Automaty a gramatiky

12

Roman Barták, KTIML

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

Co bylo minule

Uzávěrové vlastnosti bezkontextových jazyků
- zrcadlový obraz, zřetězení, (pozitivní) iterace

Uzávěrové vlastnosti obecně
- homo, homo^{-1}, průnik s RJ $\Rightarrow$ kvocienty s RJ
- sub, homo, průnik s RJ $\Rightarrow$ homo^{-1}

Uzávěrové vlastnosti deterministických BK jazyků
- jsou uzavřené na průnik s RJ, homo^{-1}, doplněk!
- nejsou uzavřené na průnik, sjednocení!

Charakteristika BKJ Dyckovými jazyky

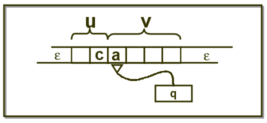
Kontextové gramatiky
- separované a monotónní gramatiky

Turingovy stroje

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
$q_0w \vdash^* upv$, $p \in F$

někdy je na konci výpočtu vyžadováno smazání pásky ($q_0w \vdash^* \lambda p\varepsilon$)

Jazyk přijímaný Turingovým strojem T
$L(T) = \{w \mid w \in (X-\{\varepsilon\})^* \ \&\ q_0w \vdash^* upv,\ p \in F\}$.

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

Příklad: $\{a^{2n}\}$

$q_0,\varepsilon \rightarrow q_F,\varepsilon,0$	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

$w\, \underline{\varepsilon}^n\, \underline{w}^R\, q_0\, \varepsilon^n$ ($\underline{\varepsilon}^n$ 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

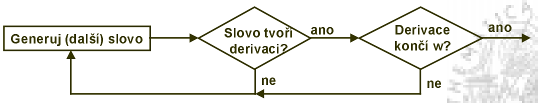
derivaci $S \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n=w$ kódujeme jako slovo $\#S\#w_1\#\ldots\#w\#$

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



Nedeterministické Turingovy stroje

Nedeterministickým Turingovým strojem nazýváme pětici $T=(Q,X,\delta,q_0,F)$, kde Q,X,q_0,F jsou jako u TS a $\delta : (Q-F)\times X \rightarrow P(Q\times X\times \{-1,0,1\})$.

Slovo w je přijímáno nedeterministickým Turingovým strojem T, pokud existuje nějaký výpočet $q_0w \mid\!\!\text{—}^* \text{upv}$, $p\in F$.

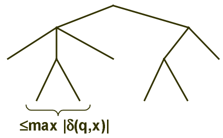
Tvrzení: **N Turingovy stroje přijímají právě rekurzivně spočetné jazyky.**

Důkaz (neformálně):

Ukážeme, že výpočty NTS lze modelovat pomocí TS.

Pozor! Nelze použít podmnožinovou konstrukci (kvůli pásce)!

TS modeluje všechny výpočty NTS prohledáváním do šířky



- Na pásce můžeme mít všechny konfigurace v hloubce *k* (páska je nekonečná), nebo
- můžeme generovat „popis“ výpočtu (posloupnost pravidel) a vždy k němu dopočítat výslednou konfiguraci

Lineárně omezené automaty

Ještě potřebujeme **ekvivalent pro kontextové gramatiky**.

Připomeňme, že kontextovou gramatiku dostaneme z libovolné monotónní gramatiky

Lineárně omezený automat (LOA) je nedeterministický TS, kde na pásce je označen levý a pravý konec (l, r). Tyto symboly nelze při výpočtu přepsat a nesmí se jít nalevo od l a napravo od r.

Slovo w je přijímáno lineárně omezeným automatem, pokud $q_0\underline{l}w\underline{r} \mid\!\!\text{—}^* \text{upv}$, $p\in F$.

Prostor výpočtu je definován vstupním slovem a automat při jeho přijímání nesmí překročit jeho délku

u monotónních (kontextových) derivací to není problém: žádné slovo v derivaci není delší než výstupní slovo

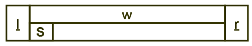
Od kontextových jazyků k LOA

Každý kontextový jazyk lze přijímat pomocí LOA.

Důkaz:

derivaci gramatiky budeme simulovat pomocí LOA

použijeme pásku se dvěma stopami (větší abeceda)

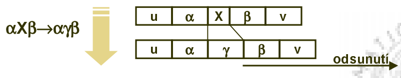


1) slovo w dáme do horní stopy a na začátek dolní stopy dáme S

2) přepisujeme slovo ve druhé stopě podle pravidel G

2.1) nedeterministicky vybereme část k přepsání

2.2) provedeme přepsání dle pravidla (pravá část se odsune)



3) pokud jsou ve druhé stopě samé terminály, porovnáme ji s první stopou (slovo přijmeme či zamítneme)

Od LOA ke kontextovým jazykům

LOA přijímají pouze kontextové jazyky.

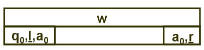
Důkaz:

potřebujeme převést LOA na monotónní gramatiku tj. gramatika nesmí generovat nic navíc!

výpočet ukryjeme do „dvoustopých“ neterminálů

1) generuj slovo ve tvaru $(a_0,[q_0,\underline{l},a_0]),(a_1,a_1),\ldots,(a_n,[a_n,\underline{r}])$

stav a okraje musíme ukrýt to neterminálů



2) simuluj práci LOA ve „druhé“ stopě (stejně jako u TS)

3) pokud je stav koncový, smaž „druhou“ stopu

speciálně je potřeba ošetřit přijímání prázdného slova

pokud LOA přijímá λ, přidáme speciální startovací pravidlo

Rekurzivní jazyky

Co se stane, když TS nepřijímá nějaké slovo?

a) výpočet skončí v nekoncovém stavu

b) výpočet nikdy neskončí

protože výpočet neskončil, nevíme, zda slovo do jazyka patří

Říkáme, že **TS T rozhoduje jazyk L**, pokud L=L(T) a pro každé slovo w je výpočet stroje nad w konečný.

Jazyky rozhodnutelné TS nazýváme **rekurzivní jazyky**.

Věta (Postova): **Jazyk L je rekurzivní, právě když L a doplněk L jsou rekurzivně spočetné.**

Důkaz:

máme Turingovy stroje T_1 pro L a T_2 pro -L

pro dané slovo w naráz simulujeme výpočet T_1 i T_2

T_1 a T_2 rozpoznávají komplementární jazyky, tedy po konečném počtu kroků víme zda $w\in L$

Problém zastavení TS

Existuje rekurzivně spočetný jazyk, který není rekurzivní?
ANO

Problém zastavení Turingova stroje (halting problem) je algoritmicky nerozhodnutelný.
Neexistuje algoritmus, který by pro daný kód TS a daný vstup rozhodl, zda se TS zastaví.

Důkaz **(neformálně):**

- **vychází z existence univerzálního TS (Turingův stroj, který simuluje výpočet jiného TS nad daným vstupem)**
  U(T,X) = T(X) T je kód stroje, X jsou vstupní data
- **můžeme udělat stroj P(X), který se na datech X zastaví právě když U(X,X) se nezastaví**
- **U(P,P) vede ke sporu: P(P)↓ ⇔ U(P,P)↑ ⇔ P(P)↑ (diagonální metoda)**

Postův korespondenční problém

Postovým korespondenčním problémem (PKP) nazýváme konečný seznam dvojic neprázdných slov $[u_1,v_1],\dots,[u_n,v_n]$.

Říkáme, že **Postův korespondenční problém má řešení**, pokud existují indexy $i_1,\dots i_k$ tak, že $1\le i_j \le n$ a $u_{i_1}u_{i_2}\dots u_{i_k} = v_{i_1}v_{i_2}\dots v_{i_k}$

Říkáme, že **Postův korespondenční problém má iniciální řešení**, pokud existují indexy $i_1,\dots i_k$ tak, že $1\le i_j \le n$ a $u_1u_{i_1}u_{i_2}\dots u_{i_k} = v_1v_{i_1}v_{i_2}\dots v_{i_k}$

Věta: **PKP je algoritmicky rozhodnutelný, právě když je algoritmicky rozhodnutelné zda PKP má iniciální řešení.**

Důkaz:

PKP s iniciálním řešením ⇒ PKP (stačí vyzkoušet všechny začátky)

PKP ⇒ PKP s iniciálním řešením

značení $\underline{a_1a_2\dots a_n}^\bullet = a_1\bullet a_2\bullet \dots a_n\bullet$ ${}^\bullet\underline{a_1a_2\dots a_n} = \bullet a_1\bullet a_2\bullet \dots a_n$

$x_1 = \bullet\,\underline{u_1}^\bullet$, $x_{j+1} = \underline{u_j}^\bullet$, $x_{n+2} = \blacklozenge$

$y_1 = {}^\bullet\underline{v_1}$, $y_{j+1} = {}^\bullet\underline{v_j}$, $y_{n+2} = \bullet\,\blacklozenge$

PKP s u,v má iniciální řešení právě když PKP s x,y má řešení

Algoritmická nerozhodnutelnost PKP

Existence iniciálního řešení PKP není algoritmicky rozhodnutelná.

Důkaz:

výpočet TS pro slovo w převedeme na PKP

u	v	
+	$+\varepsilon q_0 w+$	
x	x	$x\in X$
+	+	
px	qy	$\delta(p,x)=(q,y,0)$
p+	qy+	$\delta(p,\varepsilon)=(q,y,0)$
px	yq	$\delta(p,x)=(q,y,+1)$
p+	yq+	$\delta(p,\varepsilon)=(q,y,+1)$
zpx	qzy	$\delta(p,x)=(q,y,-1)$
zp+	qzy+	$\delta(p,\varepsilon)=(q,y,-1)$
+px	$+q\varepsilon y$	$\delta(p,x)=(q,y,-1)$

u	v	
xqy	q	$q\in F$
xq+	q+	
+qy	+q	
q++	+	

PKP má iniciální řešení ⇔ TS se zastaví nad w

u: $+\,K_0 + \;\dots\; K_{n-1} + K_n + \;\dots\; + q + +$
v: $+\,K_0 + K_1 + \;\dots\; + K_n + uqv + \;\dots\; + q + +$

výpočet *mazání*

Algoritmicky nerozhodnutelné problémy

Pro bezkontextové gramatiky G_1,G_2 je algoritmicky nerozhodnutelné zda $L(G_1)\cap L(G_2)=\emptyset$.

Důkaz: převedeme PKP na daný problém

máme PKP $[u_1,v_1],\dots,[u_n,v_n]$

zvolíme nové terminály $a_1,\dots,a_n$ pro kódy indexů

G_1: $S\to u_iSa_i \mid u_ia_i$ generuje slova $u_{i_1}\dots u_{i_k}a_{i_k}\dots a_{i_1}$

G_2: $S\to v_iSa_i \mid v_ia_i$ generuje slova $v_{i_1}\dots v_{i_k}a_{i_k}\dots a_{i_1}$

PKP má řešení právě když $L(G_1)\cap L(G_2)\neq\emptyset$

u-část = v-část + složky a_i zajišťují stejné pořadí

Je algoritmicky nerozhodnutelné, zda je bezkontextová gramatika víceznačná.

Důkaz:

$S \to S_1 \mid S_2$

$S_1 \to u_iS_1a_i \mid u_ia_i$

$S_2 \to v_iS_2a_i \mid v_ia_i$

PKP má řešení právě když je gramatika víceznačná

Další algoritmicky nerozhodnutelné problémy

Je algoritmicky nerozhodnutelné, zda L(G)=X* pro BKG G.

Důkaz:

G_1: $S\to u_iSa_i \mid u_ia_i$ generuje slova $u_{i_1}\dots u_{i_k}a_{i_k}\dots a_{i_1}$

G_2: $S\to v_iSa_i \mid v_ia_i$ generuje slova $v_{i_1}\dots v_{i_k}a_{i_k}\dots a_{i_1}$

jazyky $L(G_1)$, $L(G_2)$ jsou deterministické, tedy $-L(G_1)$ a $-L(G_2)$ jsou deterministické BKJ a $-L(G_1)\cup -L(G_2)$ je BKJ

máme BKG G takovou, že $L(G) = -L(G_1)\cup -L(G_2)$

PKP má řešení ⇔ $L(G_1)\cap L(G_2)\neq\emptyset$ ⇔ $L(G) = -L(G_1)\cup -L(G_2) \neq X^*$

Poznámka: **$L(G) = \emptyset$ je algoritmicky rozhodnutelné.**

Důsledky: **Nelze algoritmicky rozhodnout**, zda

L(G) = R, pro BKG G a regulární jazyk R (důkaz: za R zvolme X*)

$R\subseteq L(G)$, pro BKG G a regulární jazyk R (důkaz: za R zvolme X*)

$L(G_1) = L(G_2)$, pro BKG G_1 a G_2 (důkaz: nechť G_1 generuje X*)

$L(G_1)\subseteq L(G_2)$, pro BKG G_1 a G_2 (důkaz: nechť G_1 generuje X*)

Poznámka: **$L(G)\subseteq R$ je algoritmicky rozhodnutelné**

$L(G)\subseteq R \Leftrightarrow L(G)\cap -R = \emptyset$ + $(L(G)\cap -R)$ je BKJ

Shrnutí

popis nekonečných objektů konečnými prostředky

regulární jazyky
konečné automaty (NKA,2KA)
Nerode, Kleene, pumpování

bezkontextové jazyky
zásobníkové automaty (DZA)
Dyckovy jazyky, pumpování

kontextové jazyky
lineárně omezené automaty
monotonie

rekurzivně spočetné jazyky
Turingovy stroje
algoritmická nerozhodnutelnost

použití nejen pro práci s jazyky!