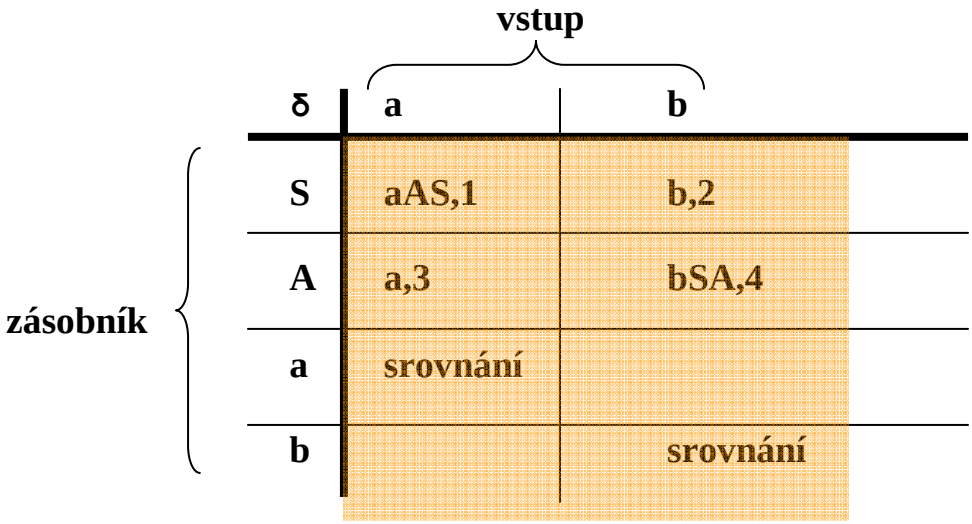
LL(k) gramatiky

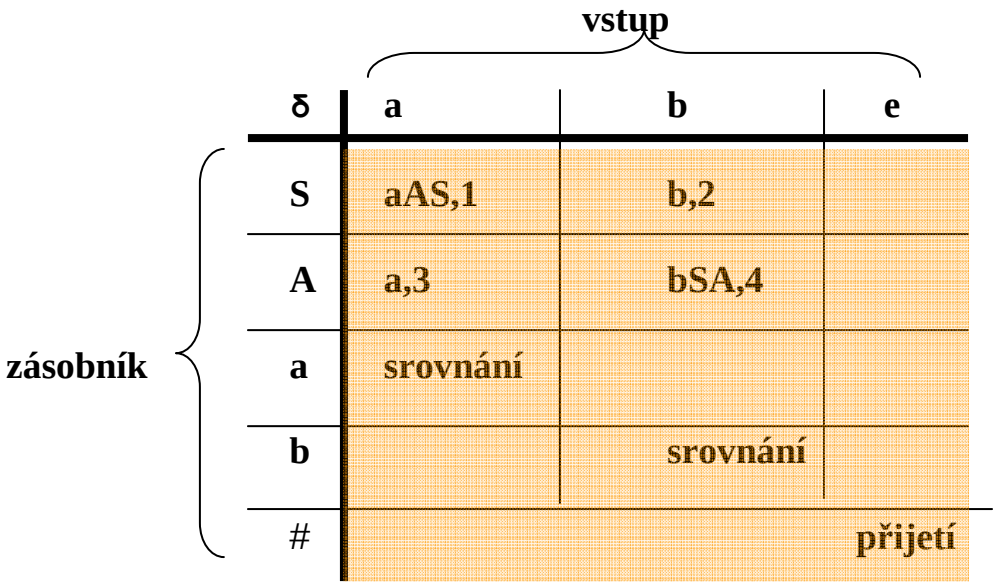
(provádí deterministický rozbor čtením textu z Leva doprava, s použitím Levé derivace a prohlédnutí k dalších symbolů vstupního textu)

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

Ekvivalentní ZA **má jediný stav q**



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



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

Je to příliš <u>jednoduchá</u> gramatika (simple S-grammer)
Jak lze pro jednoduchou gramatiku vytvořit tabulku M?
Jaké vlastnosti gramatika musí mít, aby M byla jednoznačná?

M: (N ∪ T ∪ {#}) x (T ∪ {e}) → {č.pravidla, srovnání, přijetí}

1. **Jesliže A → a α je i-té pravidlo v P, pak M(A, a) = a α, i**
2. **M(a, a) = srovnání pro všechna a ∈ T**
3. **M(#, e) = přijetí**
4. **M(ostatní) = chyba**

Algoritmus SA pro jednoduché, ..., LL(1) gramatiky

Bude provádět přechody dle 1. nebo 2., dokud nenastane 3. nebo 4.

1. **Je-li M(A, a) = β, i pak (ax, Aα, Π) ⊢ (ax, βα, Πi)**

2. **Je-li M(a, a) = srovnání pak (ax, aα, Π) ⊢ (x, α, Π)**

3. V konfiguraci (e, #, Π) končí SA přijetím, Π je levou derivací.

4. Ostatní případy končí chybou.

Konfigurací automatu je trojice (vstup,zásobník,čísla_použitých_pravidel)
!Vrchol zásobníku je vlevo. !

Př. Proveďte v G_1 analýzu věty abbab (na tabuli)
(abbab, S#, -) ⊢ (abbab, aAS#, 1) ⊢ (bbab, AS#, 1) ⊢ (bbab, bSAS#, 14) …

Expanze dle pr. 1 srovnání expanze dle pravidla 2

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

Př. G_2[S]: **1** **S → a A S**
2 **S → b**
3 **A → c S A**
4 **A → e**

(tabulku doplnit s výkladem na tabuli)

M =

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

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

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

(ab, S#, -) ⊢ (ab, aAS#, 1) ⊢ (b, AS#, 1) ⊢ (b, S#, 1 4) ⊢

Expanze dle 1 **srovnání** **expanze dle 4**

⊢ (b, b#, 1 4 2) ⊢ (e, #, 1 4 2)

Expanze dle 2 **srovnání** **výstup = levá derivace věty ab**

Zavedeme funkci **FOLLOW(X) ,** **kde X ∈ N**

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

Algoritmus výpočtu FOLLOW(A)

Vstup: Upravená gramatika G, neterminální symbol A

Metoda:

1. **Polož FOLLOW (A) = θ**
2. **Je-li A počáteční symbol G, přidej e do FOLLOW(A)**
3. **Pro všechny pravé strany pravidel z G tvaru α A β přidej FIRST (β) do FOLLOW (A), nepřidávej ale e.**
4. **Je-li v G pravidlo L → α A nebo L → α A β , kde FIRST (β) obsahuje e, pak přidej do FOLLOW (A) množinu FOLLOW (L)**

Př. Co obsahuje FOLLOW(A) v G_2 ?

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

FIRST (α) = { a; α =>* a β, a ∈ T, β ∈ (N ∪ T)* } ∪ {e; α =>* e }

Algoritmus výpočtu FIRST (α)

Vstup: Upravená gramatika, řetězec α

Metoda:

1. **Je-li α = e, pak FIRST(α) = {e}**
2. **Je-li α = a, pak FIRST(α) = {a}**
3. **Je-li α = X; X ∈ N, X → α_1 | α_2 | … | α_n jsou všechna X pravidla, pak**

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

4. **Je-li α = X β ; X ∈ Ne , β ∈ (N ∪ T)***
   pak FIRST(α) = (FIRST(X) - {e}) ∪ FIRST(β)
5. **Jinak α = X β ; X ∈ (N – Ne) ∪ T , pak FIRST(α) = FIRST(X)**

Př. Spočti FIRST a FOLLOW pro

G_3[S]: S → A B A → a A | e B → b B | 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 $\beta = X_1 X_2 \ldots X_n \in (N \cup T)^*$.**
Výstup: FIRST(β).
Metoda:

1. **Vytvoříme množinu $\mathcal{F}$ takto:**
   (a) $\mathcal{F} = \{ . X_1 X_2 \ldots X_n \}$.
   (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:
   $$\mathcal{F} = \mathcal{F} \cup \{ A \to .\alpha : B \to \gamma A \delta \in \mathcal{F}, A \in N, A \to \alpha \in P \}.$$
   (c) Jestliže v množině $\mathcal{F}$ je prvek, ve kterém je tečka na konci pravidla, tj. položka tvaru $B \to \delta.$, 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:
   $$\mathcal{F} = \mathcal{F} \cup \{ A \to \alpha B . \gamma : A \to \alpha . B \gamma \in \mathcal{F}, B \to \delta . \in \mathcal{F} \}.$$
   (d) Kroky b) a c) opakujeme tak dlouho, dokud je možno do $\mathcal{F}$ přidávat další prvky.

2. **Množinu FIRST(β) 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(β) prázdný řetězec:**
   $$\text{FIRST}(\beta) = \{ a : a \in T, A \to \alpha . a \gamma \in \mathcal{F} \} \cup \{ e : \beta . \in \mathcal{F} \}$$

Příklad

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

$E \to T\ Z$ $\qquad$ $Z \to +T\ Z \mid e$
$T \to F\ D$ $\qquad$ $D \to *\ F\ D \mid e$
$F \to (E) \mid a$

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

Na podobném principu jako výpočet funkce FIRST můžeme sestrojit 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:**
   $F := F \cup \{ C \rightarrow \alpha B.\beta : B \rightarrow \gamma. \in F, C \rightarrow \alpha B \beta \in P \}$.
   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

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

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

$F \rightarrow (E) \mid a$

Dle 1. $\quad\quad$ ***Ne*** **= {Z, D}**

Follow (Z):		**Follow (D):**		**Follow (T):**	
2a	**Z → Z.**	**2a**	**D → D .**	**2a**	**T → T .**
2b	**E → T Z .**	**2b**	**T → FD .**	**2b**	**E → T . Z**
	Z → + T Z .		**D → *FD.**		**Z → + T . Z**
	F → (E .)		**E → T . Z**	**2c**	**E → T Z .**
Proto Follow(Z) = {), e }			**Z → + T . Z**		**Z → + T Z .**
		2c	**Z → + T Z .**	**2b**	**F → (E .)**
			E → T Z .	**2e**	**Z → . + T Z**
		2b	**F → (E .)**		**Z → .**
		2e	**Z → . + T Z**		
			Z → .	**Proto Follow (T) = { +,), e}**	
		2f	**Z → + T Z . už tam je**		
		Proto Follow (D) = { +,), e }			

Algoritmus vytvoření M pro gramatiku s e-pravidly

M je definována na kartézském součinu (N ∪ T ∪ {#}) x (T ∪ {e})

1. Je-li A → a α i-té pravidlo v P, pak M(A, a) = a α, i

2. Je-li A → e i-té pravidlo v P, pak M(A, b) = e, i
pro všechna b ∈ FOLLOW(A)

3. M(a, a) = srovnání pro všechna a ∈ T

3. M(#, e) = přijetí

4. M(ostatní) = chyba

Př. G_5: S → a A
S → b
A → c S a
A → e

?zjistěme FOLLOW (A) = {a, e}
?jak vypadá rozkladová tabulka

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

Proveďme rozklad zvolené věty

LL(1) gramatiky

Algoritmus SA zůstává stejný, vytvoření M se liší.

Př. G_6[E]

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

M =

	a	+	*	(	)	e
E	TE’, 1			TE’, 1		
E’		+TE’, 2			e, 3	e,3
T	FT’, 4			FT’, 4		
T’		e, 6	*FT’, 5		e, 6	e,6
F	a, 8			(E), 7		
a						
+						
*						
(						
)						
#						

Spodní část M je stále diagonála, zahrneme ji do algoritmu analýzy

Př rozkladu zvolené věty.

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 $w = a_1\, a_2 \ldots a_n$, který je větou z $L(G)$. Pak existuje levá derivace

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

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

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

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

Vlastnosti algoritmu LL syntaktické analýzy

Předpokládejme, že $A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$ 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*, 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(α) = $\{a : \alpha \Rightarrow^* a\beta, a \in T, \beta \in (N \cup T)^*\} \cup \{e : \alpha \Rightarrow^* e\}$,
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 $A \to \alpha$ a $A \to \beta$ jsou různá pravidla v P, pak:

a) **Pro řetězce α, β musí platit, že**
   FIRST(α) $\cap$ FIRST(β)=$\varnothing$.
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(β)=$\varnothing$.
c) **Jestliže z řetězce α nelze derivovat prázdný řetězec a z β lze derivovat prázdný řetězec, pak musí platit, že**
   FIRST(α) $\cap$ FOLLOW(A)=$\varnothing$.

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 $A \to \alpha$ a $A \to \beta$ jsou různá pravidla v P, pak
FIRST(α FOLLOW(A)) $\cap$ FIRST(β FOLLOW(A)) = $\varnothing$.

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

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 $A \to \alpha$ i-té pravidlo v *P*, pak $M(A,a) = \alpha, i$ pro všechna $a \in$ FIRST(α) – {*e*}.**
2. **Je-li $A \to \alpha$ i-té pravidlo v *P* a $e \in$ FIRST(α), pak $M(A,b) = \alpha, i$ pro všechna $b \in$ FOLLOW(*A*).**
3. **M(*A*,*a*) = 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]: $E \to T\ Z$ $Z \to +T\ Z \mid e$ $T \to F\ D$
$D \to *\ F\ D \mid e$ $F \to (\ E\ ) \mid a$

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#, -) ├ **(a+a*a, TZ#, 1)** ├ **(a+a*a, FDZ#, 1 4)**
(a+a*a, aDZ#, 1 4 8) ├ **(+a*a, DZ#, 1 4 8)** ├
├ …

Silné LL(k) gramatiky

(mohou prohlédnout k symbolů ve vstupujícím řetězci)

Def.:

$$FIRST_k(\alpha) = \{ x: \quad x \in T^*, \alpha =>^* x\,y, \; y \in (N \cup T)^*, \; |x| = k\}$$
$$\cup \{ x: \quad x \in T^*, \alpha =>^* x, \; |x| < k\}$$

Def.:

$$FOLLOW_k(A) = \{ x: \; x \in T^*, S =>^* w\,A\,x\,y, \; |x| = k \}$$
$$\cup \{ x: \; x \in T^*, \; S =>^* w\,A\,x, \; |x| < k\}$$

Def.: Gramatika G se nazývá silná LL(k), když pro všechna A ∈ N platí: Jsou-li A → α , A → β různá pravidla v P , pak

$$FIRST_k(\alpha\ FOLLOW_k(A)) \cap FIRST_k(\beta\ FOLLOW_k(A)) = \theta$$

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

$$S =>^* w\,A\,\alpha => w\,\beta\,\alpha =>^* w\,x$$
$$S =>^* w\,A\,\alpha => w\,\gamma\,\alpha =>^* w\,y$$

takových, že platí $FIRST_k(x) = FIRST_k(y)$
plyne, že β = γ

Věta: G je LL(k) tehdy, platí-li pro ∀ větné formy (no jo, ale kdo je má všechny zjistit) wAα v případě, že

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

jsou dvě různá pravidla, pak

$$FIRST_k(\beta\,\alpha) \cap FIRST_k(\gamma\,\alpha) = \theta.$$

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(k) ?**

Př. Zkusme ještě G[S]: $S \rightarrow a\,A\,a\,a \mid b\,A\,b\,a$
$A \rightarrow b \mid e$

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 $\forall x \in FIRST_k(\alpha)$ taková, že $|x| = k$

2. **Je-li $A \rightarrow \alpha$ i-té pravidlo, v P a $y \in FIRST_k(\alpha)$ takové, že $|y| < k$, pak**
   $M(A, z) = \alpha, i$ pro $\forall z \in FIRST_k(y\, FOLLOW_k(A))$

3. **M(Y, x) = chyba v ostatních případech.**

Př. vytvořit M pro $G_8[S]$ na tabuli

	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 $u = FIRST_k(x)$

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

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

$$FIRST(\alpha_1) \cap FIRST(\alpha_2) \neq \theta$$

II. Kolize First Follow (FFL):

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

$$\varepsilon \in FIRST(\alpha_2), \quad FIRST(\alpha_1) \cap FOLLOW(A) \neq \theta$$

ad I) Způsobeno výskytem pravidel

$$A \rightarrow \alpha\,\alpha_1 \mid \alpha\,\alpha_2 \mid \ldots \mid \alpha\,\alpha_n$$

Lze někdy odstranit tzv. "levou faktorizací"
Transformujeme G = (N, T, P, S) , kde

$$A \rightarrow \alpha\,\alpha_1 \mid \alpha\,\alpha_2 \mid \ldots \mid \alpha\,\alpha_n \in P_A, \quad \alpha \neq \varepsilon$$

na
G' = (N', T, P', S), ve které

$$P' = (P - P_A) \cup \{ A \rightarrow \alpha\,A', \; A' \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n \}$$

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

$B \rightarrow \gamma_1 \mid \gamma_2$ **takových, že**

$\gamma_1 \Rightarrow^* e$ **a zároveň** $FIRST(\gamma_2) \cap FOLLOW(B) = T_F \neq \theta$

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

Předpokládejme, že G obsahuje pravidla:

$B \rightarrow \gamma_2$ **pro něž platí** $a \in FIRST(\gamma_2)$,

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

$A \rightarrow \alpha B a \beta$

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

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

$B \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n \in P$ jsou všechna B pravidla

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

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

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

$\cup \{[B a] \rightarrow \alpha_1 a \mid \alpha_2 a \mid \ldots \mid \alpha_n a\}$

Odstraněním FF kolize můžeme ale způsobit FFL a naopak.
Rezultativnost LL(1) transformace se nezaručuje.

Př. V pravidlech $A \rightarrow B a C$ $B \rightarrow e \mid a b C$ $C \rightarrow e \mid c B C$

odstraňte FFL kolizi

Zavedeme neterminál $[Ba]$

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

A nahradíme jím část pravé strany 1.pravidla.

$A \rightarrow [Ba] C$

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

$C \rightarrow e \mid c B C$

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

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

$E \rightarrow T \mid T E'$

$E' \rightarrow +T \mid + T E'$

$T \rightarrow F \mid F T'$

$T' \rightarrow * F \mid * F T'$

$F \rightarrow (E) \mid a$

Řešení pro pohodlné

Provedeme faktorizaci:

$E \rightarrow T\ E_1$
$E_1 \rightarrow E' \mid e$
$E' \rightarrow +\ T\ E_2$
$E_2 \rightarrow E' \mid e$
$T \rightarrow F\ T_1$
$T_1 \rightarrow T' \mid e$
$T' \rightarrow *\ F\ T_2$
$T_2 \rightarrow T' \mid e$
$F \rightarrow (\ E\ ) \mid a$

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.

$E \rightarrow T\ E_1$
$E_1 \rightarrow E' \mid e$
$E' \rightarrow +\ T\ E_1$
$T \rightarrow F\ T_1$
$T_1 \rightarrow T' \mid e$
$T' \rightarrow *\ F\ T_1$
$F \rightarrow (\ E\ ) \mid a$

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

$E \rightarrow T\ E_1$
$E_1 \rightarrow +\ T\ E_1 \mid e$
$T \rightarrow F\ T_1$
$T_1 \rightarrow *\ F\ T_1 \mid e$
$F \rightarrow (\ E\ ) \mid a$

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