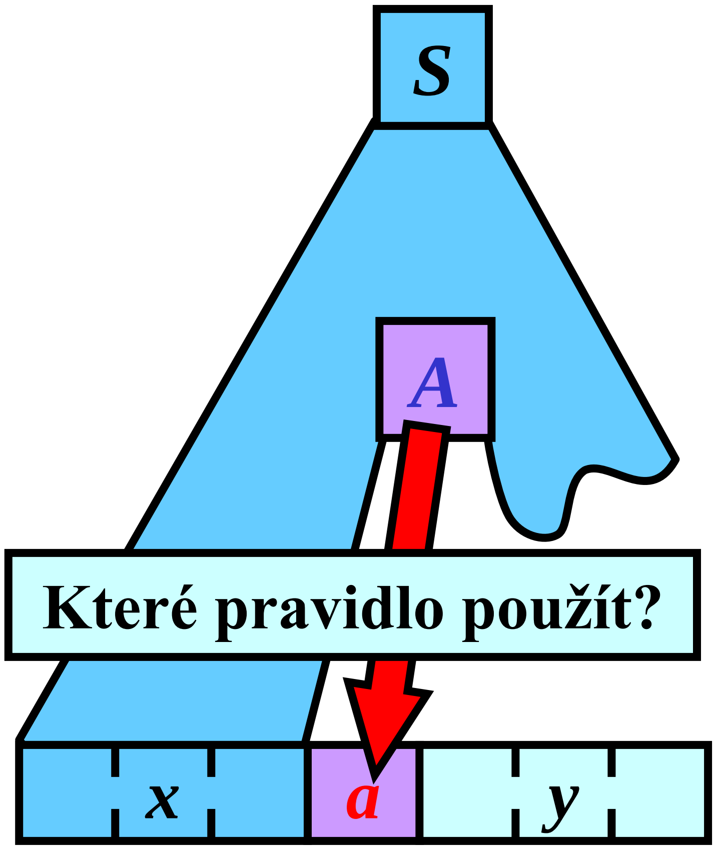
Kapitola V.
Syntaktická analýza shora dolů

SA shora dolů: Úvod

Problém:



Myšlenka:

Tabulka:

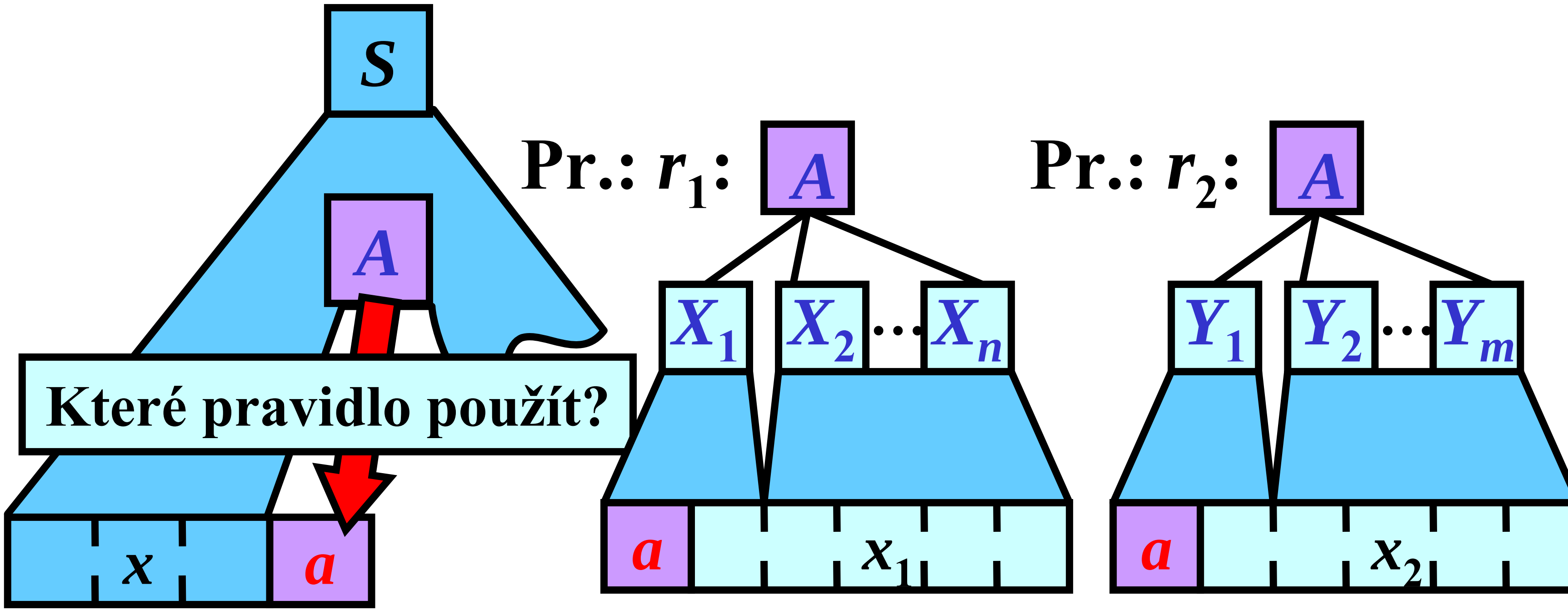
α	...	*a*	...
...			
A		α(*A*, *a*)	
...			

Použij: ***r***: $A \rightarrow x$

Otázka: Je možné sestrojit tuto tabulku pro **libovolnou** BKG?

Odpověď: NE

Výběr pravidla pomocí tabulky



Tabulka:

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

?

Použij r_1: $A \to X_1X_2\ldots X_n$

Použij r_2: $A \to Y_1Y_2\ldots Y_m$

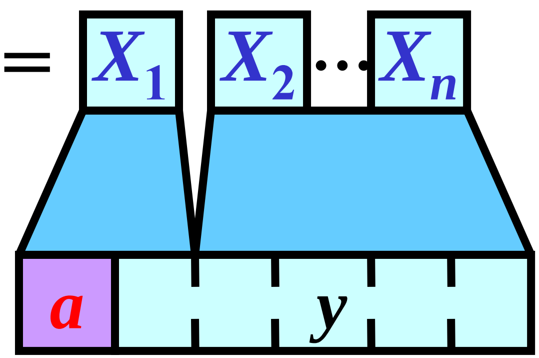
Množina *First*

Myšlenka: *First*(*x*) je množina všech terminálů, kterými může začínat větná forma derivovatelná z *x*

Definice: Nechť $G = (N, T, P, S)$ je BKG. Pro každé $x \in (N \cup T)^*$ je definováno $First(x)$ jako:
$First(x) = \{a: a \in T, x \Rightarrow^* ay; y \in (N \cup T)^*\}$.

Ilustrace:

$x =$



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

$$a \in First(x)$$

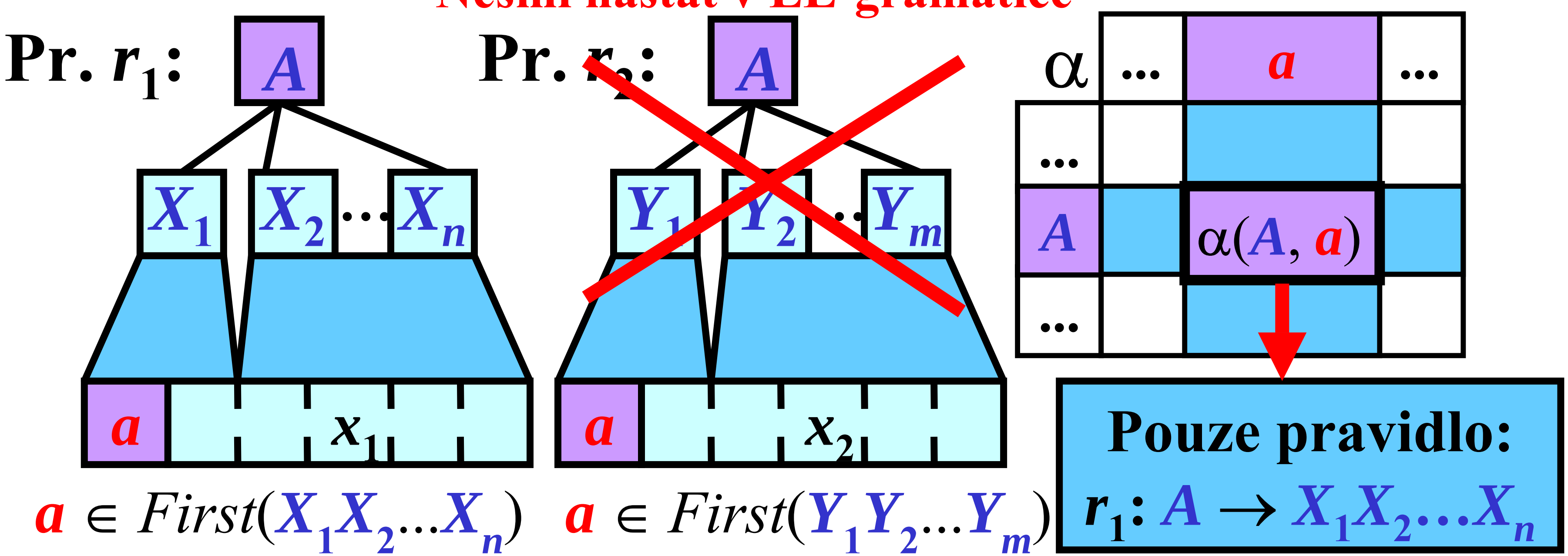
LL gramatika bez ε-pravidel

Definice: Nechť $G = (N, T, P, S)$ je BKG <u>bez ε-pravidel</u>. G je *LL gramatika*, pokud pro každé $a \in T$ a $A \in N$ existuje **maximálně jedno pravidlo** $A \rightarrow X_1X_2 \ldots X_n \in P$ takové, že: $a \in First(X_1X_2 \ldots X_n)$

Ilustrace:

Nesmí nastat v LL-gramatice

Tabulka:



Jednoduchý programovací jazyk (JPJ)

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

Pozn.: G_{JPJ} **je** <u>**LL gramatika**</u>

Příklad:

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

$\in$ JPJ

Algoritmus: *First*(X)

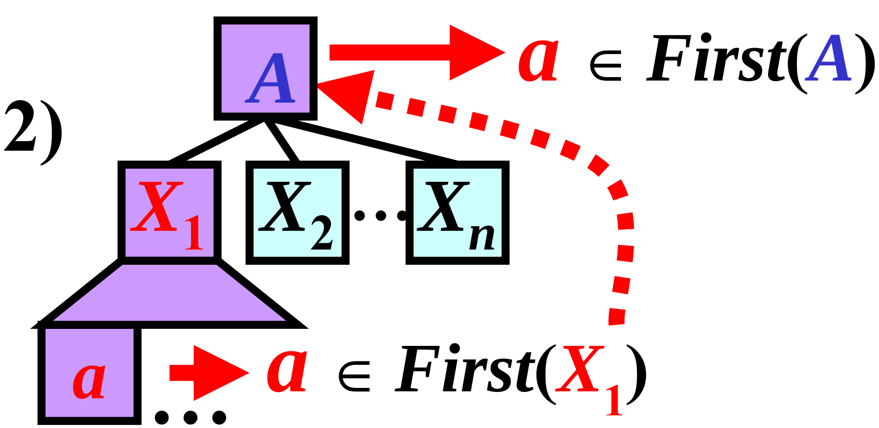
- **Vstup:** $G = (N, T, P, S)$ **bez ε-pravidel**
- **Výstup:** $First(X)$ pro každé $X \in N \cup T$

- **Metoda:**
- pro každé $a \in T$: $First(a) := \{a\}$
- **Používej následující pravidlo, dokud bude možné měnit nějakou množinu *First*:**
- **if** $A \rightarrow X_1X_2\ldots X_n \in P$, **then** přidej $First(X_1)$ do $First(A)$

Ilustrace:

1) pro každé $a \in T$:
$First(a) := \{a\}$,
protože $a \Rightarrow^0 a$

2)



First(*X*) pro JPJ: Příklad

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

<item> → **id** ∈ *P*: **přidej** *First*(**id**) **do** *First*(**<item>**)
<item> → **int** ∈ *P*: **přidej** *First*(**int**) **do** *First*(**<item>**)
Celkově: *First*(**<item>**) = {**id**, **int**}

<it-list> → **)** ∈ *P*: **přidej** *First*(**)**) **do** *First*(**<it-list>**)
<it-list> → **,** … ∈ *P*: **přidej** *First*(**,**) **do** *First*(**<it-list>**)
Celkově: *First*(**<it-list>**) = {**)**, **,**}

<stat> → **id** … ∈ *P*: **přidej** *First*(**id**) **do** *First*(**<stat>**)
<stat> → **write** … ∈ *P*: **přidej** *First*(**write**) **do** *First*(**<stat>**)
<stat> → **read** … ∈ *P*: **přidej** *First*(**read**) **do** *First*(**<stat>**)
Celkově: *First*(**<stat>**) = {**id**, **write**, **read**}

<st-list> → **end** ∈ *P*: **přidej** *First*(**end**) **do** *First*(**<st-list>**)
<st-list> → **<stat>** … ∈ *P*: **přidej** *First*(**<stat>**) **do** *First*(**<st-list>**)
Celkově: *First*(**<st-list>**) = {**id**, **write**, **read**, **end**}

<prog> → **begin** … ∈ *P*: **přidej** *First*(**begin**) **do** *First*(**<prog>**)
Celkově: *First*(**<prog>**) = {**begin**}

Konstrukce LL-tabulky

α	…	a	…
…			
A		$\alpha(A, a)$	
…			

$\alpha(A, a) = A \rightarrow X_1X_2\ldots X_n \in P$
pokud $a \in First(X_1)$; jinak
$\alpha(A, a)$ je prázdné $\Rightarrow$ **CHYBA**

Vytvořme: LL tabulku

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

id ∈ *First*(<stat>)

id ∈ *First*(id)

id ∈ *First*(id)

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

Zbytek vytvoříme analogicky.

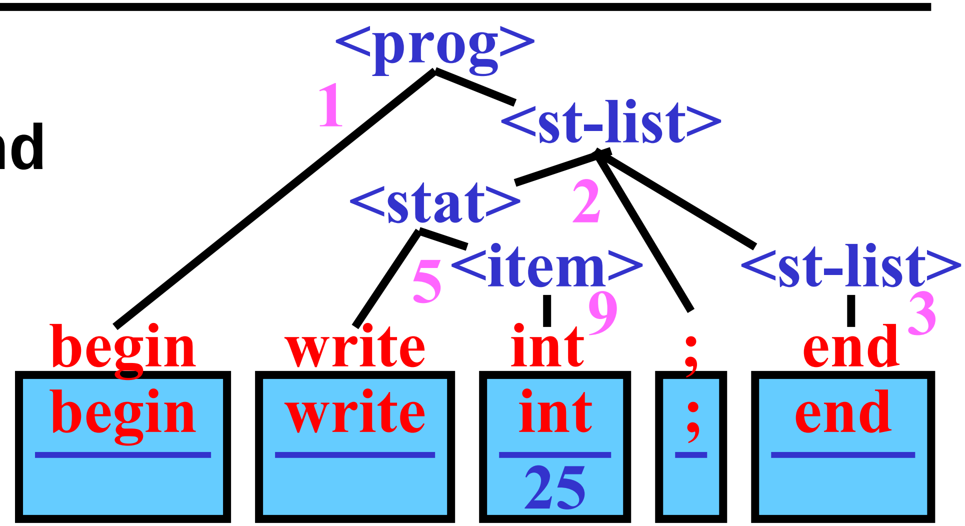
SA založená na LL-tabulce: Příklad

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

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

Zdrojový program:

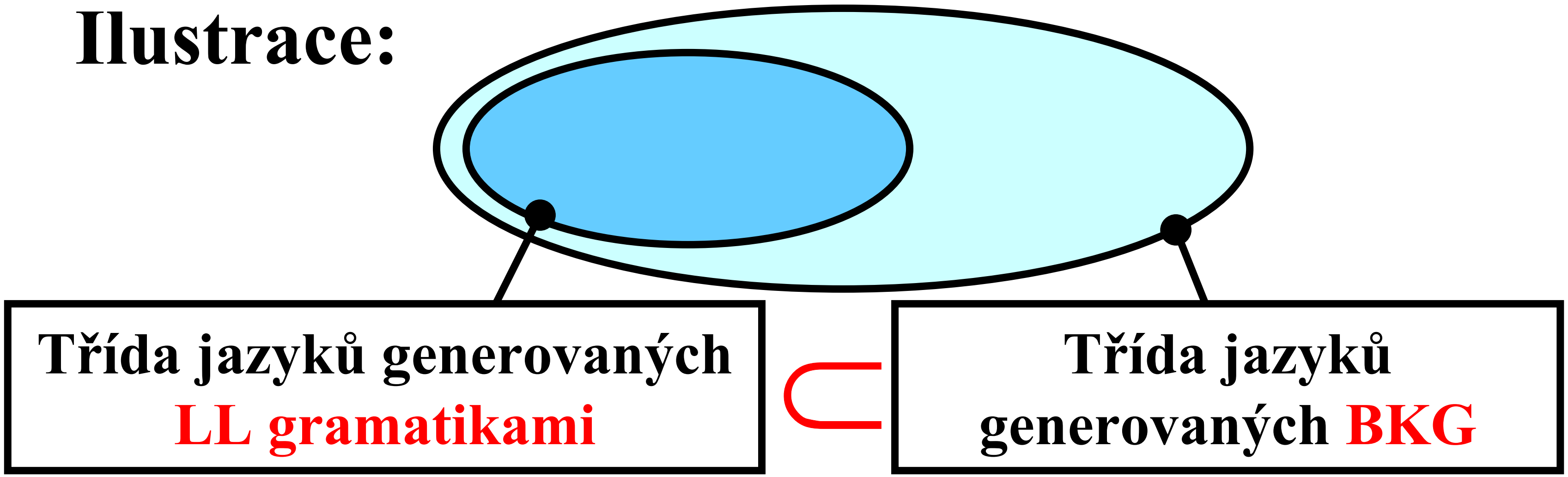
```
begin write 25; end
```



LL gramatiky: Úspěšné transformace

Obecně: BKG jsou silnější než LL-gramatiky

Ilustrace:



- **Některé** BKG mohou být převedeny na ekvivalentní LL gramatiky pomocí následujících transformací:

1) Faktorizace (vytýkání)

2) Odstranění levé rekurze

Pozn.: Pravidlo tvaru $\boldsymbol{A} \rightarrow \boldsymbol{Ax}$, kde $\boldsymbol{A} \in N$, $\boldsymbol{x} \in (N \cup T)^*$ se nazývá *levě rekurzívní pravidlo*.

Faktorizace (vytýkání)

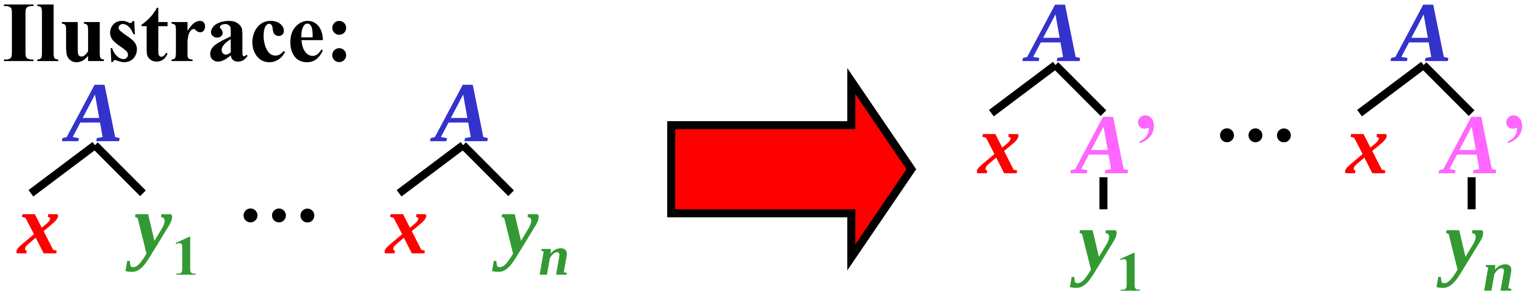
Myšlenka: Zaměnit pravidla tvaru:

$$A \rightarrow xy_1,\ A \rightarrow xy_2,\ \dots,\ A \rightarrow xy_n \text{ na:}$$

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

kde A' je nový neterminál

Ilustrace:



Příklad:

<stat> → write id
<stat> → write int

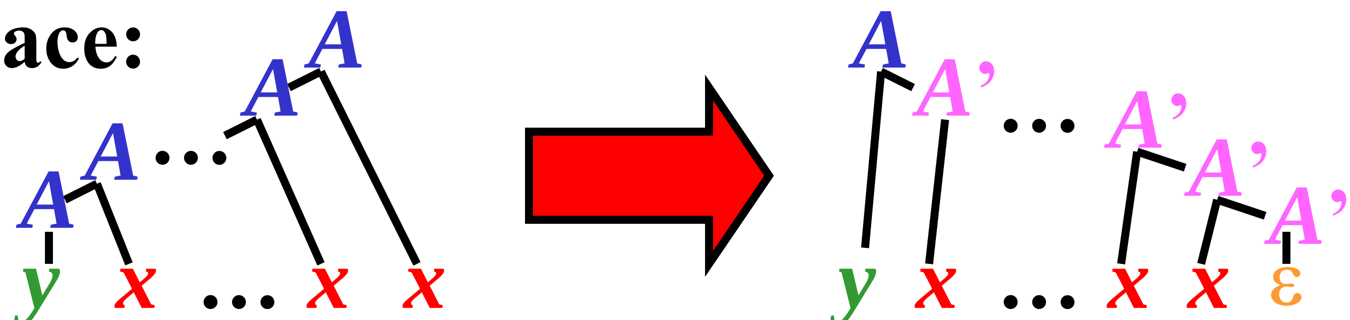
⇒

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

Odstranění levé rekurze

Myšlenka: Zaměnit pravidla tvaru: $A \rightarrow Ax$, $A \rightarrow y$ za: $A \rightarrow yA'$, $A' \rightarrow xA'$, $A' \rightarrow \varepsilon$, kde A' je nový neterminál.

Ilustrace:



Příklad:

$E \rightarrow E+T$
$E \rightarrow T$
⇒ $E \rightarrow TE'$, $E' \rightarrow +TE'$, $E' \rightarrow \varepsilon$

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

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

LL-gramatiky s ε-pravidly: Úvod

Proč ε-pravidla?

- Odstranění levé rekurze vytvoří ε-pravidla
- ε-pravidla často udělají gramatiku „čistější“

Zjednodušení této části:

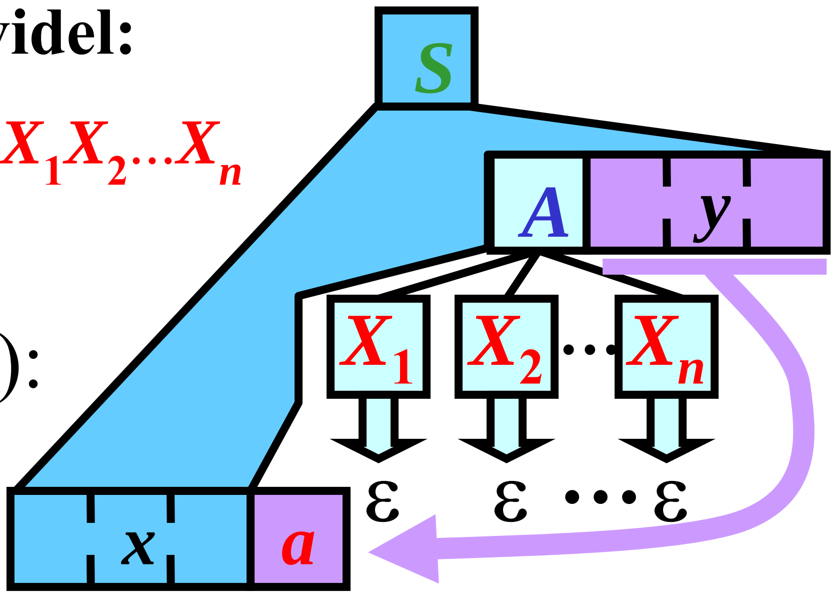
Budeme předpokládat, že každý vstupní řetězec je zakončen $.

Pozn.: $ značí „zakončovač“

Hlavní problém ε-pravidel:

Pravidlo r: $A \rightarrow X_1X_2\ldots X_n$

Možná: $a \notin First(A)$:



Pozn.: Musíme definovat další množiny: *Empty*, *Follow* a *Predict*.

Gramatika pro aritmetické výrazy

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

Příklad:

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

Množina *Empty*

Myšlenka: *Empty*(*x*) je množina, která obsahuje jediný prvek ε pokud *x* derivuje ε, jinak je prázdná

Definice: Nechť $G = (N, T, P, S)$ je BKG.
$Empty(x) = \{\varepsilon\}$ if $x \Rightarrow^* \varepsilon$; jinak
$Empty(x) = \emptyset$, kde $x \in (N \cup T)^*$.

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

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

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

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

Algoritmus: *Empty*(X)

- **Vstup:** $G = (N, T, P, S)$
- **Výstup:** *Empty*(X) pro každý symbol $X \in N \cup T$

- **Metoda:**
- pro každé $\boldsymbol{a} \in T$: *Empty*($\boldsymbol{a}$) := $\varnothing$
- pro každé $\boldsymbol{A} \in N$:

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

  else *Empty*($\boldsymbol{A}$) := $\varnothing$
- **Používej následující pravidlo, dokud bude možné měnit nějakou množinu *Empty*:**
  - **if** $\boldsymbol{A} \to \boldsymbol{X_1 X_2 \ldots X_n} \in P$ **and** *Empty*($\boldsymbol{X_i}$) $= \{\varepsilon\}$ pro všechna $i = 1, \ldots, n$ **then** *Empty*($\boldsymbol{A}$) $= \{\varepsilon\}$

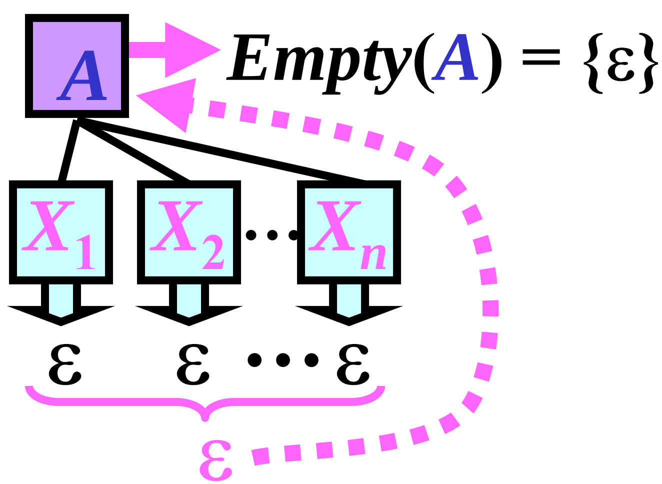
Předchozí algoritmus: Ilustrace

1) Pro každé $\boldsymbol{a} \in T$: $Empty(\boldsymbol{a}) := \emptyset$, protože $\boldsymbol{a} \Rightarrow^* \varepsilon$

2) Pro každé **r**: $\boldsymbol{A} \to \varepsilon \in \boldsymbol{P}$: $Empty(\boldsymbol{A}) := \{\varepsilon\}$, protože $\boldsymbol{A} \Rightarrow^{\mathbf{1}} \varepsilon$ **[r]**

3) Používej následující pravidlo, dokud bude možné měnit nějakou množinu *Empty* :

• **if** $A \to X_1X_2 \dots X_n \in P$ **and** $Empty(X_i) = \{\varepsilon\}$ pro všechna $i = 1,\dots,n$ **then** $Empty(A) = \{\varepsilon\}$



$Empty(X)$ pro G_{expr3}: Příklad

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

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

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

Inicializace:

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

- **Žádná *Empty* množina již nemůže být změněna**

Algoritmus: *First*(X)

- **Vstup:** $G = (N, T, P, S)$
- **Výstup:** $First(X)$ pro každé $X \in N \cup T$

- **Metoda:**
- pro každé $a \in T$: $First(a) := \{a\}$
- pro každé $A \in N$: $First(A) := \emptyset$
- **Používej následující pravidlo, dokud bude možné měnit nějakou množinu *First*:**
- **if** $A \rightarrow X_1X_2\ldots X_{k-1}X_k\ldots X_n \in P$ **then**
  - přidej všechny symboly z $First(X_1)$ do $First(A)$
  - **if** $Empty(X_i) = \{\varepsilon\}$ pro $i = 1,\ldots, k$-1, kde $k \leq n$ **then** přidej všechny symboly z $First(X_k)$ do $First(A)$

Předchozí algoritmus: Ilustrace

1) pro každé $a \in T$: $First(a) := \{a\}$, protože $a \Rightarrow^0 a$

2) pro každé $A \in N$: $First(A) := \emptyset$ (Inicializace)

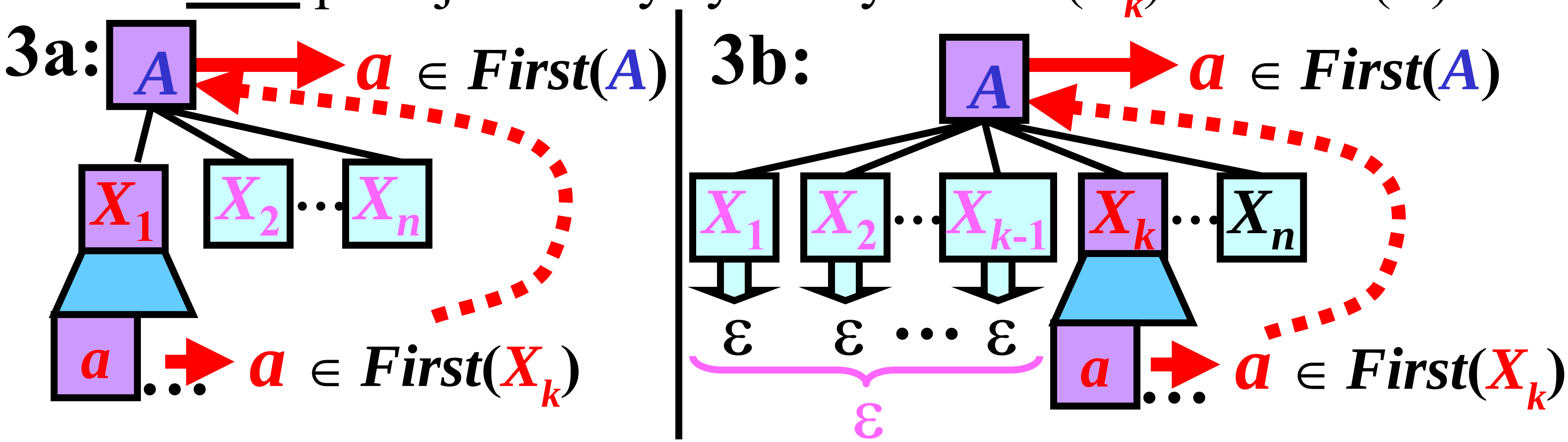
---

3) Používej následující pravidlo, dokud bude možné měnit nějakou množinu *First*:

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

3a) přidej všechny symboly z $First(X_1)$ do $First(A)$

3b) **if** $Empty(X_i) = \{\varepsilon\}$ pro $i = 1,\dots, k$-1, kde $k < n$ **then** přidej všechny symboly z $First(X_k)$ do $First(A)$:



$First(X)$ for G_{expr3}: Příklad

Inicializace:

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

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

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

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

$T' \to *FT' \in P$: **přidej** $First\,(*) = \{*\}$ **do** $First(T')$

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

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

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

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

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

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

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

- **Žádná *First* množina již nemůže být změněna.**

First(*X*) & *Empty*(*X*) pro G_{expr3}: Celkově

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

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

Množina *Empty* pro všechna $X \in N \cup T$:

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

Množina *First* pro všechna $X \in N \cup T$:

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

Pozn.: pro každé $a \in T$: $Empty(a) = \emptyset$, $First(a) = \{a\}$

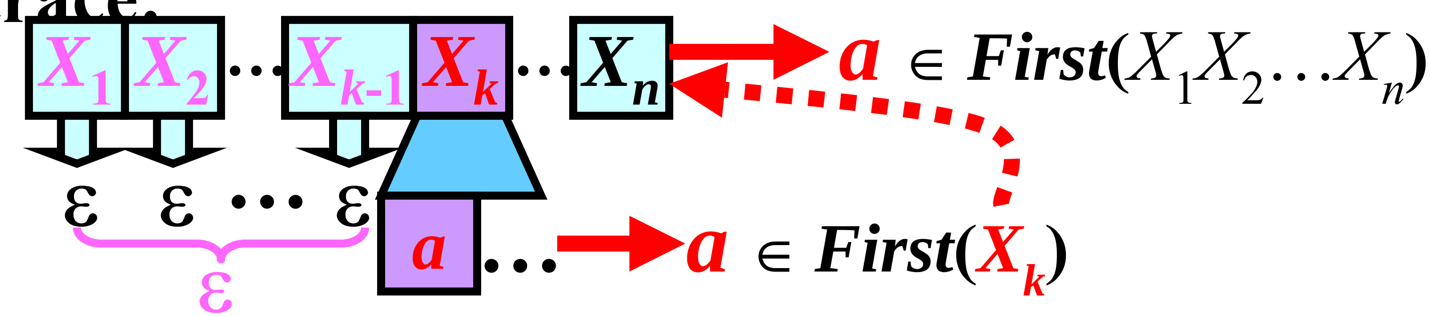
Algoritmus: $First(X_1X_2\ldots X_n)$

- **Vstup:** $G = (N, T, P, S)$; $First(X)$ & $Empty(X)$ pro každé $X \in N \cup T$; $x = X_1X_2\ldots X_n$, kde $x \in (N \cup T)^+$
- **Výstup:** $First(X_1X_2\ldots X_n)$

- **Metoda:**
- $First(X_1X_2\ldots X_n) := First(X_1)$
- **Používej následující pravidlo, dokud bude možné měnit množinu** $\boldsymbol{First(X_1X_2 \ldots X_{k-1} X_k \ldots X_n)}$**:**
  - **if** $Empty(\boldsymbol{X_i}) = \{\varepsilon\}$ pro $i = 1,\ldots,k$-1, kde $k \leq n$
    then přidej všechny symboly z $First(\boldsymbol{X_k})$ do $First(X_1X_2\ldots X_n)$

! Pozn.: $First(\varepsilon) = \emptyset$

Ilustrace:



$First(X_1X_2\ldots X_n)$: Příklad

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

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

Určeme: $First(E'T'FET)$

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

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

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

3) $First(E'T'\underline{F}ET)$: přidej $First(F) = \{i, (\}$ do $First(E'T'FET)$

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

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

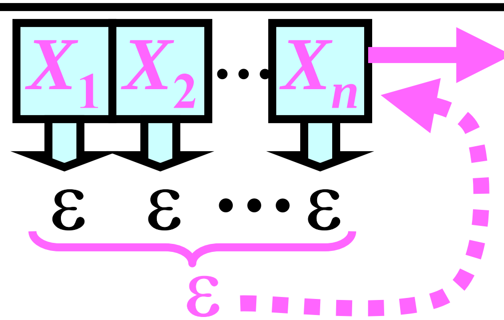
Algoritmus: *Empty*$(X_1X_2\ldots X_n)$

- **Vstup:** $G = (N, T, P, S)$; $Empty(X)$ pro všechna $X \in N \cup T$;
  $x = X_1X_2\ldots X_n$, kde $x \in (N \cup T)^+$
- **Výstup:** $Empty(X_1X_2\ldots X_n)$

- **Metoda:**
- **if** $Empty(X_i) = \{\varepsilon\}$ pro all $i = 1,\ldots,n$ **then**
  $Empty(X_1X_2\ldots X_n) := \{\varepsilon\}$
  else
  $Empty(X_1X_2\ldots X_n) := \emptyset$

! Pozn.: $Empty(\varepsilon) = \{\varepsilon\}$

Ilustrace:



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

$Empty(X_1X_2\ldots X_n)$: Příklad

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

Množina *Empty* pro všechna $X \in N$:

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

Určeme: $Empty(E'T')$

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

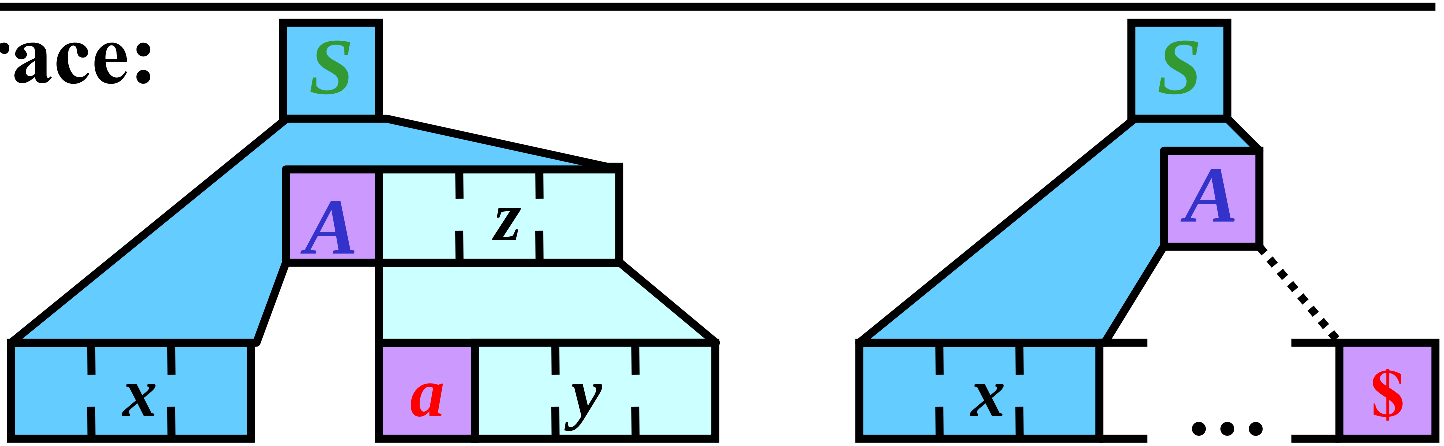
Množina *Follow*

Myšlenka: *Follow*(*A*) je množina všech terminálů, které se mohou vyskytovat vpravo od *A* ve větné formě.

Definice: Nechť $G = (N, T, P, S)$ je BKG. Pro všechna $A \in N$ definujeme množinu *Follow*(A):

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

Ilustrace:



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

$a \in Follow(A)$

$S \Rightarrow^* xA$

$\$ \in Follow(A)$

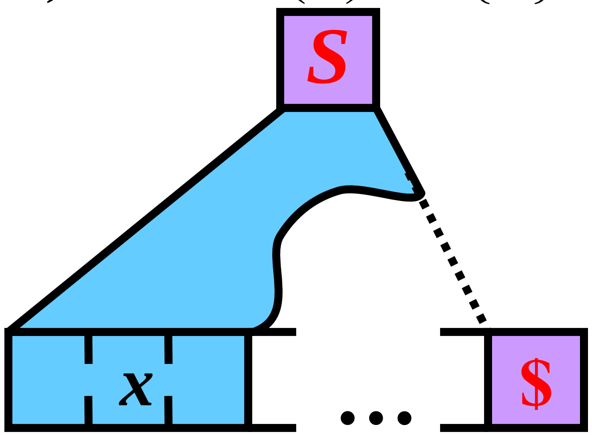
Algoritmus: *Follow*(*A*)

- **Vstup:** $G = (N, T, P, S)$;
- **Výstup:** *Follow*(*A*) pro každé $A \in N$

- **Metoda:**
- *Follow*(**S**) := {**$**};
- **Používej následující pravidlo, dokud bude možné měnit nějakou množinu *Follow*:**
- **if** $A \to xBy \in P$ **then**
  - **if** $y \neq \varepsilon$ **then**
    přidej všechny symboly z *First*(*y*) do *Follow*(**B**);
  - **if** *Empty*(*y*) = {ε} **then**
    přidej všechny symboly z *Follow*(**A**) do *Follow*(**B**);

Předchozí algoritmus: Ilustrace

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



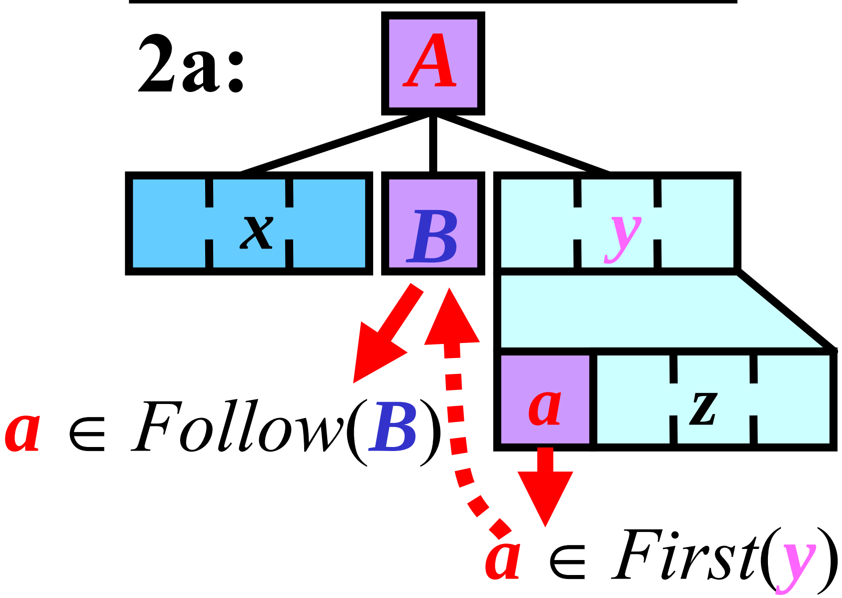
2) Používej následující pravidlo, dokud bude možné měnit *Follow*:

- **<u>if</u>** ***A*** → *x**B**y* ∈ *P* **<u>then</u>**

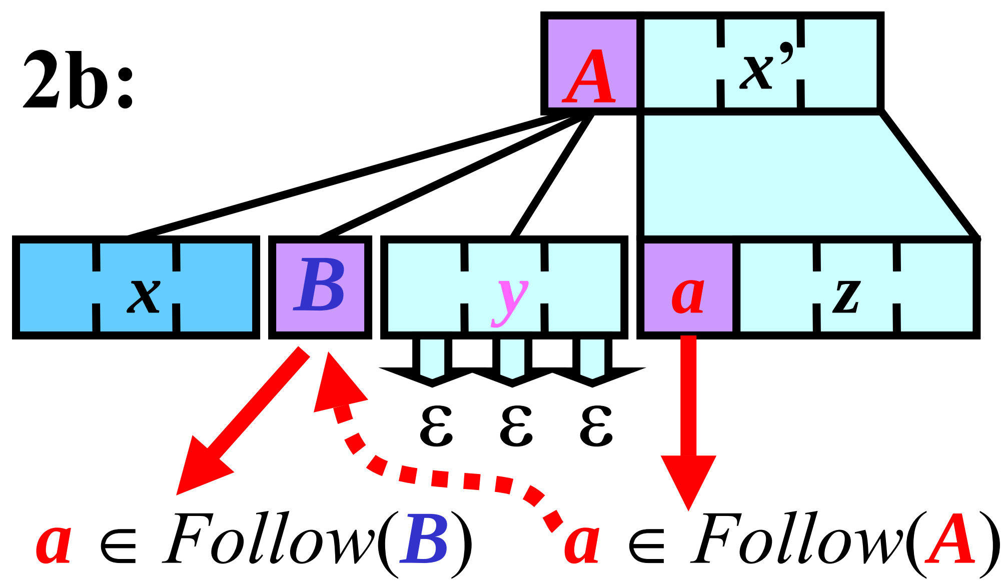
  2a) <u>if</u> *y* ≠ ε **<u>then</u>** přidej všechny symboly z *First*(*y*) do *Follow*(***B***)

  2b) <u>if</u> *Empty*(*y*) = {ε} **<u>then</u>** přidej všechny symboly z *Follow*(***A***) do *Follow*(***B***)

2a:



2b:



Follow(*X*) pro G_{expr3}: Příklad 1/3

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

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

---

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

Celkově: *Follow*(*E*) = {$,)}

---

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

ε: *Empty*(ε) = {ε}

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

$E \rightarrow T\underbrace{E'} \in P$: **přidej** *Follow*(*E*) = {$,)} **do** *Follow*(*T*)

Empty(*E'*) = {ε}

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

Follow(*X*) pro G_{expr3}: Příklad 2/3

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

---

3) $E' \to +TE'\underbrace{}_{\varepsilon} \in P$: **přidej** *Follow*(*E'*) = {**$**, **)**} **do** *Follow*(*E'*)
ε: *Empty*(ε) = {ε}

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

$E' \to +T\underbrace{E'} \in P$: **přidej** *Follow*(*E'*) = {**$**, **)**} **do** *Follow*(*T*)
Empty(*E'*) = {ε}

Celkově: Nic nezměněno

---

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

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

$T \to F\underbrace{T'} \in P$: **přidej** *Follow*(*T*) = {+, **$**, **)**} **do** *Follow*(*F*)
Empty(*T'*) = {ε}

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

$Follow(X)$ pro G_{expr3}: Příklad 3/3

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

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

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

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

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

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

Konec: Žádná množina *Follow* nemůže být změněna.

Celkově:

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

Množina *Predict*

Myšlenka: *Predict*(*A* → *x*) je množina všech terminálů, které mohou být aktuálně nejlevěji vygenerovány, pokud pro libovolnou větnou formu použijeme pravidlo *A* → *x*.

Definice: Nechť $G = (N, T, P, S)$ je BKG. Pro každé $A \rightarrow x \in P$ definujeme množinu $Predict(A \rightarrow x)$ jako:

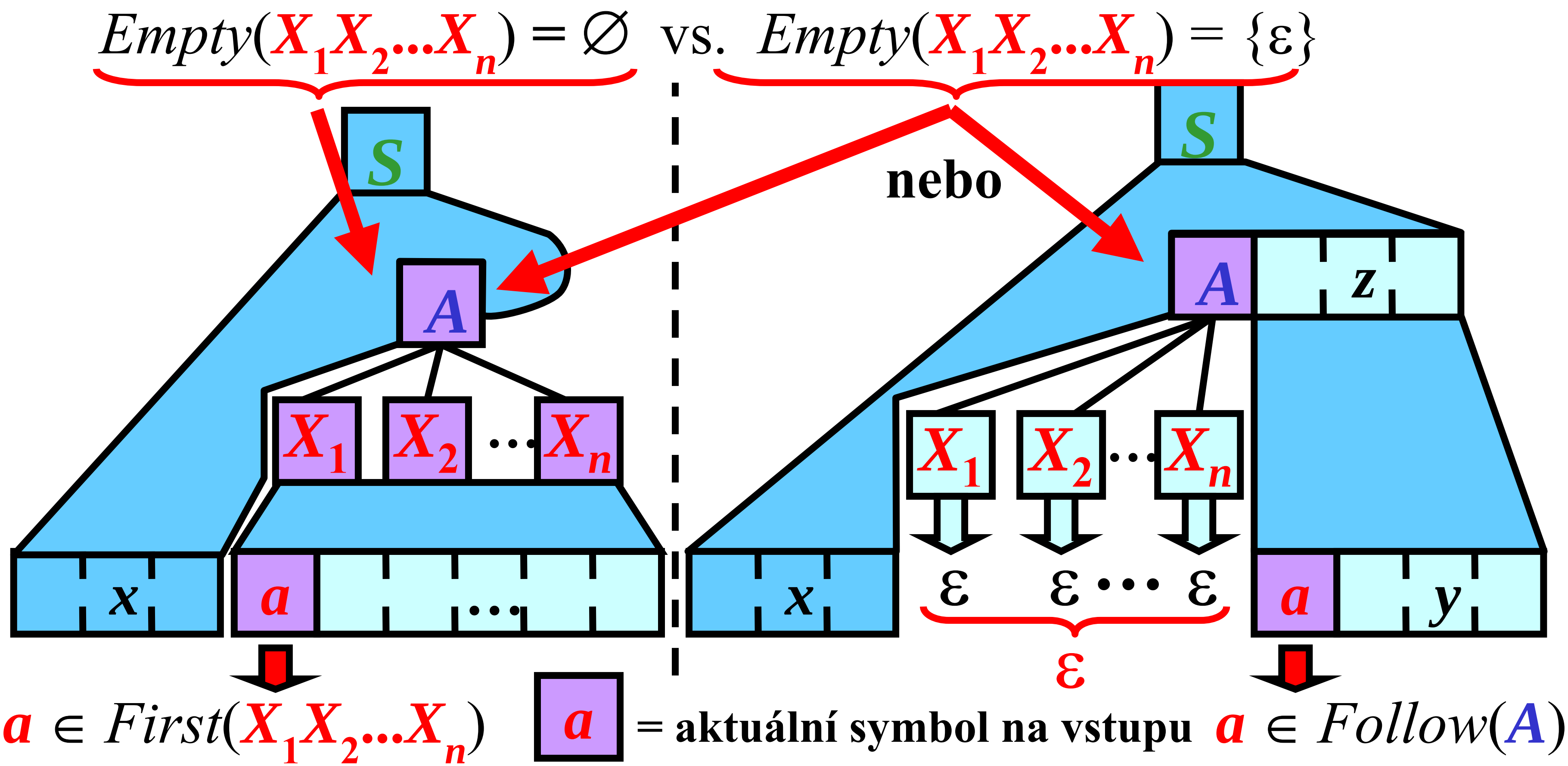
- pokud $Empty(x) = \{\varepsilon\}$ potom:

  $$Predict(A \rightarrow x) = First(x) \cup Follow(A)$$

- jinak pokud $Empty(x) = \emptyset$ potom:

  $$Predict(A \rightarrow x) = First(x)$$

Množina *Predict*($A \to X_1X_2\ldots X_n$): Ilustrace



Celkově: **<u>if</u>** $Empty(X_1X_2\ldots X_n) = \{\varepsilon\}$ **<u>then</u>**

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

<u>else</u> $Predict(A \to X_1X_2\ldots X_n) = First(X_1X_2\ldots X_n)$

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

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

1: ***E*** → ***TE'***

Empty(***TE'***) = ∅, protože *Empty*(***T***) = ∅

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

2: ***E'*** → +***TE'***

Empty(+***TE'***) = ∅, protože *Empty*(***T***) = ∅

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

3: ***E'*** → ε

Empty(ε) = {ε}

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

4: ***T*** → ***FT'***

Empty(***FT'***) = ∅, protože *Empty*(***F***) = ∅

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

$Predict(A \to x)$ pro G_{expr3}: Příklad 2/2

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

5: $T' \to *FT'$

$Empty(*FT') = \emptyset$, protože $Empty(F) = \emptyset$

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

6: $T' \to \varepsilon$

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

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

7: $F \to (E)$

$Empty((E)) = \emptyset$, protože $Empty(E) = \emptyset$

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

8: $F \to i$

$Empty(i) = \emptyset$

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

Konstrukce LL-tabulky

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

α(*A*, *a*) = $A \rightarrow X_1X_2\ldots X_n \in P$ pokud *a* ∈ *Predict*($A \rightarrow X_1X_2\ldots X_n$); jinak α(*A*, *a*) is je *prázdné*.

Určeme: LL tabulku pro G_{expr1}

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

i ∈ ***Predict***(1)

i ∈ ***Predict***(4)

i ∈ ***Predict***(8)

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

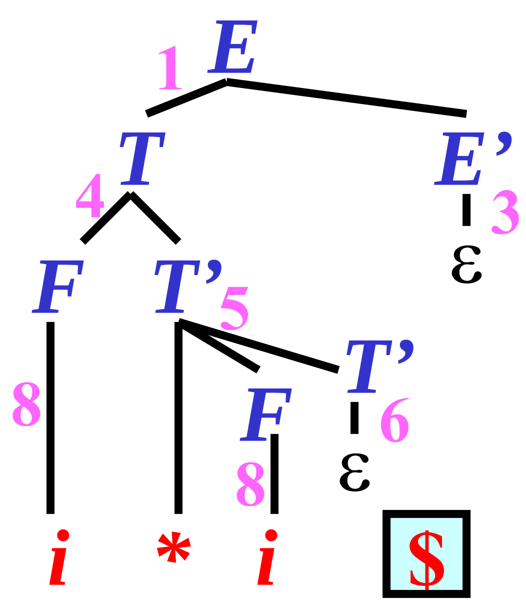
Zbytek tabulky by se sestrojil analogicky.

SA založená na LL-tabulce: Příklad

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

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

Otázka: $i * i \in L(G_{expr3})$?



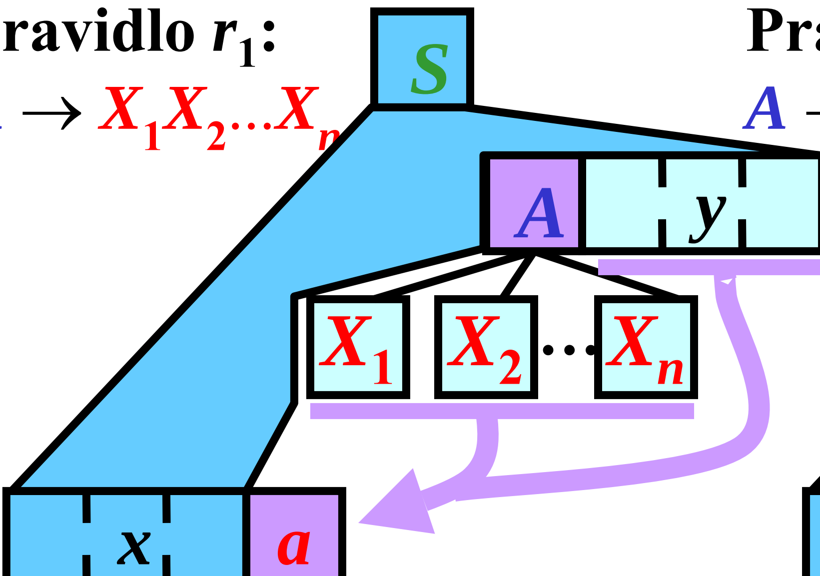
LL gramatiky s ε-pravidly: Definice

Definice: Nechť $G = (N, T, P, S)$ je BKG. G je *LL-gramatika*, pokud pro každé $a \in T$ a každé $A \in N$ existuje **maximálně jedno** A-pravidlo tvaru $A \to X_1X_2 \dots X_n \in P$ a platí: $a \in Predict(A \to X_1X_2 \dots X_n)$

Ilustrace:

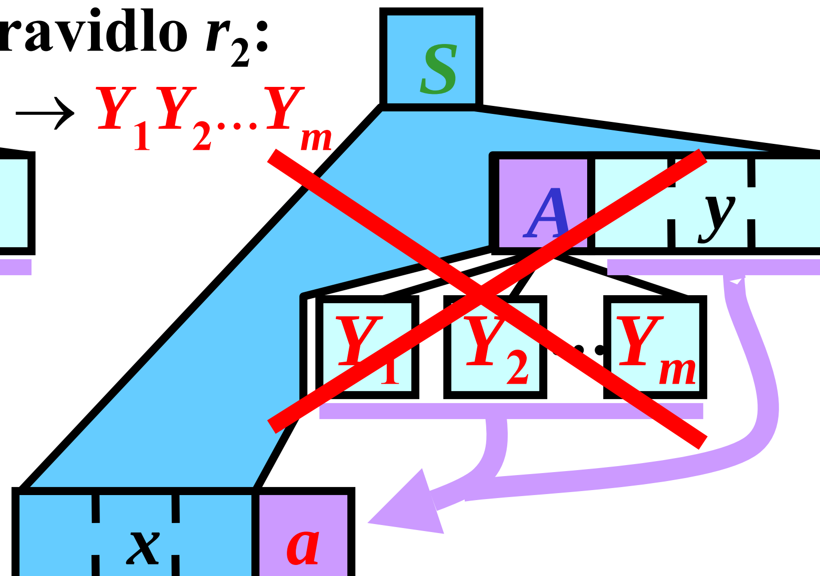
Nesmí nastat v LL-gramatice

Pravidlo r_1: $A \to X_1X_2 \dots X_n$



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

Pravidlo r_2: $A \to Y_1Y_2 \dots Y_m$



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

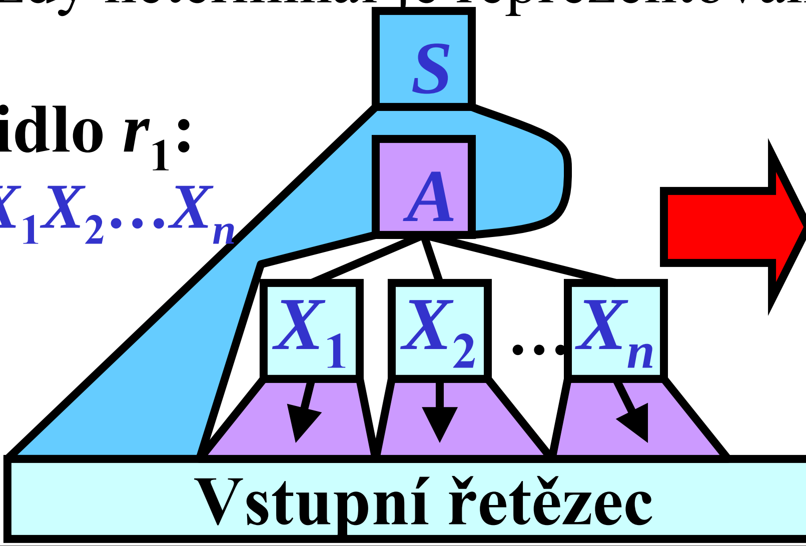
Implementace LL Analyzátoru

1) Rekurzívní sestup

- Každý neterminál je reprezentován procedurou, která řídí SA:

Pravidlo r_1:

$A \rightarrow X_1 X_2 \ldots X_n$



<u>function</u> *A*: **boolean**;
<u>begin</u>
{X_1 analýza}
{X_2 analýza}
…
{X_n analýza}
<u>end</u>

2) Prediktivní syntaktická analýza

- Syntaktický analyzátor se zásobníkem řízený tabulkou

Právě tyto symboly v tomto pořadí jsou uloženy na zásobníku.

Rekurzívní sestup: Příklad 1/4

```
Procedure GetNextToken;
begin
{ tato procedura uloží následující token do proměnné “token”}
end
```

- Pro $E \in N$: Pravidlo 1: $E \to TE'$

1

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

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

- Pro $T \in N$: Pravidlo 4: $T \to FT'$

4

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

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

Rekurzívní sestup: Příklad 2/4

- Pro *E’* ∈ *N*: Pravidla 2: *E’* → +*TE’*, 3: *E’* → ε

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

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

Rekurzívní sestup: Příklad 3/4

- Pro *T’* ∈ *N*: Pravidla 5: *T’* → **FT’*, 6: *T’* → ε

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

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

Rekurzívní sestup: Příklad 4/4

- Pro $F \in N$: Pravidla 7: $F \to (E)$, 8: $F \to i$

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

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

Hlavní tělo programu:

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

Rekurzívní sestup: Ilustrace pro *i***i*$

Start: **GetNextToken**;
Call E;

Vstupní řetězec:

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

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

TRUE

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

TRUE

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

TRUE

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

TRUE

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

TRUE

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

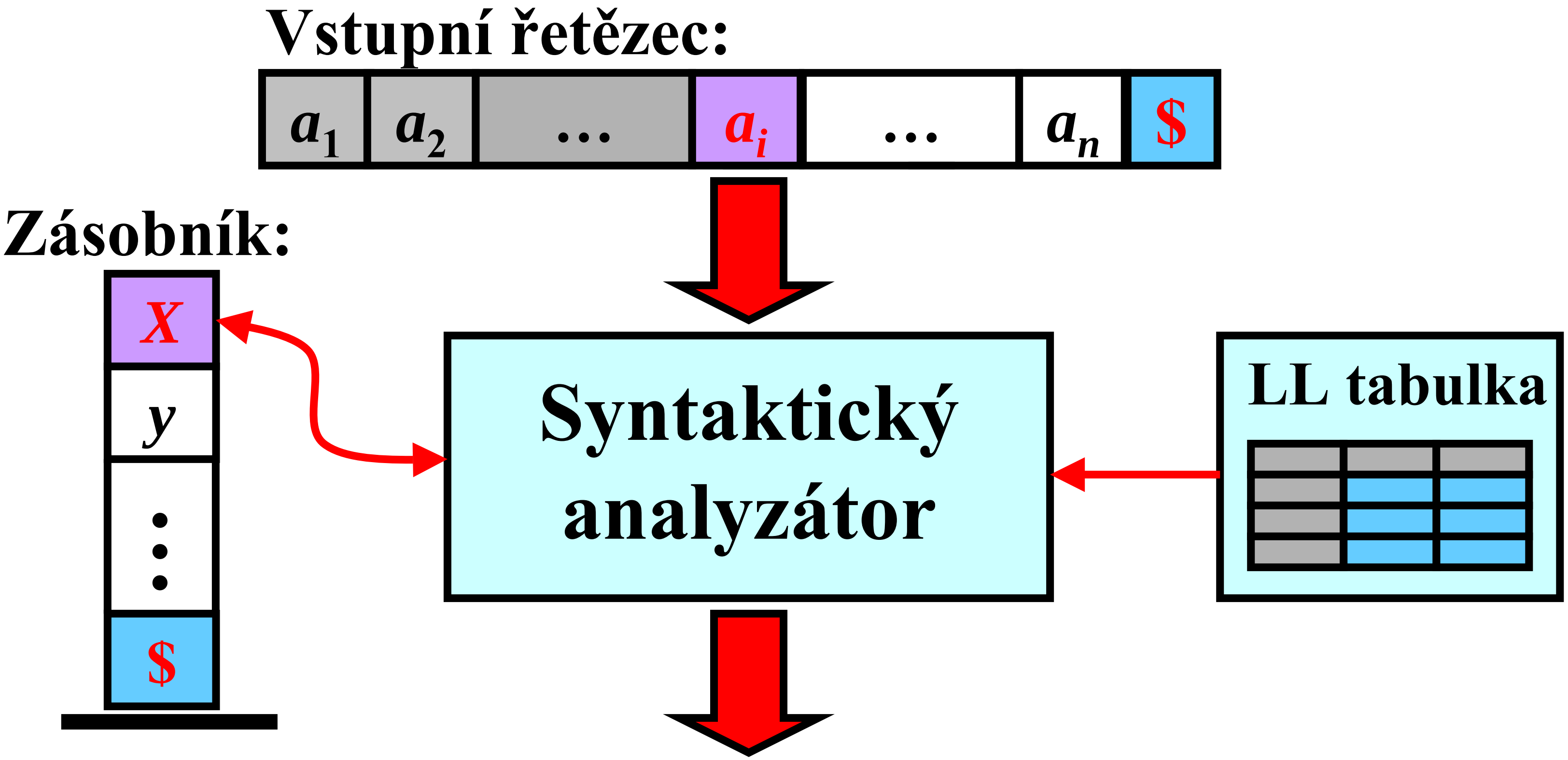
TRUE

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

TRUE

Prediktivní syntaktická analýza

- **Model pro prediktivní syntaktickou analýzu:**



Levý rozbor = posloupnost pravidel, která je použita v nejlevější derivaci pro vstupní řetězec.

Prediktivní SA: Algoritmus

- **Vstup:** LL-tabulka pro G=(N, T, P, **S**); $x \in T^*$
- **Výstup:** Levý rozbor pro x, pokud $x \in L(G)$ jinak chyba

---

- **Metoda:**
- push(**$**) & push(***S***) na zásobník
- **repeat**
  - nechť ***X*** je vrchol zásobníku a ***a*** aktuální token
  - **case** ***X*** **of:**
    - ***X*** = **$**: **if** ***a*** = **$** **then** **úspěch** **else** **chyba**;
    - ***X*** $\in T$: **if** ***X*** = ***a*** **then** pop(***X***) & přečti další ***a*** ze vstupního řetězce **else** **chyba**;
    - ***X*** $\in N$: **if** ***r***: ***X*** $\rightarrow$ ***x*** $\in$ LL-tabulka[***X***, ***a***] **then** zaměň na vrcholu zásobníku ***X*** za reversal(***x***) & zapiš ***r*** na výstup **else** **chyba**;

until **úspěch** **or** **chyba**

Prediktivní SA: Příklad

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

Pravidla:

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

Vstupní řetězec: $i * i$ \$

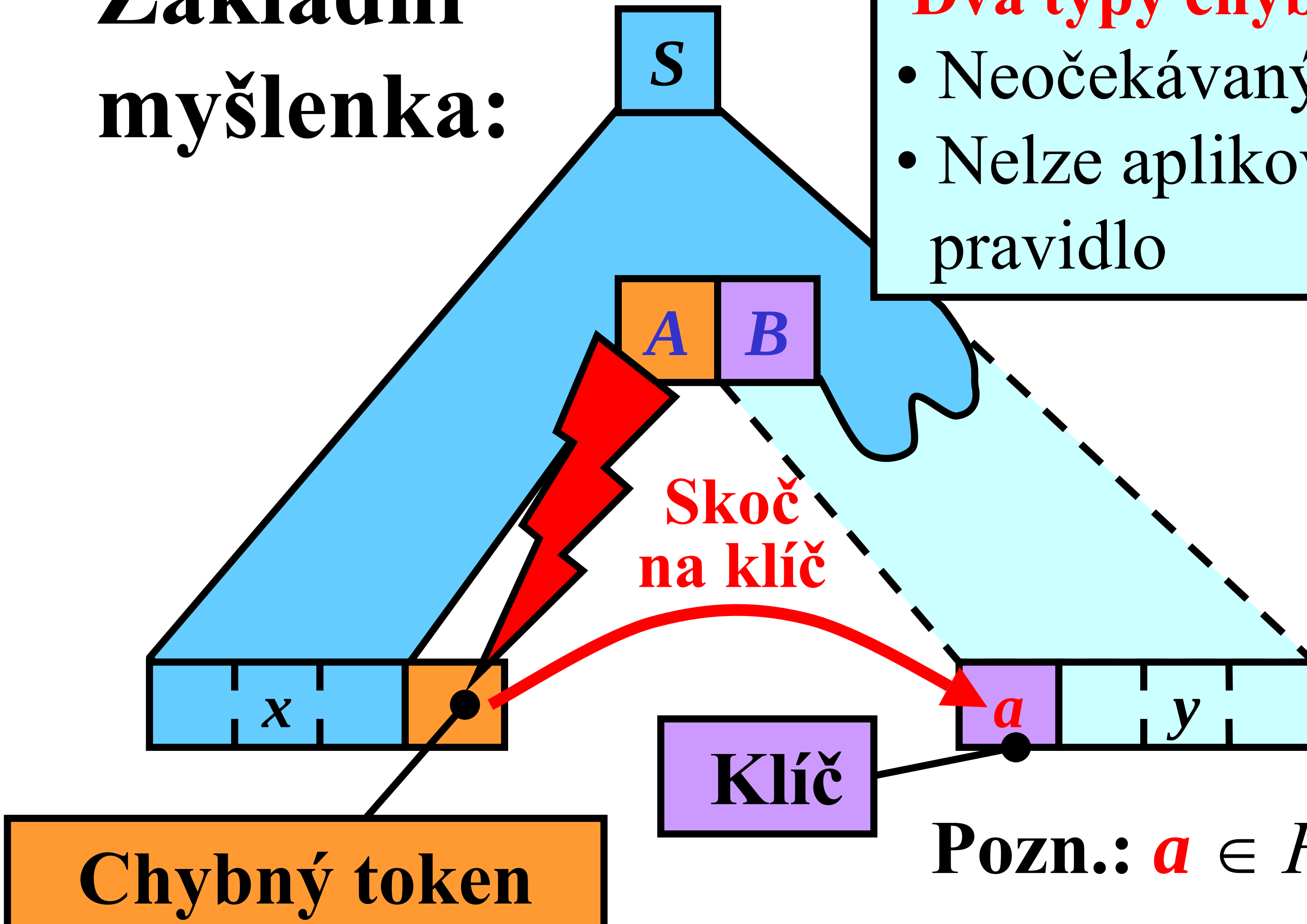
Zásobník	Vstup	Pravidlo	Derivace
$\$E$	$i*i\$$	1: $E \to TE'$	$\underline{E} \Rightarrow \underline{T}E'$
$\$E'T$	$i*i\$$	4: $T \to FT'$	$\Rightarrow \underline{F}T'E'$
$\$E'T'F$	$i*i\$$	8: $F \to i$	$\Rightarrow i\underline{T'}E'$
$\$E'T'i$	$i*i\$$		
$\$E'T'$	$*i\$$	5: $T' \to *FT'$	$\Rightarrow i*\underline{F}T'E'$
$\$E'T'F*$	$*i\$$		
$\$E'T'F$	$i\$$	8: $F \to i$	$\Rightarrow i*i\underline{T'}E'$
$\$E'T'i$	$i\$$		
$\$E'T'$	\$	6: $T' \to \varepsilon$	$\Rightarrow i*i\underline{E'}$
$\$E'$	\$	3: $E' \to \varepsilon$	$\Rightarrow i*i$
\$	\$		

Úspěch

Levý rozbor: 1485863

Zotavení z chyb: Úvod

Základní myšlenka:

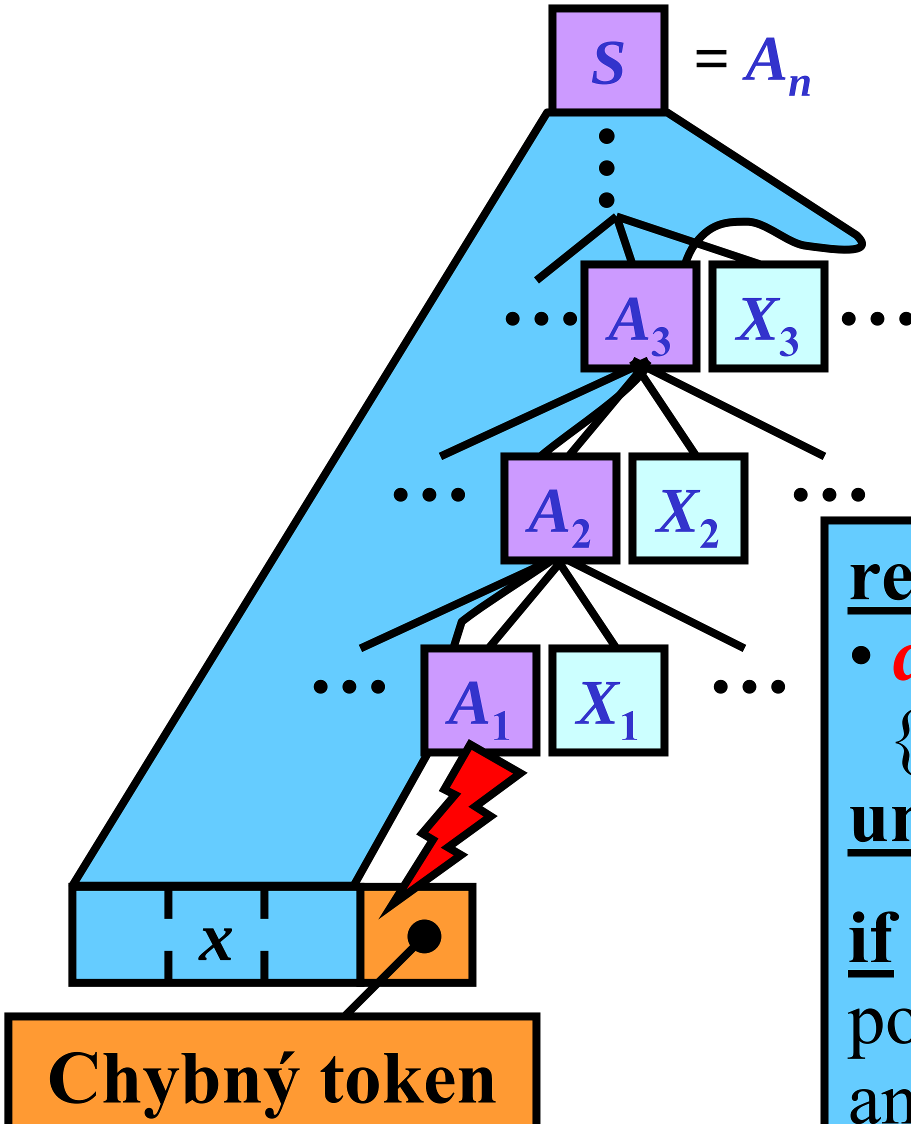


Dva typy chyb:

- Neočekávaný token
- Nelze aplikovat pravidlo

Pozn.: $a \in Follow(A)$

Hartmannova metoda: Zotavení z chyb



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

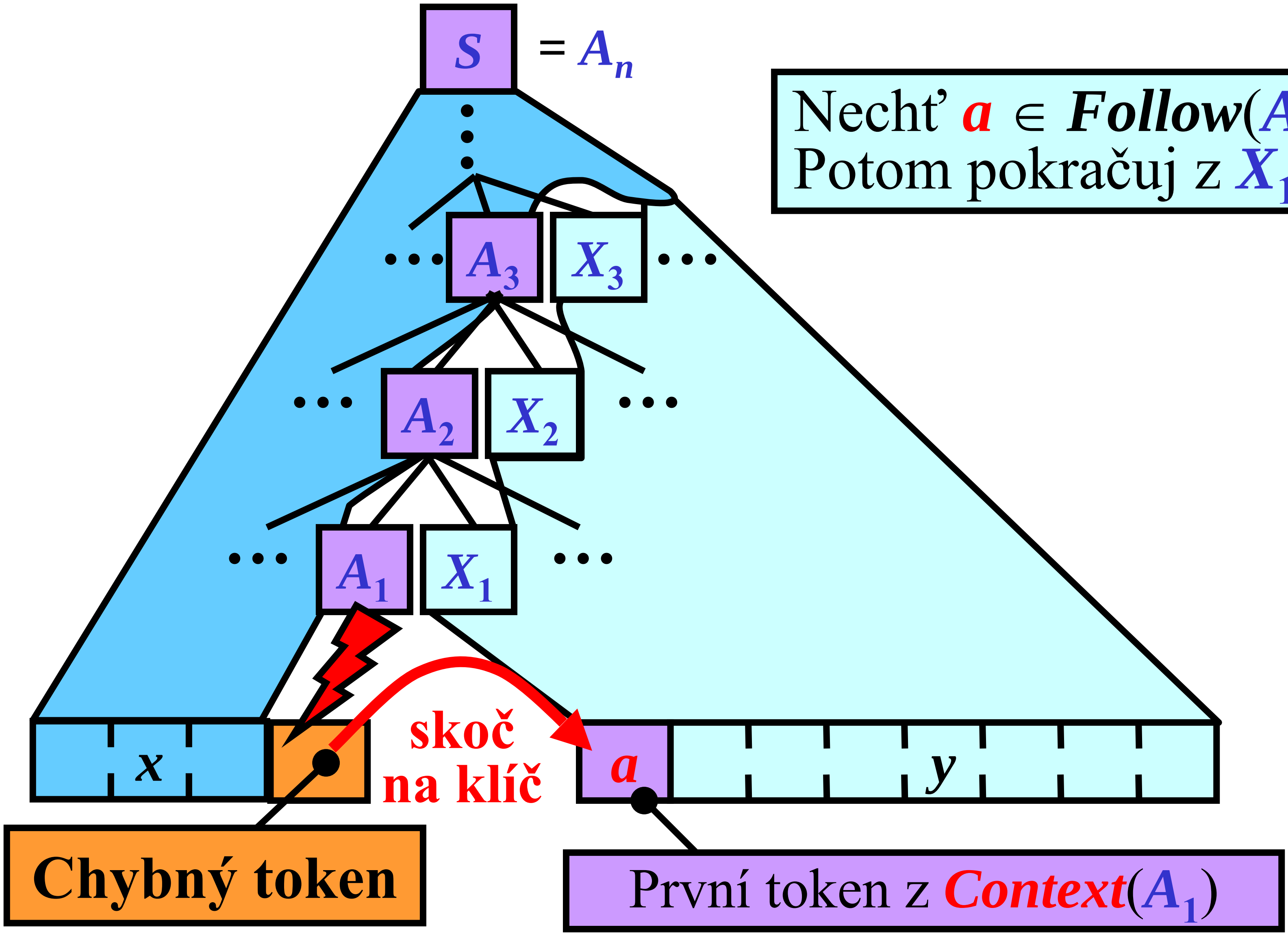
repeat
- a := GetNextToken;
  {Tyto tokeny přeskoč}

until a v množ. ***Context***(A_1)

if a v množ. ***Follow***(A_i) **then** pokračuj v syntaktické analýze od symbolu X_i.

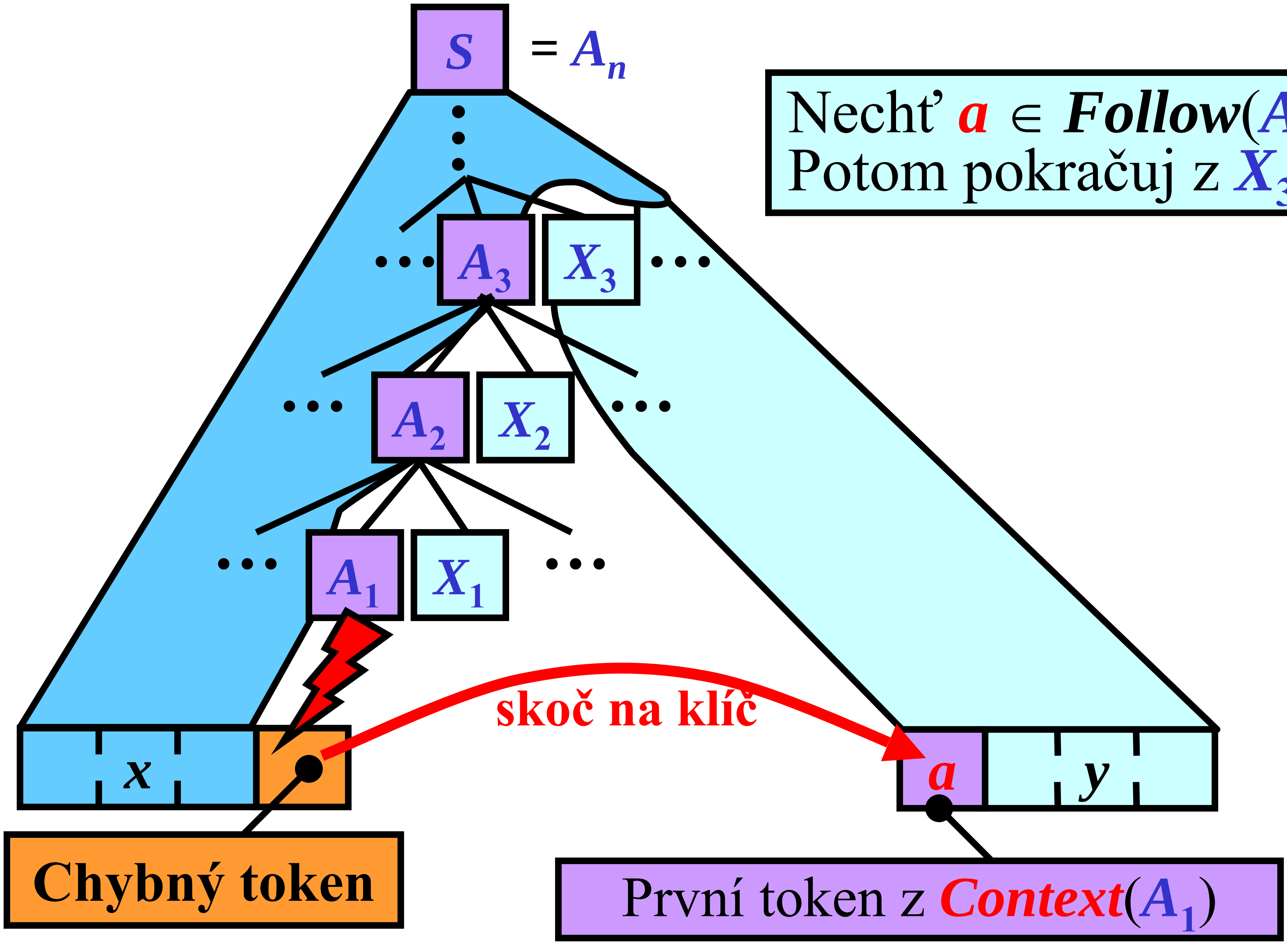
Zotavení z chyb: Ilustrace 1/2

Nechť $a \in \boldsymbol{Follow}(A_1)$.
Potom pokračuj z X_1



Zotavení z chyb: Ilustrace 2/2

Nechť $a \in \textbf{\textit{Follow}}(A_3)$.
Potom pokračuj z X_3



Context(*X*) pro prediktivní SA: Varianta I

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

***Context*(A) = *Follow*(A) pro všechna $A \in N$**

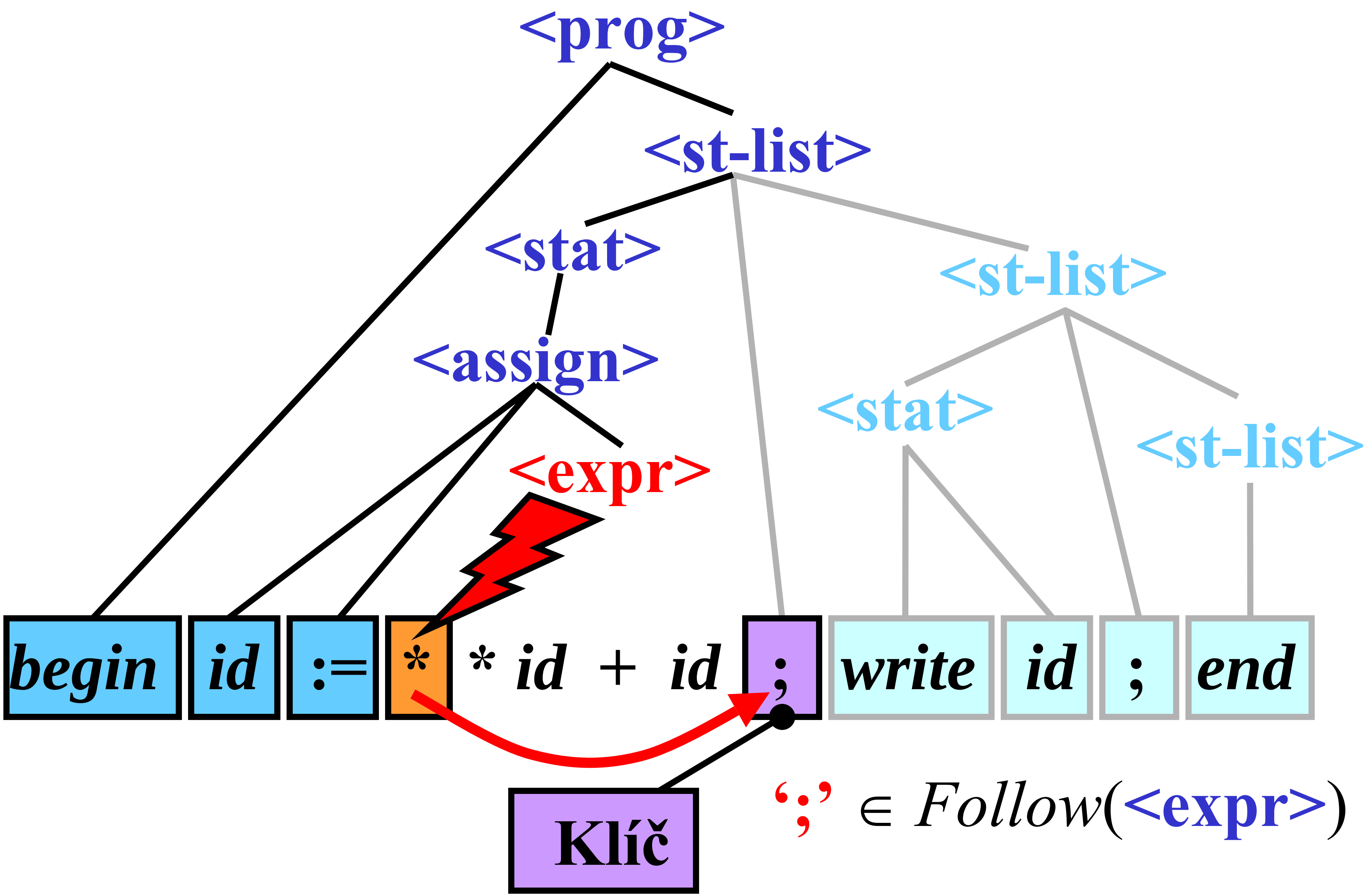
- **Metoda:**
- Nechť ***A*** je vrchol zásobníku & žádné pravidlo nelze použít:
- **repeat**

  a := GetNextToken;

  {Tyto tokeny jsou přeskočeny}

  until ***a*** v množině ***Context***(***A***)
- odstraň ***A*** ze zásobníku;

Varianta I: Příklad



Context(*X*) pro prediktivní SA: Varianta II

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

Context*(*A*) = *First*(*A*) ∪ *Follow*(*A*) pro všechna *A* ∈ *N

- **Metoda:**
- Nechť ***A*** je vrchol zásobníku & žádné pravidlo nelze použít:
- **repeat**

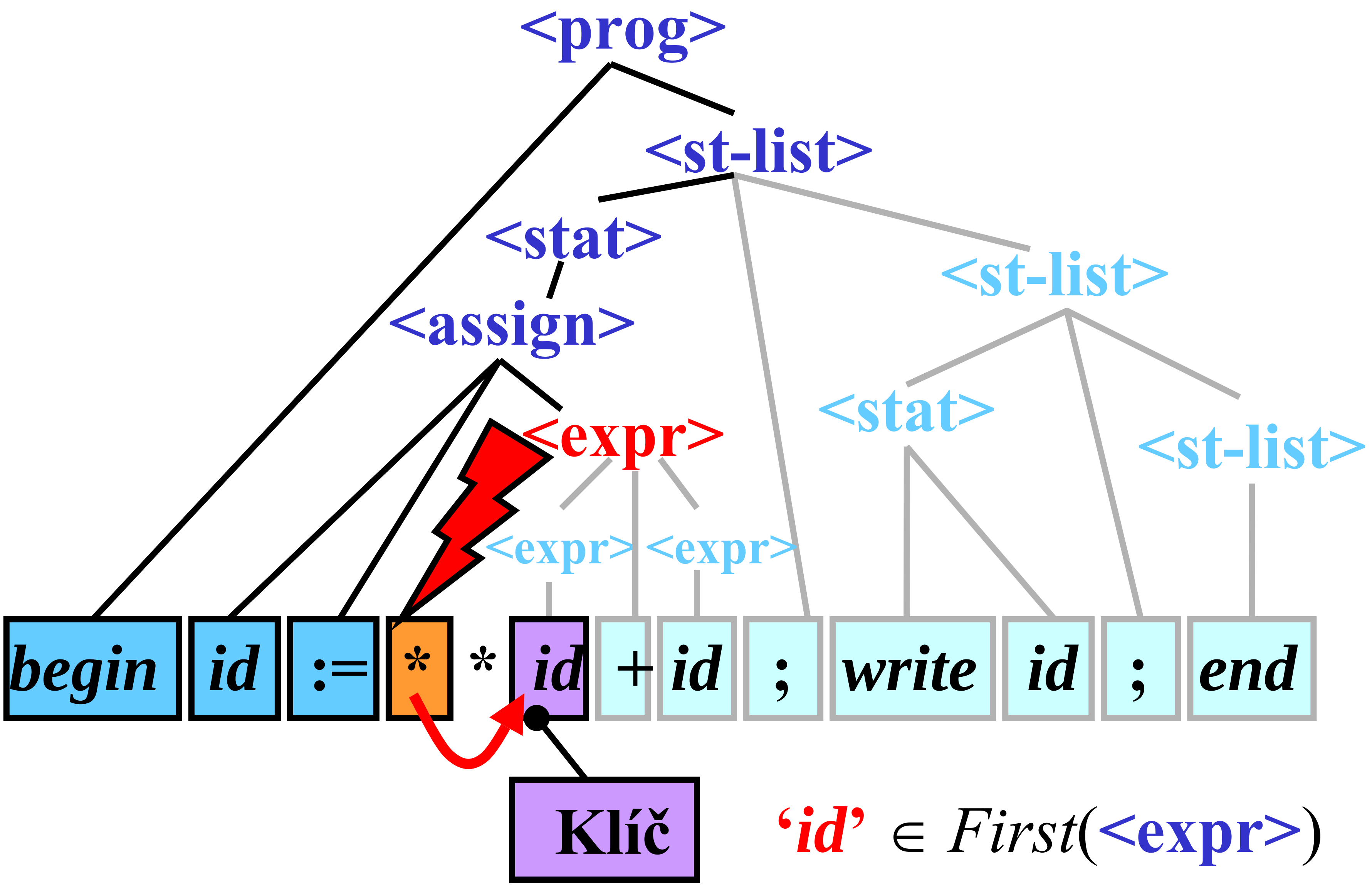
  a := GetNextToken;

  {Tyto tokeny jsou přeskočeny}

  until ***a*** v množině ***Context***(***A***)
- **if** ***a*** ∈ *First*(***A***) **then** ponech symbol ***A*** na zásobníku

  else odstraň ***A*** ze zásobníku; // ***a*** ∈ *Follow*(***A***)

Varianta II: Příklad



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