Prednáška 7 - Bezkontextové gramatiky a jazyky

Ing. Viliam Hromada, PhD.

C-510
Ústav informatiky a matematiky
FEI STU

viliam.hromada@stuba.sk

10.11.2020

Bezkontextové jazyky

- Doteraz sme sa bavili o regulárnych jazykoch, teda o jazykoch popísateľných KA alebo regulárnymi výrazmi
- V prenesenom slova zmysle šlo o rozhodovacie problémy, pre ktoré existovali výpočtové zariadenia (algoritmy) schopné riešiť ich pre rôzne inštancie.
- Na poslednej prednáške sme si povedali, že existujú rozhodovacie problémy, ktoré nie sú riešiteľné na konečných automatoch - tzv. neregulárne jazyky.
- Väčšiu triedu jazykov, než regulárne, tvoria tzv. **bezkontextové jazyky**.

Bezkontextové jazyky

- Problematika bezkontextových jazykov zohráva významnú úlohu pri prekladačoch, keďže umožňuje zostrojiť prekladače pre rôzne programovacie jazyky bez toho, že by tieto museli byť "šité na mieru".
- Taktiež zohrávajú dôležitú úlohu pri popise formátov dokumentov cez tzv. DTD (document-type definition) známe pri XML schémach.

Bezkontextové jazyky

- Bezkontextové jazyky sú (znovu) len typy jazykov (t.j. množín reťazcov nad nejakou abecedou).
- Akurát, že na rozdiel od regulárnych jazykov sú bezkontextové jazyky "všeobecnejšie", resp. patria sem aj také množiny jazyky, ktoré nepatrili do regulárnych jazykov, napr. 0^n1^n.
- Znovu sa teda budeme baviť o nejakých množinách reťazcov, ktoré vieme popísať nejakým formálnym spôsobom, napríklad zariadením, ktoré ich akceptuje - v tomto prípade to budú tzv. **zásobníkové automaty**.
- Bezkontextové jazyky sú taktiež jazyky, ktoré sú popísateľné pomocou **pravidiel ich vytvárania** - tzv, **bezkontextovými gramatikami**.

Bezkontextové gramatiky

- Uvažujme prirodzené jazyky, napr. slovenský jazyk.
- Každá (spisovná) zrozumiteľná veta v slovenskom jazyku by mala spĺňať pravidlá slovenského jazyka - tzv. gramatiku.
- Typicky napríklad holá veta v slovenskom jazyku pozostáva z prisudzovacieho skladu.
- Prisudzovací sklad je tvorený podmetom a prísudkom.
- Podmet je väčšinou podstatné meno.
- Prísudok je sloveso.
- Podstatné mená sú napríklad {mama, otec }; slovesá sú {varí, číta }.

T.j. "gramatické pravidlá" tvorby holej vety z predchádzajúceho slajdu by sme vedeli zapísať nasledovne:

1. <Holá veta> → <Prisudzovací sklad>
2. <Prisudzovací sklad> → <Podmet> <Prísudok>
3. <Podmet> → <Podstatné meno>
4. <Prísudok> → <Sloveso>
5. <Podstatné meno> → Mama
6. <Podstatné meno> → Otec
7. <Sloveso> → varí
8. <Sloveso> → číta

- Na uvedených 8 bodov by sme sa vedeli dívať ako na pravidlá, ktoré nám popisujú gramaticky správne holé vety.
- Pravidlá pozostávajú z 2 typov prvkov: prvky, ktoré predstavujú konkrétne slová v slovenských vetách (Mama, Otec, varí, číta) a prvky, ktoré len bližšie určujú štruktúru, ale vo výsledku sa nenachádzajú (sú označené symbolmi $<$, $>$, napr. $<$Holá veta$>$, $<$Podmet$>$, atď.)
- Samotné pravidlá nám špecifikujú, ako sú prvky, ktoré určujú štruktúru, ďalej definované pomocou iných prvkov.
- Jeden prvok má špecifické postavenie: $<$Holá veta$>$ je akoby "hlavný prvok", ktorý nám udáva, čo vlastne chceme popísať.

Napríklad "Otec varí", je gramaticky správna holá veta spĺňajúca uvedené pravidlá, pretože:

1. Holá veta je tvorená prisudzovacím skladom (pravidlo č. 1)
2. Prisudzovací sklad je tvorený podmetom a prísudkom (pravidlo č. 2)
3. Podmet je tvorený podstatným menom (pravidlo č. 3)
4. Prísudok je tvorený slovesom (pravidlo č. 4)
5. **Konkrétne** podstatné meno je napríklad Otec (pravidlo č. 6)
6. **Konkrétne** sloveso je napríklad varí (pravidlo č. 7)

Tak vidíme, ako táto veta vie vzniknúť v našej "gramatike" holých viet.

- T.j. na našu "gramatiku" sa vieme dívať aj ako na popis toho, ako vyzerajú gramaticky korektné holé vety, aj na spôsob, ako tieto holé vety generovať.
- Samozrejme, slovenský jazyk a jeho gramatika je komplikovanejšia, my uvádzame len jednoduchšiu verziu pre ilustráciu.
- Podobne, aj formálne jazyky sa často dajú popísať pomocou "pravidiel" ako reťazce z daného jazyka môžu vznikať, viď nasledujúci príklad.

Bezkontextové gramatiky

- Uvažujme palindrómy (reťazce, ktoré sa čítajú rovnako odpredu aj odzadu). Typické palindrómy v slovenčine sú napríklad "kobylamamalybok" (kobyla má malý bok)
- V angličtine je palindrómom napríklad postupnosť "madamimadam" (Madam, I'm Adam - vraj prvá vec, ktorú povedal Adam Eve v Raji)
- Pre jednoduchosť uvažujme palindrómy nad abecedou $\{0, 1\}$, t.j. napríklad $101, 0110, 0$.
- Dá sa ukázať, že ak uvažujeme **všetky takéto palindrómy**, t.j. jazyk

$$L = \{w \in \{0,1\}^* \mid w = w^R\}$$

tak tento jazyk **nie je regulárny**, t.j. neexistuje konečný automat, ktorý by ho vedel akceptovať.

- Avšak, existuje pomerne jednoduchý spôsob, ako popísať **pravidlá**, ako vyzerajú korektné palindrómy nad abecedou $\{0, 1\}$. Tento popis je **rekurzívny** a vyzerá nasledovne:
  1. (Základný krok:) $\varepsilon, 0, 1$ sú palindrómy
  2. (Rekurzia:) Ak w je palindróm, tak potom aj $0w0, 1w1$ sú palindrómy. Žiaden iný reťazec zložený z 0 a 1 nie je palindróm.
- Uvedený popis jazyka má rekurzívny charakter. Na formálne zachytenie rekurzívneho popisu sa v praxi používajú tzv. **bezkontextové gramatiky**.
- Napríklad nami uvedenú rekurzívnu definíciu palindrómu by sme vedeli formálne zapísať v nasledovnom tvare:

1. <Palindróm> $\rightarrow \varepsilon$
2. <Palindróm> $\rightarrow$ 0
3. <Palindróm> $\rightarrow$ 1
4. <Palindróm> $\rightarrow$ 0 <Palindróm> 0
5. <Palindróm> $\rightarrow$ 1 <Palindróm> 1

- Prvé 3 pravidlá sú vlastne formálnym zápisom základného kroku definície
- A 4., 5. pravidlo sú formálnym zápisom rekurzívneho kroku definície

1. <Palindróm> $\rightarrow \varepsilon$
2. <Palindróm> $\rightarrow$ 0
3. <Palindróm> $\rightarrow$ 1
4. <Palindróm> $\rightarrow$ 0 <Palindróm> 0
5. <Palindróm> $\rightarrow$ 1 <Palindróm> 1

- V popise máme znovu (podobne ako pri holých vetách) symboly, ktoré sa budú nachádzať v konkrétnych reťazcoch (0, 1, resp. ε)
- A symboly, ktoré nám len pomáhajú lepšie popísať štruktúru (<Palindróm>), ale vo výslednom reťazci sa nenachádzajú.
- Samotné popisovacie pravidlá majú ten tvar, že **vľavo** je "definovaný symbol" a **vpravo** je jeho "definícia", pričom vždy vľavo je 1 symbol a vpravo je ľubovoľná postupnosť symbolov.
- Takýto popis jazyka nazývame **bezkontextovou gramatikou**.

- Symboly gramatiky, ktoré predstavujú tie symboly, z ktorých sú tvorené samotné reťazce jazyka (v našom prípade 0, 1) nazývame **terminálne symboly**.
- Symboly gramatiky, ktoré nám bližšie pomáhajú popísať štruktúru reťazcov jazyka, ale **nevyskytujú sa** v samotných výsledných reťazcoch (v našom prípade symbol <Palindróm>) nazývame **neterminálne reťazce**.
- Popisovacie pravidlá bezkontextových gramatík majú **vždy tvar**, že obsahujú:
  - Jeden neterminálny symbol, ktorý je pravidlom bližšie definovaný. Nachádza sa vľavo od $\rightarrow$ a nazýva sa aj **hlava pravidla**
  - Symbol $\rightarrow$
  - Reťazec nula alebo viac terminálov a neterminálov, nazývaný **telo pravidla**, ktorý bližšie definuje hlavu pravidla.
- Jeden neterminál má **špecifické postavenie** - definuje celý **výsledný jazyk**. Takýto neterminál sa nazýva **počiatočný neterminál**. V našom prípade ním je <Palindóm>.

Bezkontextové gramatiky

Bezkontextové gramatiky sú teda spôsob špecifikácie jazykov. Každá taká gramatika G je štvorica $G = (V, T, P, S)$, ktorá obsahuje:

1. Konečnú množinu **terminálnych symbolov** T.
2. Konečnu množinu **neterminálnych symbolov (premenných)** V.
3. Konečnú množinu **prepisovacích pravidiel** P.
4. Počiatočný neterminál S, $S \in V$.

Jeden symbol nemôže byť zároveň neterminál, aj terminál, t.j. $V \cap T = \emptyset$. ε nie je ani terminál, ani neterminál.

Príklad

Gramatika $G = (V, T, P, S)$, kde $V = \{<\text{Palindróm}>\}$, $T = \{0, 1\}$, $S =<\text{Palindróm}>$ a prepisovacie pravidlá:

1. $<\text{Palindróm}> \rightarrow \varepsilon$
2. $<\text{Palindróm}> \rightarrow 0$
3. $<\text{Palindróm}> \rightarrow 1$
4. $<\text{Palindróm}> \rightarrow 0\ <\text{Palindróm}>\ 0$
5. $<\text{Palindróm}> \rightarrow 1\ <\text{Palindróm}>\ 1$

je bezkontextová gramatika popisujúca palindrómy nad abecedou $\{0, 1\}$.

Iný príklad

Skúsme popísať pomocou bezkontextovej gramatiky matematické výrazy obsahujúce:

- Operátory $+$, $*$ a ľavé a pravé okrúhle zátvorky, t.j. $(,)$
- Ako operandy budú slúžiť alebo znovu samotné výrazy, alebo konkrétne identifikátory (t.j. mená premenných), pričom tie budú predstavovať reťazce zložené zo symbolov $a, b, 0, 1$, kde na prvom mieste môže byť len a alebo b.

Z uvedeného popisu možno vyčítať pravidlá, ako by takéto matematické výrazy mali vyzerať. Najprv si uveďme pravidlá pre samotné výrazy:

1. $<\text{Výraz}> \rightarrow <\text{Identifikátor}>$
2. $<\text{Výraz}> \rightarrow <\text{Výraz}> + <\text{Výraz}>$
3. $<\text{Výraz}> \rightarrow <\text{Výraz}> * <\text{Výraz}>$
4. $<\text{Výraz}> \rightarrow (<\text{Výraz}>)$

1. Pravidlo č. 1 hovorí, že výraz môže predstavovať 1 identifikátor (1 premenná).
2. Pravidlo č. 2 rekurzívne hovorí, že sčítanie 2 matematických výrazov je zase matematický výraz.
3. Pravidlo č. 3 rekurzívne hovorí, že násobenie 2 matematických výrazov je zase matematický výraz.
4. Pravidlo č. 4 rekurzívne hovorí, že ozátvorkovaný matematický výraz je zase matematický výraz.

Z uvedeného je taktiež možné odvodiť, ako budú vyzerať identifikátory:

1. <Identifikátor> $\rightarrow a$
2. <Identifikátor> $\rightarrow b$
3. <Identifikátor> $\rightarrow$ <Identifikátor>a
4. <Identifikátor> $\rightarrow$ <Identifikátor>b
5. <Identifikátor> $\rightarrow$ <Identifikátor>0
6. <Identifikátor> $\rightarrow$ <Identifikátor>1

1. Pravidlo č. 1,2 uvádzajú, že *a* alebo *b* môžu byť identifikátor.
2. Pravidlá č. 3-6 uvádzajú rekurzívne pravidlá pre identifikátory, t.j. že ak *I* je identifikátor, tak aj *Ia*, *Ib*, *I*0, *I*1 je identifikátor (inými slovami, ak k existujúcemu identifikátoru pripíšeme $0, 1, a, b$, stále máme platný identifikátor)
3. Všimnite si, že pravidlá nedovoľujú, aby bol identifikátor len postupnosť 0 alebo 1.

Preznačme neterminál <Výraz> ako E a neterminál <Identifikátor> ako I a zlúčme pravidlá dokopy. Dostávame množinu pravidiel:

1. $E \to I$
2. $E \to E + E$
3. $E \to E * E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

T.j. uvažujeme gramatiku $G = (V, T, P, S)$, kde pravidlá P sú uvedené na predchádzajúcom slajde, $V = \{E, I\}$ sú neterminály, $I = \{a, b, 0, 1, +, *, (,)\}$ sú terminály. Počiatočným neterminálom bude E pretože **tento** neterminál popisuje práve matematické výrazy.

- V bezkontextových gramatikách platí, že každý neterminál popisuje nejaké reťazce.
- Počiatočný neterminál popisuje práve reťazce jazyka, ktorý chceme gramatikou popísať. V našom prípade E popisuje požadované matematické výrazy.
- Neterminál I popisuje taktiež reťazce, avšak "len" tie, ktoré predstavujú identifikátory (mená premenných). Teda nie všetky matematické výrazy, ale len ich súčasti.

V praxi môžeme zápis pravidiel zjednodušiť tak, že ak máme viacero pravidiel s rovnakou ľavou stranou, tak tieto pravidlá zapíšeme do zjednodušeného zápisu s 1 ľavou stranou a na pravú stranu uvedieme jednotlivé pravé strany oddelené zvislou čiarou, t.j.:

$$E \to I \mid E + E \mid E * E \mid (E)$$

$$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

Derivácie reťazcov v gramatikách

- Ako sme povedali, gramatiky teda slúžia na popis toho, **ako môžu vyzerať reťazce** patriace do nejakého jazyka.
- Ak máme teda daný nejaký reťazec, patriaci do jazyka, tak tento reťazec musí spĺňať pravidlá dané gramatikou, t.j. vieme ukázať, že ho vieme vyskladať z reťazcov, ktoré sú popisané **jednotlivými neterminálmi**. Tento prístup nazývame **rekurzívna inferencia**
- Pravidlá gramatiky sa však dajú použiť aj opačne - pomocou nich vieme **odvádzať (derivovať)** reťazce patriace do jazyka, ktorý je gramatikou popísaný. V tomto postupe začneme s počiatočným neterminálom a postupnou aplikáciou pravidiel gramatiky vieme odvodiť nejaký reťazec terminálov. Tento prístup nazývame **derivácia**.

Príklad rekurzívnej inferencie

- Chceme ukázať, že reťazec $a * (a + b00)$ je korektný matematický výraz podľa gramatiky zo slajdu 21.
- Prvým spôsobom je, že to dokážeme podľa rekurzívnej inferencie.
- Tento spôsob spočíva v tom, že postupne ukazujeme, **aké reťazce** patria do jazykov **neterminálov gramatiky**, až sa nám podarí ukázať, že daný reťazec patrí do jazyka počiatočného neterminálu.

	Odvodený reťazec	Neterminál	Číslo pravidla	Použité reťazce
(i)	a	I	5	
(ii)	b	I	6	
(iii)	$b0$	I	9	(ii)
(iv)	$b00$	I	9	(iii)
(v)	a	E	1	(i)
(vi)	$b00$	E	1	(iv)
(vii)	$a + b00$	E	2	$(v), (vi)$
$(viii)$	$(a + b00)$	E	4	(vii)
(ix)	$a * (a + b00)$	E	3	$(v), (viii)$

- V tabuľke postupne uvádzame rôzne reťazce terminálov, ku ktorým uvádzame neterminály, do ktorých jazykov patria
- Na záver sa nám podarí zistiť, že reťazec $a * (a + b00)$ patrí do jazyka počiatočného neterminálu E, čím vlastne dokážeme, že príslušný reťazec spĺňa pravidlá gramatiky.
- Uvedený spôsob **končí**, keď sa nám podarí zistiť, že zadaný reťazec patrí do jazyka **počiatočného neterminálu**.
- **Alternatívny spôsob** zistenia toho, či daný reťazec spĺňa pravidlá gramatiky je **začať s počiatočným neterminálom** a **postupnou aplikáciou pravidiel odvodiť** zadaný reťazec, t.j. formou **derivácie**.

Derivácia v gramatike

- Nech je daná bezkontextová gramatika $G = (V, T, P, S)$.
- Nech $A \in V$ je neterminál, $\alpha, \beta, \gamma \in (V \cup T)^*$ sú reťazce terminálov a neterminálov a nech v pravidlách gramatiky je pravidlo $A \to \gamma$, t.j. neterminál A je možné prepísať na postupnosť γ.
- Nech je daná postupnosť symbolov gramatiky $\alpha A \beta$. Keďže podľa pravidiel je možné zobrať neterminál A a nahradiť ho postupnosťou γ, tak potom je možné urobiť tzv. **deriváciu** a prepísať $\alpha A \beta \Rightarrow \alpha \gamma \beta$.
- To znamená, **derivácia** znamená aplikáciu pravidla v reťazci symbolov, kde neterminál z **hlavy** pravidla nahradíme **telom** z pravidla.
- Jeden krok derivácie (t.j. jednu aplikáciu pravidla gramatiky) označujeme $\Rightarrow$.

Napríklad uvažujme znovu gramatiku zo slajdu č. 21. Potom je možné podľa pravidiel robiť rôzne derivácie, napríklad:

- $E \Rightarrow E * E$ (podľa pravidla č. 3 na neterminál E)
- $E * E \Rightarrow I * E$ (podľa pravidla č. 1 na prvý neterminál E)
- $I * E \Rightarrow a * E$ (podľa pravidla č. 5 na neterminál I)
- $a * E \Rightarrow a * (E)$ (podľa pravidla č. 4 na neterminál E)
- $a * (E) \Rightarrow a * (E + E)$ (podľa pravidla č. 2 na neterminál E)
- $a * (E + E) \Rightarrow a * (I + E)$ (podľa pravidla č. 1 na prvý neterminál E)
- $a * (I + E) \Rightarrow a * (a + E)$ (podľa pravidla č. 5 na neterminál I)
- $a * (a + E) \Rightarrow a * (a + I)$ (podľa pravidla č. 1 na neterminál E)
- $a * (a + I) \Rightarrow a * (a + I0)$ (podľa pravidla č. 9 na neterminál I)
- $a * (a + I0) \Rightarrow a * (a + I00)$ (podľa pravidla č. 9 na neterminál I)
- $a * (a + Ib0) \Rightarrow a * (a + b00)$ (podľa pravidla č. 6 na neterminál I)
- Z reťazca $a * (a + b00)$ nevieme ďalej odvodiť nič, lebo **neobsahuje neterminály!**.

V prípade, že chceme popísať **viacero aplikácií** pravidiel (viacero krokov derivácie), používame symbol $\Rightarrow^*$ definovaný nasledovne:

1. (Základný krok:) Pre každý reťazec neterminálov a terminálov α platí, že $\alpha \Rightarrow^* \alpha$. (t.j. každý reťazec sa dá odvodiť sám zo seba na **nula krokov**).
2. (Rekurzívny krok:) Ak $\alpha \Rightarrow^* \beta$ a $\beta \Rightarrow \gamma$, potom $\alpha \Rightarrow^* \gamma$. T.j. ak vieme z α odvodiť β na nula alebo viac krokov a z β vieme odvodiť γ na 1 krok, tak z α vieme odvodiť γ.

T.j. $\alpha \Rightarrow^* \gamma$ znamená, že vieme odvodiť reťazec neterminálov a terminálov γ z reťazca neterminálov a terminálov α na **nula alebo viac krokov**.

Návrat k príkladu

Pohľadom na slajd č. 29 vidíme, že v našej gramatike zo slajdu č. 21 vieme urobiť nasledovnú postupnosť derivácií:

$$\mathbf{E} \Rightarrow \mathbf{E} * E \Rightarrow \mathbf{I} * E \Rightarrow a * \mathbf{E} \Rightarrow a * (\mathbf{E}) \Rightarrow a * (\mathbf{E} + E) \Rightarrow a * (\mathbf{I} + E) \Rightarrow$$

$$a * (a + \mathbf{E}) \Rightarrow a * (a + \mathbf{I}) \Rightarrow a * (a + \mathbf{I}0) \Rightarrow a * (a + \mathbf{I}00) \Rightarrow a * (a + b00)$$

V skrátenom zápise teda:

$$E \Rightarrow a * (a + b00)$$

Mimochodom, všimnite si, že v uvedenej postupnosti derivácií sme **vždy** aplikovali pravidlo na ten neterminál, ktorý bol v príslušnom reťazci **najviac vľavo - preto sú uvedené boldom**.

Ľavé a pravé derivácie

- Ak pri derivácii reťazca **vždy** aplikujeme prepisovacie pravidlo na ten neterminál, ktorý sa nachádza najviac **vľavo** v reťazci, hovoríme o **ľavej derivácii**.
- Takúto deriváciu môžeme označovať aj $\Rightarrow_l$
- Ak pri derivácii reťazca **vždy** aplikujeme prepisovacie pravidlo na ten neterminál, ktorý sa nachádza najviac **vpravo** v reťazci, hovoríme o **pravej derivácii**.
- Takúto deriváciu môžeme označovať aj $\Rightarrow_p$
- Platí, že ak má nejaký reťazec ľavú deriváciu, tak určite má aj pravú deriváciu, a naopak.

Príklad

Derivácia na slajde č. 31 je zároveň aj ľavou deriváciou reťazca $a * (a + b00)$.
Pravá derivácia by vyzerala nasledovne.

$$\mathbf{E} \Rightarrow_p E*\mathbf{E} \Rightarrow_p E*(\mathbf{E}) \Rightarrow_p E*(E+\mathbf{E}) \Rightarrow_p E*(E+\mathbf{I}) \Rightarrow_p E*(E+\mathbf{I}0) \Rightarrow_p E*(E+\mathbf{I}00) \Rightarrow_p$$

$$E * (\mathbf{E} + b00) \Rightarrow_p E * (\mathbf{I} + b00) \Rightarrow_p \mathbf{E} * (a + b00) \Rightarrow_p \mathbf{I} * (a + b00) \Rightarrow_p a * (a + b00)$$

T.j. v tejto gramatike platí:

$$E \Rightarrow_p^* a * (a + b00)$$

Jazyky a gramatiky

- Ako je možno vidieť, gramatiky sú znovu len iný spôsob **špecifikácie jazykov**, keďže v každej gramatike je možné z neterminálov odvádzať rôzne reťazce.
- Najdôležitejšie reťazce sú tie, ktoré sú odvoditeľné z **počiatočného neterminálu**.
- Tieto reťazce predstavujú **jazyk, generovaný gramatikou**.

Definícia

Nech $G = (V, T, P, S)$ je bezkontextová gramatika. Jazyk generovaný gramatikou, ozn. $L(G)$ je množina reťazcov z terminálov, ktorú je možné odvodiť z počiatočného neterminálu gramatiky, t.j.:

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

- Každý jazyk ktorý je generovaný nejakou bezkontextovou gramatikou sa nazýva **bezkontextový jazyk**.
- Gramatiky teda zohrávajú analogickú úlohu ako regulárne výrazy / konečné automaty v tom, že tiež je pomocou nich možné **popisovať rôzne množiny reťazcov (jazykov)**.
- Akurát, ako bolo možné vidieť na príklade palindrómov, bezkontextové gramatiky vedia popisovať aj také jazyky, pre ktoré **neexistuje** konečný automat / regulárny výraz (ako napríklad palindrómy).
- Ak sme pre nejaký jazyk zostrojovali konečný automat, ktorý ho akceptuje, bolo dobré **dokázať**, že naša konštrukcia je korektná.
- **To isté je možné urobiť aj pri bezkontextových gramatikách**.

Dôkaz správnosti gramatiky

- Ak je našou úlohou pre nejaký jazyk L nájsť bezkontextovú gramatiku G, ktorá ho generuje, t.j. $L(G) = L$, správna gramatika je taká, pre ktorú platí:
  1. Každý reťazec generovaný gramatikou patrí do jazyka L, t.j. $L(G) \subseteq L$. Inými slovami gramatika negeneruje **nič navyše** / **mimo jazyka** L
  2. Každý reťazec z jazyka L má v gramatike odvodenie, t.j. $L \subseteq L(G)$, inými slovami **neexistuje reťazec jazyka L, ktorý by sa v gramatike nedal generovať**.
- Korektnosť gramatiky je možné aj dokázať, viď ďalší príklad.

Príklad dôkazu korektnosti gramatiky

Nech jazyk L sú všetky palindrómy nad abecedou $\{0, 1\}$. Tvrdíme, že tento jazyk je generovateľný gramatikou G s neterminálom P, ktorý je zároveň aj počiatočný, s terminálmi $\{0, 1\}$ a s pravidlami:

1. $P \to \varepsilon$
2. $P \to 0$
3. $P \to 1$
4. $P \to 0P0$
5. $P \to 1P1$

Teda tvrdíme, že $L(G) = L$. Korektnosť gramatiky dokážeme tak, že ukážeme, že $L \subseteq L(G)$ a že $L(G) \subseteq L$.

Dôkaz $L \subseteq L(G)$

Chceme dokázať, že $L \subseteq L(G)$, t.j. že ak je reťazec w **palindróm**, tak má v gramatike odvodenie. Dokážeme to indukciou na dĺžke $|w|$.

1. Základný krok: $|w| = 0$. Palindróm dĺžky 0 je $w = \varepsilon$, Keďže $P \Rightarrow \varepsilon$, tak odvodenie existuje.
2. Základný krok: $|w| = 1$. Palindrómy dĺžky 1 sú $w = a$ alebo $w = b$. Oba majú odvodenie, lebo $P \Rightarrow 0$ alebo $P \Rightarrow 1$.
3. Indukčný predpoklad: Každý palindróm dĺžky n **alebo menej** má v gramatike odvodenie.
4. Treba dokázať: Nech w je palindróm dĺžky aspoň $n + 1$. Potom má v gramatike odvodenie.

Dôkaz $L \subseteq L(G)$

Pokračujeme v dokazovaní:

1. Treba dokázať: Nech w je palindróm dĺžky aspoň $n+1$. Potom má v gramatike odvodenie.
2. Nech w je palindróm dĺžky $n+1$. Keďže je palindróm, musí začínať a končiť rovnakým symbolom, t.j. $w = 0x0$ alebo $w = 1x1$, kde x musí byť znovu palindróm, $|x| = n-1$.
3. Ak $w = 0x0$, tak z indukčného predpokladu predpokladáme, že $P \Rightarrow^* x$ (lebo predpokladáme, že každý palindróm kratší ako n má v G deriváciu). Keďže $w = 0x0$, tak môžeme uvažovať nasledovnú deriváciu:
   $P \Rightarrow 0P0 \Rightarrow^* 0x0 = w$. Čiže ak w je palindróm začínajúci a končiaci 0, musí mať v gramatike deriváciu.
4. Analogicky ak $w = 1x1$.
5. Tým sme dokázali, že aj palindróm dĺžky $n+1$ má v gramatike G deriváciu.

- Podarilo sa nám teda dokázať, že každý palindróm má v gramatike G deriváciu, teda platí $L \subseteq L(G)$.
- Ešte musíme dokázať, že $L(G) \subseteq L$, t.j. že všetko, čo gramatika generuje, je zároveň aj palindrómom, teda negeneruje nič navyše.

Dôkaz $L(G) \subseteq L$

Dôkaz $L(G) \subseteq L$ urobíme indukciou na počte derivačných krokov reťazca $w \in L(G)$. Musíme ukázať, že každý generovaný reťazec je zároveň aj palindróm.

1. Základný krok: derivácia na 1 krok. Na 1 krok vieme v gramatike odvodiť nasledovné reťazce:
   - 1.1 $P \Rightarrow \varepsilon$. Je odvodený reťazec ε palindróm? - áno.
   - 1.2 $P \Rightarrow 0$. Je odvodený reťazec 0 palindróm? - áno.
   - 1.3 $P \Rightarrow 1$. Je odvodený reťazec 1 palindróm? - áno.
2. Indukčný predpoklad: Každý reťazec w, ktorý vznikne deriváciou na n krokov je zároveň aj palindróm. T.j. ak $P \Rightarrow^* x$ na n krokov, potom x je palindróm.
3. Treba dokázať: Každý reťazec w ,ktorý vznikne deriváciou na $n+1$ krokov je zároveň aj palindróm.

Dôkaz $L(G) \subseteq L$

Pokračujeme v dokazovaní:

1. Treba dokázať: Každý reťazec w ,ktorý vznikne deriváciou na $n+1$ krokov je zároveň aj palindróm.
2. Uvažujme, ako vyzerá derivácia v gramatike na $n+1$ krokov. Taká derivácia musí byť v tvare:

   $$P \Rightarrow 0P0 \Rightarrow^* 0x0 = w$$

   alebo

   $$P \Rightarrow 1P1 \Rightarrow^* 1x1 = w$$

   Z indukčného predpokladu vieme, že x je palindróm. Lenže aj $0x0$, resp. $1x1$ je palindróm, teda **všetko**, čo gramatika generuje, sú palindrómy.

- Podarilo sa nám teda dokázať, že všetko, čo gramatika generuje, sú palindrómy, t.j. $L(G) \subseteq L$. V kombinácii so skôr dokázaným tvrdením:
- Podarilo sa nám teda dokázať, že každý palindróm má v gramatike G deriváciu, teda platí $L \subseteq L(G)$.
- Dostávame, že naša gramatika G je správna, pretože platí $L(G) = L$.

Dôkaz správnosti gramatiky

- Ak máme dokázať, že gramatika G, generuje nejaký zadaný jazyk L, t.j. $L = L(G)$, tak musíme dokázať:
  1. $L \subseteq L(G)$
  2. $L(G) \subseteq L$
- Tvrdenie $L \subseteq L(G)$ sa často dokazuje indukciou na **počte symbolov slova z jazyka** L
- Tvrdenie $L(G) \subseteq L$ sa často dokazuje indukciou na **počte krokov derivácie slova z** $L(G)$.
- Tak, ako sme to ukázali v predchádzajúcom príklade.

Vetné formy

Nech $G = (V, T, P, S)$ je bezkontextová gramatika.

- Povedali sme si, že každý reťazec w terminálov, $w \in T^*$, ktorý je v gramatike možné odvodiť z počiatočného neterminálu, $S \Rightarrow^* w$, je reťazcom z jazyka generovaného gramatikou, t.j. $w \in L(G)$.
- Zaveďme ešte jeden pojem: každý reťazec neterminálov a terminálov $\alpha \in (V \cup T)^*$, ktorý je možné odvodiť z počiatočného neterminálu, sa nazýva **vetná forma**.
- V prípade ľavých (pravých) derivácií hovoríme o ľavých (pravých) vetných formách.
- Každý reťazec $w \in L(G)$ je teda zároveň špeciálnou vetnou formou, zloženou len z terminálov.

Vetné formy - príklad

Nech $G = (V, T, P, S)$ je bezkontextová gramatika, $V = \{S\}$, $T = \{a, b\}$, pravidlá:

1. $S \to \varepsilon$
2. $S \to aSb$

V tejto gramatike existuje o.i. napríklad derivácia:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

Teda $aaabbb \in L(G)$ je reťazec generovaný gramatikou G. Navyše:
$S, aSb, aaSbb, aaaSbbb, aaabbb$ sú taktiež **vetné formy** danej gramatiky.

Mimochodom, táto gramatika generuje jazyk $L(G) = \{a^n b^n \mid n \geq 0\}$, teda jazyk, ktorý **nebol regulárny** a pre ktorý **neexistoval konečný automat**. Jedným zo spôsobov, ako takýto jazyk popísať, je teda uvedená bezkontextová gramatika!

Praktické využitie bezkontextových gramatík

- Bezkontextové gramatiky boli pôvodné zavedené americkým lingvistom Noamom Chomskym, ktorý chcel pomocou nich popísať prirodzené jazyky. Žiaľ, to sa nepodarilo, lebo sa zistilo, že prirodzené jazyky sa nedajú popísať bezkontextovými jazykmi.
- Bezkontextové gramatiky však našli uplatnenie v informatike pri popise rekurzívne definovaných konceptov.
- Uvedieme si 2 dôležité príklady využitia bezkontextových gramatík v praxi:
  1. Popis syntaxe programovacích jazykov
  2. Popis značkovacích jazykov.

Popis syntaxe programovacích jazykov

- Každý programovací jazyk (C, C++, Python, Java, ...) má pravidlá, ktoré musia spĺňať programy v ňom písané - tzv. syntaktické pravidlá.
- Každý reťazec symbolov spĺňajúci tieto pravidlá je potom platný, syntakticky korektný, program napísaný v tomto jazyku.
- Syntaktické pravidlá bývajú v praxi popísané práve pomocou bezkontextových gramatík.
- Množina všetkých syntakticky korektných programov napísaných v danom programovacom jazyku potom tvorí jazyk v zmysle definície, akú poznáme.

Popis syntaxe programovacích jazykov

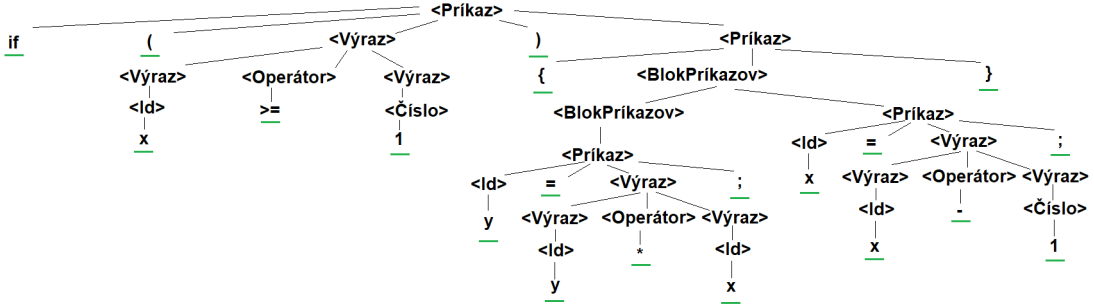
- Každý prekladač / interpretér daného programovacieho jazyka musí v istej fáze prekladu / vykonávania programu zisťovať, či je vstupný program syntakticky korektný.
- Ak je program syntakticky korektný, tak sa navyše hľadá **presný postup**, ako sa dá program v programovacom jazyku odvodiť (t.j. aplikáciou **ktorých pravidiel gramatiky** popisujúcej syntax viem odvodiť môj zadaný **program**)
- Na základe toho, ktoré pravidlá musím použiť, totižto prekladač / interpretér vie vôbec "pochopiť", čo môj program robí a teda vie alebo vygenerovať ekvivalentný program v inom jazyku (preklad), alebo vie korektne vykonať môj program (interpretácia).

Príklad

Uvažujme zjednodušený príklad popisu syntaxe príkazu v programovacom jazyku:

1. <Príkaz> → <Id> = <Výraz> ;
2. <Príkaz> → { <BlokPríkazov> }
3. <Príkaz> → if (<Výraz>) <Príkaz>
4. <BlokPríkazov> → <Príkaz>
5. <BlokPríkazov> → <BlokPríkazov> <Príkaz>
6. <Výraz> → <Id>
7. <Výraz> → <Číslo>
8. <Výraz> → <Výraz> <Operátor> <Výraz>
9. <Id> → a | b | ... | z
10. <Číslo> → 0 | 1 | ... | 9
11. <Operátor> → + | − | ⋆ | / | < | > | == | != | <= | >=

Derivácia: `if (x >= 1) {y = y * x; x = x - 1;}` vyzerá:



Uvedená "grafická reprezentácia" derivácie sa nazýva aj **derivačný strom**.

- Vidíme, že existuje derivácia, teda uvedený "program" je syntakticky korektný.
- Navyše je zo samotnej derivácie zrejmé, ktorá časť programu predstavuje podmienku vetvenia a ktorá časť telo vetvenia.
- To je samozrejme docielené aj tým, ako je samotná gramatika skonštruovaná.
- Viac na predmete Automaty a formálne jazyky :).

Vidíme, že napríklad reťazec (program) `if (x >= 1 {y = y * x; x = x - 1;}` by deriváciu nemohol mať, pretože:

- V "podmienke" `(x >= 1` chýba pravá zátvorka
- Pravidlo gramatiky číslo 3, ktoré generuje podmienený príkaz, však vždy pre ľavú zátvorku vygeneruje aj pravú.
- Preto uvedený reťazec by v gramatike nemal odvodenie, teda uvedený program **nie je syntakticky korektný** vzhľadom na bezkontextovú gramatiku popisujúcu syntax programovacieho jazyka.

Popis syntaxe programovacích jazykov

Ďalšie dôvody použitia bezkontextových gramatík pre popis syntaxe programovacích jazykov:

- Bezkontextové gramatiky sú vhodné na popis korektne ozátvorkovaných výrazov :

$$B \to BB \mid (B) \mid \varepsilon$$

- Ozátvorkované výrazy sú súčasťou de facto každého programovacieho jazyka.
- Taktiež sa používajú na popis štruktúry matematických výrazov (viď zjednodušený príklad na slajde č. 21)
- Rôzne konštrukcie používané v programovacích jazykoch sa správajú "podobne" ako zátvorky v tom, že sú **párové** - napríklad $\{, \}$ na označenie bloku príkazov v jazkoch C, Java, resp. dvojica **begin,end** v jazykoch Pascal, Lazarus

Popis syntaxe programovacích jazykov

Ďalšie dôvody použitia bezkontextových gramatík pre popis syntaxe programovacích jazykov:

- Iné konštrukcie sú podobné zátvorkám s tým rozdielom, že pripúšťajú aj nebalansovanú ľavú zátvorku, t.j. napríklad **if-else** konštrukcia, kde teoreticky môže existovať len samotný príkaz **if**.

$$S \to \varepsilon \mid SS \mid iS \mid iSeS$$

Popis syntaxe programovacích jazykov

Navyše platí veľmi dôležitá praktická vlastnosť:

Ak je gramatika popisujúca syntax programovacieho jazyka **vhodne skonštruovaná**, potom **existuje rýchly (s lineárnou zložitosťou) algoritmus**, ktorý dokáže pre zadaný program zistiť:

- Či sa dá v gramatike odvodiť (t.j. či je syntakticky korektný)
- Ako samotná derivácia vyzerá - dôležité, aby sa korektne označili v kóde podmienky, bloky príkazov, atď. keďže sú to dôležité informácie z hľadiska toho, čo vôbec program robí.

V podstate presne o tomto bude predmet Automaty a formálne jazyky.

Bezkontextové gramatiky a značkovacie jazyky

- Značkovacie jazyky (markup languages) predstavujú systém zostrojenia anotácie dokumentu, ktorá je syntakticky odlíšiteľná od ostatného textu.
- T.j. dokument sa na základe týchto značiek spracuje a tieto značky následne napríklad ovplyvňujú formátovanie samotného dokumentu.
- Typickým príkladom sú jazyky HTML (Hypertext Markup Language) alebo systém LaTeX
- Iné značkovacie jazyky používajú značky na označenie jeho sémantického obsahu, napríklad XML (Extensible Markup Language).

Bezkontextové gramatiky a HTML

- V prípade jazyka HTML je možné jeho syntax taktiež popísať bezkontextovou gramatikou.
- Následne sa táto bezkontextová gramatika môže použiť na spracovanie a riadenie zobrazenia HTML dokumentu na obrazovke.
- Uvažujme nasledovný jednoduchý príklad HTML dokumentu, v ktorom je text a značky, použité na jeho formátovanie. Ako viete, značky v HTML sa uvádzajú medzi $<, >$

HTML príklad

HTML zdroj

```
<P> There are <EM>10 types of people</EM>:
<OL>
<LI> Those who understand binary.
<LI> Those who don't.
</OL>
```

Výstup

There are *10 types of people*:

1. Those who understand binary.
2. Those who don't.

Pre jednoduchosť uvažujme nasledovné značky v HTML:

- Párové značky <EM>,</EM> pre ktoré platí, že čo je uvedené medzi nimi sa formátuje ako *zvýraznené*, v našom prípade ako *italics*.
- Párové značky <OL>,</OL> označujúce Ordered List (usporiadaný očíslovaný zoznam)
- (voliteľne párovú) značku <P> označujúcu začiatok odseku
- (voliteľné párovú) značku <LI> označujúcu jednu položku zoznamu

HTML príklad

HTML zdroj

```
<P> There are <EM>10 types of people</EM>:
<OL>
<LI> Those who understand binary.
<LI> Those who don't.
</OL>
```

Výstup

There are *10 types of people*:

1. Those who understand binary.
2. Those who don't.

Pre jednoduchosť uvažujme nasledovné značky v HTML:

- Párové značky <EM>,</EM> pre ktoré platí, že čo je uvedené medzi nimi sa formátuje ako *zvýraznené*, v našom prípade ako *italics*.
- Párové značky <OL>,</OL> označujúce Ordered List (usporiadaný očíslovaný zoznam)
- (voliteľne párovú) značku <P> označujúcu začiatok odseku
- (voliteľné párovú) značku <LI> označujúcu jednu položku zoznamu

Aby sme popísali syntax HTML, resp. korektného **HTML dokumentu** pomocou bezkontextovej gramatiky, zaveďme si neterminály označujúce niektoré základné triedy reťazcov v HTML jazyku:

- *Text* budú ľubovoľné reťazce predstavujúce klasický text, t.j. reťazce neobsahujúce značky.
- *Char* bude ľubovoľný symbol, ktorý môže byť súčasťou klasického textu.
- *Doc* bude dokument, ktorý je tvorený postupnosťou elementov.
- *Element* je alebo *Text*, alebo párové značky s nejakým dokumentom (*Doc*) medzi nimi, alebo nepárová značka nasledovaná dokumentom (*Doc*).
- *ListItem* je značka <LI> za ktorou nasleduje dokument (*Doc*), ktorý predstavuje 1 položku zoznamu.
- *List* je postupnosť nula alebo viacerých položiek zoznamu (t.j. viacerých *ListItem*-ov).

S ich pomocou vieme popísať syntax HTML dokumentu nasledovne:

1. *Char* $\rightarrow$ *a* | *A* | ...
2. *Text* $\rightarrow \varepsilon$ | *CharText*
3. *Doc* $\rightarrow \varepsilon$ | *ElementDoc*
4. *Element* $\rightarrow$ *Text*
5. *Element* $\rightarrow$<EM> *Doc* </EM>
6. *Element* $\rightarrow$<P> *Doc*
7. *Element* $\rightarrow$<OL> *Doc* </OL>| ...
8. *ListItem* $\rightarrow$<LI> *Doc*
9. *List* $\rightarrow \varepsilon$ | *ListItemList*

Neterminál *Doc* by predstavoval počiatočný neterminál pre samotný HTML dokument. Na základe gramatiky je možné potom zistiť, či je dokument HTML validný a taktiež, ako ho je potrebné zobraziť.

Bezkontextové gramatiky a XML

- V prípade HTML je teda možné použiť bezkontextové gramatiky na popis syntaxe HTML a taktiež je nimi možné riadiť spracovanie HTML dokumentu.
- V prípade XML majú bezkontextové gramatiky využitie aj pri konštrukcii tzv. DTD (Document-type Definition).
- Ako pravdepodobne viete, XML je značkovací jazyk na popis sémanticky textu (dát). Samotné značky v jazyku označujú "typ" dát. Keďže značky (t.j. typ) môžu byť ľubovoľné, je možné tým pádom vyrobiť rôzne XML schémy pre rôzne dáta.

XML príklad

Napríklad XML reprezentácia "lístočku" s odosielateľom, prijímateľom, nadpisom a textom:

```
 <?xml version="1.0" encoding="UTF-8"?>
<motak>
<odosielatel>Maroš</odosielatel>
<prijimatel>Peto</prijimatel>
<nadpis>Pokyn</nadpis>
<telo>Zober od rodiny môj mobil</telo>
</motak>
```

DTD

- Aby bolo pri spracovaní XML zrejmé, aké rôzne značky môže obsahovať a aké štruktúry sa môžu medzi značkami nachádzať, tak je vhodné, aby bol súčasťou XML aj dokument, ktorý popisuje štruktúru príslušného XML dokumentu.
- V praxi sa na tieto účely používa alebo XSD alebo DTD.
- My sa tu budeme ďalej venovať DTD, t.j. Document-type definition.
- DTD je de facto **bezkontextová gramatika**, ktorá preddefinovaným spôsobom popisuje štruktúru príslušného XML dokumentu, na ktorý sa viaže.

DTD

- Aby bolo pri spracovaní XML zrejmé, aké rôzne značky môže obsahovať a aké štruktúry sa môžu medzi značkami nachádzať, tak je vhodné, aby bol súčasťou XML aj dokument, ktorý popisuje štruktúru príslušného XML dokumentu.
- V praxi sa na tieto účely používa alebo XSD alebo DTD.
- My sa tu budeme ďalej venovať DTD, t.j. Document-type definition.
- DTD je de facto **bezkontextová gramatika**, ktorá preddefinovaným spôsobom popisuje štruktúru príslušného XML dokumentu, na ktorý sa viaže.

DTD

DTD dokument pozostáva z nasledovných časti: Na začiatok je definícia tzv. koreňového elementu (name-of-DTD):

```
<!DOCTYPE name-of-DTD [
zoznam definicii elementov
]>
```

Vnútri v zozname definícií sú uvedené definície elementov v tvare:

```
<!ELEMENT element-name (popis elementu)>
```

Popis elementu je v podstate regulárny výraz, popisujúci, ako je element ďalej definovaný. Pre jednoduchosť uvažujme, že môže obsahovať **mená iných elementov**, špeciálny člen `#PCDATA` označujúci ľubovoľný text a operátory, ktoré ich spájajú, kam patrí zjednotenie označené $|$, zreťazenie označené čiarkou a špeciálne iterácie $*, +, ?$.

DTD príklad

Pre lístoček zo slajdu č. 64 by DTD dokument vyzeral nasledovne:

```
<!DOCTYPE motak
[
<!ELEMENT motak (odosielatel,prijimatel,nadpis,telo)>
<!ELEMENT odosielatel (#PCDATA)>
<!ELEMENT prijimatel (#PCDATA)>
<!ELEMENT nadpis (#PCDATA)>
<!ELEMENT telo (#PCDATA)>
]>
```

DTD ako gramatika

DTD z predchádzajúceho slajdu vlastne predstavuje bezkontextovú gramatiku v tvare:

1. $motak \rightarrow odosielatel\ prijimatel\ nadpis\ telo$
2. $odosielatel \rightarrow \#PCDATA$
3. $prijimatel \rightarrow \#PCDATA$
4. $nadpis \rightarrow \#PCDATA$
5. $telo \rightarrow \#PCDATA$

kde *motak* je zároveň počiatočný neterminál. $\#PCDATA$ je "terminál" predstavujúci reťazce znakov.

Validácia XML voči DTD

- XML dokumenty sú tzv. "well-formed", ak sú syntakticky korektné v zmysle, že majú koreňový element, elementy majú ukončovaciu značku (t.j. <motak> a </motak>), elementy sú korektne vnorené a podobne.
- Ak je k dispozícii DTD definícia, tak je možné XML voči DTD validovať - také XML nazývame "valid".
- Validáciou XML je vlastne proces overenia, či je možné XML "derivovať" v gramatike popísanej v DTD dokumente.

Použitá literatúra

1 Hopcroft, Motwani, Ullman - Introduction to Automata Theory, Languages and Computations, 3rd Ed.