Automaty a gramatiky

11

Roman Barták, KTIML

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

Uzávěrové vlastnosti v kostce

	RJ	BKJ	DBKJ
Sjednocení	✓	✓	✗
Průnik	✓	✗	✗
Průnik s RJ	✓	✓	✓
Doplněk	✓	✗	✓
Substituce/ homomorfismus	✓	✓	✗
Inverzní homomorfismus	✓	✓	✓

Kvocienty s regulárním jazykem

Bezkontextové jazyky jsou uzavřené na levý (pravý) kvocient s regulárním jazykem.

$R \backslash L = \{ w \mid \exists u \in R\ uw \in L \}$, $L/R = \{ u \mid \exists w \in R\ uw \in L \}$

idea:

- ZA běží na prázdno (nečte vstup) paralelně s KA
- je-li KA v koncovém stavu, můžeme začít číst vstup

formálně:

konečný automat $A_1 = (Q_1, X, \delta_1, q_1, F_1)$

zásobníkový automat $M_2 = (Q_2, X, Y, \delta_2, q_2, Z_0, F_2)$ (přijímání koncovým stavem)

definujme nový automat $M = (Q', X, Y, \delta, (q_1, q_2), Z_0, F_2)$, kde

$Q' = (Q_1 \times Q_2) \cup Q_2$ (dvojice stavů pro paralelní běh ZA a KA)

$\delta'((p,q), \lambda, Z) = \{ ((p',q'),u) \mid \exists a \in X\ p' \in \delta_1(p, a) \wedge (q',u) \in \delta_2(q, a, Z) \}$
$\cup \{ ((p,q'),u) \mid (q',u) \in \delta_2(q, \lambda, Z) \}$
$\cup \{ (q,Z) \mid p \in F_1 \}$

$\delta'(q, a, Z) = \delta_2(q, a, Z)$ $a \in X \cup \{\lambda\}$, $q \in Q_2$

zřejmě $L(M) = L(A_1) \backslash L(M_2)$

Uzávěrové vlastnosti deterministických BKJ

Rozumné programovací jazyky jsou deterministické BKJ.

Deterministické bezkontextové jazyky:

- **nejsou uzavřené na průnik,**
- **jsou uzavřené na průnik s regulárním jazykem,**
- **jsou uzavřené na inverzní homomorfismus.**

Doplněk deterministického BKJ je opět deterministický BKJ!

„prohodíme koncové a nekoncové stavy“

potíže:

- **nemusí přečíst celé vstupní slovo**
  - **krok není definován (např. vyprázdnění zásobníku)**
    snadno ošetříme „podložkou“ na zásobníku
  - **cyklus (zásobník roste, zásobník pulsuje)**
    odhalíme pomocí čítače
- **po přečtení slova prochází koncové a nekoncové stavy**
  stačí si pamatovat, zda prošel koncovým stavem

Uzávěrové vlastnosti DBKJ v praxi

DBKJ nejsou uzavřené na sjednocení (BKJ ano)!

Příklad:

$L = \{a^i b^j c^k \mid i \neq j \vee j \neq k \vee i \neq k\}$ je BKJ, ale není DBKJ

sporem: nechť L je DBKJ

potom -L (doplněk) je DBKJ

$-L \cap a^*b^*c^* = \{a^i b^j c^k \mid i=j=k\}$ je DBKJ - SPOR

DBKJ nejsou uzavřené na homomorfismus (BKJ ano)!

Příklad:

$L_1 = \{a^i b^j c^k \mid i=j\}$ je DBKJ

$L_2 = \{a^i b^j c^k \mid j=k\}$ je DBKJ

$0L_1 \cup 1L_2$ je DBKJ, $1L_1 \cup 1L_2$ není DBKJ

položme $h(0) = 1$

$h(x) = x$ pro ostatní symboly

$h(0L_1 \cup 1L_2) = 1L_1 \cup 1L_2$

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

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

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

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

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

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

Kontextové gramatiky

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

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

Poznámky:

- neterminál X se přepisuje na w pouze v kontextu α a β
- pravidlo $S \rightarrow \lambda$ slouží pouze pro přidání λ do jazyka

Příklad:

$L = \{a^n b^n c^n \mid n \geq 1\}$ je kontextový jazyk (není BKJ)

$S \rightarrow aSBC \mid abC$

$CB \rightarrow BC$ pozor, není kontextové pravidlo!

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

Separované gramatiky

Gramatika je separovaná, pokud obsahuje pouze pravidla tvaru $\alpha \rightarrow \beta$, kde:
buď $\alpha,\beta \in V^+_N$ (neprázdné posloupnosti neterminálů)
nebo $\alpha \in V_N$ a $\beta \in V_T \cup \{\lambda\}$.

Lemma: Ke každé gramatice G lze sestrojit ekvivalentní separovanou gramatiku G‘.

Důkaz:

nechť $G = (V_N, V_T, S, P)$

pro každý terminál $x \in V_T$ zavedeme nový neterminál X‘

v pravidlech z P nahradíme terminály odpovídajícími neterminály a přidáme pravidla $X‘ \rightarrow x$

$G‘ = (V_N \cup V‘_T, V_T, S, P‘ \cup \{X‘ \rightarrow x \mid x \in V_T\})$

zřejmě L(G) = L(G’)

Od monotonie ke kontextovosti

Gramatika je **monotónní** (nevypouštějící), jestliže pro každé pravidlo $(u \to v) \in P$ platí $|u| \le |v|$.

Monotónní gramatiky slovo v průběhu generování nezkracují.

Věta: **Ke každé monotónní gramatice lze nalézt ekvivalentní gramatika kontextovou.**

Důkaz:

nejprve převedeme gramatiku na separovanou
tím se monotonie neporuší (+ pravidla X→x jsou kontextová)

zbývají pravidla $A_1 \dots A_m \to B_1 \dots B_n$ (kde $m \le n$)

převedeme na kontextová pravidla s novými neterminály C

$A_1A_2 \dots A_m \to C_1A_2 \dots A_m$
$C_1A_2 \dots A_m \to C_1C_2 \dots A_m$ …
$C_1 \dots C_{m-1}A_m \to C_1 \dots C_{m-1}C_m$
$C_1 \dots C_m \to B_1 \dots C_m$
$B_1C_2 \dots C_m \to B_1B_2 \dots C_m$ …
$B_1 \dots B_{m-1}C_m \to B_1 \dots B_{m-1}B_m \dots B_n$

Příklad kontextového jazyka

$L = \{a^i b^j c^k \mid 1 \le i \le j \le k\}$ je kontextový jazyk (není BKJ)

$S \to aSBC \mid aBC$	*generování symbolů a*
$B \to BBC$	*množení symbolů B*
$C \to CC$	*množení symbolů C*
$CB \to BC$	*uspořádání symbolů B a C*
$aB \to ab$	*začátek přepisu B na b*
$bB \to bb$	*pokračování přepisu B na b*
$bC \to bc$	*začátek přepisu C na c*
$cC \to cc$	*pokračování přepisu C na c*

$CB \to BC$ není kontextové pravidlo, nahradíme ho:

$CB \to XB, XB \to XY, XY \to BY, BY \to BC$

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 $\mathcal{L}_3$)

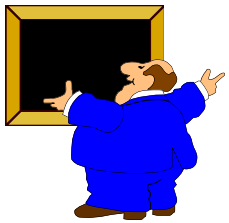
pouze pravidla ve tvaru $X \to wY$, $X \to w$, $X,Y \in V_N$, $w \in V_T^*$

Turingovy stroje - historie a motivace

1931 - 1936 pokusy o formalizaci pojmu algoritmu
Gödel, Kleene, Church, Turing

Turingův stroj

zachycení práce matematika
- (nekonečná) tabule
  - lze z ní číst a lze na ni psát
- mozek (řídící jednotka)



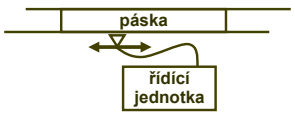
Formalizace TS:
- místo tabule oboustranně nekonečná páska
- místo křídy čtecí a zapisovací hlava, kterou lze posouvat
- místo mozku konečná řídící jednotka (jako u ZA)

Další formalizace:
- λ-kalkul, částečně rekurzivní funkce, RAM

Definice Turingova stroje

Turingovým strojem nazýváme pětici $T=(Q,X,\delta,q_0,F)$, kde

- Q - neprázdná konečná množina stavů
- X - neprázdná konečná množina symbolů obsahuje symbol ε pro prázdné políčko
- δ - přechodová funkce $\delta : (Q-F)\times X \rightarrow Q\times X\times \{-1,0,1\}$ popisuje změnu stavu, zápis na pásku a posun hlavy
- $q_0 \in Q$ - počáteční stav
- $F \subseteq Q$ - množina koncových stavů



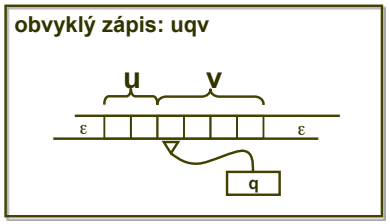
1) výpočet začíná ve stavu q_0
2) v každém taktu dojde
   - ke změně stavu
   - k přepisu políčka na pásce
   - k posunu hlavy
3) výpočet končí, když není definována žádná instrukce (speciálně platí pro koncové stavy)

Turingovy stroje - konfigurace a modifikace

Konfigurace Turingova stroje je souhrn údajů přesně popisující stav výpočtu. Obsahuje:

- nejmenší souvislou část pásky, která obsahuje
  - všechny neprázdné buňky
  - čtenou buňku
- vnitřní stav
- polohu čtené buňky (hlavy)

TS postupně přepracovává konfigurace.



Modifikace Turingova stroje:

- více pásek, více hlav
- jednostranná páska
- omezené činnosti v taktu
- omezený počet stavů, omezená abeceda
- dva zásobníky

Příklad Turingova stroje

Navrhněte Turingův stroj převádějící konfiguraci q_0w na q_Fw^R, kde $w\in\{a_1,\ldots,a_n\}^*$ (tj. obrácení slova).

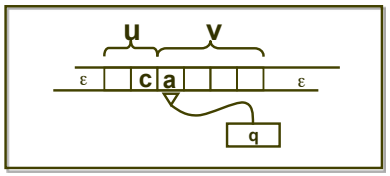
$q_0,\varepsilon \rightarrow q_F,\varepsilon,0$	prázdné slovo
$q_0,a_i \rightarrow q_{i,r},a'_i,+1$	přečte písmeno, pamatuje si ve stavu
$q_0,a'_i \rightarrow q_R,a'_i,+1$	konec (slovo sudé délky)
$q_{i,r},a_j \rightarrow q_{i,r},a_j,+1$	běží doprava
$q_{i,r},\varepsilon \rightarrow q_{i,w},\varepsilon,-1$	na konci se otočí
$q_{i,r},a'_j \rightarrow q_{i,w},a'_j,-1$	“
$q_{i,w},a_j \rightarrow q_{j,l},a'_i,-1$	vymění písmena
$q_{i,w},a'_i \rightarrow q_R,a'_i,+1$	konec (slovo liché délky)
$q_{i,l},a_j \rightarrow q_{i,l},a_j,-1$	a běží zpět (doleva)
$q_{i,l},a'_j \rightarrow q_0,a'_i,+1$	na zarážce uloží písmeno a začne znova
$q_R,a'_j \rightarrow q_R,a'_j,+1$	běží doprava
$q_R,\varepsilon \rightarrow q_C,\varepsilon,-1$	na konci se otočí
$q_C,a'_j \rightarrow q_C,a_j,-1$	při běhu doleva ruší označení
$q_C,\varepsilon \rightarrow q_F,\varepsilon,+1$	slovo je obráceno

Výpočet Turingova stroje

Turingovým strojem nazýváme pětici $T=(Q,X,\delta,q_0,F)$

- prázdné políčko ε

Konfigurace TS popisuje aktuální stav výpočtu - uqv.



Krok výpočtu (přímá změna konfigurace): $uqv \vdash wpz$

$v=av'$, $w=u$, $z=bv'$	$q,a \rightarrow p,b,0$
$v=av'$, $w=ub$, $z=v'$	$q,a \rightarrow p,b,+1$
$v=av'$, $u=wc$, $z=cbv'$	$q,a \rightarrow p,b,-1$

Poznámky:

- technicky je potřeba ošetřit případy, kdy $v=\lambda$ nebo $u=\lambda$
- s *u* a *v* lze pracovat jako se dvěma zásobníky

Výpočet je posloupnost přímých kroků $uqv \vdash^* wpz$

Turingovy stroje a jazyky

Slovo w je přijímáno Turingovým strojem T, pokud
$q_0 w \vdash\!\!\!\!-^* upv, p \in F$
někdy je na konci výpočtu vyžadováno smazání pásky ($q_0 w \vdash\!\!\!\!-^* \lambda p \varepsilon$)

Jazyk přijímaný Turingovým strojem T
$L(T) = \{w \mid w \in (X-\{\varepsilon\})^* \wedge q_0 w \vdash\!\!\!\!-^* upv, p \in F\}$.

Jazyk L nazveme rekurzivně spočetným, pokud je přijímán nějakým Turingovým strojem T (L=L(T)).

***Příklad*: $\{a^{2^n}\}$**

q_0,ε	$\rightarrow q_F,\varepsilon,0$	prázdné slovo (konec výpočtu)
q_0,a	$\rightarrow q_1,a,+1$	zvětší čítač (2k+1 symbolů)
q_1,a	$\rightarrow q_0,a,+1$	nuluje čítač (2k symbolů)

Od Turingova stroje ke gramatice

Každý rekurzivně spočetný jazyk je typu 0.

Důkaz:

pro Turingův stroj T najdeme gramatiku G tak, že L(T)=L(G)
- gramatika nejdříve vygeneruje pásku stroje + kopii slova
- potom simuluje výpočet (stavy jsou součástí slova)
- v koncovém stavu smažeme pásku, necháme pouze kopii slova

$w\ \underline{\varepsilon}^n\ \underline{w}^R\ q_0\ \underline{\varepsilon}^n$ ($\underline{\varepsilon}^n$ představují volný prostor pro výpočet)

I) $S \rightarrow D\ Q_0\ E$
$D \rightarrow x\ D\ \underline{x} \mid E$ generuje slovo a jeho reverzní kopii pro výpočet
$E \rightarrow \underline{\varepsilon}\ E \mid \underline{\varepsilon}$ generuje volný prostor pro výpočet

II) $\underline{X}\ P\ \underline{Y} \rightarrow \underline{X'}\ Q\ \underline{Y}$ pro $\delta(p,x)=(q,x',0)$
$\underline{X}\ P\ \underline{Y} \rightarrow Q\ \underline{X'}\ \underline{Y}$ pro $\delta(p,x)=(q,x',+1)$
$\underline{X}\ P\ \underline{Y} \rightarrow \underline{X'}\ \underline{Y}\ Q$ pro $\delta(p,x)=(q,x',-1)$

III) $P \rightarrow C$ pro $p \in F$
$C\ \underline{A} \rightarrow C$ mazání pásky
$\underline{A}\ C \rightarrow C$ mazání pásky
$C \rightarrow \lambda$ konec výpočtu

Od Turingova stroje ke gramatice - pokračování

Ještě L(T) = L(G)?

w∈L(T)

existuje konečný výpočet stroje T (konečný prostor)
gramatika vygeneruje dostatečně velký prostor pro výpočet
simulujeme výpočet a smažeme dvojníky

w∈L(G)

pravidla v derivaci nemusí být v pořadí, jakém chceme
derivaci můžeme přeuspořádat tak, že pořadí je I, II, III
podtržené symboly smazány, tj. vygenerován koncový stav

Příklad:

$\delta(q_0,\varepsilon) = (q_F,\varepsilon,0)$
$\delta(q_0,a) = (q_1,a,+1)$
$\delta(q_1,a) = (q_0,a,+1)$

Gramatika po zjednodušení

$S \rightarrow D\ q_0$
$D \rightarrow a\ D\ \underline{a} \mid \underline{\varepsilon}$
$\underline{\varepsilon}\ q_0 \rightarrow C$
$\underline{a}\ q_0 \rightarrow q_1\ \underline{a}$
$\underline{a}\ q_1 \rightarrow q_0\ \underline{a}$
$C\ \underline{a} \rightarrow C$
$C \rightarrow \lambda$

Od gramatik k Turingově stroji

Každý jazyk typu 0 je rekurzivně spočetný.

***Důkaz* (neformálně):**

idea: Turingův stroj postupně generuje všechny derivace
derivaci $S \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n = w$ kódujeme jako slovo $\#S\#w_1\#\ldots\#w\#$
TS postupně generuje všechna slova $\#S\#\ w_1\#\ldots\#w_k\#$
pokud $w_n=w$, výpočet končí
jinak, TS generuje další derivaci
- umíme udělat TS, který přijímá slova #u#v#, kde $u \Rightarrow v$
- umíme udělat TS, který přijímá slova $\#w_1\#\ldots\#w_k\#$, kde $w_1 \Rightarrow^* w_k$
- umíme udělat TS postupně generující všechna slova
- stroje spojíme do „while“ cyklu