Automaty a gramatiky

6

Roman Barták, KTIML

bartak@ktiml.mff.cuni.cz
http://ktiml.mff.cuni.cz/~bartak

Úvod do formálních gramatik

Gramatiky, všichni je známe, ale co to je?

Popis jazyka pomocí pravidel, podle kterých se vytvářejí všechny řetězce daného jazyka.

Původně pro popis přirozených jazyků

<věta> → <podmětná část> <přísudková část>

Zadání syntaxe vyšších programovacích jazyků

od dob Algolu 60

Backus-Naurova normální forma (BNF)

<číslo> :== <číslo bez zn.> | +<číslo bez zn.> |-<číslo bez zn.>

<číslo bez zn.> :== <číslice> | <číslice><číslo bez znam>.

<číslice> :== 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Příklady gramatik

1) Gramatika správných uzávorkování

V → VV | (V) | ()

Výraz (()())(()) je generován posloupností přepisů:

V → VV → (V)V → (VV)V → (()V)V → (()())V → (()())(V) → (()())(())

2) Gramatika generující všechny výrazy s operacemi + a *, závorkami a jedinou konstantou c.

V → T+V | T

T → F*T | F

F → (V) | c

Výraz c+c*c+c je generován posloupností přepisů:

V → T+V → F+V → c+V → c+T+V → c+F*T+V → c+c*T+V → c+c*F+V → c+c*c+V → c+c*c+T → c+c*c+F → c+c*c+c

Přepisovací systémy - základní pojmy

Přepisovacím (produkčním) systémem nazýváme dvojici

R = (V,P),

kde

V - konečná abeceda

P - konečná množina přepisovacích pravidel

přepisovací pravidlo (produkce) je uspořádaná dvojice (u,v), kde u,v∈V* (zpravidla píšeme u → v)

Říkáme, že *w se přímo přepíše na z* (píšeme $w \Rightarrow z$), jestliže:

$\exists u,v,x,y \in V^*$ tž. $w=xuy$, $z=xvy$ a $(u \rightarrow v) \in P$.

Říkáme, že *w se přepíše na z* (píšeme $w \Rightarrow^* z$), jestliže:

$\exists u_1,\ldots,u_n \in V^*$ $\quad w=u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_n=z$.

Posloupnost $u_1,\ldots,u_n$ nazýváme *odvozením* (derivací).

Pokud $\forall i \neq j \; u_i \neq u_j$, potom hovoříme o *minimálním odvození*.

Přepisovací systémy

Příklad:

V = {0,1}

P = {01→10, 10→01}

00110 ⇒* 00011 dostaneme z 00<u>11</u>0 ⇒ 00<u>10</u>1 ⇒ 00011

00110 ⇒* 01100 dostaneme z 0<u>01</u>10 ⇒ 0<u>10</u>10 ⇒ 01100

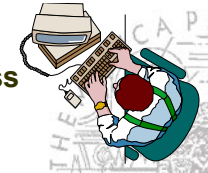
libovolné slovo přepíše na libovolné jiné slovo (se stejným počtem výskytů 0 a 1)

Produkční systémy slouží jako programovací nástroj v UI

program = systém produkcí

data = slova v abecedě

- OPS5, TOPS, CLIPS, JBoss Rules, Jess
- Constraint Handling Rules (CHR)
- Definite Clause Grammars (DCG)



Formální (generativní) gramatiky

Generativní gramatikou nazýváme čtveřici $G=(V_N,V_T,S,P)$:

V_N - konečná množina neterminálních symbolů

V_T - konečná množina terminálních symbolů

obě abecedy jsou neprázdné a disjunktní!

$S \in V_N$ - počáteční neterminální symbol

P - systém produkcí $u \to v$, kde $u,v \in (V_N \cup V_T)^*$ a u obsahuje alespoň jeden neterminální symbol.

Jazyk L(G) generovaný gramatikou G definujeme takto:

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

Gramatiky G_1 a G_2 jsou *ekvivalentní*, jestliže $L(G_1)=L(G_2)$.

Příklad:

$G = (\{S\},\{0,1\},S,\{S \to 0S1, S \to 01\})$, $L(G) = \{0^i1^i \mid i \geq 1\}$

Chomského hierarchie

Klasifikace gramatik podle tvaru přepisovacích pravidel.

gramatiky typu 0 (rekurzivně spočetné jazyky $\mathcal{L}_0$)

pravidla v obecné formě

gramatiky typu 1 (kontextové jazyky $\mathcal{L}_1$)

pouze pravidla ve tvaru $\alpha X \beta \to \alpha w \beta$,

$X \in V_N$, $\alpha,\beta \in (V_N \cup V_T)^*$, $w \in (V_N \cup V_T)^+$

jedinou výjimkou je pravidlo $S \to \lambda$, potom se ale S nevyskytuje na pravé straně žádného pravidla

gramatiky typu 2 (bezkontextové jazyky $\mathcal{L}_2$)

pouze pravidla ve tvaru $X \to w$, $X \in V_N$, $w \in (V_N \cup V_T)^*$

gramatiky typu 3 (regulární/pravé lineární jazyky $\mathcal{L}_3$)

pouze pravidla ve tvaru $X \to wY$, $X \to w$, $X,Y \in V_N$, $w \in V_T^*$

Uspořádanost Chomského hierarchie

Chomského hierarchie definuje uspořádání tříd jazyků:

$$\mathcal{L}_0 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_3$$

Dokonce vlastní podmnožiny (později):

$$\mathcal{L}_0 \supset \mathcal{L}_1 \supset \mathcal{L}_2 \supset \mathcal{L}_3$$

$\mathcal{L}_0 \subseteq \mathcal{L}_1$ (rekurzivně spočetné jazyky zahrnují kontextové jazyky)

obecná pravidla ⊆ pravidla tvaru $\alpha X \beta \to \alpha w \beta$

$\mathcal{L}_2 \subseteq \mathcal{L}_3$ (bezkontextové jazyky zahrnují regulární jazyky)

„$X \to w$, $w \in (V_N \cup V_T)^*$ “ ⊆ „$X \to wY$, $X \to w$, $Y \in V_N$, $w \in V_T^*$ “

$\mathcal{L}_1 \subseteq \mathcal{L}_2$ (kontextové jazyky zahrnují bezkontextové jazyky)

$\alpha X \beta \to \alpha w \beta$, $|w|>0$ vs. $X \to w$, $|w| \geq 0$

problém s pravidly tvaru $X \to \lambda$

Můžeme z bezkontextových gramatik vyřadit pravidla $X \to \lambda$?

Nevypouštějící bezkontextové gramatiky

Bezkontextová gramatika G je ***nevypouštějící*** právě tehdy, když nemá pravidla ve tvaru $X \to \lambda$.

Věta: Ke každé bezkontextové gramatice G existuje nevypouštějící bezkontextová gramatika G_1 taková, že $L(G_1) = L(G) - \{\lambda\}$ (jazyky se liší maximálně o prázdné slovo).

Je-li $\lambda \in L(G)$, potom existuje BKG G_2 tak, že $L(G_2) = L(G)$ a jediné pravidlo s λ na pravé straně je $S' \to \lambda$ a S' (počáteční neterminál) se nevyskytuje na pravé straně žádného pravidla G_2 (tedy $\mathcal{L}_1 \subseteq \mathcal{L}_2$).

Příklad:

G: $S \to 0S1 \mid \lambda$

G_1: $S \to 0S1 \mid 01$

G_2: $S' \to S \mid \lambda$, $S \to 0S1 \mid 01$

Převod na nevypouštějící BKG

… aneb, jak se zbavit pravidel ve tvaru $X \to \lambda$?

Základní myšlenka:

- pravidlo $X \to \lambda$ se používá pro vyhození X ze slova
- co když X do slova vůbec nezařadíme?

…, $Y \to uXv$, $X \to \lambda$, … $\Rightarrow$ …, $Y \to uXv$, $Y \to uv$, …

1) Nejprve zjistíme neterminály, které se přepisují na λ:

$U = \{X \mid X \in V_N \wedge X \Rightarrow^* \lambda\}$

Proč tak silně (nestačilo by $X \to \lambda$ místo $X \Rightarrow^* \lambda$)?

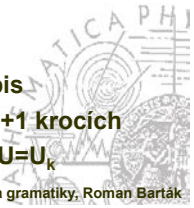
Řešení derivací $X \Rightarrow^{X \to Y} Y \Rightarrow^{Y \to Z} Z \Rightarrow^{Z \to \lambda} \lambda$

Iterační algoritmus pro získání U:

$U_1 = \{X \mid X \in V_N \wedge (X \to \lambda) \in P\}$ přímý přepis

$U_{i+1} = \{X \mid X \in V_N \wedge (X \to w) \in P, w \in U_i^*\}$ přepis po i+1 krocích

$U_1 \subseteq U_2 \subseteq \ldots \subseteq V_N$ + stabilizace ($\exists k\ U_k = U_{k+1} = \ldots$) + $U = U_k$



Převod na nevypouštějící BKG - pokračování

2) Úprava pravidel

do P_1 dáme pravidla tvaru $X \to u$ taková, že:

- $u \neq \lambda$
- v P je pravidlo $X \to v_1 Y_1 v_2 \ldots v_m Y_m v_{m+1}$, $Y_i \in U$, $v_i \in ((V_N - U) \cup V_T)^*$ a u vzniká z $(v_1 Y_1 v_2 \ldots v_m Y_m v_{m+1})$ vypuštěním některých (všech, žádného) symbolů Y_i.

3) Ještě $L(G_1) = L(G) - \{\lambda\}$

zřejmé: G_1 je nevypouštějící BKG, $L(G_1) \subseteq L(G)$, $\lambda \notin L(G_1)$

nechť $w \in L(G)$ a $w \neq \lambda$, tj. $S \Rightarrow^* w$,

pokud se použilo pravidlo z $P\text{-}P_1$, pak má tvar $X \to \lambda$

v derivaci před ním muselo být užito pravidlo $Y \to uXv$

uděláme novou derivaci s $Y \to uv$ a bez $X \to \lambda$

4) Zbývá situace $\lambda \in L(G)$

$G_2 = (V_N \cup \{S'\}, V_T, S', P_1 \cup \{S' \to \lambda, S' \to S\})$

Příklad - nevypouštějící BKG

$S \to aSc \mid A$

$A \to bAc \mid \lambda$

1) Nejprve zjistíme neterminály, které se přepisují na λ:

$U = \{A, S\}$

2) Upravíme pravidla:

$S \to aSc \mid A$

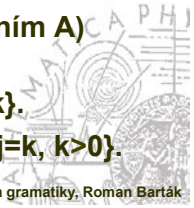
$S \to ac$ (vzniklo z $S \to aSc$ vypuštěním S)

$A \to bAc$ (pravidlo $A \to \lambda$ nepřevádíme)

$A \to bc$ (vzniklo z $A \to bAc$ vypuštěním A)

Původní gramatika generuje jazyk $\{a^i b^j c^k \mid i+j=k\}$.

Převedená gramatika generuje jazyk $\{a^i b^j c^k \mid i+j=k, k>0\}$.



Chomského hierarchie

gramatiky typu 0 (rekurzivně spočetné jazyky $\mathcal{L}_0$)

pravidla v obecné formě

gramatiky typu 1 (kontextové jazyky $\mathcal{L}_1$)

pouze pravidla ve tvaru $\alpha X\beta \to \alpha w\beta$,
$X \in V_N$, $\alpha,\beta \in (V_N \cup V_T)^*$, $w \in (V_N \cup V_T)^+$

jedinou výjimkou je pravidlo $S \to \lambda$, potom se ale S nevyskytuje na pravé straně žádného pravidla

gramatiky typu 2 (bezkontextové jazyky $\mathcal{L}_2$)

pouze pravidla ve tvaru $X \to w$, $X \in V_N$, $w \in (V_N \cup V_T)^*$

gramatiky typu 3 (regulární/pravé lineární jazyky $\mathcal{L}_3$)

pouze pravidla ve tvaru $X \to wY$, $X \to w$, $X,Y \in V_N$, $w \in V_T^*$

Gramatiky typu 3 a regulární jazyky

pouze pravidla ve tvaru $X \to wY$, $X \to w$, $X,Y \in V_N$, $w \in V_T^*$

Podívejme se na derivace generované gramatikami typu 3

P: $S \to 0S \mid 1A \mid \lambda$, $A \to 0A \mid 1B$, $B \to 0B \mid 1S$

$S \Rightarrow 0S \Rightarrow 01A \Rightarrow 011B \Rightarrow 0110B \Rightarrow 01101S \Rightarrow 01101$

Pozorování:

- každé slovo derivace obsahuje právě jeden neterminál
- tento neterminál je vždy umístěn zcela vpravo
- aplikací pravidla $X \to w$ se derivace uzavírá
- krok derivace = generuje symbol(y) +změní neterminál

Idea vztahu gramatiky a konečného automatu:

neterminál = stav konečného automatu

pravidla = přechodová funkce

Převod konečného automatu na gramatiku

$L \in \mathcal{F} \Rightarrow L \in \mathcal{L}_3$

Důkaz:

$L=L(A)$ pro nějaký konečný automat $A=(Q,X,\delta,q_0,F)$

definujme gramatiku $G=(Q,X,q_0,P)$, kde pravidla mají tvar

$p \to aq$, když $\delta(p,a)=q$

$p \to \lambda$, když $p \in F$

ještě $L(A)=L(G)$?

1) $\lambda \in L(A) \Leftrightarrow q_0 \in F \Leftrightarrow (q_0 \to \lambda) \in P \Leftrightarrow \lambda \in L(G)$

2) $a_1 \ldots a_n \in L(A) \Leftrightarrow \exists q_0,\ldots,q_n \in Q$ tž. $\delta(q_i,a_{i+1})=q_{i+1}$, $q_n \in F$
$\Leftrightarrow (q_0 \Rightarrow a_1q_1 \Rightarrow \ldots \Rightarrow a_1 \ldots a_nq_n \Rightarrow a_1 \ldots a_n)$ je derivace pro $a_1 \ldots a_n$
$\Leftrightarrow a_1 \ldots a_n \in L(G)$

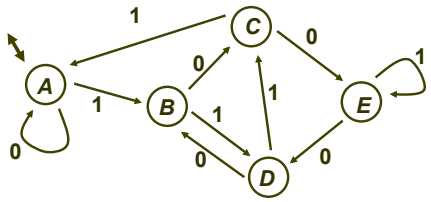
QED

A co naopak?

- pravidla $X \to aY$ kódujeme do přechodové funkce a $X \to \lambda$ je konec
- ale co pravidla $X \to a_1 \ldots a_nY$, $X \to Y$, $X \to a_1 \ldots a_n$?

Příklad převodu KA na gramatiku

$A \to 1B \mid 0A \mid \lambda$
$B \to 0C \mid 1D$
$C \to 0E \mid 1A$
$D \to 0B \mid 1C$
$E \to 0D \mid 1E$



Příklady derivací:

$A \Rightarrow 0A \Rightarrow 0$ (0)

$A \Rightarrow 1B \Rightarrow 10C \Rightarrow 101A \Rightarrow 101$ (5)

$A \Rightarrow 1B \Rightarrow 10C \Rightarrow 101A \Rightarrow 1010A \Rightarrow 1010$ (10)

$A \Rightarrow 1B \Rightarrow 11D \Rightarrow 111C \Rightarrow 1111A \Rightarrow 1111$ (15)

$L = \{ w \mid w \in \{0,1\}^* \wedge w$ je binární zápis čísla dělitelného 5$\}$

Standardizace pravidel regulární gramatiky

Ke každé gramatice $G=(V_N,V_T,S,P)$ typu 3 existuje ekvivalentní gramatika G‘, která obsahuje pouze pravidla ve tvaru: $X\rightarrow aY$ a $X\rightarrow\lambda$.

Důkaz:

definujme $G'=(V'_N,V_T,S,P')$, kde pravidla P‘ získáme takto

P	P‘
$X\rightarrow aY$	$X\rightarrow aY$
$X\rightarrow\lambda$	$X\rightarrow\lambda$
$X\rightarrow a_1\ldots a_nY$	$X\rightarrow a_1Y_2,\ Y_2\rightarrow a_2Y_3,\ \ldots,\ Y_n\rightarrow a_nY$
$Z\rightarrow a_1\ldots a_n$	$Z\rightarrow a_1Z_1,\ Z_1\rightarrow a_2Z_2,\ \ldots,\ Z_n\rightarrow\lambda$

($Y_2,\ldots,Y_n$, $Z_1,\ldots,Z_n$ jsou nové neterminály - pro každé pravidlo jiná sada)

zbývá $X\rightarrow Y$

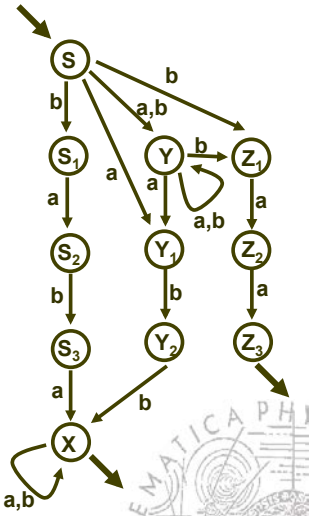
definujme $U(X) = \{Y \mid Y\in V_N \wedge X\Rightarrow^* Y\}$

efektivní postup $U_1=\{Y \mid (X\rightarrow Y)\in P\}$, $U_{i+1} = U_i \cup \{Y \mid (Z\rightarrow Y)\in P, Z\in U_i\}$

$X\rightarrow Y$	$X\rightarrow w$ pro všechna $Z\rightarrow w$ z P‘ a $Z\in U(X)$

Příklad standardizace regulární gramatiky

Originální	*Převedená*
$S\rightarrow babaX$	$S\rightarrow bS_1$ $S_1\rightarrow aS_2$ $S_2\rightarrow bS_3$ $S_3\rightarrow aX$
$S\rightarrow Y$	$S\rightarrow aY \mid bY \mid aY_1 \mid bZ_1$
$Y\rightarrow aY \mid bY$	$Y\rightarrow aY \mid bY$
$Y\rightarrow abbX$	$Y\rightarrow aY_1$ $Y_1\rightarrow bY_2$ $Y_2\rightarrow bX$
$Y\rightarrow baa$	$Y\rightarrow bZ_1$ $Z_1\rightarrow aZ_2$ $Z_2\rightarrow aZ_3$ $Z_3\rightarrow\lambda$
$X\rightarrow aX \mid bX \mid \lambda$	$X\rightarrow aX \mid bX \mid \lambda$



$L=\{ w \mid w=babau \vee w=uabbv \vee w=ubaa,\ u,v\in\{a,b\}^* \}$

Převod gramatiky na konečný automat

$L\in\mathcal{L}_3 \Rightarrow L\in\mathcal{F}$

Důkaz:

$L=L(G)$ pro nějakou gramatiku $G = (V_N,V_T,S,P)$ typu 3 obsahující pouze pravidla ve tvaru: $X\rightarrow aY$ a $X\rightarrow\lambda$

definujme nedeterministický konečný automat

$A = (V_N,V_T,\delta,\{S\},F)$, kde:

$F = \{ X \mid (X\rightarrow\lambda)\in P\}$

$\delta(X,a) = \{ Y \mid (X\rightarrow aY)\in P\}$

ještě $L(G)=L(A)$?

1) $\lambda\in L(G) \Leftrightarrow (S\rightarrow\lambda)\in P \Leftrightarrow S\in F \Leftrightarrow \lambda\in L(A)$

2) $a_1\ldots a_n \in L(G)$

$\Leftrightarrow$ existuje derivace $(S \Rightarrow a_1X_1 \Rightarrow \ldots \Rightarrow a_1\ldots a_nX_n \Rightarrow a_1\ldots a_n)$

$\Leftrightarrow \exists X_0,\ldots,X_n\in V_N$ tž. $\delta(X_i,a_{i+1})\ni X_{i+1}$, $X_0=S$, $X_n\in F \Leftrightarrow a_1\ldots a_n \in L(A)$

Levé (a pravé) lineární gramatiky

Gramatiky typu 3 nazýváme také *pravé lineární* (neterminál je vždy vpravo).

Obdobně - gramatika G je *levá lineární*, jestliže má pouze pravidla tvaru $X\rightarrow Yw$, $X\rightarrow w$, $X,Y\in V_N$, $w\in V_T^*$ (neterminál je vždy vlevo).

Věta: Jazyky generované levou lineární gramatikou jsou právě regulární jazyky.

Důkaz:

- „otočením“ pravidel dostaneme pravou lineární gramatiku $X\rightarrow Yw$, $X\rightarrow w$ převedeme na $X\rightarrow w^RY$, $X\rightarrow w^R$
- získaná gramatika generuje jazyk L^R
- víme, že regulární jazyky jsou uzavřené na reverzi tudíž protože L^R je regulární, je i L $(=(L^R)^R)$ regulární
- takto lze získat všechny regulární jazyky

Lineární gramatiky (a jazyky)

Můžeme levě a právě lineární pravidla používat najednou?

Další zobecnění - *gramatika je lineární*, jestliže má pouze pravidla tvaru $X \to uYv$, $X \to w$, $X,Y \in V_N$, $u,v,w \in V_T^*$ (na pravé straně vždy maximálně jeden neterminál).

Lineární jazyky jsou právě jazyky generované lineárními gramatikami.

Zřejmě: regulární jazyky $\subseteq$ lineární jazyky

Platí také: regulární jazyky $\subseteq$ lineární jazyky?

NE!

$\{0^n1^n \mid n \geq 1\}$ není regulární jazyk, ale je lineární ($S \to 0S1 \mid 01$)

Pozorování:

lineární pravidla lze rozložit na levě a právě lineární pravidla

$S \to 0A$, $A \to S1$