Bezkontextové jazyky, derivačné stromy, redukovaná gramatika

Ing. Viliam Hromada, PhD.

C-510
Ústav informatiky a matematiky
FEI STU

viliam.hromada@stuba.sk

Pre pripomenutie:

- **Regulárne jazyky** - jazyky generovateľné pomocou tzv. regulárnych gramatík.
- **Bezkontextové jazyky** - jazyky generovateľné pomocou tzv. bezkontextových gramatík.

Bezkontextové jazyky

- Regulárne jazyky sa ľahko spracujú z výpočtového hľadiska, ich použitie je však obmedzené - napr. popis a identifikácia základných symbolov programovacích jazykov.
- Na špecifikáciu zložitejších konštrukcií programovacích jazykov (výrazy, deklarácie, podprogramy) sa používajú bezkontextové jazyky, resp. gramatiky.
- Pre pripomenutie, bezkontextová gramatika $G = (N, T, P, S)$ obsahuje pravidlá v tvare:

$$A \to \alpha,$$

kde $A \in N, \alpha \in (N \cup T)^*$.

Príklad bezkontextovej gramatiky

Nech $G = (\{E, T, F\}, \{(,), +, *, \mathrm{id}\}, P, E)$, kde pravidlá P:

$E \to E + T \mid T$

$T \to T * F \mid F$

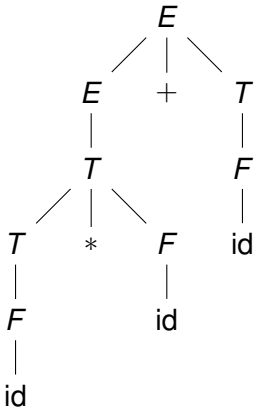
$F \to (E) \mid \mathrm{id}.$

Zjednodušená gramatika na generovanie výrazov v programovacích jazykoch.
Existuje v nej napr. odvodenie:

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow T * F + T \Rightarrow F * F + T \Rightarrow \mathrm{id} * F + T$$

$$\Rightarrow \mathrm{id} * F + F \Rightarrow \mathrm{id} * F + \mathrm{id} \Rightarrow \mathrm{id} * \mathrm{id} + \mathrm{id}$$

V bezkontextovej gramatike vieme ku každému odvodeniu kresliť **derivačný strom** (strom odvodenia):



Strom odvodenia

Definícia

*Nech $G = (N, T, P, S)$ je bezkontextová gramatika. Ohodnotený strom sa nazýva **strom odvodenia** (derivačný strom) slova w v gramatike G, ak spĺňa nasledovné podmienky:*

- *Každý vrchol stromu je ohodnotený symbolom z množiny $N \cup T_\varepsilon$.*
- *Koreň stromu je ohodnotený začiatočným symbolom S.*
- *Ak má nejaký vrchol stromu potomkov, potom je ohodnotený symbolom z množiny N.*
- *Ak $X_1, X_2, ..., X_k$ sú ohodnotenia priamych potomkov vrcholu A, potom v P musí existovať pravidlo $A \to X_1 X_2 ... X_k$.*
- *Listy stromu sú ohodnotené symbolmi z množiny T_ε.*
- *Slovo $w \in T^*$ sa dá získať zreťazením ohodnotení listov stromu zľava doprava.*

Definícia

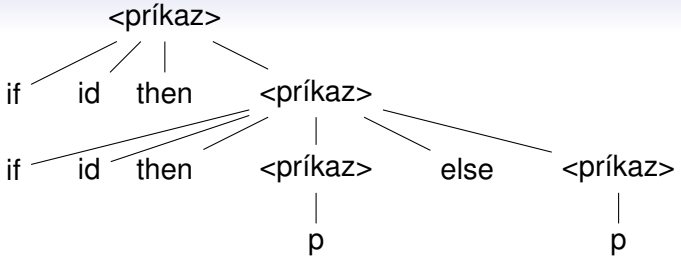
*Bezkontextová gramatika G sa nazýva **jednoznačná**, ak ku každému slovu $w \in L(G)$ existuje jediný strom odvodenia. V opačnom prípadne sa gramatika nazýva **nejednoznačná** (viacznačná, ambiguous).*

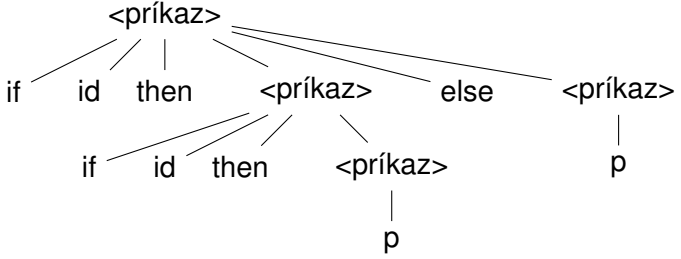
Nejednoznačná gramatika - príklad

Gramatika $G = (\{\text{<príkaz>}\}, \{\text{id, p, if, then, else}\}, P, \text{<príkaz>})$, pravidlá P:

1. <príkaz> $\rightarrow$ if id then <príkaz>
2. <príkaz> $\rightarrow$ if id then <príkaz> else <príkaz>
3. <príkaz> $\rightarrow$ p

Zjednodušená gramatika reprezentujúca podmienené príkazy, **id** zastupuje identifikátor (premennú) predstavujúci podmienku, **p** zastupuje nejaký príkaz. Potom pre slovo (reťazec terminálov): **if id then if id then p else p** existujú 2 rôzne stromy odvodenia:


<príkaz>
if
id
then
<príkaz>
if
id
then
<príkaz>
else
<príkaz>
p
p


<príkaz>
if
id
then
<príkaz>
else
<príkaz>
if
id
then
<príkaz>
p
p

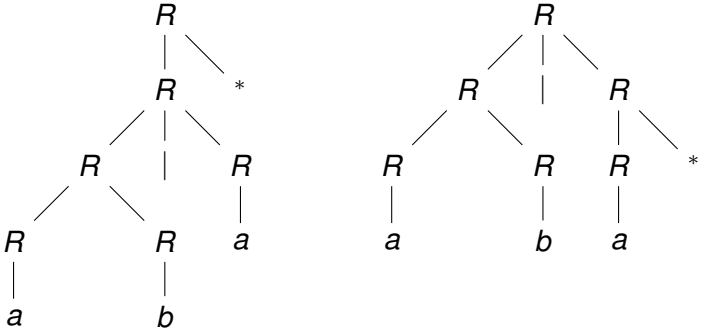
Gramatika pre regulárne výrazy nad $A = \{a, b\}$

Gramatika $G = \{\{R\}, \{^*, |, a, b\}, P, R\}$, kde pravidlá:

$R \to R\|R$	(zjednotenie)
$R \to RR$	(zreťazenie)
$R \to R^*$	(iterácia)
$R \to a$	
$R \to b$	
$R \to \varepsilon$	

Gramatika pre regulárne výrazy nad $A = \{a, b\}$

V takejto gramatike by mal regulárny výraz $ab|a^*$ viacero derivačných stromov, napr.



Čo by teoreticky mohlo znamenať nejednoznačnosť príslušného regulárneho výrazu pri jeho počítačovom spracovaní, pretože prvý regulárny výraz vyzerá ako $((ab)|a)^*$ a druhý ako $(ab)|(a^*)$.

Problém určenia nejednoznačnosti gramatiky

- Ak máme zadanú nejakú bezkontextovú gramatiku $G = (N, T, P, S)$, tak problém zistenia, či je gramatika **jednoznačná**/**nejednoznačná** je vo všeobecnosti **nerozhodnuteľný**!
- To znamená, že **neexistuje** algoritmus, ktorý by pre ľubovoľnú bezkontextovú gramatiku vedel v konečnom čase povedať, či je jednoznačná alebo nejednoznačná!

- Nejednoznačné gramatiky predstavujú problém - ak existujú pre 1 slovo viaceré stromy odvodenia, asi má slovo viacero sémantických interpretácií.
- Je zrejmé, že odvodenie slova a derivačný strom sú úzko prepojené - ak existuje odvodenie, existuje i derivačný strom (a naopak).
- Pri bezkontextových gramatikách si môžeme vybrať, na ktorý neterminál vo vetnej forme aplikujeme príslušné prepisovacie pravidlo v danom kroku odvodenia - bez rizika, že tým ovplyvníme výsledok.

2 špeciálne odvodenia

Príklad: Všimnite si tieto 2 odvodenia:

1. $E \Rightarrow E + T \Rightarrow T + T \Rightarrow T * F + T \Rightarrow F * F + T \Rightarrow \mathrm{id} * F + T \Rightarrow \mathrm{id} * \mathrm{id} + T \Rightarrow \mathrm{id} * \mathrm{id} + F \Rightarrow \mathrm{id} * \mathrm{id} + \mathrm{id}$
2. $E \Rightarrow E + T \Rightarrow E + F \Rightarrow E + \mathrm{id} \Rightarrow T + \mathrm{id} \Rightarrow T * F + \mathrm{id} \Rightarrow T * \mathrm{id} + \mathrm{id} \Rightarrow F * \mathrm{id} + \mathrm{id} \Rightarrow \mathrm{id} * \mathrm{id} + \mathrm{id}$

V odvodení č. 1 sme nahradili vždy prvý neterminál zľava, v odvodení č. 2 sme nahradili vždy prvý neterminál sprava. Oba derivačné stromy by však boli rovnaké (a také isté, ako derivačný strom na slajde č. 5).

Ľavé (pravé) odvodenie

Definícia

Nech G je bezkontextová gramatika. Potom odvodenie $\alpha_0 \Rightarrow \alpha_1 \Rightarrow ... \Rightarrow \alpha_k$ *v G, kde každý krok odvodenia* $\alpha_i \Rightarrow \alpha_{i+1}$ *pre* $0 \leq i \leq k$ *realizujeme tak, že v reťazci* α_i *nahradíme prvý neterminál zľava, nazývame* ***ľavé odvodenie (ľavá derivácia)****; ak nahrádzame prvý neterminál sprava, nazývame* ***pravé odvodenie (pravá derivácia)****. Príslušné vznikajúce vetné formy nazývame* ***ľavé (pravé) vetné formy****. Krok odvodenia v ľavom (pravom) odvodení budeme označovať symbolom* $\Rightarrow_l$ *(*$\Rightarrow_p$*).*

Ak G je jednoznačná gramatika, potom ľavé (pravé) odvodenie každého slova je určené jednoznačne. Túto vlastnosť využívajú **algoritmy hľadajúce deriváciu slova v gramatike - tzv. syntaktické analyzátory.**

Transformácie gramatík

- Gramatiky možno **transformovať** na **ekvivalentné** gramatiky (t.j. generujúce ten istý jazyk), ktoré budú navyše **spĺňať** dodatočné **požiadavky** (napr. na tvar pravidiel).
- Základom pri transformácii gramatík je, aby sa po transformácii **nezmenil jazyk generovaný gramatikou**.

3 základné úpravy

Existujú 3 základné úpravy bezkontextových gramatík:

1. Odstránenie nadbytočných symbolov - toto si vysvetlíme.
2. Odstránenie ε-pravidiel - možno ku koncu semestra, ak vyjde čas
3. Odstránenie jednoduchých pravidiel - možno ku koncu semestra, ak vyjde čas

Odstránenie nadbytočných symbolov

Môže sa stať, že pri definícii gramatiky sa použijú symboly (či už N alebo T), ktoré sa alebo nikdy počas derivácie nepoužijú, alebo ak sa použijú, nikdy sa nepodarí odvodiť reťazec terminálov. Ich prítomnosť v gramatike potom zbytočne zväčšuje jej veľkosť a navyše môžu viesť k zlým výsledkom rôznych algoritmov, pracujúcich s gramatikami.

Definícia

Nech $G = (N, T, P, S)$ je bezkontextová gramatika. Symbol $X \in N \cup T$ sa nazýva:

- ***nedostupný**, ak v G neexistuje odvodenie $S \Rightarrow^* \alpha X \beta$ pre nejaké $\alpha, \beta \in (N \cup T)^*$,*
- ***nadbytočný**, ak v G neexistuje odvodenie $S \Rightarrow^* xXz \Rightarrow^* xyz$ pre nejaké $x, y, z \in T^*$.*

Odstránenie nadbytočných symbolov

Definícia

*Gramatika bez nadbytočných symbolov sa nazýva **redukovaná** gramatika.*

Každý nedostupný symbol je zároveň aj nadbytočný. Vytvorenie redukovanej gramatiky pozostáva z 2 hlavných fáz:

1. Odstránenie neterminálov, z ktorých nie je možné odvodiť reťazec terminálov alebo ε.
2. Odstránenie neterminálov a terminálov, ktoré sa nemôžu vyskytnúť vo vetných formách gramatiky.

Symboly sa odstránia spolu s príslušnými pravidlami, v ktorých vystupujú.

Príklad

Nech gramatika G má pravidlá:

$S \to AB \mid A$

$A \to AA \mid a$

$B \to bB$

$C \to c$

Neterminál B je nadbytočný symbol. Neterminál C a terminál c sú nedostupné symboly. Po ich odstránení dostávame ekvivalentnú redukovanú gramatiku:

$S \to A$

$A \to AA \mid a$

Odstránenie nadbytočných symbolov - množina N_T

V prvej fáze hľadáme neterminály, z ktorých je možné odvodiť reťazec terminálov (prípadne ε). Hľadáme množinu:

$$N_T = \{A \mid A \Rightarrow^* w, w \in T^*\} \tag{1}$$

Odstránenie nadbytočných symbolov - množina N_T

Vstup: bezkontextová gramatika $G = (N, T, P, S)$
Výstup: množina N_T podľa (1)

1: $N_T \leftarrow \emptyset$
2: **opakuj**
3: $\acute{N_T} \leftarrow N_T$
4: $N_T \leftarrow \acute{N_T} \cup \{A \mid A \rightarrow \alpha \in P \wedge \alpha \in (\acute{N_T} \cup T)^*\}$
5: **pokiaľ** $N_T \neq \acute{N_T}$

Odstránenie nadbytočných symbolov - množina N_T

Príklad: Nech $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, kde P obsahuje pravidlá:

$S \to ABc \mid abA$

$A \to aAa \mid bBb \mid b$

$B \to aBb \mid caBC$

$C \to \varepsilon \mid ab \mid BcCa$

Podľa algoritmu by sa premenná N_T postupne menila: $\emptyset, \{A, C\}, \{S, A, C\}$. Neterminál B je teda taký, že sa z neho nedá odvodiť terminálny reťazec. Jeho výskyt v gramatike je teda **nadbytočný**, pretože jeho odstránením sa **nijako nezmení jazyk** $L(G)$, ktorý gramatika generuje.

Odstránenie nadbytočných symbolov - množina V_D

V druhej fáze hľadáme neterminály a terminály, ktoré sa môžu vyskytnúť v nejakej vetnej forme danej gramatiky, t.j. sú to **dostupné symboly gramatiky**. Hľadáme množinu:

$$V_D = \{X \mid S \Rightarrow^* \alpha X \beta, \alpha, \beta \in (N \cup T)^*\}. \quad (2)$$

Odstránenie nadbytočných symbolov - množina V_D

Vstup: bezkontextová gramatika $G = (N, T, P, S)$
Výstup: množina V_D podľa (2)

1: $V_D \leftarrow \{S\}$
2: **opakuj**
3: $\acute{V}_D \leftarrow V_D$
4: $V_D \leftarrow \acute{V}_D \cup \{X \mid A \rightarrow \alpha X \beta \in P \wedge A \in \acute{V}_D\}$
5: **pokiaľ** $V_D \neq \acute{V}_D$

Odstránenie nadbytočných symbolov - množina V_D

Príklad: Nech $G = (\{S, A, B, C, D\}, \{a, b, c, d\}, P, S)$, kde P obsahuje pravidlá:

$S \to AB \mid abS \mid \varepsilon$

$A \to aAa \mid b$

$B \to aCb \mid cC$

$C \to cS$

$D \to abC \mid \varepsilon \mid dD$

Podľa algoritmu by sa premenná V_D postupne menila:
$\{S\}, \{S, A, B, a, b\}, \{S, A, B, C, a, b, c\}$. Nedostupné symboly sú teda D, d.

Odstránenie nadbytočných symbolov - výsledný algoritmus

Vstup: bezkontextová gramatika $G = (N, T, P, S)$
Výstup: gramatika bez nadbytočných symbolov

1: Vytvor množinu N_T.
2: **ak** $S \in N_T$ **potom**
3: odstráň z gramatiky G všetky neterminály, ktoré nepatria do N_T;
4: vytvor množinu V_D
5: odstráň z gramatiky G všetky symboly, ktoré nepatria do V_D;
6: **koniec ak**

Poradie krokov je **dôležité**!. Najprv sa musia odstrániť neterminály, ktoré nevedú na terminálne reťazce (N_T) a až potom sa odstránia nedostupné symboly (V_D).

Odstránenie nadbytočných symbolov

Príklad: Pokračujme v gramatike, pre ktorú sme hľadali množinu N_T na slajde č. 24. Po odstránení neterminálu B (pretože nebol v množine N_T) dostávame gramatiku $G_1 = (\{S, A, C\}, \{a, b, c\}, P_1, S)$, kde P_1 obsahuje pravidlá:

$S \to abA$

$A \to aAa \mid b$

$C \to ab \mid \varepsilon$

Pre túto gramatiku je množina $V_D = \{S, A, a, b\}$, t.j. symboly C, c sú nedostupné. Po ich odstránení dostávame redukovanú gramatiku:

$$G_{red} = (\{S, A\}, \{a, b\}, \{S \to abA, A \to aAa \mid b\}, S).$$

T.j. $L(G) = L(G_1) = L(G_{red}) = \{aba^n ba^n \mid n \in \mathbb{Z}_0^+\}$.

Ak by sme gramatiku zo slajdu č. 24 upravovali **v opačnom poradí**, potom:

1. Množina $V_D = \{S, A, B, C, a, b, c\}$, t.j. všetky symboly sú dostupné a nič sa neodstráni.
2. Množina $N_T = \{A, C, S\}$, t.j. odstránime len neterminál B a výsledná gramatika:

   $S \to abA$

   $A \to aAa \mid b$

   $C \to \varepsilon \mid ab$

pričom vidíme, že neterminál C nie je dostupný a terminál c ani len nevystupuje v niektorom z pravidiel.

Použitá literatúra

Dedera, Ľ.: Počítačové jazyky a ich spracovanie.

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

Molnár, Ľ.: Gramatiky a jazyky.