Prednáška 1 - Formálne jazyky a gramatiky

Ing. Viliam Hromada, PhD.

C-510
Ústav informatiky a matematiky
FEI STU

viliam.hromada@stuba.sk

Organizácia semestra

- Prednášajúci: Ing. Viliam Hromada, PhD., C 510, `viliam.hromada@stuba.sk`
- Cvičiaci: Ing. Viliam Hromada, PhD., C 510

- `https://uim.fei.stuba.sk/predmet/i-afj/`
- Cez semester: 2 testy, každý za 20 bodov.
- Pre udelenie zápočtu je nutné získať **minimálne** 20 bodov.
- Na záver 60 bodová skúška.

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

Motivácia

- Rozlišujeme 2 základné typy jazykov (v zmysle "prostriedkov na dorozumievanie") - prirodzené a umelé (formálne).
- Prirodzený jazyk je jazyk, ktorý prechádza prirodzeným (historickým) vývojom, štandardne ide o jazyky, ktorými sa dorozumievajú ľudia medzi sebou (slovenský jazyk, anglický jazyk, ...)
- Umelý (formálny) jazyk je jazyk so špecifickou aplikáciou, ktorého pravidlá sú vopred jasne stanovené, štandardne sem patria napr. programovacie jazyky.
- Aj prirodzené, aj formálne jazyky majú svoju **syntax**, t.j. pravidlá vytvárania "gramaticky správnych" *viet*.

Motivácia

- Umelé (formálne) jazyky sú spravidla navrhnuté za určitým cieľom - napríklad programovacie jazyky uľahčujú "prikazovanie" činnosti počítaču.
- Aby však počítač "rozumel" nejakej vete, napísanej vo formálnom jazyku (napríklad programu v jazyku C), musia byť stanovené (a vo vete dodržané) pravidlá pre gramaticky korektné vety v danom jazyku.
- T.j. musím definovať nejakú vetnú konštrukciu, ktorú budeme používať na komunikáciu.
- Na základe tejto konštrukcie potom dochádza k spracovaniu vety ako z hľadiska jej štruktúry (syntaxe), tak aj významu (sémantiky).
- Aplikácií toho, o čom je tento predmet, je ooooooooooooooveľa viac...

Motivácia

- Cieľom tohto predmetu je poskytnúť Vám základné znalosti z oblasti spracovania formálnych (umelých, počítačových) jazykov.
- Znalosti sú využiteľné pri spracovaní ľubovoľných štruktúrovaných dát, keďže zdrojový kód nejakého programu je len špeciálny prípad štruktúrovaných dát - textového reťazca spĺňajúceho syntax programovacieho jazyka.
- Zaveďme si teraz niekoľko základných pojmov, ktoré vychádzajú z analógie s prirodzenými jazykmi - aby sme si vytvorili s počítačom "spoločný jazyk", najprv sa musíme dohodnúť, z akých symbolov sa budú vôbec skladať slová, vety, ktorými budeme komunikovať.

Úvodné pojmy

Definícia

Abeceda *je konečná neprázdna množina prvkov, ktoré sa nazývajú* ***symboly****.*

Príklady abecedy:

- {a, b, c, ..., z}, t.j. malé písmená anglickej abecedy,
- {a, b, c, .., z, A, B, C, ..., Z}, t.j. malé a veľké písmená anglickej abecedy,
- {0, 1, ..., 9}, t.j. číslice,
- { **begin**, **end**, =, +, (,), ;, **id**, **konšt** } - tzv. lexikálne elementy programovacieho jazyka. V tomto prípade je **begin** považované za 1 symbol, nie za postupnosť 5 symbolov!

Definícia

Nech A je abeceda. Potom ľubovoľnú konečnú postupnosť symbolov z abecedy A nazývame ***reťazcom*** *(slovom, vetou) nad abecedou A. Počet symbolov v reťazci x nazývame* ***dĺžkou reťazca*** *a označujeme* $|x|$.

Napríklad:

- $A = \{a, b, c, ..., z\}$. Ak $x = mama$, potom $|x| = 4$.
- $A = \{a, b, c, ..., z\}$. Ak $x = Mama$, potom x nie je reťazec nad abecedou A, pretože obsahuje symbol M, ktorý nie je z abecedy A.
- $A = \{0, 1, ..., 9\}$. Ak $x = 11$, potom $|x| = 2$.
- $A =$ { **begin**, **end**, =, +, (,), ;, **id**, **konšt** }. Ak $x =$ **beginid;end** potom $|x| = 4$

Ďalšie pojmy:

- Reťazec s nulovou dĺžkou sa nazýva **prázdny reťazec**, označuje sa ε, $|\varepsilon| = 0$.
- Nech $x = a_1...a_n$ je reťazec nad abecedou A. Potom **obrátený reťazec** k reťazcu x je reťazec $x^R = a_n...a_1$.
- Nech $u = a_1...a_l$ a $v = b_1...b_k$ sú reťazce nad abecedou A. Potom **zreťazenie** reťazcov u a v je reťazec $u \cdot v$ (skrátene uv), $u \cdot v = a_1...a_l b_1...b_k$.

Príklady

- Nech $A = \{a, b, c, d, e\}$. Ak $x = abeceda$, potom $x^R = adeceba$. $|x| = |x^R| = 7$.
- Nech $A = \{a, b, c, ..., z\}$. Ak $x = kobylamamalybok$, potom $x^R = kobylamamalybok$.
- Nech $A = \{a, b\}$. Ak $u = aa$, $v = bb$, potom $uv = aabb$, $uu = aaaa$, $vu = bbaa$, $vv = bbbb$. $|u| = 2, |uu| = 2|u| = 4$.
- Nech $A = \{a, b\}$. Ak $u = aa$, $v = \varepsilon$, potom $uv = aa\varepsilon = aa$, $|u| = 2, |v| = 0, |uv| = 2$.

Ďalšie definície:

- Nech x je reťazec a $i \geq 0$ je prirodzené číslo. Potom reťazec $x^i = \underbrace{xx...x}_{i}$ je reťazec, ktorý predstavuje i-násobné zreťazenie reťazca x za sebou, $x^0 = \varepsilon$.
- Ak x, y, z sú 3 reťazce, potom x je predpona, y podreťazec a z prípona reťazca xyz.
- Symbol A^* označuje **množinu všetkých reťazcov** nad abecedou A a symbol A^+ označuje **množinu všetkých neprázdnych reťazcov** nad abecedou A.

Príklady:

- $A = \{a, b, c\}$. $x = ab$
  - $x^0 = \varepsilon, |x^0| = 0$.
  - $x^2 = abab, |x^2| = 2|x| = 4$.
  - $x^3 = ababab, |x^3| = 3|x| = 6$.
- $A = \{a, b, c\}$. Reťazec abc má:
  - Predpony: $\{\varepsilon, a, ab, abc\}$
  - Prípony: $\{\varepsilon, c, bc, abc\}$
  - Podreťazce: $\{\varepsilon, a, b, c, ab, bc, abc\}$
- $A = \{a, b, c\}$. $A^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, ...\}$
- $A = \{a, b, c\}$. $A^+ = \{a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, ...\}$

Definícia

Nech A je abeceda. Potom ľubovoľnú podmnožinu L množiny A^* nazývame (formálnym) ***jazykom*** *nad abecedou A. Ak $L = \emptyset$, potom L nazývame* ***prázdnym jazykom****; ak L obsahuje konečný počet reťazcov, potom L nazývame* ***konečným jazykom****.*

Nech L, L_1, L_2 sú jazyky. Potom:

- **zreťazením** L_1 a L_2 nazývame jazyk $L_1 L_2 = \{xy | x \in L_1 \wedge y \in L_2\}$,
- **doplnkom** L nazývame jazyk $L^C = A^* - L$,
- **n-tou mocninou** jazyka L nazývame jazyk L^n definovaný takto:
  - $L^0 = \{\varepsilon\}$.
  - $L^{k+1} = LL^k$.

Nech L je jazyk. Potom:

- **iteráciou** jazyka L nazývame jazyk $L^* = \bigcup_{n=0}^{\infty} L^n$,
- **pozitívnou iteráciou** jazyka L nazývame jazyk $L^+ = \bigcup_{n=1}^{\infty} L^n$.

Zrejme platí, že: $L^* = L^+ \cup \{\varepsilon\}$.

Príklad: Nech $A = \{a, b, c\}$ je abeceda; nech $L_1 = \{b, c\}$, $L_2 = \{aa, ba, ca\}$ sú (konečné) jazyky nad abecedou A. Potom:

- $L_1 L_2 = \{baa, bba, bca, caa, cba, cca\}$,
- $L_2 L_1 = \{aab, aac, bab, bac, cab, cac\}$,
- $L_1^0 = \{\varepsilon\}$
- $L_1^1 = L_1$
- $L_1^2 = L_1 L_1 = \{bb, bc, cb, cc\}$
- $L_1^3 = L_1 L_1^2 = \{bbb, bbc, bcb, bcc, cbb, cbc, ccb, ccc\}$
- $L_1^* = \{\varepsilon, b, c, bb, bc, cb, cc, bbb, ...\}$,
- $L_1^+ = \{b, c, bb, bc, cb, cc, bbb, ...\}$,
- $L_1^C = \{\varepsilon, a, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, ...\}$
- $L_2^C = \{\varepsilon, a, b, c, ab, ac, bb, bc, cb, cc, aaa, ...\}$.

Abeceda, reťazce, jazyky

Všimnite si, akým spôsobom sme v definíciách postupovali:

1. Definovali sme množinu symbolov (abecedu)
2. Definovali sme postupnosť symbolov (reťazec)
3. Definovali sme množinu reťazcov (jazyk)

Analogicky v prirodzenom jazyku (napr. slovenčine)

1. Slovenská abeceda má 46 písmen (bez rozlišovania malých a veľkých)
2. Slová (vety) v slovenskom jazyku graficky pozostávajú z týchto písmen (a prípadných znamienok, čiarok, atď.)
3. "Slovenský jazyk" pozostáva zo zmysluplných slov, prípadne viet.

Programovacie jazyky

Uvažujme predchádzajúce definície v kontexte programovacích jazykov:

- Abeceda je množina všetkých elementov, z ktorých sa môže skladať zdrojový kód (kľúčové slová, operátory, separátory, identifikátory, konštanty,...).
- Z týchto symbolov vytvárame reťazce - teda potenciálne zdrojové kódy.
- V prípade programovacích jazykov je *jazykom* množina **všetkých** takých zdrojových kódov, ktorým počítač "rozumie", t.j. dokáže ich spracovať (napr. prekladom do iného jazyka alebo interpretovaním).
- Základným kritériom toho, či je reťazec "zrozumiteľný" z hľadiska použitého programovacieho jazyka je, či spĺňa sadu pravidiel vytvárania programov, tzv. **syntax**.

- Preto teda, ak sa bavíme o programovacom jazyku ako o prostriedku "dorozumievania" s počítačom, resp. prikazovaniu počítaču, čo má robiť, môžeme hovoriť o "jazyku" ako o množine platných reťazcov, ktorým počítač "rozumie", pretože spĺňajú nejakú preddefinovanú štruktúru (syntax).
- Počítač totižto na základe tejto štruktúry (syntaxe) vykonáva spracovanie programu (kompiláciu/interpretáciu).
- V prípade programovacích jazykov je teda jazyk **množina všetkých syntakticky platných programov**. Ako však počítač zistí, či danému programu (reťazcu symbolov) rozumie, teda či daný program patrí do jazyka?
- Jedna možnosť by bola mať niekde zoznam **všetkých platných zdrojových kódov** spĺňajúcich syntax a následne ho len porovnať so zadaným...

Niečo kompaktnejšie

- Popis jazyka ako množiny všetkých reťazcov nie je vždy vhodný.
- Čo ak je jazyk nekonečný?
- Lepšie je popísať, **ako** vyzerajú slová (vety) z jazyka.
- Napr. v prípade zdrojových kódov je lepší popis cez prípustné "tvary" zdrojákov, než cez vymenovanie všetkých prípustných možností...

Motivačný príklad

Predstavme si veľmi jednoduchý "programovací" jazyk s nasledujúcou syntaxou:

- Každý program tvorí postupnosť príkazov uzatvorená medzi kľúčové slová **begin** a **end**.
- V postupnosti príkazov je každý príkaz ukončený bodkočiarkou ";".
- Každý príkaz pozostáva z jediného kľúčového slova - buď **p1** alebo **p2**.

Abecedu (t.j. množinu prípustných symbolov), z ktorej budeme tvoriť "vety" (t.j. programy) bude tvoriť množina lexikálnych elementov - kľúčové slová **begin**, **end**, **p1**, **p2** a separátor ;. Tieto symboly, z ktorých pozostáva výsledný program, nazveme **terminálne symboly**.

Motivačný príklad

Uvedenú syntax by sme mohli schematicky zapísať nasledovne:

1. <program> → **begin** <postupnosť príkazov> **end**
2. <postupnosť príkazov> → <príkaz> ; <postupnosť príkazov>
3. <postupnosť príkazov> → ε
4. <príkaz> → **p1**
5. <príkaz> → **p2**

Máme vlastne definovaných 5 rôznych pravidiel pre tvorbu korektných programov.

Motivačný príklad

Vezmime uvedené pravidlá a skúsme pomocou nich zostrojiť niečo, čo ich spĺňa...

- Vieme, že chceme vytvoriť nejaký <program>. Ten je ďalej definovaný pomocou pravidla č. 1: <program> ⇒ **begin**<postupnosť príkazov>**end**
- Keďže <postupnosť príkazov> je ďalej definovaná napríklad pomocou pravidla č. 2, tak by sme mohli urobiť:
  <program> ⇒ **begin**<postupnosť príkazov>**end** ⇒
  begin<príkaz>;<postupnosť príkazov>**end**
- <príkaz> je bližšie definovaný pomocou pravidla 4, resp. 5 ako **p1**, resp. **p2**. Uvažujme, že ho definujeme ako **p1**:
  <program> ⇒ **begin**<postupnosť príkazov>**end** ⇒
  begin<príkaz>;<postupnosť príkazov>**end** ⇒
  begin p1;<postupnosť príkazov>**end**

(pokračovanie)

- A predpokladajme, že zvyšná postupnosť príkazov je prázdna (t.j. pravidlo č. 3): <program> ⇒ **begin**<postupnosť príkazov>**end** ⇒
  begin<príkaz>;<postupnosť príkazov>**end** ⇒
  begin p1;<postupnosť príkazov>**end** ⇒ **begin p1**; **end**
- Pomocou nami definovaných pravidiel sme teda zostrojili reťazec z terminálnych symbolov: **begin p1 ; end**
- Tento reťazec určite spĺňa uvedených 5 pravidiel - pretože bol podľa nich vytvorený. Teda by spĺňal syntax nášho programovacieho jazyka.

Podobným spôsobom vieme definovať aj pravidlá pre vytváranie iných jazykov.

Výrazy medzi znakmi "<" a ">" **neboli** súčasťou finálneho programu, avšak pomáhali nám lepšie definovať štruktúru (syntax). Nazývame ich **neterminálne** symboly.

Neterminálny symbol <program> má špeciálny význam - definuje celkový výsledok (v našom prípade program), resp. sa z neho odvádzal výsledný program - tzv. **začiatočný symbol**.

Všimnite si, **ako sme používali pravidlá!** Vždy sme reťazec z **ľavej strany** nejakého pravidla nahradili **pravou stranou** toho istého pravidla.

Samotná gramatika určuje len **syntax** jazyka, t.j. štruktúru prípustných reťazcov (viet). Ich význam určuje **sémantika**.

Gramatiky

Definícia

Gramatika je usporiadaná štvorica $G = (N, T, P, S)$, kde:

- *N je konečná neprázdna množina neterminálnych symbolov (alebo neterminálov),*
- *T je konečná množina terminálnych symbolov (alebo terminálov), pričom $N \cap T = \emptyset$,*
- *$S \in N$ je začiatočný (štartovací) symbol gramatiky a*
- *P je množina (prepisovacích) pravidiel, ktorá je konečnou podmnožinou množiny $(N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$.*

Prázdny reťazec ε nepatrí ani medzi terminálne, ani medzi neterminálne symboly gramatiky.

Príklad gramatiky: Usporiadaná štvorica $G = (N, T, P, S)$, kde:

- Množina neterminálov $N = \{$<program>, <postupnosť príkazov>, <príkaz>$\}$
- Množina terminálov $T = \{$**begin**, **end**, **p1**, **p2**, ; $\}$
- Počiatočný neterminál $S =$ <program>
- Pravidlá P:
  1. <program> $\rightarrow$ **begin** <postupnosť príkazov> **end**
  2. <postupnosť príkazov> $\rightarrow$ <príkaz> ; <postupnosť príkazov>
  3. <postupnosť príkazov> $\rightarrow \varepsilon$
  4. <príkaz> $\rightarrow$ **p1**
  5. <príkaz> $\rightarrow$ **p2**

Pri zápise pravidiel, ktoré majú rovnakú ľavú stranu (reťazec symbolov gramatiky pred šípkou) môžeme používať skrátený zápis, kde zlúčime pravé strany a oddelíme ich zvislou čiarou. V predchádzajúcom príklade by sme tak zlúčili pravidlá č. 2 a 3 a pravidlá č. 4 a 5:

- <program> $\rightarrow$ **begin** <postupnosť príkazov> **end**.
- <postupnosť príkazov> $\rightarrow$ <príkaz> ; <postupnosť príkazov> | ε
- <príkaz> $\rightarrow$ **p1** | **p2**

Derivácie reťazcov

- Okrem toho, že pravidlá gramatík nám slúžia ako zápis štruktúry toho, čo gramatika popisuje, nám môžu slúžiť aj na vytváranie reťazcov terminálnych symbolov, ktoré príslušnú štruktúru spĺňajú.
- Proces, v ktorom pomocou pravidiel gramatiky vytvárame reťazce terminálov, nazývame **derivácia**.

Definícia

Nech $G = (N, T, P, S)$ je gramatika. Reláciou ***krok odvodenia*** *(alebo* ***relácia derivácie****) $\Rightarrow$ nazývame reláciu definovanú na množine $(N \cup T)^*$ (t.j. terminálov a neterminálov) nasledovným spôsobom:*

$$\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$$

ak v P existuje pravidlo $\beta \to \delta$, pričom $\alpha, \gamma, \delta \in (N \cup T)^*$ a $\beta \in (N \cup T)^* N (N \cup T)^*$. (Čítaj: z reťazca $\alpha\beta\gamma$ je možné priamo odvodiť/derivovať reťazec $\alpha\delta\gamma$).

Inverzná relácia ku kroku odvodenia (derivácie) sa nazýva **redukcia**. Relácia **krok odvodenia** formalizuje aplikáciu prepisovacieho pravidla.

Definícia

Nech $G = (N, T, P, S)$ je gramatika a $\alpha_i \in (N \cup T)^*$ pre $i = 0, 1, ...k$. Potom hovoríme, že reťazec α_k sa dá odvodiť z reťazca α_0 (na k krokov), ak platí, že $\alpha_i \Rightarrow \alpha_{i+1}$ pre $i = 0, 1, ..., k - 1$.

*Postupnosť reťazcov $\alpha_0 \Rightarrow \alpha_1 \Rightarrow ... \Rightarrow \alpha_k$ sa nazýva **odvodenie/derivácia** reťazca α_k z reťazca α_0. Zapisujeme $\alpha_0 \Rightarrow^k \alpha_k$. Číslo k nazývame **dĺžka odvodenia**.*

Triviálne platí, že každý reťazec $\alpha \in (N \cup T)^*$ sa dá odvodiť z α na 0 krokov.

Definícia

Nech $G = (N, T, P, S)$ je gramatika. Potom:

- ***tranzitívnym uzáverom relácie krok odvodenia*** *nazývame reláciu $\Rightarrow^+$ definovanú na množine $(N \cup T)^*$ nasledovne:*
  $\alpha \Rightarrow^+ \beta$ práve vtedy, keď existuje $n \geq 1$ také, že $\alpha \Rightarrow^n \beta$ a
- ***reflexívnym a tranzitívnym uzáverom relácie krok odvodenia*** *reláciu $\Rightarrow^*$ definovanú:*
  $\alpha \Rightarrow^* \beta$ práve vtedy, keď existuje $n \geq 0$ také, že $\alpha \Rightarrow^n \beta$,

pričom $\alpha, \beta \in (N \cup T)^*$.

T.j. v prípade tranzitívneho uzáveru sú α, β v relácii, ak sa dá odvodiť β z α na min. 1 krok. V prípade reflexívneho a tranzitívneho uzáveru pripúšťame aj odvodenie na 0 krokov.

Príklad: Uvažujme gramatiku $G = (N, T, P, S)$, $N = \{S, \text{<PP>}, \text{<P>}\}$, $T = \{\mathbf{b}, \mathbf{e}, \mathbf{p1}, \mathbf{p2}, ;\}$:

- S $\rightarrow$ **b**<PP>**e**
- <PP> $\rightarrow$ <P>; <PP> $\mid \varepsilon$
- <P> $\rightarrow$ **p1** $\mid$ **p2**

Potom:

1. <P> $\Rightarrow$ **p1**
2. <P> $\not\Rightarrow$ **p1 p1 p1**
3. <P>; <PP> $\Rightarrow$ **p1**; <PP>
4. <PP> $\Rightarrow$ <P>; <PP> $\Rightarrow$ **p1**; <PP> $\Rightarrow$ **p1**; ε
5. <PP> $\Rightarrow^3$ **p1**;
6. S $\Rightarrow$ **b**<PP>**e** $\Rightarrow$ **b**<P>; <PP>**e** $\Rightarrow^*$ **b p1**; **e**
7. $S \Rightarrow^*$ **b p1 ; e**
8. $S \Rightarrow^+$ **b p1 ; e**
9. $S \not\Rightarrow^*$ **b p1 e** (*t.j. bez bodkočiarky za* ***p1***)

Definícia

Nech $G = (N, T, P, S)$ je gramatika. Potom každý reťazec $\alpha \in (N \cup T)^*$, ktorý možno odvodiť zo začiatočného neterminálu S gramatiky G, sa nazýva ***vetná forma****;*

Formálne, α je ***vetná forma*** *práve vtedy, keď $S \Rightarrow^* \alpha$.*

Jazyk generovaný gramatikou

Definícia

*Nech $G = (N, T, P, S)$ je gramatika. Potom **jazykom generovaným gramatikou G** nazývame množinu:*

$$L(G) = \{w | S \Rightarrow^* w, w \in T^*\}.$$

*Reťazec patriaci do daného jazyka sa nazýva **slovo** alebo **veta jazyka**.*

Príklad gramatiky: V prípade našej gramatiky:

1. <program> $\rightarrow$ **begin** <postupnosť príkazov> **end**
2. <postupnosť príkazov> $\rightarrow$ <príkaz> ; <postupnosť príkazov>
3. <postupnosť príkazov> $\rightarrow \varepsilon$
4. <príkaz> $\rightarrow$ **p1**
5. <príkaz> $\rightarrow$ **p2**

je teda jazyk generovaný gramatikou $L(G)$ množina takých "programov", ktoré začínajú kľúčovým slovom **begin**, končia kľúčovým slovom **end** a medzi nimi sa alebo nenachádza nič, alebo sa tam nachádzajú príkazy **p1** alebo **p2**, vždy ukončené bodkočiarkou.

Jazyk generovaný gramatikou

Uvedomte si 2 veci z hľadiska vzťahu gramatiky G a jazyka $L(G)$, ktorý generuje:

- **Každé slovo, ktoré gramatika generuje, patrí do $L(G)$!** (z definície)
- **Každé slovo z množiny $L(G)$ má v gramatike odvodenie!** (t.j. existuje postupnosť aplikácie pravidiel, ktorá z počiatočného neterminálu odvodí dané slovo)

Príklad: Uvažujme nasledovnú gramatiku: $G = (\{S, A\}, \{0, 1\}, P, S)$, kde pravidlá P:

$S \to 0A$

$A \to 0A \mid 1A \mid 0$

Skúsme zistiť, či nasledovné reťazce patria do jazyka generovaného gramatikou G, t.j. či patria do $L(G)$:
$\varepsilon, 0, 1, 00, 01, 10, 11$

Na základe pravidiel teda už tušíme, že do $L(G)$ patria **len tie** reťazce, ktoré **začínajú a končia nulou**, medzi ktorými môže byť **ľubovoľná postupnosť z 0 a 1**.

Jazyk -> Gramatika

- Častou úlohou je úloha, kde je zadaný jazyk L nad nejakou abecedou A, $L \subseteq A^*$ a je potrebné nájsť gramatiku G, ktorá daný jazyk generuje, t.j. $L(G) = L$.
- Tu si je potrebné uvedomiť 2 veci:
  - Každé slovo z jazyka L **musí mať** v gramatike G odvodenie, t.j. $L \subseteq L(G)$.
  - Každé slovo, ktoré gramatika G generuje **musí zároveň** patriť do jazyka L, t.j. $L(G) \subseteq L$.
  - Ak sú splnené podmienky $L \subseteq L(G)$ a $L(G) \subseteq L$, potom $L(G) = L$.

Príklad: Hľadajme gramatiku G ku zadanému jazyku L. Nech jazyk L tvoria také reťazce zo **symbolov** $\{a, b\}$, v ktorých je **párny počet písmen** a**.**

Riešenie (jedno z mnohých správnych!!!): $G = (\{S, A\}, \{a, b\}, P, S)$, kde pravidlá P:

$S \to Sb \mid \varepsilon \mid Aa$

$A \to Sa \mid Ab$

Pri overení správnosti riešenia si musíme vždy položiť tieto 2 otázky:

1. Má každý reťazec $x \in L$ v gramatike G deriváciu, teda $x \in L(G)$? Ak áno, potom $L \subseteq L(G)$.
2. Platí pre každý reťazec, ktorý má v gramatike G deriváciu, $y \in L(G)$, že patrí do zadaného jazyka L, teda $y \in L$? Ak áno, potom $L(G) \subseteq L$.
3. Ak platí $L \subseteq L(G) \wedge L(G) \subseteq L$, tak potom $L = L(G)$.

Ekvivalencia gramatík

Definícia

*Gramatiky G_1, G_2 sa nazývajú **ekvivalentné gramatiky**, ak pre jazyky, ktoré generujú, platí vzťah $L(G_1) = L(G_2)$.*

Klasifikácie gramatík (Chomsky)

Definícia

Nech $G = (N, T, P, S)$ je gramatika. Potom hovoríme, že gramatika G je:

- ***regulárna*** *(typu 3), ak každé prepisovacie pravidlo z P má jeden z tvarov $A \to uB$ alebo $A \to w$, kde $A, B \in N$, $u \in T^+$ a $w \in T^*$,*
- ***bezkontextová*** *(typu 2), ak každé prepisovacie pravidlo z P má tvar $A \to \alpha$, kde $A \in N$, $\alpha \in (N \cup T)^*$,*
- ***kontextová*** *(typu 1), ak každé prepisovacie pravidlo z P má tvar $\alpha \to \beta$, kde $\alpha \in (N \cup T)^* N (N \cup T)^*, \beta \in (N \cup T)^+$ a platí $|\alpha| \leq |\beta|$,*
- ***frázová*** *(typu 0), ak sa na tvar prepisovacích pravidiel nekladú žiadne obmedzenia.*

Keďže každá gramatika G generuje nejakú množinu $L(G)$, tak podľa typov gramatiky rozlišujeme aj typy jazykov.

Definícia

Jazyk nazývame (postupne) regulárny, bezkontextový, kontextový alebo rekurzívne vyčísliteľný, ak ho možno generovať regulárnou, bezkontextovou, kontextovou alebo frázovou gramatikou.

Chomského hierarchia formálnych jazykov

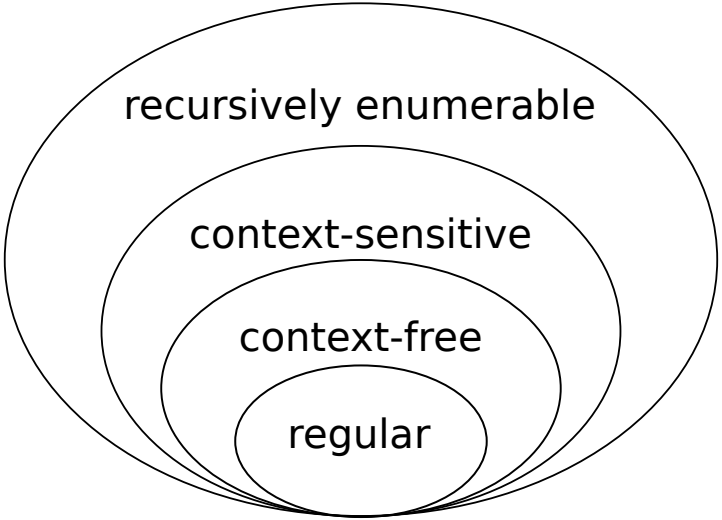

Figure: https://commons.wikimedia.org/wiki/File:Chomsky-hierarchy.svg

Zhrnutie

- Abeceda $\Rightarrow$ reťazce $\Rightarrow$ jazyk
- Gramatika, derivácia slova v gramatike.
- Špecifikácia jazyka gramatikou.
- Rôzne typy gramatík.