Automaty a gramatiky

12

Roman Barták, KTIML

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

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

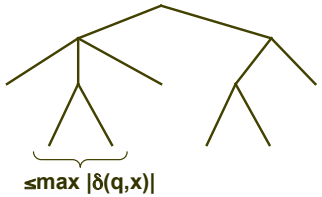
Tvrzení: **N Turingovy stroje přijímají právě rekurzivně spočetné jazyky.**

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 $q_0lwr \vdash^* upv$, $p\in F$.

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

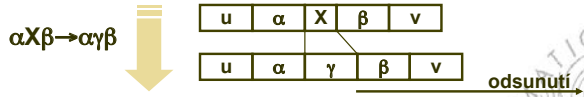
Každý kontextový jazyk lze přijímat pomocí LOA.

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

w		
$q_0,\underline{l},a_0$		$a_0,\underline{r}$

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

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 $X \rightarrow wY$, $X \rightarrow w$, $X,Y \in V_N$, $w \in V_T^*$

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: $P(P)\downarrow \Leftrightarrow U(P,P)\uparrow \Leftrightarrow P(P)\uparrow$ (diagonální metoda)

Postův korespondenční problém

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

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

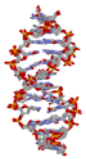
Říkáme, že **Postův korespondenční problém má iniciální řešení**, pokud existují indexy $i_1,\ldots i_k$ tak, že $1\le i_j \le n$ a $u_1u_{i_1}u_{i_2}\ldots u_{i_k} = v_1v_{i_1}v_{i_2}\ldots v_{i_k}$

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 $\underline{a_1a_2\ldots a_n}^{\bullet} = a_1\bullet a_2\bullet \ldots a_n\bullet$ $\quad {}^{\bullet}\underline{a_1a_2\ldots a_n} = \bullet a_1\bullet a_2\bullet \ldots a_n$

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

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

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: $+\,K_0\,+ \quad \ldots \quad K_{n-1}+K_n+ \quad \ldots \quad +q++$

v: $+\,K_0+K_1+ \quad \ldots \quad +K_n+uqv+ \quad \ldots \quad +q++$

výpočet *mazání*

Algoritmická rozhodnutelnost u BKJ

Pro bezkontextové jazyky je algoritmicky rozhodnutelné, zda dané slovo patří či nepatří do jazyka.

- umíme $\lambda \in L(G)$ $(S\Rightarrow^*\lambda)$
- pro ostatní slova použijeme ChNF ($X \to YZ$, $X \to a$)
  - v každé derivaci se délka slova zvětšuje nebo roste počet terminálních symbolů (tj. v derivaci není cyklus!)
  - na terminální slovo délky n použijeme právě (2n-1) pravidel
  - derivací pro všechna slova délky n je konečně
  - můžeme postupně vyzkoušet všechny derivace vedoucí ke slovům dané délky, například prohledáváním do hloubky

Pro bezkontextové jazyky je algoritmicky rozhodnutelné, zda je jazyk prázdný.

umíme zjistit, zda z S lze generovat terminální slovo (algoritmus redukce)

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 $[u_1,v_1],\ldots,[u_n,v_n]$

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

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

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

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: $S \rightarrow u_i S a_i \mid u_i a_i$ generuje slova $u_{i_1} \ldots u_{i_k} a_{i_k} \ldots a_{i_1}$

G_2: $S \rightarrow v_i S a_i \mid v_i a_i$ generuje slova $v_{i_1} \ldots v_{i_k} a_{i_k} \ldots a_{i_1}$

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 $\Leftrightarrow L(G_1) \cap L(G_2) \neq \emptyset \Leftrightarrow L(G) = -L(G_1) \cup -L(G_2) \neq X^*$

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!