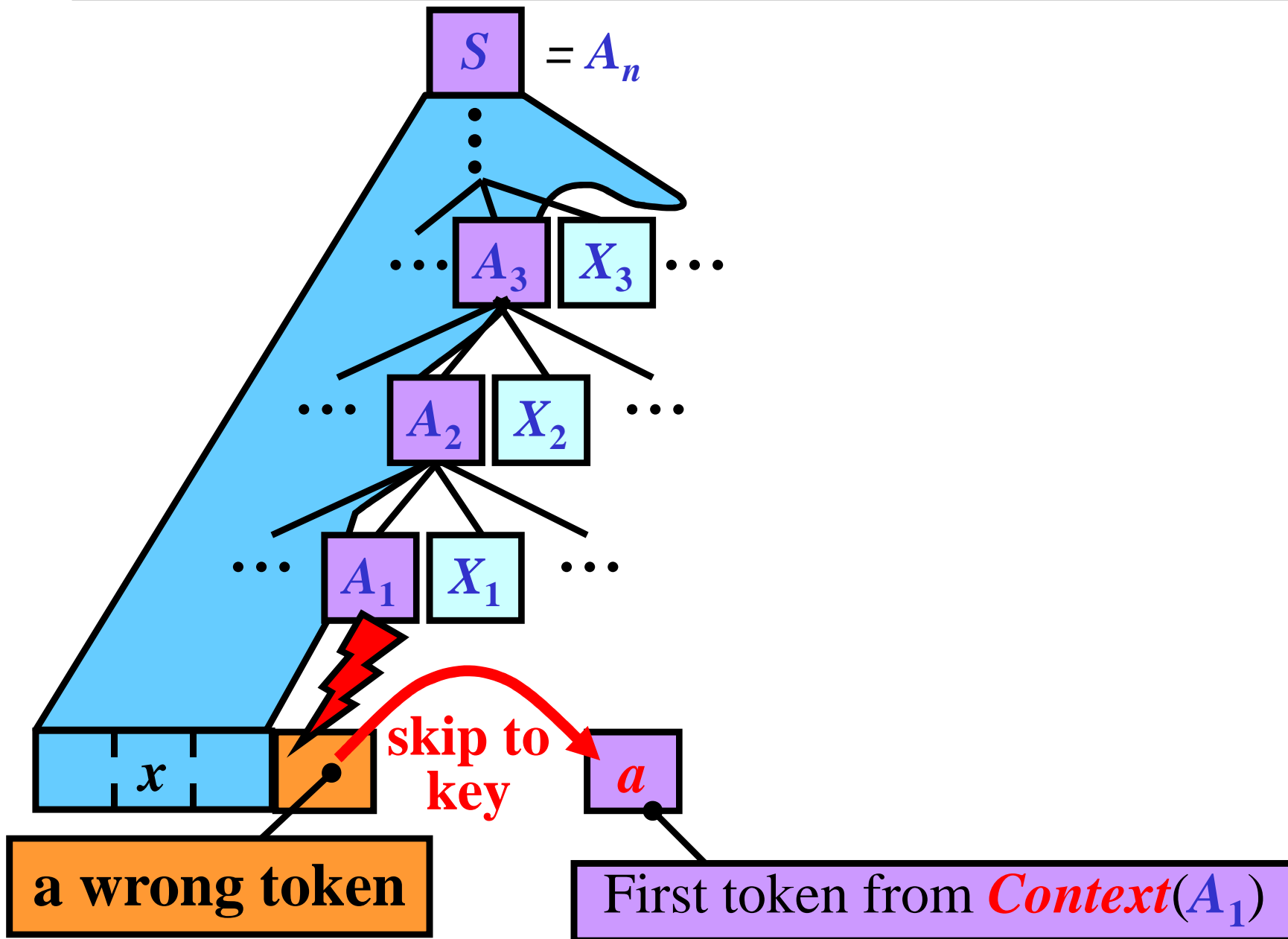
Panic-Mode Recovery: Illustration 1/2



Panic-Mode Recovery: Illustration 1/2

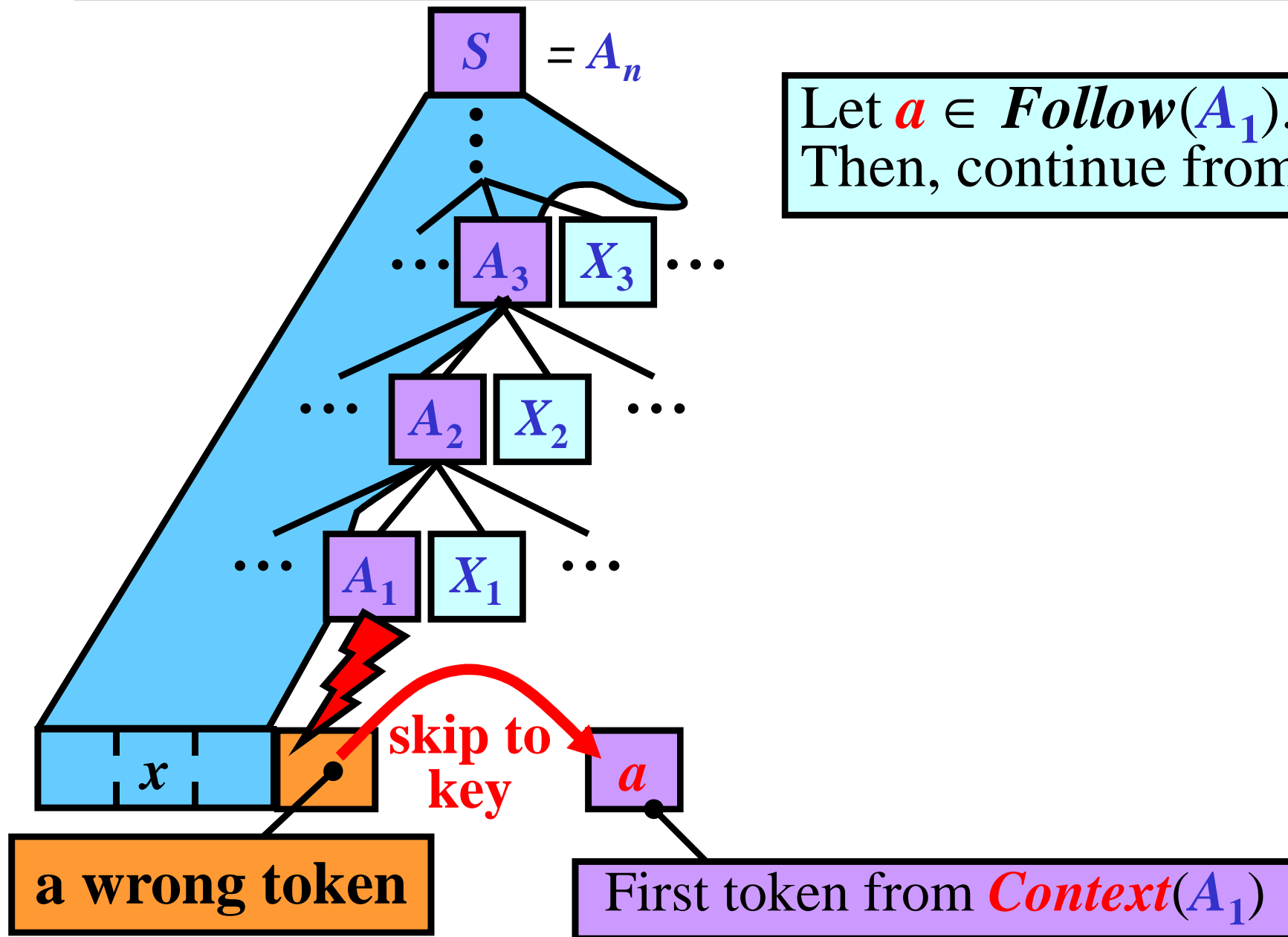


Panic-Mode Recovery: Illustration 1/2

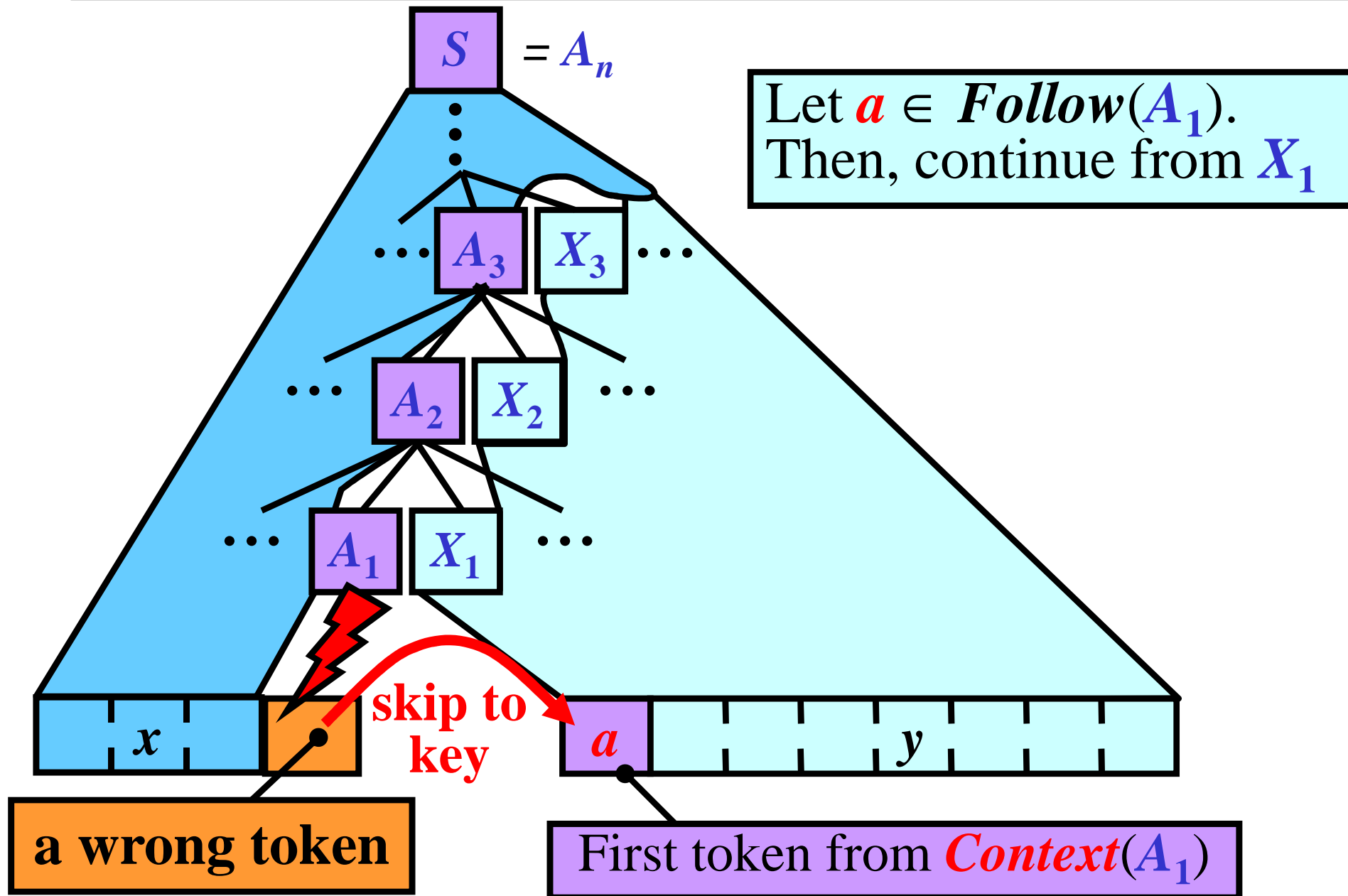


Panic-Mode Recovery: Illustration 1/2

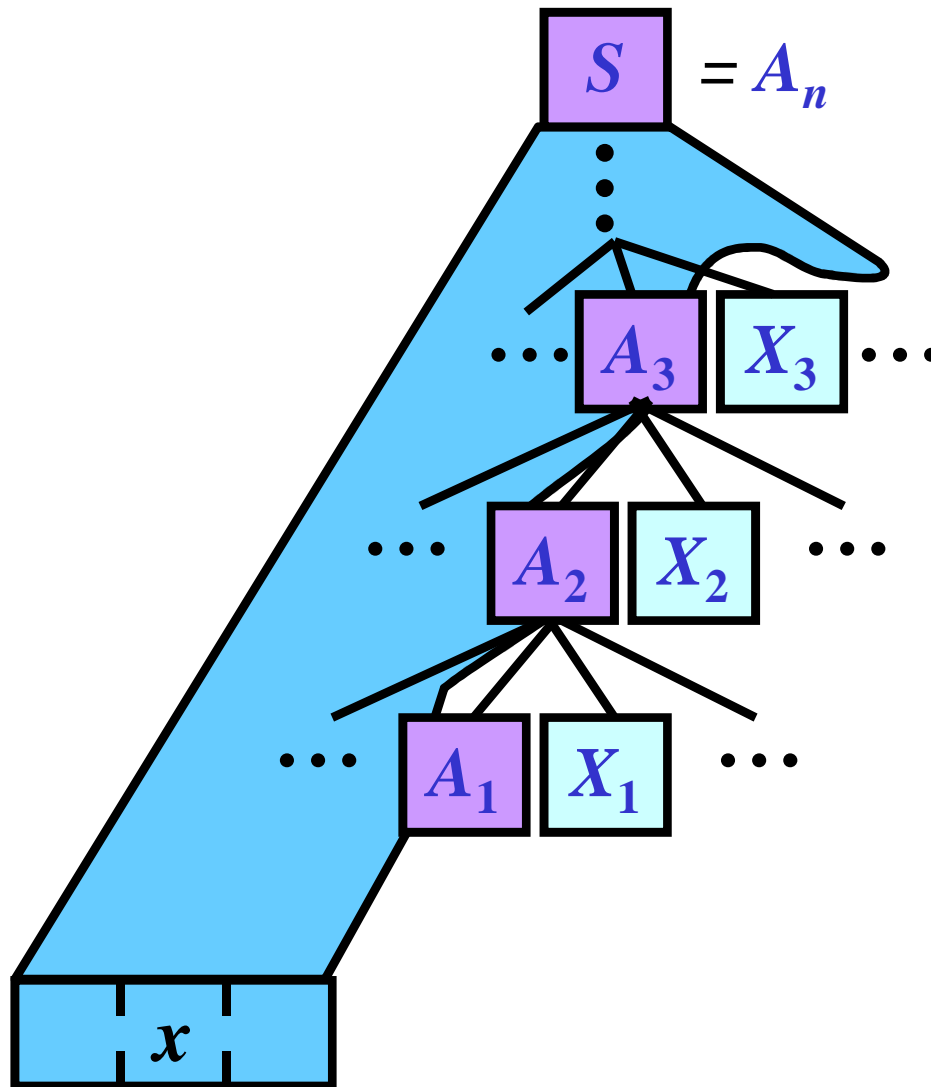
Let $a \in \text{Follow}(A_1)$.
Then, continue from X_1



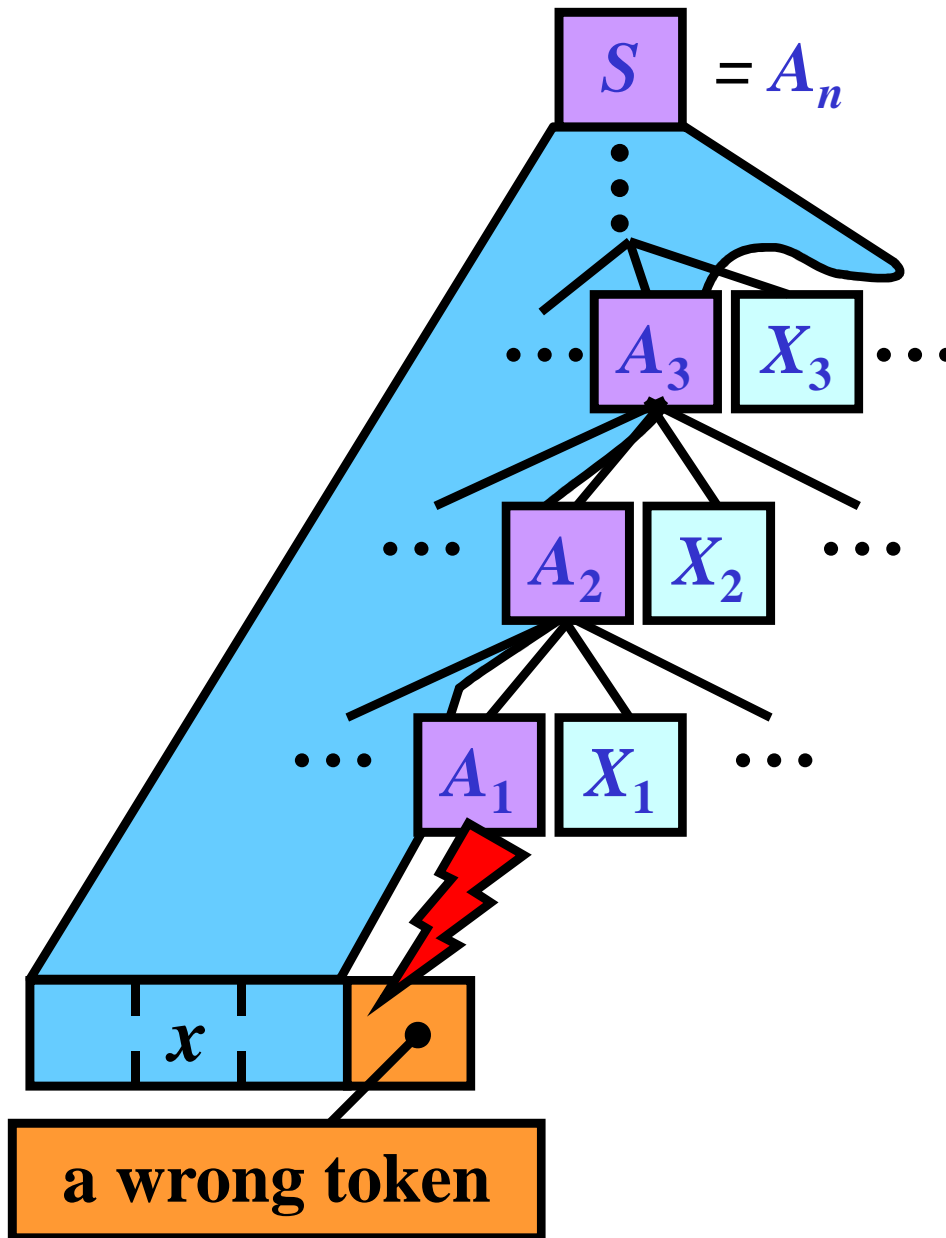
Panic-Mode Recovery: Illustration 1/2



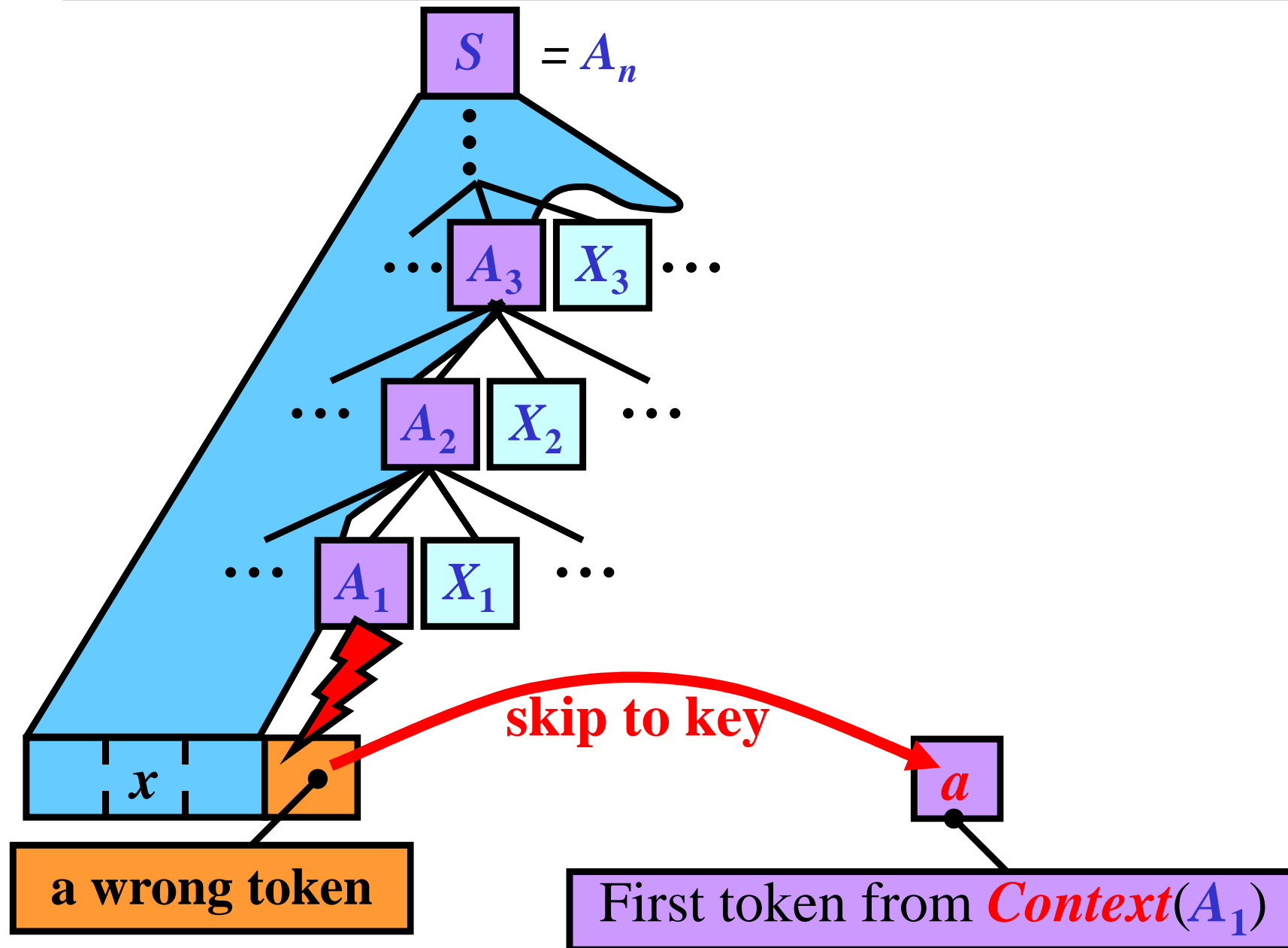
Panic-Mode Recovery: Illustration 2/2



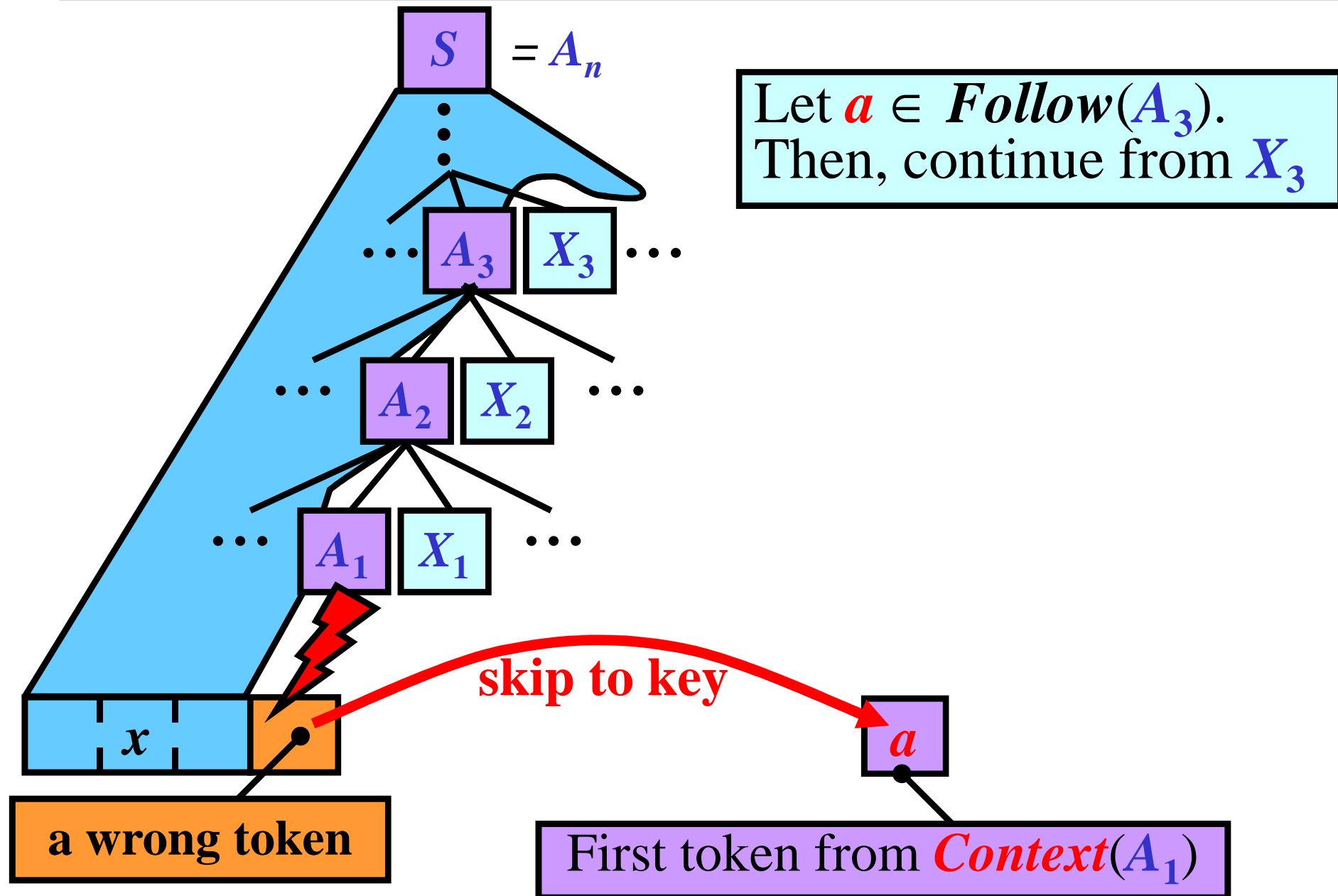
Panic-Mode Recovery: Illustration 2/2



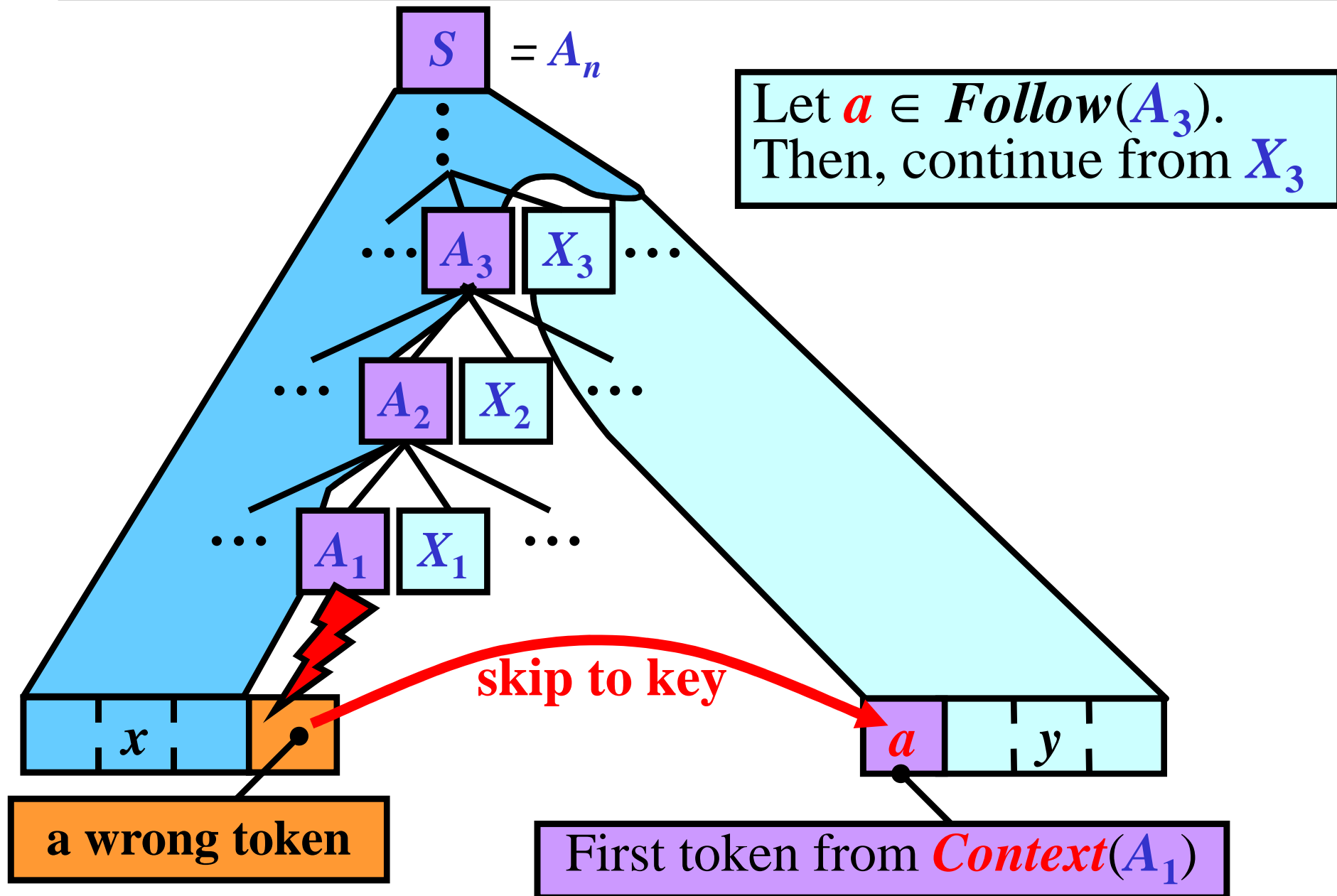
Panic-Mode Recovery: Illustration 2/2



Panic-Mode Recovery: Illustration 2/2



Panic-Mode Recovery: Illustration 2/2



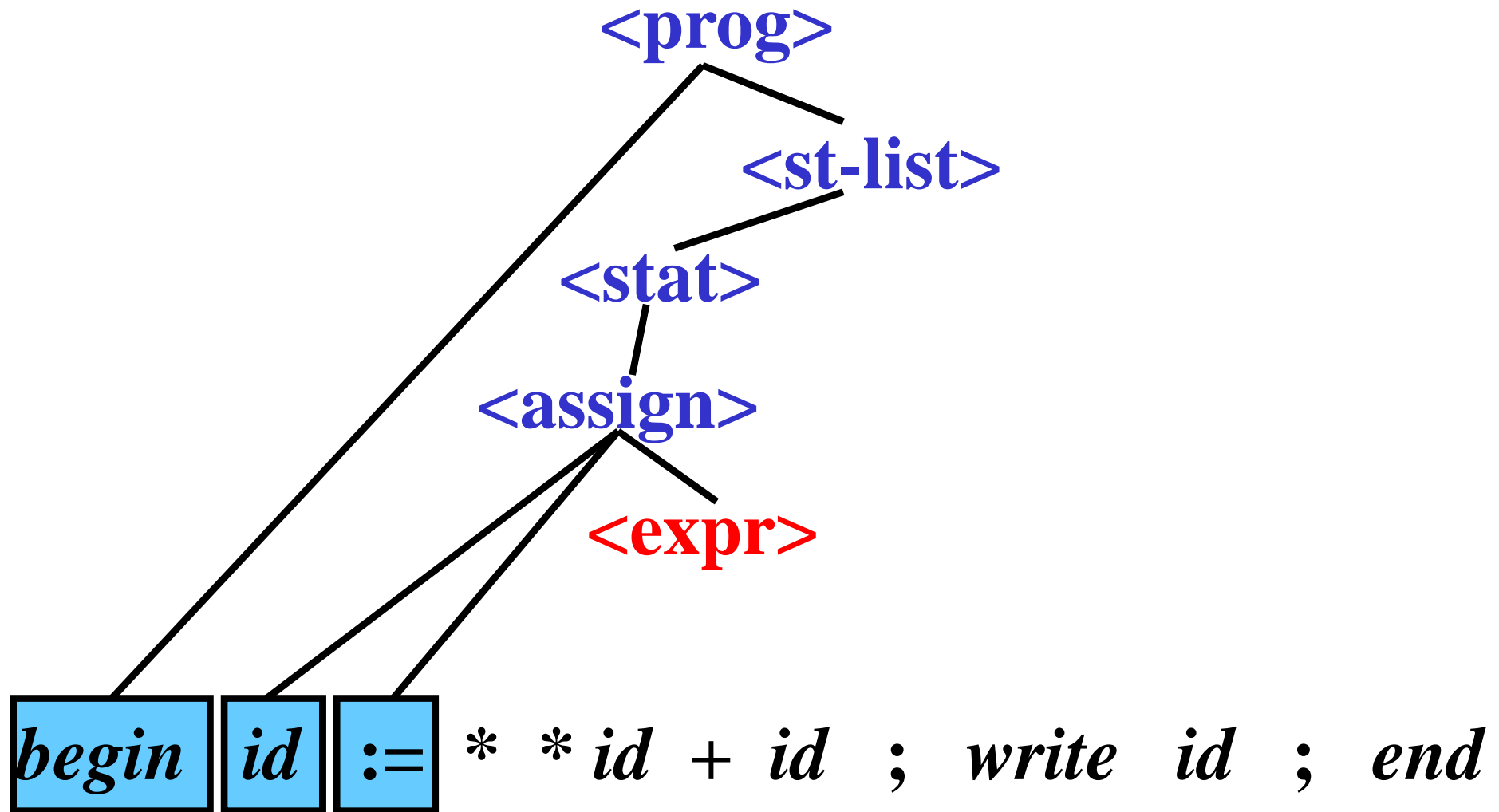
Context(X) for Predictive Parser: Variant I

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

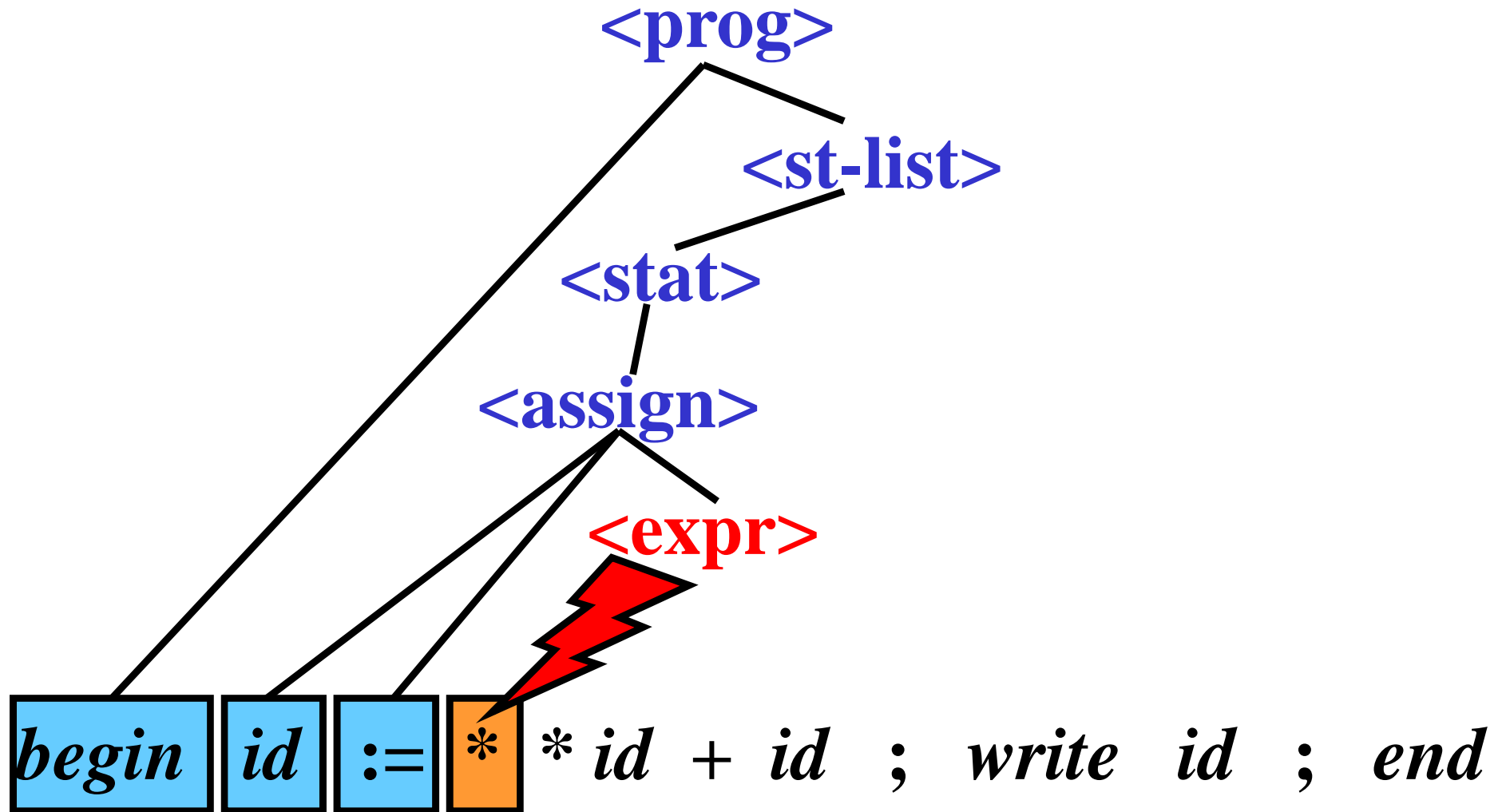
Context(A) = *Follow*(A) for every $A \in N$

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

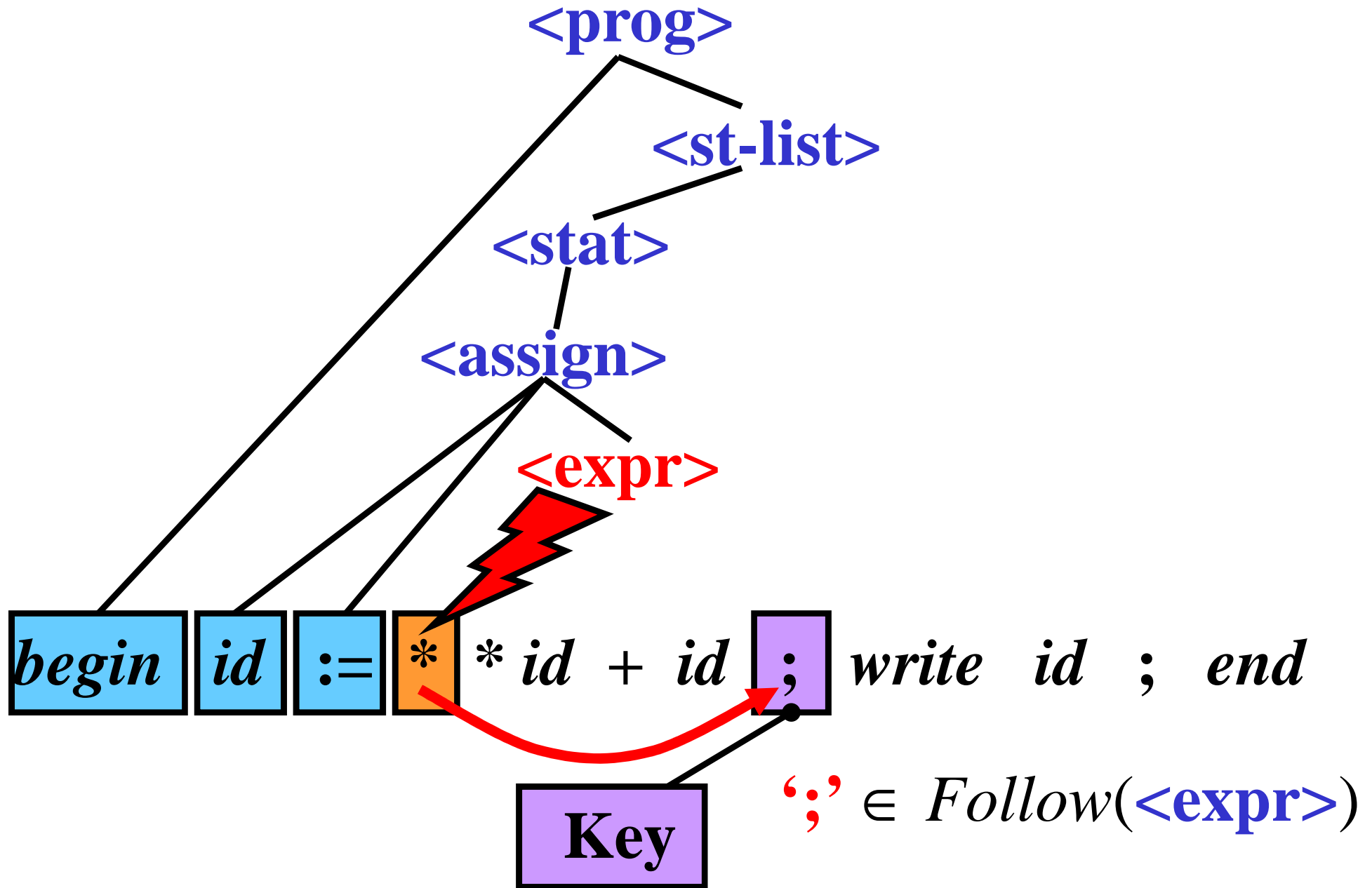
Variant I: Example



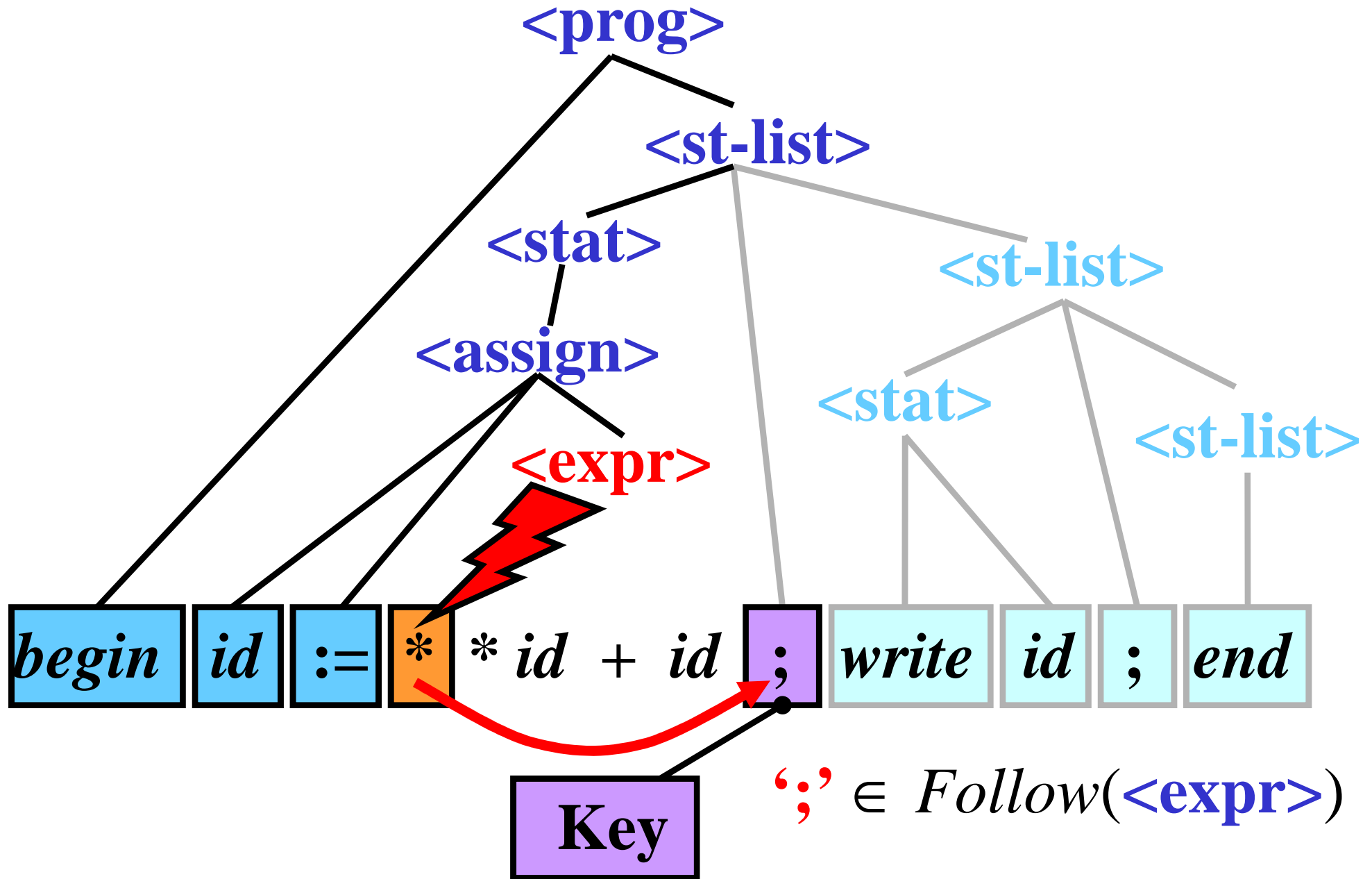
Variant I: Example



Variant I: Example



Variant I: Example



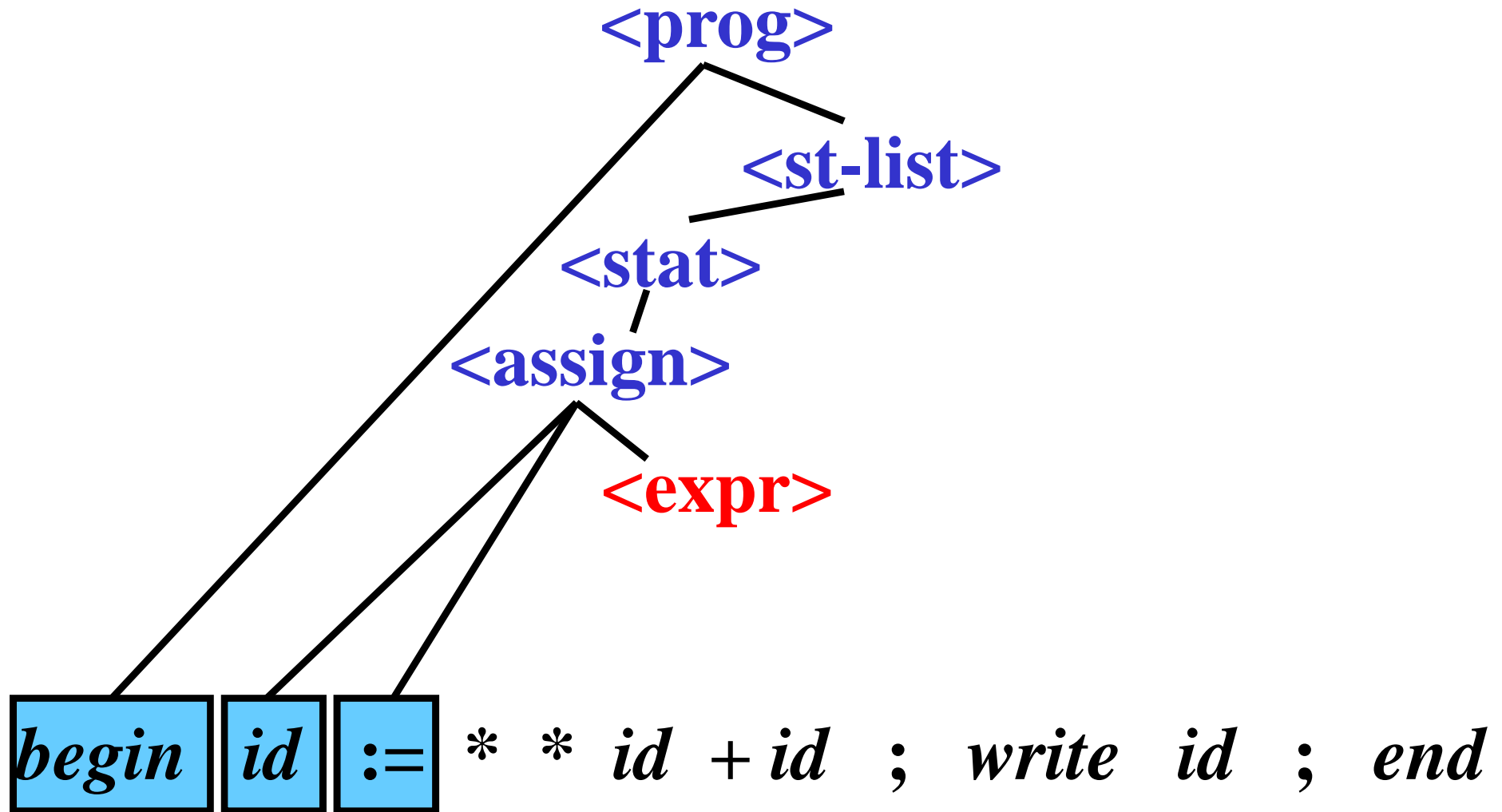
Context(X) for Predictive Parser: Variant II

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

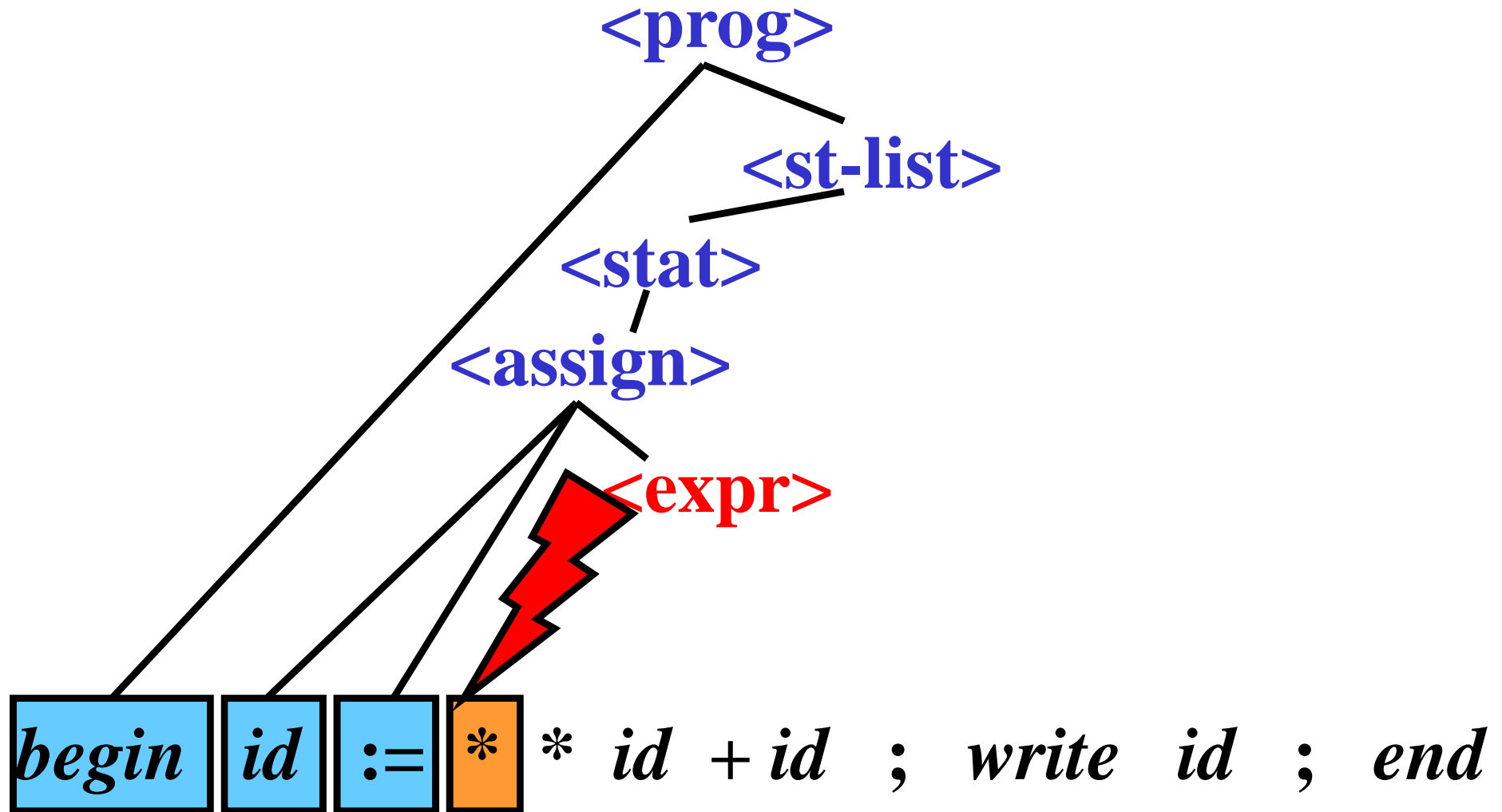
Context(A) = $First(A) \cup Follow(A)$ for every $A \in N$

- **Method:**
- Let A be pushdown top & no rule is applicable:
- **repeat**
 - $a := \text{GetNextToken};$
 - { 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 // $a \in Follow(A)$

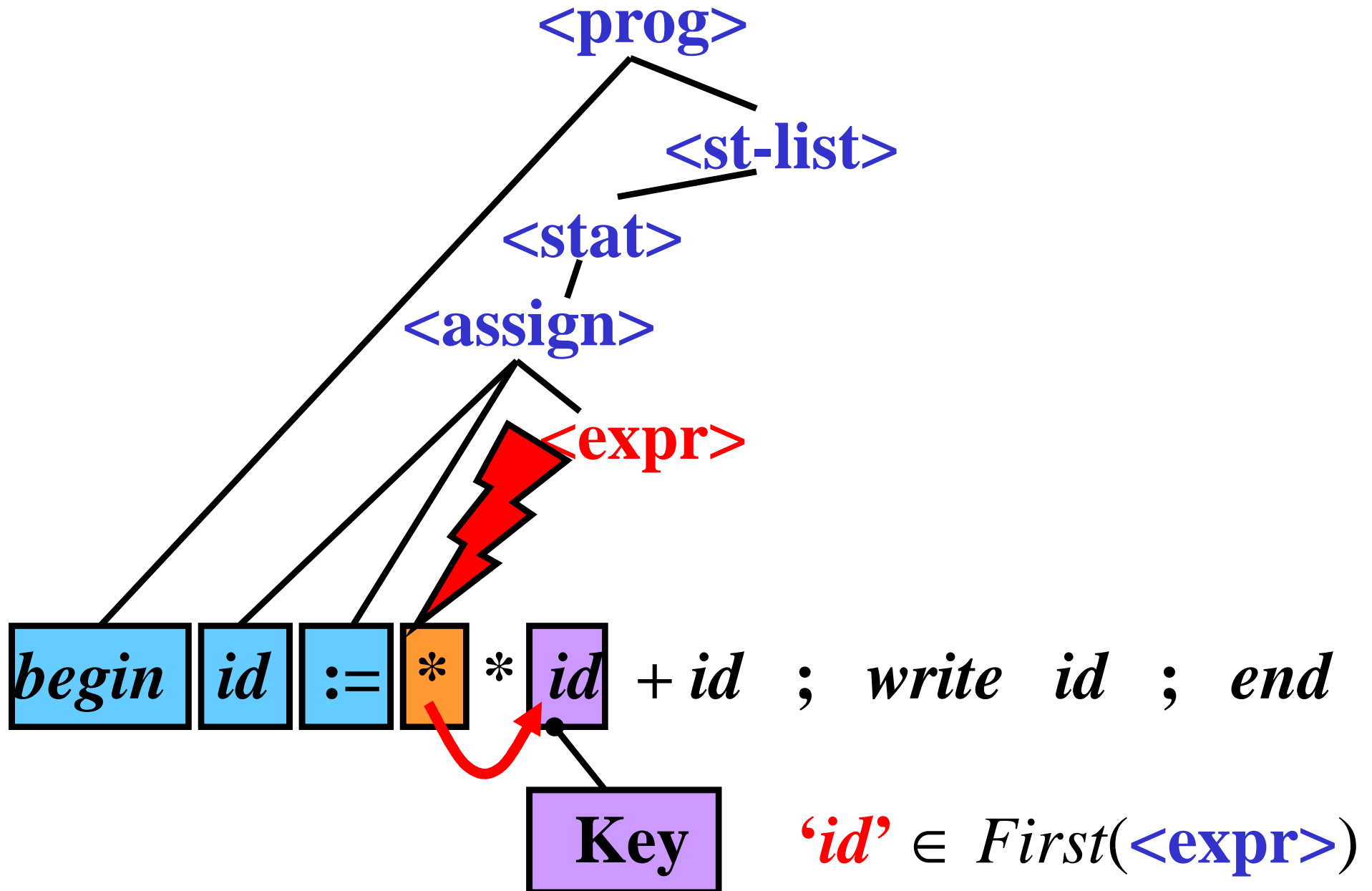
Variant II: Example



Variant II: Example



Variant II: Example



Variant II: Example

