Dnešní téma

Chomského NF, Pumping Lemma pro CFL, alg. CYK

Marta Vomlelová

MS 303

Bezkontextové gramatiky (CFG) v Chomského normální formě

- Chomského normální forma bezkontextové gramatiky:
  - neobsahuje zbytečné symboly (S není nikdy zbytečný)
  - všechna pravidla tvaru $A \to BC$ nebo $A \to a$, A, B, C jsou neterminály, a terminál,
    nebo $S \to \epsilon$ pro startovní symbol S, který se pak nesmí vyskytovat na pravé straně žádného pravidla.
  - Pro každý bezkontextový jazyk L existuje gramatika v Chomského normálním tvaru, která generuje L.

Postupně provedeme zjednodušení gramatiky, nejdřív:

- Eliminace *zbytečných symbolů*
- eliminace ϵ*–pravidel* $A \to \epsilon$; $A \in V$
- eliminace *jednotkových pravidel* $A \to B$ pro $A, B \in V$.

- To vše potřebujeme k formulaci iteračního lemmatu pro bezkontextové jazyky.
- A ověření, zda zadaný řetězec patří do jazyka gramatiky pomocí Cocke-Younger-Kasami algoritmu.

Eliminace zbytečných symbolů

Definition 6.40 (zbytečný, užitečný, generující, dosažitelný symbol)

- Symbol X je **užitečný** v gramatice $G = (V, T, P, S)$ pokud existuje derivace tvaru $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ kde $w \in T^*, X \in (V \cup T)$, $\alpha, \beta \in (V \cup T)^*$.
- Startovní symbol S je užitečný vždy (i když negeneruje žádné $w \in T^*$).
- Pokud X není užitečný, říkáme, že je **zbytečný**.
- X je **generující** pokud $X \Rightarrow^* w$ pro nějaké slovo $w \in T^*$. Vždy $w \Rightarrow^* w$ v nula krocích.
- X je **dosažitelný** pokud $S \Rightarrow^* \alpha X \beta$ pro nějaká $\alpha, \beta \in (V \cup T)^*$.

Chceme eliminovat ne–generující a ne–dosažitelné symboly.

Example 6.51

Uvažujme gramatiku:
$S \to AB|a$
$A \to b$

Eliminujeme B (ne–generující):
$S \to a$
$A \to b$.

Eliminujeme A (nedosažitelný):
$S \to a$.

Lemma 6.2 (Eliminace zbytečných symbolů)

Nechť $G = (V, T, P, S)$ je CFG, předpokládejme $L(G) \neq \emptyset$. Zkonstruujeme $G_1 = (V_1, T_1, P_1, S)$ následovně:

- Eliminujeme ne–generující symboly a pravidla je obsahující
- poté eliminujeme všechny nedosažitelné symboly

Pak G_1 nemá zbytečné symboly a $L(G_1) = L(G)$.

Algorithm: Generující symboly

Základ Každý $a \in T$ je generující.

Indukce Pro každé pravidlo $A \to \alpha$, kde každý symbol v α je generující. Pak i A je generující.
(Včetně $A \to \epsilon$).

Algorithm: Dosažitelné symboly

Základ S je dosažitelný.

Indukce Je–li A dosažitelný, pro všechna pravidla $A \to \alpha$ jsou všechny symboly v α dosažitelné.

Lemma 6.3 (generující/dosažitelné symboly)

Výše uvedené algoritmy najdou právě všechny generující / dosažitelné symboly.

Eliminace ϵ pravidel

Definition 6.41 (nulovatelný neterminál)

Neterminál A je **nulovatelný** pokud $A \Rightarrow^* \epsilon$.

Pro nulovatelné neterminály na pravé straně pravidla $B \to CAD$, vytvoříme dvě verze pravidla – s a bez nulovatelného neterminálu.

Algorithm: Nalezení nulovatelných symbolů v G

Základ Pokud $A \to \epsilon$ je pravidlo G, pak A je nulovatelné.

Indukce Pokud $B \to C_1 \dots C_k$, kde jsou všechna C_i nulovatelná, je i B nulovatelné (terminály nejsou nulovatelné nikdy).

Algorithm: Konstrukce gramatiky bez ϵ–pravidel z G

- Najdi nulovatelné symboly
- Pro každé pravidlo $A \to X_1 \dots X_k \in P, k \geq 1$, nechť m z X_i je nulovatelných. Nová gramatika G_1 bude mít 2^m verzí tohoto pravidla s/bez každého nulovatelného symbolu kromě ϵ v případě $m = k$.

Příklad eliminace ϵ–pravidel

Algorithm: $S \to \epsilon$

Pokud S je startovní symbol a gramatika obsahuje pravidlo $S \to \epsilon$, zavedeme nový počáteční symbol S_1, přidáme $S_1 \to S|\epsilon$, toto pravidlo necháváme a s $S \to \epsilon$ eliminujeme.

Example 6.52

Mějme gramatiku $(\{A, B, S\}, \{a, b\}, P, S)$:

$$P = \left\{ \begin{array}{l} S \to AB \\ A \to AA|B|\epsilon \\ B \to bBB|\epsilon \end{array} \right\}.$$

- Pro každé pravidlo přidáme verze s/bez nulovatelných symbolů

$$\left\{ \begin{array}{l} S \to AB|A|B \\ A \to AA|A|A|B \\ B \to bBB|bB|bB|b \end{array} \right\}.$$

- Výsledná gramatika:

$$\left\{ \begin{array}{l} S \to AB|A|B \\ A \to AA|A|B \\ B \to bBB|bB|b \end{array} \right\}.$$

Eliminace jednotkových pravidel

Definition 6.42 (jednotkové pravidlo)

Jednotkové pravidlo je $A \to B \in P$ kde A, B jsou oba neterminály.

Example 6.53

$$\left\{\begin{array}{l} I \to a|b|Ia|Ib|I0|I1 \\ F \to I|(E) \\ T \to F|T*F \\ E \to T|E+T \end{array}\right\}$$

Expanze T v $E \to T$

$E \to F|T*F$

Expanze $E \to F$

$E \to I|(E)$

Expanze $E \to I$

$E \to a|b|Ia|Ib|I0|I1$

Dohromady: $E \to a|b|Ia|Ib|I0|I1|(E)|T*F|E+T$.

Dále se musíme vyhnout možným cyklům.

Definition 6.43 (jednotkový pár)

Dvojici $A, B \in V$ takovou, že $A \Rightarrow^* B$ pouze jednotkovými pravidly nazýváme **jednotkový pár** (jednotková dvojice).

Algorithm: Nalezení jednotkových párů

Základ (A, A) pro každý $A \in V$ je jednotkový pár.

Indukce Je–li (A, B) jednotkový pár a $(B \to C) \in P$, pak (A, C) je jednotkový pár.

Example 6.54 (Jednotkové páry z předchozí gramatiky)

$(E, E), (T, T), (F, F), (I, I), (E, T), (E, F), (E, I), (T, F), (T, I), (F, I)$.

Algorithm: Eliminace jednotkových pravidel z G

- najdi všechny jednotkové páry v G
- pro každý jednotkový pár (A, B) dáme do nové gramatiky všechna pravidla $A \to \alpha$ kde $B \to \alpha \in P$ a $B \to \alpha$ není jednotkové pravidlo.

Example 6.55

$$\left\{\begin{array}{l} I \to a|b|Ia|Ib|I0|I1 \\ F \to I|(E) \\ T \to F|T * F \\ E \to T|E + T \end{array}\right\} \text{ převedeme } \left\{\begin{array}{l} I \to a|b|Ia|Ib|I0|I1 \\ F \to (E)|a|b|Ia|Ib|I0|I1 \\ T \to T * F|(E)|a|b|Ia|Ib|I0|I1 \\ E \to E + T|T * F|(E)|a|b|Ia|Ib|I0|I1 \end{array}\right\}$$

Gramatiky v normálním tvaru

Lemma 6.4 (Gramatika v normálním tvaru, redukovaná)

Mějme bezkontextovou gramatiku G,pak existuje CFG G_1 taková že $L(G_1) = L(G)$ a G_1 neobsahuje ϵ–pravidla kromě $S \to \epsilon$, neobsahuje jednotková pravidla ani zbytečné symboly. Gramatika G_1 se nazývá **redukovaná**.

Redukce bezkontextové gramatiky.

Idea důkazu:

- Začneme eliminací ϵ–pravidel.
- Eliminujeme jednotková pravidla. Tím nepřidáme ϵ–pravidla.
- Eliminujeme zbytečné symboly. Tím nepřidáme žádná pravidla.
  - Nejdříve negenerující (kromě počátečního symbolu S),
  - pak nedosažitelné.

Definition 6.44 (Chomského normální tvar)

O bezkontextové gramatice $G = (V, T, P, S)$ bez zbytečných symbolů kde jsou všechna pravidla v jednom ze tří tvarů

- $A \to BC$, $A, B, C \in V$,
- $A \to a$, $A \in V$, $a \in T$,
- případně $S \to \epsilon$, pak S není na pravé straně žádného pravidla,

říkáme, že je v **Chomského normálním tvaru (ChNF)**.

Potřebujeme dva další kroky:

- pravé strany délky 2 a více předělat na samé neterminály
- rozdělit pravé strany délky 3 a více neterminálů na více pravidel

Algorithm: neterminály

- Pro každý terminál a vytvoříme nový neterminál, řekněme A,
- přidáme pravidlo $A \to a$,
- použijeme A místo a na pravé straně pravidel délky 2 a více.

Algorithm: rozdělení pravidel

- Pro pravidlo $A \to B_1 \dots B_k$ zavedeme $k-2$ neterminálů C_i
- Přidáme pravidla $A \to B_1 C_1, C_1 \to B_2 C_2$, $\dots, C_{k-2} \to B_{k-1} B_k$.

Theorem 6.16 (Chomského normální tvar bezkontextové gramatiky)

Mějme bezkontextovou gramatiku G.Pak existuje CFG G_1 v Chomského normálním tvaru taková, že $L(G_1) = L(G)$.

Example 6.56

$$\{I \to a|b|Ia|Ib|I0|I1, F \to I|(E), T \to F|T * F, E \to T|E + T\}$$

$$\begin{cases} I \to a|b|IA|IB|IZ|IU \\ F \to LER|a|b|IA|IB|IZ|IU \\ T \to TMF|LER|a|b|IA|IB|IZ|IU \\ E \to EPT|TMF|LER|a|b|IA|IB|IZ|IU \\ A \to a \\ B \to b \\ Z \to 0 \\ U \to 1 \\ P \to + \\ M \to * \\ L \to (\\ R \to) \end{cases} \quad \begin{cases} F \to LC_3|a|b|IA|IB|IZ|IU \\ T \to TC_2|LC_3|a|b|IA|IB|IZ|IU \\ E \to EC_1|TC_2|LC_3|a|b|IA|IB|IZ|IU \\ C_1 \to PT \\ C_2 \to MF \\ C_3 \to ER \\ I, A, B, Z, U, P, M, L, R \text{ jako předchozí} \end{cases}$$

Příprava na (pumping) lemma o vkládání

Lemma (Velikost derivačního stromu gramatiky v CNF)

Mějme derivační strom podle gramatiky $G = (V, T, P, S)$ v Chomského normálním tvaru, který dává slovo w. Je–li délka nejdelší cesty n, pak $|w| \leq 2^{n-1}$.

Proof.

Indukcí podle n,

Základ $|a| = 1 = 2^0$

Indukce $2^{n-2} + 2^{n-2} = 2^{n-1}$.

□

Lemma (Důsledek)

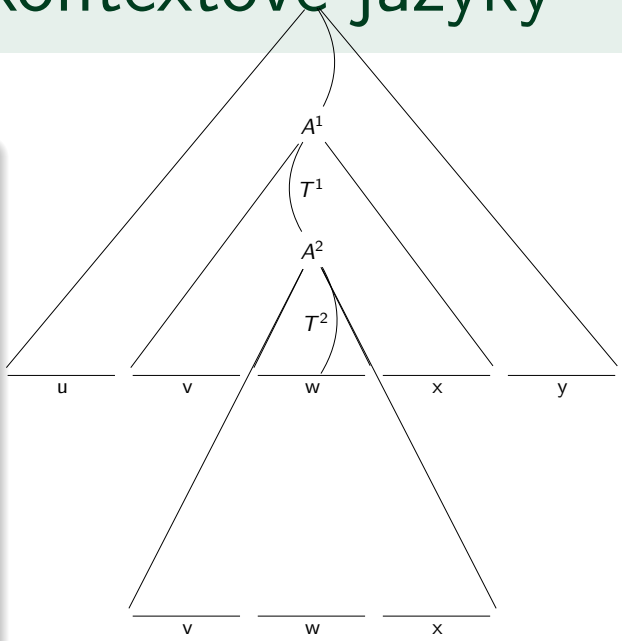
Mějme derivační strom podle gramatiky $G = (V, T, P, S)$ v Chomského normální formě, který dává slovo w, $|w| > p = 2^{n-1}$. Pak ve stromě existuje cesta delší než n.

Lemma o vkládání (pumping) pro bezkontextové jazyky

Theorem 6.17 (!!Lemma o vkládání (pumping) pro bezkontextové jazyky)

Mějme bezkontextový jazyk L. Pak existuje přirozené číslo $n \in \mathbb{N}$ takové, že každé $z \in L, |z| > n$ lze rozložit na $z = uvwxy$ kde:

- $|vwx| \leq n$
- $vx \neq \epsilon$
- $\forall i \geq 0$, $uv^i wx^i y \in L$.



Idea důkazu:

- vezmeme derivační strom pro z
- najdeme nejdelší cestu
- na ní dva stejné neterminály
- tyto neterminály určí dva podstromy
- podstromy definují rozklad slova
- nyní můžeme větší podstrom posunout ($i > 1$)
- nebo nahradit menším podstromem ($i = 0$)

Proof: $|z| > n : z = uvwxy, |vwx| \leq n, vx \neq \epsilon, \forall i \geq 0 uv^i wx^i y \in L$

- vezmeme gramatiku v Chomského NF (pro $L = \{\epsilon\}$ a $\emptyset$ zvol $n = 1$).
- Nechť $|V| = k$. Položíme $n = 2^k$.
- Pro $z \in L, |z| > 2^k$, má v derivačním stromu z cestu délky $> k$
- vezmeme cestu maximální délky; terminál kam vede označíme t
- Aspoň dva z posledních $(k+1)$ neterminálů na cestě do t jsou stejné
- vezmeme dvojici A^1, A^2 nejblíže k t (určuje podstromy T^1, T^2)
- cesta z A^1 do t je nejdelší v podstromu T^1 a má délku maximálně $(k+1)$

  tedy slovo dané stromem T^1 není delší než 2^k (tedy $|vwx| \leq n$)
- z A^1 vedou dvě cesty (ChNF), jedna do T^2 druhá do zbytku vx

  ChNF je nevypouštějící, tedy $vx \neq \epsilon$
- derivace slova $(A^1 \Rightarrow^* vA^2x, A^2 \Rightarrow^* w)$

  $S \Rightarrow^* uA^1y \Rightarrow^* uvA^2xy \Rightarrow^* uvwxy$
- posuneme–li A^2 do A^1 $(i = 0)$

  $S \Rightarrow^* uA^2y \Rightarrow^* uwy$
- posuneme–li A^1 do A^2 $(i = 2, 3, \ldots)$

  $S \Rightarrow^* uA^1y \Rightarrow^* uvA^1xy \Rightarrow^* uvvA^2xxy \Rightarrow^* uvvwxxy.$

□

Použití lemma o vkládání

Example 6.57 (ne–bezkontextový jazyk)

Následující jazyk není bezkontextový

- $\{0^i 1^i 2^i | i \geq 1\}$

- důkaz sporem: předpokládejme bezkontextovost
- z lemmatu o vkládání máme n
- zvolme $z = |0^n 1^n 2^n| > n$

žádné dělení nesplňuje PL neboť

- pumpovací slovo $|vwx| \leq n$
- tj. vždy lze pumpovat maximálně dva různé symboly
- $vx \neq \epsilon$, iterací se slovo změní
- poruší se rovnost počtu symbolů – SPOR.

Example 6.58 (ne–bezkontextový jazyk)

Následující jazyk není bezkontextový

- $\{0^i 1^j 2^k | 0 \leq i \leq j \leq k\}$

- důkaz sporem: předpokládejme bezkontextovost
- z lemmatu o vkládání máme n
- zvolme $z = |0^n 1^n 2^n| > n$

žádné dělení nesplňuje PL neboť

- pumpovací slovo $|vwx| \leq n$
- tj. vždy lze pumpovat maximálně dva různé symboly
  - pokud 0 (nebo 1), pumpujeme nahoru – SPOR $i \leq j$ (nebo $j \leq k$)
  - pokud 2 (nebo 1), pumpujeme dolů – SPOR $j \leq k$ (nebo $i \leq j$)

Použití lemma o vkládání

Example 6.59 (ne–bezkontextový jazyk)

Následující jazyk není bezkontextový

- $\{0^i 1^j 2^i 3^j | i, j \geq 1\}$

- důkaz sporem: předpokládejme bezkontextovost
- z lemmatu o vkládání máme n
- zvolme $z = |0^n 1^n 2^n 3^n| > n$
- pumpovací slovo $|vwx| \leq n$
- tj. vždy lze pumpovat maximálně dva různé sousední symboly
- poruší se rovnost počtu symbolů 0 a 2 nebo 1 a 3 – SPOR.

Example 6.60 (ne–bezkontextový jazyk)

Následující jazyk není bezkontextový

- $\{ww | w \in \{0,1\}^*\}$

- důkaz sporem: předpokládejme bezkontextovost
- z lemmatu o vkládání máme n
- zvolme $z = |0^n 1^n 0^n 1^n| > n$
- pumpovací slovo $|vwx| \leq n$
- tj. vždy lze pumpovat maximálně dva různé sousední symboly
- poruší se buď rovnost nul či jedniček
  - rozeberete 4 případy, vx obsahuje znak z prvních nul, prvních jedniček, druhých nul, druhých jedniček.

Kdy lemma o vkládání nezabere

- Lemma o vkládání je pouze implikace!

Example 6.61 (pumpovatelný, ne–bezkontextový jazyk)

$L = \{a^i b^j c^k d^l | i = 0 \vee j = k = l\}$ není bezkontextový jazyk, přesto lze pumpovat.

$i = 0 : b^j c^k d^l$ lze pumpovat v libovolném písmenu
$i > 0 : a^i b^n c^n d^n$ lze pumpovat v části obsahující a

Co s tím?

- zobecnění pumping lemmatu (Ogdenovo lemma)
  - pumpování vyznačených symbolů
- uzávěrové vlastnosti.

Příklad gramatiky

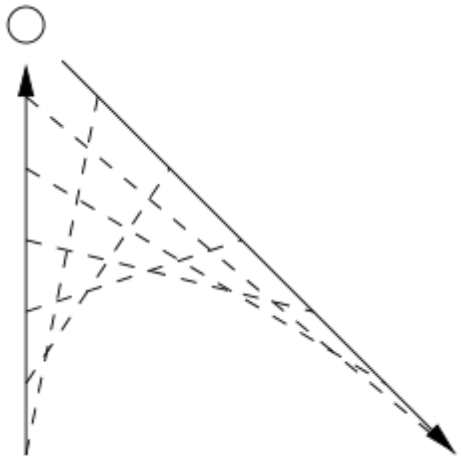
Example 6.62 (Uzávorkování v ChNF)

Mějme gramatiku $G = (\{S, L, R\}, \{(,)\}, P, S)$, kde $P = \left\{ \begin{array}{lcl} S & \to & LR|SS|LA \\ A & \to & SR \\ L & \to & (\\ R & \to &) \end{array} \right\}$.

- $S \Rightarrow LR \Rightarrow (R \Rightarrow ()$
- $S \Rightarrow SS \Rightarrow LRS \Rightarrow (RS \Rightarrow ()S \Rightarrow ()LR \Rightarrow ()(R \Rightarrow ()()$
- $S \Rightarrow LA \Rightarrow (A \Rightarrow (SR \Rightarrow (LRR \Rightarrow ((RR \Rightarrow (()R \Rightarrow (())$

Příklad CYK: pro slovo najít derivační strom

- Chceme rozhodnout, zda dané slovo w patří do jazyka bezkontextové gramatiky
  - Vezmeme gramatiku v Chomského normální formě
  - exponenciální složitost: prozkoušíme všechny derivační stromy do hloubky $|n|$,
  - polynomiálně: Cocke-Younger-Kasami algoritmus.



Example 6.63 (CYK algoritmus)

Gramatika
$G = (\{S, A, L, R\}, \{(,)\}, P, S)$:

$$P = \left\{ \begin{array}{lcl} S & \to & LR|SS|LA \\ A & \to & SR \\ L & \to & (\\ R & \to &) \end{array} \right\}$$

Tabulku vyplňujeme odspodu:

$\{S\}$					
$\{\}$	$\{A\}$				
$\{\}$	$\{S\}$	$\{\}$			
$\{\}$	$\{\}$	$\{\}$	$\{A\}$		
$\{\}$	$\{S\}$	$\{\}$	$\{S\}$	$\{\}$	
$\{L\}$	$\{L\}$	$\{R\}$	$\{L\}$	$\{R\}$	$\{R\}$
(	(	)	(	)	)

Cocke-Younger-Kasami algoritmus náležení slova do CFL

Algorithm: CYK algoritmus, v čase $O(n^3)$

- Mějme gramatiku v ChNF $G = (V, T, P, S)$ pro jazyk L a slovo $w = a_1 a_2 \dots a_n \in T^*$.
- Vytvořme trojúhelníkovou tabulku (vpravo),
  - horizontální osa je w
  - X_{ij} jsou množiny neterminálů A takových, že $A \Rightarrow^* a_i a_{i+1} \dots a_j$.

  Základ: $X_{ii} = \{A; A \to a_i \in P\}$

  Indukce: $X_{ij} = \{A \to BC; B \in X_{ik}, C \in X_{k+1,j}\}$
- Vyplňujeme tabulku zdola nahoru.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

Theorem 6.18 (CYK)

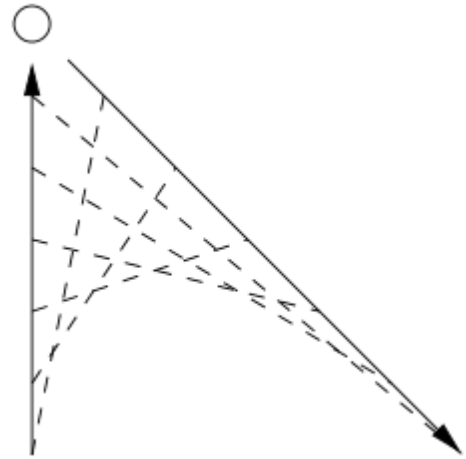
Slovo w patří do jazyka gramatiky G právě když je v CYK algoritmu $S \in X_{1,n}$.

Obecný příklad CYK

Example 6.64 (CYK algoritmus)

Uvažujme gramatiku
$G = (\{S, A, B, C\}, \{a, b\}, P, S)$:

$$P = \left\{ \begin{array}{lcl} S & \to & AB|BC \\ A & \to & BA|a \\ B & \to & CC|b \\ C & \to & AB|a \end{array} \right\}.$$



Pravidla pozpátku:

$$\begin{array}{rcl} AB & \to & \{S, C\} \\ BA & \to & \{A\} \\ BC & \to & \{S\} \\ CC & \to & \{B\} \\ b & \to & \{B\} \\ a & \to & \{A, C\} \end{array}$$

Tabulku vyplňujeme odspodu:

$\{S, A, C\}$				
-	$\{S, A, C\}$			
-	$\{B\}$	$\{B\}$		
$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

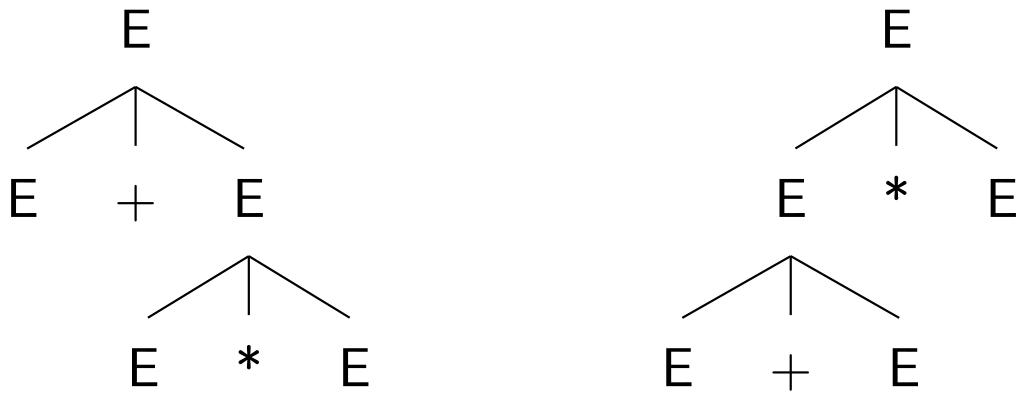
Shrnutí

- Pro každou bezkontextovou gramatiku existuje ekvivalentní gramatika v Chomského normálním tvaru generující stejný jazyk.
- Iterační lemma pro bezkontextové jazyky.
- CYK- algoritmus nalezení derivačního stromu pro slovo z gramatiky v ChNF.

Víceznačnost gramatik

Dvě derivace téhož výrazu:

$E \Rightarrow E + E \Rightarrow E + E * E$ $\quad$ $E \Rightarrow E * E \Rightarrow E + E * E$



- Rozdíl je důležitý, vlevo $1 + (2 * 3) = 7$, vpravo $(1 + 2) * 3 = 9$.
- Tato gramatika může být modifikovaná na jednoznačnou.

Example 6.65

Různé derivace mohou reprezentovat stejný derivační strom, pak není problém.

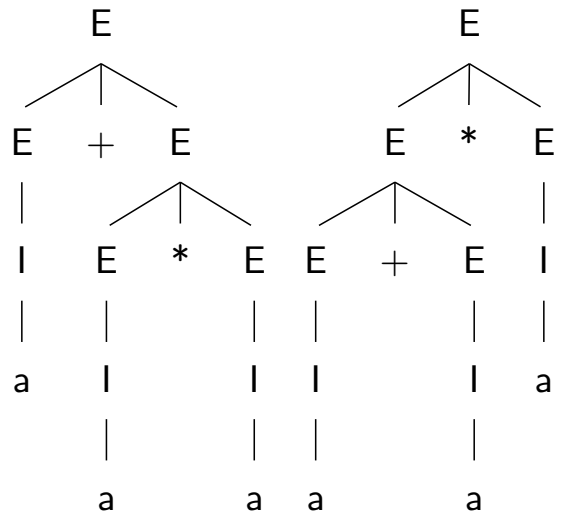
1. $E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$
2. $E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow I + b \Rightarrow a + b$.

Definition 6.45 (Jednoznačnost a víceznačnost CFG)

- Bezkontextová gramatika $G = (V, T, P, S)$ je **víceznačná** pokud existuje aspoň jeden řetězec $w \in T^*$ pro který můžeme najít dva různé derivační stromy, oba s kořenem S dávající slovo w.
- V opačném případě nazýváme gramatiku **jednoznačnou**.
- Bezkontextový jazyk L je **jednoznačný**, jestliže existuje jednoznačná CFG G tak, že $L = L(G)$.
- **Bezkontextový jazyk L je (podstatně) nejednoznačný**, jestliže každá CFG G taková, že $L = L(G)$, je nejednoznačná. Takovému jazyku říkáme i **víceznačný**.

Example 6.66 (nejednoznačnost CFG)

Dva derivační stromy dávající $a + a * a$ ukazující víceznačnost gramatiky.



Příklad víceznačného jazyka

Example 6.67 (Víceznačný jazyk)

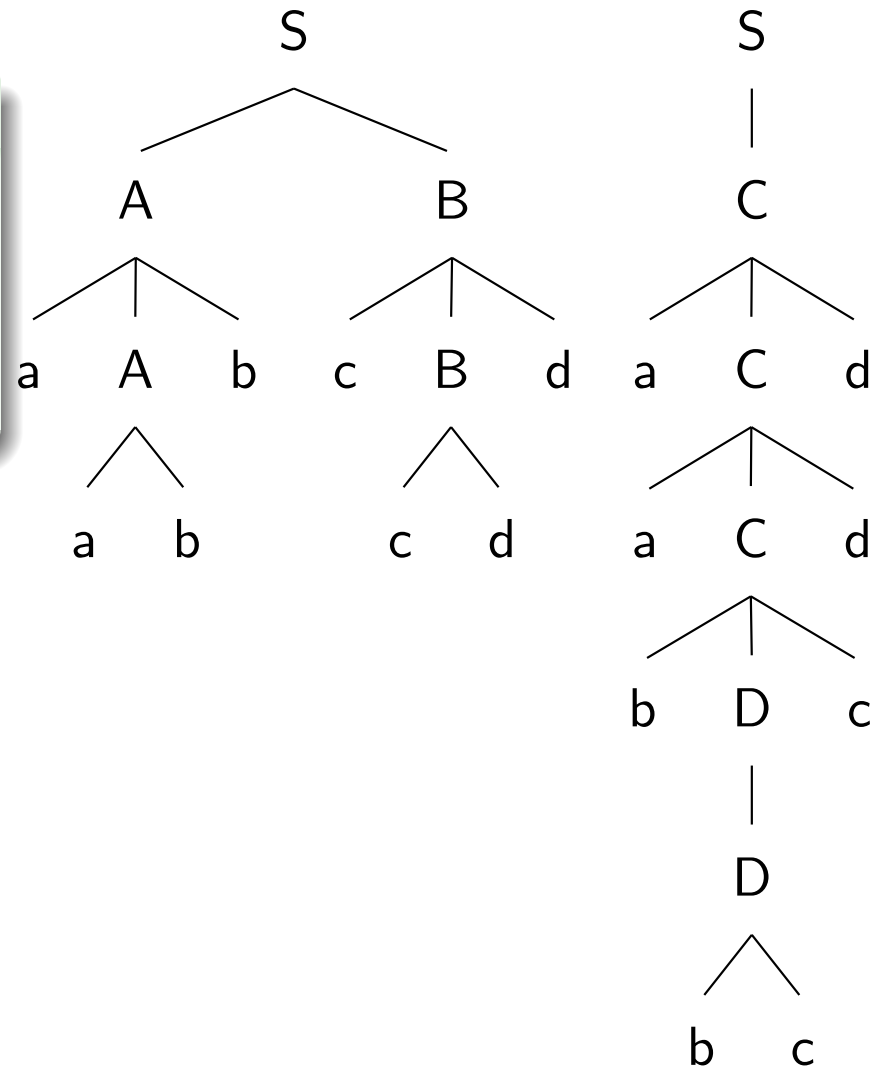
Příklad víceznačného jazyka:

$$L = \{a^n b^n c^m d^m | n \geq 1, m \geq 1\} \cup \cup \{a^n b^m c^m d^n | n \geq 1, m \geq 1\}.$$

1. $S \to AB|C$
2. $A \to aAb|ab$
3. $B \to cBd|cd$
4. $C \to aCd|aDd$
5. $D \to bDc|bc$.

Jakákoli gramatika pro daný jazyk bude generovat pro některá slova typu $a^n b^n c^n d^n$ dva různé derivační stromy.

Dva derivační stromy pro *aabbccdd*.



Odstanění víceznačnosti gramatiky

- Neexistuje algoritmus, který nám řekne, zda je daná gramatika víceznačná.
- Existují bezkontextové jazyky, pro které neexistuje jednoznačná bezkontextová gramatika, pouze víceznačné CFG.
- Existují určitá doporučení pro odstranění víceznačnosti.

Víceznačnost má různé příčiny:

- Není respektovaná priorita operátorů.
- Posloupnost identických operátorů lze shlukovat zleva i zprava.
- $S \rightarrow$ if then S else $S|$ if then $S|\epsilon$

  slovo 'if then if then else' má dva významy

  'if then (if then else)' nebo 'if then (if then) else'

  Řešení:
  - syntaktická chyba (Algol 60)
  - else patří k bližšímu if (preference pořadí pavidel)
  - závorky begin-end,odsazení v Python (asi nejčistší řešení).

Vynucení priority

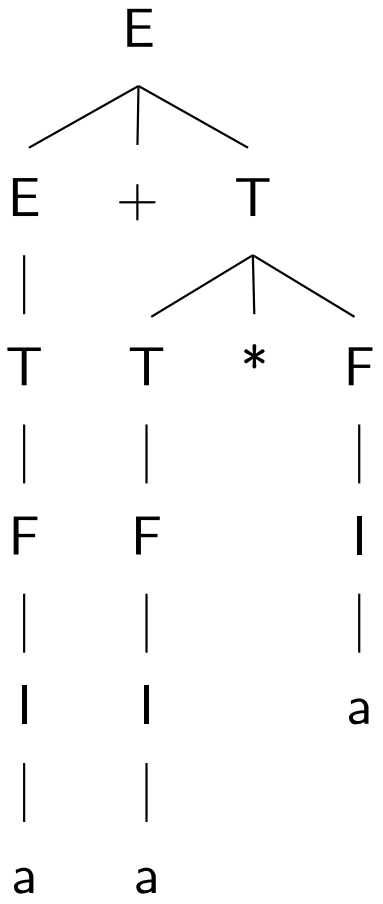
Řešením je zavést více různých proměnných, každou pro jednu úroveň 'priority'.
Konkrétně:

- **Faktor** je výraz který nesmí rozdělit žádný operátor.
  - identifikátory
  - výraz v závorkách
- **Term** je výraz, který nemůže rozdělit operátor $+$.
- **Výraz** může být rozdělen $*$ i $+$.

Jednoznačná gramatika pro výrazy:

1. $I \to a|b|Ia|Ib|I0|I1$
2. $F \to I|(E)$
3. $T \to F|T * F$
4. $E \to T|E + T$.

Jediný derivační strom pro $a + a * a$.



Jednoznačnost a kompilátory

Kompilace výrazu (zásobník na mezivýsledky + dva registry):

(1) $E \to E + T$ ... pop r1; pop r2; add r1,r2; push r2
(2) $E \to T$
(3) $T \to T * F$ ... pop r1; pop r2; mul r1,r2; push r2
(4) $T \to F$
(5) $F \to (E)$
(6) $F \to a$ ... push a

- 'a+a*a' získáme postupnou aplikací pravidel 1,2,4,6,3,4,6,6
- posloupnost obrátíme a vybereme pouze pravidla generující kód

  6,6,3,6,1
- nyní nahradíme pravidla příslušným kódem

  push a; push a; pop r1; pop r2; mul r1,r2; push r2; push a; pop r1; pop r2; add r1,r2; push r2

Shrnutí

- Gramatiky
  - obecné
  - kontextové
  - bezkontextové
  - regulární, pravé lineární
- jazyk gramatiky, derivace, derivace dává slovo, derivační strom (pro bezkontextové gramatiky), ekvivalentní gramatiky
- ne každá lineární gramatika má ekvivalentní pravou lineární
- bezkontextové gramatiky
- jednoznačné a (podstatně) víceznačné gramatiky.