Prednáška 5 - Chomského normálny tvar, CYK algoritmus; Množiny *FIRST*, *FOLLOW*

Ing. Viliam Hromada, PhD.

C-510
Ústav informatiky a matematiky
FEI STU

viliam.hromada@stuba.sk

Chomského normálny tvar (forma) gramatiky

Definícia

*Bezkontextová gramatika $G = (N, T, P, S)$ je v **Chomského normálnom tvare**, ak každé pravidlo je v jednom z tvarov:*

1. $A \to BC$ *pre* $A \in N; B, C \in (N - \{S\})$;
2. $A \to a$ *pre* $A \in N, a \in T$;
3. $S \to \varepsilon$.

Úprava do CHNT

Vstup: Bezkontextová gramatika $G = (N, T, P, S)$
Výstup: Gramatika G v CHNT

1: Odstráň z gramatiky ε-pravidlá.
2: Uprav gramatiku, aby počiatočný neterminál nebol v pravej strane žiadneho pravidla.
3: Odstráň z gramatiky jednoduché pravidlá.
4: Odstráň z gramatiky nadbytočné a nedostupné symboly.
5: **pre všetky** pravidlá tvaru $A \to X_1X_2...X_n$, $n \geq 3$ **rob**
6: nahraď pravidlami $A \to X_1A_1, A_1 \to X_2A_2, ..., A_{n-2} \to X_{n-1}X_n$, kde $A_1, ..., A_{n-2}$ sú nové neterminály.
7: **koniec pre**
8: **pre všetky** pravidlá tvaru $A \to Y_1Y_2$, kde $Y_1 \in T \vee Y_2 \in T$ **rob**
9: nahraď na pravej strane pravidiel terminály $Y_1(Y_2)$ novými neterminálmi $V_1(V_2)$ a pridaj pravidlo (pravidlá) $V_1 \to Y_1$, resp. $V_2 \to Y_2$
10: **koniec pre**

Krok č. 2 z algoritmu

V algoritme sa po odstránení ε-pravidiel robí úprava: *Uprav gramatiku, aby počiatočný neterminál nebol v pravej strane žiadneho pravidla.*.

- V prípade, že sa pri odstránení ε pridal do gramatiky nový neterminál $\acute{S}$ a pravidlá $\acute{S} \to \varepsilon \mid S$, tak tento krok je irelevantný.
- V prípade, že sa v gramatike jej **aktuálny** počiatočný neterminál S nachádza v niektorej pravej strane niektorého pravidla, **pridá sa nový počiatočný neterminál** $\acute{S}$ a pravidlo $\acute{S} \to S$.

CHNT - príklad

Gramatika $G = (\{S, A\}, \{a, b, c\}, P, S)$ s pravidlami:

$S \to aASa \mid c$

$A \to Abc \mid c$

1) ε-pravidlá nie je potrebné odstrániť - žiadne neobsahuje.
2) Zavedieme symbol $\acute{S}$ a pridáme pravidlo:

$\acute{S} \to S$

$S \to aASa \mid c$

$A \to Abc \mid c$

CHNT - príklad (pokr.)

3,4) Po odstránení jednoduchých pravidiel a nadbytočných symbolov:

$\acute{S} \to aASa \mid c$

$S \to aASa \mid c$

$A \to Abc \mid c$

5-7) Úprava pravidiel dlhších ako 3 symboly:

$\acute{S} \to aA_1 \mid c$

$S \to aA_1 \mid c$

$A \to AA_3 \mid c$

$A_1 \to AA_2$

$A_2 \to Sa$

$A_3 \to bc$

CHNT - príklad (pokr.)

8-10) Odstránenie terminálov z pravidiel s pravou stranou dĺžky 2

$\acute{S} \to V_1 A_1 \mid c$

$S \to V_1 A_1 \mid c$

$A \to A A_3 \mid c$

$A_1 \to A A_2$

$A_2 \to S V_1$

$A_3 \to V_2 V_3$

$V_1 \to a$

$V_2 \to b$

$V_3 \to c$

CHNT - poznámky

- Ak vieme v danej gramatike odvodiť slovo w, potom dĺžka jeho odvodenia v CHNT je $2|w| - 1$.
- Strom odvodenia (derivačný strom) slova v gramatike v CHNT je binárny strom.

Praktické využitie Chomského normálneho tvaru

- Chomského normálny tvar sa v praxi využíva v tzv. CYK algoritme.
- CYK algoritmus je **polynomiálnym** algoritmom na zistenie, či **ľubovoľná bezkontextová gramatika** $G = (N, T, P, S)$ generuje zadaný **reťazec terminálov** $w \in T^*$, teda či $w \in L(G)$.

CYK algoritmus

- Predpokladajme, že máme danú bezkontextovú gramatiku $G = (N, T, P, S)$ v **Chomského normálnom tvare**.
- Chceme zistiť, či slovo $w \in L(G)$, kde $w = a_1 ... a_n, a_i \in T$.
- Ak $w = \varepsilon$, stačí sa pozrieť, či je v gramatike pravidlo $S \to \varepsilon$.
- Pre $w \in T^+$ vie CYK algoritmus (Cocke-Younger-Kasami) zistiť či $w \in L(G)$ so zložitosťou $O(n^3 \cdot |P|)$, kde n je dĺžka analyzovaného slova pre **hocijakú** bezkontextovú gramatiku a $|P|$ počet pravidiel gramatiky.
- Základnou ideou algoritmu je konštrukcia množín:

$$N_{i,j} = \{A \mid A \Rightarrow^* a_i a_{i+1} ... a_j, A \in N\},$$

$1 \leq i \leq j \leq n$.

- Ak $S \in N_{1,n}$ potom slovo $w \in L(G)$.

CYK algoritmus

- Postupne sa hľadajú neterminály, ktoré vedia generovať podreťazce daného slova w.
- Na začiatku sa určia neterminály generujúce podreťazce dĺžky 1.
  - Keďže gramatika je v CHNT, dané neterminály sa určia priamo z pravidiel.
- Následne sa využíva nasledovná vlastnosť:
  1. Ak $B \in N_{j,k}$, t.j. $B \Rightarrow^* a_j ... a_k$,
  2. ak $C \in N_{k+1,j+i}$, t.j. $C \Rightarrow^* a_{k+1} ... a_{j+i}$,
  3. a zároveň existuje pravidlo $A \rightarrow BC$, potom
  4. $A \in N_{j,j+i}$, pretože $A \Rightarrow BC \Rightarrow^* a_j ... a_k a_{k+1} ... a_{j+i}$.

CYK algoritmus I

Vstup: Bezkontextová gramatika $G = (N, T, P, S)$ v CNF, slovo $w \neq \varepsilon$
Výstup: Zistenie, či $w \in L(G)$

1: $w = a_1 ... a_n$
2: **pre všetky** $i \in \{1, ..., n\}$ **rob**
3: **pre všetky** $j \in \{i, ..., n\}$ **rob**
4: $N_{i,j} \leftarrow \emptyset$
5: **koniec pre**
6: **koniec pre**
7: **pre všetky** $i \in \{1, ..., n\}$ **rob**
8: **pre všetky** pravidlá tvaru $A \rightarrow a_i$ **rob**
9: $N_{i,i} \leftarrow N_{i,i} \cup \{A\}$
10: **koniec pre**
11: **koniec pre**
12: **pre všetky** $i \in \{1, 2, ..., n-1\}$ **rob**
13: **pre všetky** $j \in \{1, 2, ..., n-i\}$ **rob**

CYK algoritmus II

14: **pre všetky** $k \in \{j, ..., j+i-1\}$ **rob**
15: **pre všetky** pravidlá $A \to BC$, kde $B \in N_{j,k}$ a zároveň $C \in N_{k+1,j+i}$ **rob**
16: $N_{j,j+i} \leftarrow N_{j,j+i} \cup \{A\}$
17: **koniec pre**
18: **koniec pre**
19: **koniec pre**
20: **koniec pre**
21: **ak** $S \in N_{1,n}$ **potom**
22: **vráť** „Slovo patrí do jazyka. “
23: **inak**
24: **vráť** „Slovo nepatrí do jazyka. “
25: **koniec ak**

CYK algoritmus - príklad

Je daná gramatika:

$S \to AB \mid AC \mid a$

$A \to DA \mid a$

$B \to b \mid d$

$C \to BC \mid c$

$D \to d$

Zistite pomocou CYK algoritmu, či slovo $dab \in L(G)$.

CYK algoritmus - príklad (pokr.)

Najprv sa zostroja množiny

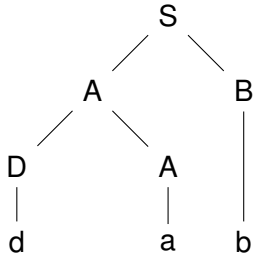
$N_{1,1} = \{B, D\}$, kvôli pravidlám $B \to d$ a $D \to d$.
$N_{2,2} = \{S, A\}$, kvôli pravidlám $S \to a$ a $A \to a$.
$N_{3,3} = \{B\}$, kvôli pravidlu $B \to b$.

Ďalej podľa iterácii:

$i = 1, j = 1, k = 1$	$A \to DA$, $D \in N_{1,1}$, $A \in N_{2,2}$ teda $A \in N_{1,2}$
$i = 1, j = 2, k = 2$	$S \to AB$, $A \in N_{2,2}$, $B \in N_{3,3}$ teda $S \in N_{2,3}$
$i = 2, j = 1, k = 1$	$N_{1,1} = \{B, D\}, N_{2,3} = \{S\}$ do $N_{1,3}$ nepribudne nič
$i = 2, j = 1, k = 2$	$S \to AB$, $A \in N_{1,2}$, $B \in N_{3,3}$ teda $S \in N_{1,3}$

Keďže $S \in N_{1,3}$, tak $dab \in L(G)$.

Na základe príslušných množín vieme teoreticky aj nájsť derivačný strom, resp. odvodenie:



- Základnou úlohou CYK je zistiť, či slovo patrí do jazyka generovaného gramatikou, avšak v podstate sa dá použiť aj na zostrojenie derivačného stromu (na základe množín $N_{i,j}$).
- Výhodou CYK algoritmu je, že vie spracovať všetky bezkontextové gramatiky.
  - Aj nejednoznačné, aj jednoznačné
  - Aj nedeterministické, aj deterministické
- Nevýhodou CYK algoritmu je, že je potrebné, aby gramatika bola v CHNT a taktiež má kubickú zložitosť vzhľadom na dĺžku vstupného slova.
- Existujú aj ďalšie algoritmy, ktoré síce nevedia spracovať všetky bezkontextové gramatiky, avšak majú lineárnu zložitosť vzhľadom na dĺžku vstupného slova - viac si povieme pri syntaktickej analýze.

Analýza vlastností bezkontextových gramatík

- Je zrejmé, že to, aké reťazce generujú bezkontextové gramatiky - a či existuje derivácia nejakého reťazca v gramatike - úzko súvisí s pravidlami danej gramatiky.
- Preto sa pri bezkontextových gramatikách vyšetrujú tzv. množiny *FIRST* a *FOLLOW*, ktoré bližšie špecifikujú, aké vlastnosti majú vetné formy, resp. reťazce vznikajúce počas derivácií.
- Následne má potom zmysel tieto vlastnosti využívať pri zisťovaní, či derivácie reťazcov v gramatike existujú alebo nie.
- Množina *FIRST* pojednáva o tom, aké terminálne symboly môžu stáť **na prvom mieste** reťazca, ktorý viem derivovať z nejakej postupnosti symbolov gramatiky.
- Množina *FOLLOW* pojednáva o tom, aké terminálne symboly môžu **nasledovať** nejaký neterminál vo vetnej forme.

Množina *FIRST*

Definícia

Nech $G = (N, T, P, S)$ je redukovaná bezkontextová gramatika a $\alpha \in (N \cup T)^*$. Potom

$$FIRST(\alpha) = \{a \in T \mid \alpha \Rightarrow^* a\beta, \beta \in (N \cup T)^*\} \cup \{\varepsilon \mid \alpha \Rightarrow^* \varepsilon\}. \quad (1)$$

T.j. pre reťazec zložený z neterminálov a terminálov α je $FIRST(\alpha)$ množina terminálov, ktorými môžu začínať reťazce odvodené z α. A ak je možné z α odvodiť ε, tak do $FIRST(\alpha)$ patrí aj ε.

Množina *FIRST* - rôzne situácie

- Definícia hovorí, že množinu *FIRST* môžeme hľadať pre ľubovoľný reťazec nad symbolmi gramatiky.
- V praxi sa najprv nájde množina *FIRST* pre jednotlivé neterminály gramatiky, t.j. $FIRST(A)$ pre všetky neterminály $A \in N$.
- Následne sa pri hľadaní množiny $FIRST(\alpha)$ pre ľubovoľné reťazce $\alpha \in (N \cup T)^*$ vychádza z jednotlivých množín *FIRST* pre neterminály.

Množina *FIRST* - rôzne situácie

1. ak gramatika obsahuje pravidlo $A \to a\alpha$, potom $a \in FIRST(A)$
2. ak $A \in N_\varepsilon$, potom $\varepsilon \in FIRST(A)$
3. ak gramatika obsahuje pravidlo $A \to \alpha$, potom $FIRST(\alpha) \subseteq FIRST(A)$

Množina *FIRST* - príklad

Nech je daná gramatika $G = (N, T, P, S)$, kde
$N = \{S, A, B, C\}$, $T = \{a, b, +, (,), \$\}$ a pravidlá:

$S \to C\$$

$A \to b \mid \varepsilon$

$B \to +S \mid \varepsilon$

$C \to A(C) \mid aB$

Potom:

- $FIRST(B) = \{+, \varepsilon\}$
- $FIRST(C) = \{b, a, (\}$
- $FIRST(A) = \{b, \varepsilon\}$
- $FIRST(S) = \{a, b, (\}$

$FIRST(aBA) = \{a\}$

$FIRST(AB) = \{b, \varepsilon, +\}$

$FIRST(A\$B) = \{b, \$\}$

Množina $FIRST(X)$ I

Vstup: Redukovaná bezkontextová gramatika $G = (N, T, P, S)$.
Výstup: Množiny $FIRST(X)$ pre $X \in N \cup T$

1: **pre všetky** $A \in N$ **rob**
2: $\quad FIRST(A) \leftarrow \emptyset$
3: **koniec pre**
4: **pre všetky** $a \in T$ **rob**
5: $\quad FIRST(a) \leftarrow \{a\}$
6: **koniec pre**
7: **pre všetky** pravidlá $A \rightarrow a\alpha$ **rob**
8: $\quad FIRST(A) \leftarrow FIRST(A) \cup \{a\}$
9: **koniec pre**
10: **pre všetky** $A \in N_\varepsilon$ **rob**
11: $\quad FIRST(A) \leftarrow FIRST(A) \cup \{\varepsilon\}$
12: **koniec pre**

Množina $FIRST(X)$ II

13: **opakuj**
14: **pre všetky** pravidlá $A \to B\alpha$ **rob**
15: $FIRST(A) \leftarrow FIRST(A) \cup FIRST(B\alpha)$
16: **koniec pre**
17: **pokiaľ** zmenila sa niektorá z množín $FIRST(A), A \in N$

Množina $FIRST(\alpha)$

Vstup: množiny $FIRST(X), X \in N \cup T$; reťazec $\alpha \in (N \cup T)^*$.
Výstup: $FIRST(\alpha), \alpha \in (N \cup T)^*$

ak $\alpha = \varepsilon$ **potom**
  vráť $\{\varepsilon\}$
inak
  nech $\alpha = X_1 X_2 ... X_n$
  $FIRST(\alpha) \leftarrow FIRST(X_1) - \{\varepsilon\}$
  $i \leftarrow 1$
  pokiaľ $\varepsilon \in FIRST(X_i) \wedge i < n$ **rob**
    $i \leftarrow i + 1$
    $FIRST(\alpha) \leftarrow FIRST(\alpha) \cup (FIRST(X_i) - \{\varepsilon\})$
  koniec pokiaľ
  ak $i = n \wedge \varepsilon \in FIRST(X_n)$ **potom**
    $FIRST(\alpha) \leftarrow FIRST(\alpha) \cup \{\varepsilon\}$
  koniec ak
  vráť $FIRST(\alpha)$
koniec ak

Množina *FIRST* - príklad

Nech je daná gramatika $G = (N, T, P, S)$, kde
$N = \{S, A, B, C\}$, $T = \{a, b, +, (,), \$\}$ a pravidlá:

$S \to C\$$

$A \to b \mid \varepsilon$

$B \to +S \mid \varepsilon$

$C \to A(C) \mid aB$

Určte $FIRST(X)$ pre všetky $X \in N \cup T$, $FIRST(BS)$, $FIRST(AB)$, $FIRST(CB)$

Množina *FIRST* - príklad

FIRST(X):

FIRST	S	A	B	C	a	b	$+$	$($	$)$	\$
Terminály:					a	b	$+$	$($	$)$	\$
Pravidlá $A \to a\alpha$		b	$+$	a						
Neterminály $A \in N_\varepsilon$		ε	ε							
1. iterácia:	a			b, (						
2. iterácia:	b, (									
3. iterácia:										
Celkom:	$a, b, ($	b, ε	$+, \varepsilon$	$a, b, ($	a	b	$+$	$($	$)$	\$

Množina *FIRST* - príklad

$$FIRST(BS) = \{+, a, b, (\}$$
$$FIRST(AB) = \{b, +, \varepsilon\}$$
$$FIRST(CB) = \{a, b, (\}$$

Množina *FOLLOW*

Definícia

Nech $G = (N, T, P, S)$ je redukovaná bezkontextová gramatika a $A \in N$. Potom

$$FOLLOW(A) = \{a \in T \mid S \Rightarrow^{+} \alpha A a \beta\} \cup \{\varepsilon \mid S \Rightarrow^{*} \gamma A\}, \tag{2}$$

kde $\alpha, \beta, \gamma \in (N \cup T)^{*}$.

T.j. pre neterminál *A* je množina *FOLLOW*(*A*) množina takých terminálov, ktoré môžu v nejakej vetnej forme stáť **hneď za neterminálom** *A*. Do množiny patrí aj ε, ak môže neterminál *A* stáť **na konci** nejakej vetnej formy.

Množina $FOLLOW(A)$

- Predpokladajme, že gramatika obsahuje pravidlo $A \to \alpha B \beta; \alpha, \beta \in (N \cup T)^*; A, B \in N$
- Predpokladajme, že existuje odvodenie: $S \Rightarrow^* ...Aa... \Rightarrow^* ...\alpha B \beta a... \Rightarrow^* ...$
  1. Vždy platí, že $\varepsilon \in FOLLOW(S)$
  2. $a \in FOLLOW(A)$
  3. $(FIRST(\beta) - \{\varepsilon\}) \subseteq FOLLOW(B)$
  4. (ak $\varepsilon \in FIRST(\beta)$, potom $FOLLOW(A) \subseteq FOLLOW(B)$.

Príklad: Nech je daná gramatika $G = (N, T, P, S)$, kde $N = \{S, A, B, C\}$, $T = \{a, b, +, (,), \$\}$ a pravidlá:

$S \to C\$$

$A \to b \mid \varepsilon$

$B \to +S \mid \varepsilon$

$C \to A(C) \mid aB$

Potom:

$FOLLOW(S) = \{\varepsilon, \$,)\}$

$FOLLOW(A) = \{(\}$

$FOLLOW(B) = \{\$,)\}$

$FOLLOW(C) = \{\$,)\}$

Množina $FOLLOW(A)$ pre $A \in N$

Vstup: Redukovaná bezkontextová gramatika $G = (N, T, P, S)$, množiny $FIRST(X)$ pre $X \in N \cup T$

Výstup: množiny $FOLLOW(A)$ pre $A \in N$

1: **pre všetky** $A \in N$ **rob**
2: $FOLLOW(A) \leftarrow \emptyset$
3: **koniec pre**
4: $FOLLOW(S) \leftarrow \{\varepsilon\}$
5: **opakuj**
6: **pre všetky** $B \in N$ v pravých stranách pravidiel $A \rightarrow \alpha B \beta$ **rob**
7: $FOLLOW(B) \leftarrow FOLLOW(B) \cup (FIRST(\beta) - \{\varepsilon\})$
8: **ak** $\varepsilon \in FIRST(\beta)$ **potom**
9: $FOLLOW(B) \leftarrow FOLLOW(B) \cup FOLLOW(A)$
10: **koniec ak**
11: **koniec pre**
12: **pokiaľ** sa zmenila niektorá z množín $FOLLOW(B), B \in N$

Množina $FOLLOW(A)$ - príklad

Príklad: Nech je daná gramatika $G = (N, T, P, S)$, kde $N = \{S, A, B, C\}$, $T = \{a, b, +, (,), \$\}$ a pravidlá:

$S \to C\$$

$A \to b \mid \varepsilon$

$B \to +S \mid \varepsilon$

$C \to A(C) \mid aB$

Množina $FOLLOW(A)$ - príklad (pokr.)

$FOLLOW$	S	A	B	C
Začiatočný symbol	ε			
1. iterácia: $S \to \underline{C}\$$				\$
$B \to +\underline{S}$				
$C \to \underline{A}(C)$		(		
$C \to A(\underline{C})$				)
$C \to a\underline{B}$			\$,)	
2. iterácia: $S \to \underline{C}\$$				
$B \to +\underline{S}$	\$,)			
...				

Zvyšok tabuľky už budú prázdne riadky, pričom po 3. iterácii sa už množiny nezmenia a algoritmus končí. **V tabuľke uvádzame len novo-zistené symboly.**

Použitá literatúra

Aho, A., Lam, M., Sethi, R., Ullman, J.: *Compilers: Principles, techniques and tools.*

Dedera, Ľ: *Počítačové jazyky a ich spracovanie*.

Linz, P.: *An Introduction to Formal Languages and Automata*.