

LL(k) gramatiky

(provádí deterministický rozbor čtením textu z Leva doprava, s použitím Levé derivace a prohlédnutí k dalších symbolů vstupního textu)

Př. $G_1[S]$:
 $S \rightarrow a A S$ (1)
 $S \rightarrow b$ (2)
 $A \rightarrow a$ (3)
 $A \rightarrow bSA$ (4)

Ekvivalentní ZA

má jediný stav q

		vstup	
		a	b
zásobník	δ		
	S	aAS,1	b,2
	A	a,3	bSA,4
	a	srovnání	
	b		srovnání

Jak zjistit konec analýzy? (musí být prázdný zásobník i vstup, tzn. v zásobníku bude pouze dno # a prázdný vstup signalizuje e)

		vstup		
		a	b	e
zásobník	δ			
	S	aAS,1	b,2	
	A	a,3	bSA,4	
	a	srovnání		
	b		srovnání	
	#			přijetí

To je tzv. rozkladová tabulka M, prázdná políčka znamenají chybu

Problém s e-řetězcí (co bude na řadě ve vstupu při expanzi $A \rightarrow e$?

Př. $G_2[S]$: 1 $S \rightarrow a A S$

2 $S \rightarrow b$

3 $A \rightarrow c S A$

4 $A \rightarrow e$

(tabulku doplnit s výkladem na tabuli)

M =

δ	a	b	c	e
S				
A				
a				
b				
c				
#				

Problém s e-řetězcí (co bude na řadě ve vstupu při expanzi $A \rightarrow e$?
 To, co se ve větných formách může objevit za A)

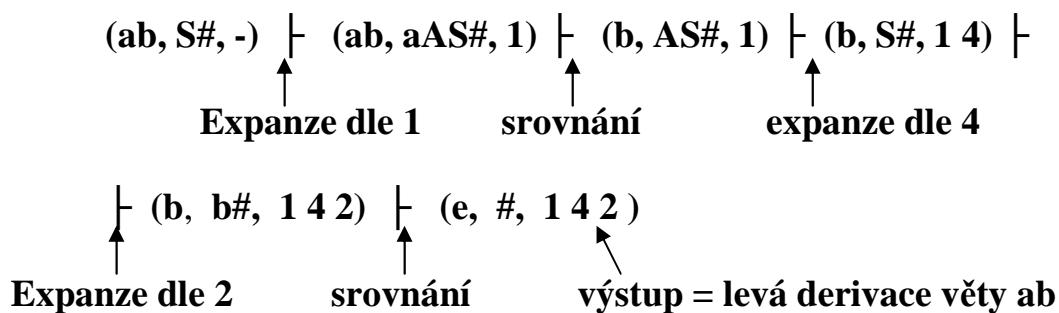
- Př. $G_2[S]$:
 1 $S \rightarrow a A S$
 2 $S \rightarrow b$
 3 $A \rightarrow c S A$
 4 $A \rightarrow e$

(tabulku doplnit s výkladem na tabuli)

δ	a	b	c	e
S	aAS, 1	b, 2		
A	e, 4	e, 4	cSA, 3	
a	srovnání			
b		srovnání		
c			srovnání	
#				přijetí

Algoritmus SA je stejný jako u jednoduchých, jen konstrukce M se liší

Např. (na tabuli) analyzuj v G_2 řetězec a b



Zavedeme funkci FOLLOW(X), kde $X \in N$

FOLLOW (X) =
 $\{ c : S \Rightarrow^* \alpha X \beta, \beta \Rightarrow^* c \gamma, c \in T, \gamma \in (N \cup T)^* \} \cup \{ e; S \Rightarrow^* \alpha X \}$

Algoritmus výpočtu FOLLOW(A)

Vstup: Upravená gramatika G, neterminální symbol A

Metoda:

1. Polož $FOLLOW(A) = \emptyset$
2. Je-li A počáteční symbol G, přidej e do FOLLOW(A)
3. Pro všechny pravé strany pravidel z G tvaru $\alpha A \beta$ přidej $FIRST(\beta)$ do FOLLOW(A), nepřidávej ale e.
4. Je-li v G pravidlo $L \rightarrow \alpha A$ nebo $L \rightarrow \alpha A \beta$, kde $FIRST(\beta)$ obsahuje e, pak přidej do FOLLOW(A) množinu FOLLOW(L)

Př. Co obsahuje FOLLOW(A) v G_2 ?

Použili jsme funkci $FIRST(\alpha)$ kde $\alpha \in (N \cup T)^*$

$$FIRST(\alpha) = \{ a; \alpha \Rightarrow^* a \beta, a \in T, \beta \in (N \cup T)^* \} \cup \{ e; \alpha \Rightarrow^* e \}$$

Algoritmus výpočtu FIRST(α)

Vstup: Upravená gramatika, řetězec α

Metoda:

1. Je-li $\alpha = e$, pak $FIRST(\alpha) = \{e\}$
2. Je-li $\alpha = a$, pak $FIRST(\alpha) = \{a\}$
3. Je-li $\alpha = X$; $X \in N$, $X \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$ jsou všechna X pravidla, pak

$$FIRST(\alpha) = \bigcup_{i=1}^n FIRST(\alpha_i)$$

4. Je-li $\alpha = X \beta$; $X \in N - \{e\}$, $\beta \in (N \cup T)^*$
pak $FIRST(\alpha) = (FIRST(X) - \{e\}) \cup FIRST(\beta)$
5. Jinak $\alpha = X \beta$; $X \in (N - \{e\}) \cup T$, pak $FIRST(\alpha) = FIRST(X)$

Př. Spočti FIRST a FOLLOW pro

$$G_3[S]: \quad S \rightarrow AB \quad A \rightarrow aA | e \quad B \rightarrow bB | e$$

Výpočet hodnot funkcí FIRST a FOLLOW

(alternativa pro programovou realizaci)

Algoritmus

Výpočet funkce FIRST.

Vstup: Bezkontextová gramatika $G=(N, T, P, S)$ a řetězec $\beta= X_1 X_2 \dots X_n \in (N \cup T)^*$.

Výstup: $FIRST(\beta)$.

Metoda:

1. Vytvoříme množinu \mathcal{F} takto:

(a) $\mathcal{F} = \{ . X_1 X_2 \dots X_n \}$.

(b) Jestliže v množině \mathcal{F} je prvek, ve kterém je bezprostředně za tečkou neterminální

symbol A , přidáme do množiny \mathcal{F} všechna pravidla z P se symbolem A na levé straně a tečku umístíme před první symbol pravé strany:

$$\mathcal{F} = \mathcal{F} \cup \{ A \rightarrow .\alpha : B \rightarrow \gamma A \ \delta \in \mathcal{F}, A \in N, A \rightarrow \alpha \in P \}.$$

(c) Jestliže v množině \mathcal{F} je prvek, ve kterém je tečka na konci pravidla, tj. položka tvaru $B \rightarrow \delta$, vložíme do \mathcal{F} nové položky vytvořené tak, že posuneme tečky za B

ve všech položkách z \mathcal{F} takových, kde tečka byla před B :

$$\mathcal{F} = \mathcal{F} \cup \{ A \rightarrow \alpha B . \gamma : A \rightarrow \alpha . B \ \gamma \in \mathcal{F}, B \rightarrow \delta \in \mathcal{F} \}.$$

(d) Kroky b) a c) opakujeme tak dlouho, dokud je možno do \mathcal{F} přidávat další prvky.

2. Množinu $FIRST(\beta)$ vytvoříme tak, že do ní vložíme všechny terminální symboly, které se vyskytují bezprostředně za tečkou v některém prvku množiny \mathcal{F} . Jestliže v množině \mathcal{F} je prvek, kde se vyskytuje tečka na konci řetězce β , přidáme do $FIRST(\beta)$ prázdný řetězec:

$$FIRST(\beta) = \{ a : a \in T, A \rightarrow \alpha . a \ \gamma \in \mathcal{F} \} \cup \{ e : \beta . \in \mathcal{F} \}$$

Příklad

Je dána gramatika $G_4=(\{E,Z,T,D,F\},\{+,*,(,),a\},P,S)$, kde P obsahuje pravidla:

$$E \rightarrow T Z \quad Z \rightarrow + T Z \mid e$$

$$T \rightarrow F D \quad D \rightarrow * F D \mid e$$

$$F \rightarrow (E) \mid a$$

$FIRST(E)$ vypočteme takto: $\mathcal{F} = \{ . E, E \rightarrow . T Z, T \rightarrow . F D, F \rightarrow . (E), F \rightarrow . a \}$, a proto $FIRST(E) = \{ (, a \}$.

$FIRST(Z)$ vypočteme takto: $\mathcal{F} = \{ . Z, Z \rightarrow . + T Z, Z \rightarrow ., Z . \}$, a proto $FIRST(Z) = \{ +, e \}$.

$FIRST(DZ)$ vypočteme takto: $\mathcal{F} = \{ . DZ, D \rightarrow . * F D, D \rightarrow ., D . Z, Z \rightarrow . + T Z, Z \rightarrow ., DZ . \}$ a proto $FIRST(DZ) = \{ *, +, e \}$.

Na podobném principu jako výpočet funkce FIRST můžeme sestavit algoritmus pro výpočet funkce FOLLOW.

Algoritmus

Výpočet funkce FOLLOW

Vstup: Bezkontextová gramatika $G=(N,T,P,S)$ a neterminální symbol A

Výstup: FOLLOW(A).

Metoda:

1. Vytvoříme množinu $Ne = \{ B : B \Rightarrow^* e, B \in N \}$, tj. neterminálních symbolů, ze kterých je možno generovat prázdné řetězce.
2. Vytvoříme množinu F takto:
 - a) Vytvoříme fiktivní pravidlo $A \rightarrow A$ a $F := \{ A \rightarrow A. \}$.
 - b) Jestliže v množině F je položka, ve které je tečka na konci pravidla, tj. položka $B \rightarrow \gamma$, vložíme do F nové položky vytvořené tak, že vezmeme všechna pravidla z P , ve kterých se na pravých stranách vyskytuje symbol B a tečku v nich umístíme právě za tento symbol B :
 $F := F \cup \{ C \rightarrow \alpha B. \beta : B \rightarrow \gamma. \beta \in F, C \rightarrow \alpha B \beta \in P \}$.
 - c) Jestliže v množině F je prvek, ve kterém je bezprostředně za tečkou neterminální symbol, který patří do množiny Ne , přidáme do F další položku, kterou vytvoříme z uvažované položky posunutím tečky o jeden symbol doprava:
 $F := F \cup \{ A \rightarrow \alpha B. \beta : A \rightarrow \alpha. B \beta \in F, B \in Ne \}$.
 - d) Kroky b) a c) opakujeme tak dlouho, dokud je možno do F přidávat další prvky.
 - e) Jestliže v množině F je prvek, ve kterém je bezprostředně za tečkou neterminální symbol B , přidáme do množiny F všechna pravidla z P se symbolem B na levé straně a tečku umístíme před první symbol pravé strany:
 $F := F \cup \{ B \rightarrow . \alpha : C \rightarrow \gamma. B \beta \in F, B \in N, B \rightarrow \alpha \in P \}$.
 - f) Jestliže v množině F je prvek, ve kterém je bezprostředně za tečkou neterminální symbol, který patří do množiny Ne , přidáme do F další položku, kterou vytvoříme z uvažované položky posunutím tečky o jeden symbol doprava:
 $F := F \cup \{ A \rightarrow \alpha B. \beta : A \rightarrow \alpha. B \beta \in F, B \in Ne \}$.
 - g) Kroky e) a f) opakujeme tak dlouho, dokud je možno do F přidávat další prvky.
3. Množinu FOLLOW(A) vytvoříme tak, že do ní vložíme všechny terminální symboly, které se vyskytují bezprostředně za tečkou v některém prvku množiny F . Jestliže je v množině F prvek, ve kterém se vyskytuje tečka na konci pravidla a na levé straně je symbol S (tj. počáteční symbol gramatiky), přidáme do FOLLOW(A) prázdný řetězec:
 $FOLLOW(A) := \{ a : a \in T, B \rightarrow \alpha. a \beta \in F \} \cup \{ e : S \rightarrow \alpha. \in F \}$.

Př. Spočti FOLLOW pro symboly z G_4

$$\begin{array}{ll}
 E \rightarrow T Z & Z \rightarrow + T Z \mid e \\
 T \rightarrow F D & D \rightarrow * F D \mid e \\
 F \rightarrow (E) \mid a &
 \end{array}$$

Dle 1. $N_e = \{Z, D\}$

Follow (Z):

2a $Z \rightarrow Z.$

2b $E \rightarrow T Z.$
 $Z \rightarrow + T Z.$
 $F \rightarrow (E.)$

Proto Follow(Z) = { }, e }

Follow (D):

2a $D \rightarrow D.$

2b $T \rightarrow F D.$
 $D \rightarrow * F D.$
 $E \rightarrow T. Z$
 $Z \rightarrow + T. Z$

2c $Z \rightarrow + T Z.$

$E \rightarrow T Z.$

2b $F \rightarrow (E.)$

2e $Z \rightarrow . + T Z$

$Z \rightarrow .$

2f $Z \rightarrow + T Z. \text{ už tam je}$

Proto Follow (D) = { +,), e }

Follow (T):

2a $T \rightarrow T.$

2b $E \rightarrow T. Z$
 $Z \rightarrow + T. Z$

2c $E \rightarrow T Z.$

$Z \rightarrow + T Z.$

2b $F \rightarrow (E.)$

2e $Z \rightarrow . + T Z$

$Z \rightarrow .$

Proto Follow (T) = { +,), e }

Algoritmus vytvoření M pro gramatiku s e-pravidly

M je definována na kartézském součinu $(N \cup T \cup \{\#\}) \times (T \cup \{e\})$

1. Je-li $A \rightarrow a \alpha$ i-té pravidlo v P, pak $M(A, a) = a \alpha, i$

2. Je-li $A \rightarrow e$ i-té pravidlo v P, pak $M(A, b) = e, i$
pro všechna $b \in \text{FOLLOW}(A)$

3. $M(a, a) = \text{srovnání}$ pro všechna $a \in T$

3. $M(\#, e) = \text{přijetí}$

4. $M(\text{ostatní}) = \text{chyba}$

Př. G_5 : $S \rightarrow a A$
 $S \rightarrow b$
 $A \rightarrow c S a$
 $A \rightarrow e$

?jak vypadá rozkladová tabulka

	a	b	c	e
S				
A				
a	sr			
b		sr		
c			sr	
#				přij

zjistěme $\text{FOLLOW}(A) = ?$

FOLLOW (A) = {a, e}

	a	b	c	e
S	aA,1	b, 2		
A	e, 4		cSa, 3	e, 4
a	sr			
b		sr		
c			sr	
#				přij

**Proved'me rozklad zvolené věty
(a c a a, S#, e) ⊢ (...**

LL(1) gramatiky

Algoritmus SA zůstává stejný jako výše uvedený, vytvoření M se liší.
Princip syntaktické analýzy LL gramatik

Budeme se zabývat algoritmem syntaktické analýzy, který vytváří derivační strom analyzovaného řetězce směrem shora dolů.

Základní princip syntaktické analýzy můžeme v tomto případě formulovat takto:

Je dána bezkontextová gramatika $G = (N, T, P, S)$ a řetězec $w = a_1 a_2 \dots a_n$, který je větou z $L(G)$. Pak existuje levá derivace

$$S = \gamma_1 \Rightarrow \gamma_2 \Rightarrow \dots \Rightarrow \gamma_n = w.$$

Vzhledem k tomu, že derivace je levá, má každá větná forma γ_i tvar:

$$\gamma_i = a_1 a_2 \dots a_j A_i \beta_i,$$

kde a_1, a_2, \dots, a_j jsou terminální symboly, A_i je neterminální symbol, β_i je řetězec terminálních a neterminálních symbolů. Přitom řetězec $a_1 a_2 \dots a_j$ je předponou věty w , $j \geq 0$.

Vlastnosti algoritmu LL syntaktické analýzy

Předpokládejme, že $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ jsou všechna pravidla v P s neterminálním symbolem A na levé straně. Pak základní problém syntaktické analýzy metodou shora dolů spočívá v nalezení toho pravidla $A \rightarrow \alpha_k$, jehož aplikací dostaneme z větné formy γ_i větnou formu γ_{i+1} .

Pro výběr pravidla $A \rightarrow \alpha_k$, je možno použít:

- a) informaci o dosavadním průběhu (historii) analýzy,
- b) informaci o dosud nepřečtené části vstupního řetězce (dopředu prohlíženém řetězci omezené délky).

Pokud tyto informace vždy stačí k jednoznačnému výběru pravidla $A \rightarrow \alpha_k$, pak se gramatika G nazývá *LL* gramatika. Název je odvozen od toho, že při čtení vstupního řetězce zleva je vytvářen levý rozklad.

Při syntaktické analýze *LL* gramatik jsou do zásobníku ukládány řetězce, které odpovídají levým větným formám nebo takovým jejich příponám, které vzniknou odejmutím předpony tvořené řetězcem terminálních symbolů.

Základními operacemi syntaktického analyzátoru pro *LL* gramatiky (*LL* analyzátoru) jsou:

- a) *Expanze* – neterminální symbol na vrcholu zásobníku je nahrazen pravou stranou vybraného pravidla
- b) *Srovnání* – terminální symbol na vrcholu zásobníku se ze zásobníku vyloučí, jestliže je shodný se symbolem, který byl ze vstupního řetězce přečten.
- c) *Přijetí* – vstupní řetězec je přečten a zásobník je prázdný.
- d) *Chyba* – ve všech ostatních případech.

Pokud pro danou gramatiku G vystačíme při rozhodování o výběru pravidla pro expanzi s informací o dopředu prohlíženém řetězci délky nejvýše k a s gramatickými symboly v zásobníku, pak se gramatika G nazývá

silná $LL(k)$ gramatika.

Při analýze silných $LL(k)$ gramatik jsou do zásobníku ukládány přímo symboly gramatiky a syntaktický analyzátor je řízen rozkladovou tabulkou.

V případě, že je nutno použít při syntaktické analýze pro rozhodování o dalším postupu informace o historii analýzy, je $LL(k)$ analyzátor řízen rozkladovým automatem a do zásobníku jsou ukládány stavy tohoto automatu.

Z hlediska praktických aplikací jsou nejzajímavější $LL(1)$ gramatiky.

Zopakujme- Dána gramatika $G = (N, T, P, S)$, $\alpha \in (N \cup T)^*$, $X \in N$, pak:

$FIRST(\alpha) = \{a : \alpha \Rightarrow^* a \beta, a \in T, \beta \in (N \cup T)^*\} \cup \{e : \alpha \Rightarrow^* e\}$,

$FOLLOW(X) = \{a : S \Rightarrow^* \alpha X \beta, \beta \Rightarrow^* a \gamma, \gamma \in (N \cup T)^*\} \cup \{e : S \Rightarrow^* \alpha X\}$.

Bezkontextová gramatika $G=(N, T, P, S)$ se nazývá $LL(1)$ gramatika, když platí následující podmínka pro každé $A \in N$: jestliže $A \rightarrow \alpha$ a $A \rightarrow \beta$ jsou různá pravidla v P , pak:

a) Pro řetězce α, β musí platit, že

$FIRST(\alpha) \cap FIRST(\beta) = \emptyset$.

b) Jestliže z řetězce α lze derivovat prázdný řetězec a z řetězce β nelze derivovat prázdný řetězec, pak musí platit, že

$FOLLOW(A) \cap FIRST(\beta) = \emptyset$.

c) Jestliže z řetězce α nelze derivovat prázdný řetězec a z β lze derivovat prázdný řetězec, pak musí platit, že

$FIRST(\alpha) \cap FOLLOW(A) = \emptyset$.

Poznámka: Předchozí definici je možno zkráceně formulovat takto:

Bezkontextová gramatika $G = (N, T, P, S)$ se nazývá $LL(1)$ gramatika, když pro každé $A \in N$ platí podmínka:

Jestliže $A \rightarrow \alpha$ a $A \rightarrow \beta$ jsou různá pravidla v P , pak

$FIRST(\alpha FOLLOW(A)) \cap FIRST(\beta FOLLOW(A)) = \emptyset$.

Př. na tabuli ověřit, je-li $G_6[E]$ gramatikou $LL(1)$

1 $E \rightarrow T E'$

2 $E' \rightarrow + T E'$

3 $E' \rightarrow e$

4 $T \rightarrow F T'$

5 $T' \rightarrow * F T'$

6 $T' \rightarrow e$

7 $F \rightarrow (E)$

8 $F \rightarrow a$

$FIRST (T E') = \{a, (\}$

$FIRST (+ T E') = \{+\}$

$FOLLOW (E') = \{e,)\}$

$FIRST (F T') = \{a, (\}$

$FIRST (* F T') = \{*\}$

$FOLLOW (T') = \{e,), +\}$

$FIRST ((E)) = \{(\}$

$FIRST (a) = a$

Algoritmus

Vytvoření rozkladové tabulky pro LL(1) gramatiku.

Vstup: LL(1) gramatika $G = (N, T, P, S)$.

Výstup: Rozkladová tabulka M pro gramatiku G .

Metoda: Rozkladová tabulka M je definována na $N \times (T \cup \{e\})$.

1. Je-li $A \rightarrow \alpha$ i -té pravidlo v P , pak $M(A,a) = \alpha, i$ pro všechna $a \in \text{FIRST}(\alpha) - \{e\}$.
2. Je-li $A \rightarrow \alpha$ i -té pravidlo v P a $e \in \text{FIRST}(\alpha)$, pak $M(A,b) = \alpha, i$ pro všechna $b \in \text{FOLLOW}(A)$.
3. $M(A,a) = \text{chyba}$ ve všech ostatních případech.

Algoritmus

Syntaktická analýza LL(1) gramatik.

Vstup: Rozkladová tabulka M pro LL(1) gramatiku $G=(N, T, P, S)$, vstupní řetězec $\omega \in T^*$.

Výstup: Levý rozklad v případě, že $\omega \in L(G)$, jinak chybová signalizace.

Algoritmus čte vstupní řetězec, používá zásobník a vytváří výstupní řetězec.

Konfigurace algoritmu je trojice (x, α, π) , kde $x \in T^*$ je dosud nepřečtená část vstupního řetězce, $\alpha \in (N \cup T)^*$ je obsah zásobníku a $\pi \in C^*$ je posloupnost čísel pravidel použitých při dosud provedených expanzích. Na začátku je v zásobníku symbol S (zásobník má vrchol vlevo), výstupní řetězec je prázdný.

Metoda: Algoritmus provádí přechody podle 1) a 2), dokud se neobjeví situace podle 3) nebo 4).

1. *Expanze*: $(ax, A\alpha, \pi) \vdash (ax, \beta\alpha, \pi i)$, jestliže $A \in N$ a $M(A,a) = \beta, i$. Symbol A na vrcholu zásobníku je nahrazen řetězcem β a číslo pravidla i je připojeno k výstupní posloupnosti.
2. *Srovnání*: $(ax, a\alpha, \pi) \vdash (x, \alpha, \pi)$, jestliže $a \in T$. Symbol na vstupu se srovnává se symbolem v zásobníku a v případě rovnosti se vstupní symbol přečte a zásobníkový symbol se ze zásobníku vyloučí.
3. *Přijetí*: V konfiguraci $(e, \#, \pi)$ analýza končí a π je levý rozklad vstupního řetězce.
4. *Chyba*: Ve všech ostatních případech analýza končí chybovou signalizací.

Př. na tabuli $G_7[E]$: $E \rightarrow T Z$ $Z \rightarrow +T Z \mid e$ $T \rightarrow F D$
 $D \rightarrow * F D \mid e$ $F \rightarrow (E) \mid a$

Pro nepřítomné je výsledek na další straně

Výsledek pro $G_7[E]$:

	a	+	*	()	e
E	TZ,1			TZ,1		
T	FD,4			FD,4		
Z		+TZ,2			e,3	e,3
D		e,6	*FD,5		e,6	e,6
F	a,8			(E),7		

Zkusit analyzovat nějakou větu

$(a+a*a, E\#, -) \vdash (a+a*a, TZ\#, 1) \vdash (a+a*a, FDZ\#, 1\ 4)$
 $(a+a*a, aDZ\#, 1\ 4\ 8) \vdash (+a*a, DZ\#, 1\ 4\ 8) \vdash$
 $\vdash \dots$

Silné LL(k) gramatiky

(mohou prohlédnout k symbolů ve vstupujícím řetězci)

Def.:

$$\text{FIRST}_k(\alpha) = \{ x: x \in T^*, \alpha \Rightarrow^* x y, y \in (N \cup T)^*, |x| = k \} \\ \cup \{ x: x \in T^*, \alpha \Rightarrow^* x, |x| < k \}$$

Def.:

$$\text{FOLLOW}_k(A) = \{ x: x \in T^*, S \Rightarrow^* w A x y, |x| = k \} \\ \cup \{ x: x \in T^*, S \Rightarrow^* w A x, |x| < k \}$$

Def.: Gramatika G se nazývá silná LL(k), když pro všechna $A \in N$ platí: Jsou-li $A \rightarrow \alpha, A \rightarrow \beta$ různá pravidla v P, pak

$$\text{FIRST}_k(\alpha \text{ FOLLOW}_k(A)) \cap \text{FIRST}_k(\beta \text{ FOLLOW}_k(A)) = \emptyset$$

Def. Gramatika je LL(k) pro $k \geq 0$, jestliže v případě existence dvou levých derivací

$$S \Rightarrow^* w A \alpha \Rightarrow w \beta \alpha \Rightarrow^* w x \\ S \Rightarrow^* w A \alpha \Rightarrow w \gamma \alpha \Rightarrow^* w y$$

takových, že platí $\text{FIRST}_k(x) = \text{FIRST}_k(y)$
plyne, že $\beta = \gamma$

Věta: G je LL(k) tehdy, platí-li pro \forall větné formy (no jo, ale kdo je má všechny zjistit) $wA\alpha$ v případě, že

$$A \rightarrow \gamma, A \rightarrow \beta$$

jsou dvě různá pravidla, pak

$$\text{FIRST}_k(\beta \alpha) \cap \text{FIRST}_k(\gamma \alpha) = \emptyset.$$

Př. na tabuli zjistit, zda $G_8[S]$:
 $S \rightarrow a b A \mid e$
 $A \rightarrow S a a \mid b$ je silná LL(2) ?

$FIRST_2(abA FOLLOW_2(S)) =$

$FIRST_2(e FOLLOW_2(S)) =$

$FIRST_2(Saa FOLLOW_2(A)) =$

$FIRST_2(bFOLLOW_2(A)) =$

Př. Zkusme ještě $G_{81}[S]$:
 $S \rightarrow a A a a \mid b A b a$
 $A \rightarrow b \mid e$

je silná LL(2) ?

$FIRST_2(aAaa FOLLOW_2(S)) \cap FIRST_2(bAba FOLLOW_2(S)) =$

$FIRST_2(b FOLLOW_2(A)) \cap FIRST_2(e FOLLOW_2(A)) =$

je LL(2) ?

pro \forall větné formy $wA\alpha$, zde jednou je $a A a a$, druhou je $b A b a$

pro pravidla $A \rightarrow \gamma$, $A \rightarrow \beta$ zde $A \rightarrow b$ $A \rightarrow e$

platí $FIRST_2(\beta \alpha) \cap FIRST_2(\gamma \alpha) = \emptyset$

$FIRST_2(baa) \cap FIRST_2(eaa) =$

$FIRST_2(bba) \cap FIRST_2(eba) =$

Algoritmus vytvoření rozkladové tabulky pro silné LL(k) gramatiky

1. Je-li $A \rightarrow \alpha$ i-té pravidlo, v P, pak

$M(A, x) = \alpha, i$ pro $\forall x \in \text{FIRST}_k(\alpha)$ taková, že $|x| = k$

2. Je-li $A \rightarrow \alpha$ i-té pravidlo, v P a $y \in \text{FIRST}_k(\alpha)$ takové, že $|y| < k$, pak

$M(A, z) = \alpha, i$ pro $\forall z \in \text{FIRST}_k(y \text{ FOLLOW}_k(A))$

3. $M(Y, x) = \text{chyba}$ v ostatních případech.

Př. vytvořit M pro $G_8[S]$ na tabuli

$S \rightarrow a b A \mid e$

$A \rightarrow S a a \mid b$

$\text{FIRST}_2(abA \text{ FOLLOW}_2(S)) = \{ ab \}$

$\text{FIRST}_2(e \text{ FOLLOW}_2(S)) = \{ aa, e \}$

$\text{FIRST}_2(Saa \text{ FOLLOW}_2(A)) = \{ aa, ab \}$

$\text{FIRST}_2(b \text{ FOLLOW}_2(A)) = \{ ba, b \}$

	aa	ab	a	ba	bb	b	e
S							
A							
a							
b							
#							

	aa	ab	a	ba	bb	b	e
S	e,1	abA,2					e,1
A	Saa,3	Saa,3		b,4		b,4	
a	sr	sr	sr				
b				sr	sr	sr	
#							přj

Algoritmus syntaktické analýzy pro silné LL(k) gramatiky:

Vykonává kroky 1. a 2., dokud nenastane 3. nebo 4.

Označme $u = \text{FIRST}_k(x)$

1. Expanze: $(x, A, \alpha, \Pi) \vdash (x, \beta, \alpha, \Pi i)$, je-li $M(A, u) = \beta, i$

2. Srovnání: $(ax, a, \alpha, \Pi) \vdash (x, \alpha, \Pi)$

3. Přijetí: V konfiguraci $(e, \#, \Pi)$ končíme přijetím. Π je levou derivací.

4. Chyba: Ve všech ostatních případech.

Př. analýzy věty na tabuli

$(ababbaa, S\#, -) \vdash (ababbaa, abA\#, 2) \vdash^2 (abbaa, A\#, 2) \vdash$
 $\vdash \dots$

LL(1) transformace gramatik

Hledáme k BKG (nelevorekurzivní, bez zbytečných symbolů) ekvivalentní LL(1) gramatiku

Důvody nesplnění LL(1) podmínky mohou být dva:

I. Kolize First First (FF):

Existují alespoň dvě pravidla $A \rightarrow \alpha_1 \mid \alpha_2$, pro která platí

$$\text{FIRST}(\alpha_1) \cap \text{FIRST}(\alpha_2) \neq \emptyset$$

II. Kolize First Follow (FFL):

Existují alespoň dvě pravidla $A \rightarrow \alpha_1 \mid \alpha_2$, pro která platí:

$$e \in \text{FIRST}(\alpha_2), \quad \text{FIRST}(\alpha_1) \cap \text{FOLLOW}(A) \neq \emptyset$$

ad I) Způsobeno výskytem pravidel

$$A \rightarrow a \alpha_1 \mid a \alpha_2 \mid \dots \mid a \alpha_n$$

Lze někdy odstranit tzv. "levou faktorizací"

Transformujeme $G = (N, T, P, S)$, kde

$$A \rightarrow a \alpha_1 \mid a \alpha_2 \mid \dots \mid a \alpha_n \in P_A, \quad a \neq \epsilon$$

na

$G' = (N', T, P', S)$, ve které

$$P' = (P - P_A) \cup \{ A \rightarrow a A', A' \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n \}$$

Na faktorizovatelný tvar lze převést eliminací pravidel.

Rezultativnost odstranění FF kolize se nezaručuje

Př. V pravidlech $A \rightarrow a B \mid C D$ $C \rightarrow a E \mid b F$...

odstraňme FF kolizi (se zdůvodněním na tabuli)

Nejprve eliminujeme 2. pravidlo: $A \rightarrow aB \mid aED \mid bFD$

Pak již lze faktorizovat: $A \rightarrow aA_1 \mid bFD$

$$A_1 \rightarrow B \mid ED$$

$$C \rightarrow aE \mid bF$$

...

ad II) Způsobeno výskytem dvou pravidel

$B \rightarrow \gamma_1 | \gamma_2$ takových, že

$\gamma_1 \Rightarrow^* e$ a zároveň $\text{FIRST}(\gamma_2) \cap \text{FOLLOW}(B) = T_F \neq \emptyset$

Lze odstranit tzv. "pohlčením terminálu", který patří do FOLLOW

Předpokládejme, že G obsahuje pravidla:

$B \rightarrow \gamma_2$ pro něž platí $a \in \text{FIRST}(\gamma_2)$,

$B \rightarrow \gamma_1$ s vlastností $\gamma_1 \Rightarrow^* e$ a také pravidlo

$A \rightarrow \alpha B a \beta$

Transformujeme původní $G = (N, T, P, S)$, kde

$A \rightarrow \alpha B a \beta \in P$

$B \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n \in P$ jsou všechna B pravidla

na novou gramatiku $G' = (N', T, P', S)$, ve které

$N' = N \cup [Ba]$, kde $[Ba]$ je nový neterminál

$P' = P - \{A \rightarrow \alpha B a \beta\} \cup \{A \rightarrow \alpha [Ba] \beta\} \cup$

$\cup \{[Ba] \rightarrow \alpha_1 a | \alpha_2 a | \dots | \alpha_n a\}$

Odstraněním FF kolize můžeme ale způsobit FFL a naopak.

Rezultativnost LL(1) transformace se nezaručuje.

Př. V pravidlech $A \rightarrow B a C$ $B \rightarrow e | a b C$ $C \rightarrow e | c B C$
odstraňte FFL kolizi

Zavedeme neterminál $[Ba]$

$[Ba] \rightarrow a | a b C a$

A nahradíme jím část pravé strany 1.pravidla.

$A \rightarrow [Ba] C$

$B \rightarrow e | a b C$ zde již kolize FFL není

$C \rightarrow e | c B C$

$[Ba] \rightarrow a | a b C a$ tady ale vznikla FF

Př. Odstraňte kolize vzniklé v $G[E]$ po eliminaci levé rekurze. Na tabuli:

$E \rightarrow T | T E'$

$E' \rightarrow +T | +T E'$

$T \rightarrow F | F T'$

$T' \rightarrow *F | *F T'$

$F \rightarrow (E) | a$

Řešení pro pohodlné

Provedeme faktorizaci:

$$\begin{aligned} E &\rightarrow T E_1 \\ E_1 &\rightarrow E' | e \\ E' &\rightarrow + T E_2 \\ E_2 &\rightarrow E' | e \\ T &\rightarrow F T_1 \\ T_1 &\rightarrow T' | e \\ T' &\rightarrow * F T_2 \\ T_2 &\rightarrow T' | e \\ F &\rightarrow (E) | a \end{aligned}$$

E_1 a E_2 mají tutéž generační schopnost, jedno z nich je proto zbytečné a můžeme je nahradit druhým.

T_1 a T_2 mají tutéž generační schopnost, jedno z nich je proto zbytečné a můžeme je nahradit druhým.

$$\begin{aligned} E &\rightarrow T E_1 \\ E_1 &\rightarrow E' | e \\ E' &\rightarrow + T E_1 \\ T &\rightarrow F T_1 \\ T_1 &\rightarrow T' | e \\ T' &\rightarrow * F T_1 \\ F &\rightarrow (E) | a \end{aligned}$$

Vyloučíme E' a T' dosazením jejich pravých stran a dostaneme

$$\begin{aligned} E &\rightarrow T E_1 \\ E_1 &\rightarrow + T E_1 | e \\ T &\rightarrow F T_1 \\ T_1 &\rightarrow * F T_1 | e \\ F &\rightarrow (E) | a \end{aligned}$$

Což je výsledek, který při použití kratší verze odstraňování levé rekurze bychom dostali rovnou.