Automaty a gramatiky 5

Roman Barták, KTIML

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

Co bylo minule

Regulární jazyk jako sloučenina

- elementární jazyky (prázdný, jednopísmenný) + operace sjednocení, zřetězení, iterace
- Kleeneova věta
- popis jazyků pomocí regulárních výrazů
- převod regulárního výrazu na automat

Dvousměrné (dvoucestné) automaty

- automatem řízený pohyb hlavy
- na pásku nic nepíšeme!
- dva způsoby převodu na konečný automat

Automaty s výstupem (motivace)

... aneb jak zaznamenat výpočet automatu?

Dosud jediná zpráva z automatu - jsme v přijímajícím stavu.
Můžeme z konečného automatu získat více informací?
Můžeme zaznamenat trasu výpočtu?

1) indikace stavů (všech, nejen koncových)
   v každé chvíli víme, kde se automat nachází
   Příklad: různé (regulární) čítače

2) indikace přechodů
   po přečtení každého symbolu víme, co automat udělal
   Příklad: (regulární) překlad slov

Automat až není tak docela černá skříňka.

Mooreův stroj

Mooreovým (sekvenčním) strojem nazýváme šestici
$A = (Q,X,Y,\delta,\mu,q_0)$ resp. pětici $A = (Q,X,Y,\delta,\mu)$, kde:

- Q - konečná neprázdná množina stavů (*stavový prostor*)
- X - konečná neprázdná množina symbolů (*vstupní abeceda*)
- Y - konečná neprázdná množina symbolů (*výstupní abeceda*)
- δ - zobrazení $Q \times X \rightarrow Q$ (*přechodová funkce*)
- μ - zobrazení $Q \rightarrow Y$ (*značkovací funkce*)
- $q_0 \in Q$ (*počáteční stav*)

Poznámky:

- někdy nás nezajímá počáteční stav, ale jen práce automatu
- značkovací funkce umožňuje suplovat roli koncových stavů
  $F \subseteq Q$ nahradíme značkovací funkcí $\mu : Q \rightarrow \{0,1\}$ takto:
  $\mu(q) = 0$, pokud $q \notin F$
  $= 1$, pokud $q \in F$

Příklad Mooreova stroje

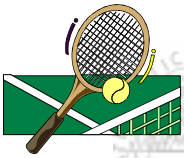
Navrhněte automat počítající tenisové skóre.

Vstupní abeceda: ID hráče, který uhrál bod

Výstupní abeceda/stavy: skóre (tj. $Q=Y$ a $\mu(q)=q$)

Stav/výstup	A	B
00:00	15:00	00:15
15:00	30:00	15:15
15:15	30:15	15:30
00:15	15:15	00:30
30:00	40:00	30:15
30:15	40:15	30:30
30:30	40:30	30:40
15:30	30:30	15:40
00:30	15:30	00:40
40:00	A	40:15
40:15	A	40:30
40:30	A	shoda
30:40	shoda	B
15:40	30:40	B
00:40	15:30	B

Stav/výstup	A	B
shoda	A:40	40:A
A:40	A	shoda
40:A	shoda	B
A	15:00	00:15
B	15:00	00:15



Mealyho stroj

Mealyho (sekvenčním) strojem nazýváme šestici
$A = (Q,X,Y,\delta,\lambda,q_0)$ resp. pětici $A = (Q,X,Y,\delta,\lambda)$, kde:

- Q - konečná neprázdná množina stavů (*stavový prostor*)
- X - konečná neprázdná množina symbolů (*vstupní abeceda*)
- Y - konečná neprázdná množina symbolů (*výstupní abeceda*)
- δ - zobrazení $Q \times X \rightarrow Q$ (*přechodová funkce*)
- λ - zobrazení $Q \times X \rightarrow Y$ (*výstupní funkce*)
- $q_0 \in Q$ (*počáteční stav*)

Poznámka:

výstup je určen stavem a vstupním symbolem
tj. Mealyho stroj je obecnějším prostředkem než stroj Mooreův
značkovací funkci $\mu : Q \rightarrow Y$ lze nahradit výstupní funkcí $\lambda: Q \times X \rightarrow Y$
například takto:
$\forall x \in X \ \lambda(q,x) = \mu(q)$

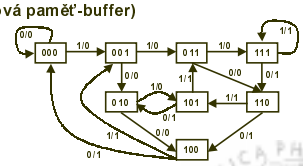
Příklad Mealyho stroje

Navrhněte automat, který dělí vstupní slovo v binárním tvaru číslem 8 (celočíselně).

Realizace:

- posun o tři bity doprava (1101010 → 0001101)
- potřebujeme si pamatovat poslední trojici bitů (vlastně dynamická tříbitová paměť-buffer)

Stav\symbol	0	1
000	000/0	001/0
001	010/0	011/0
010	100/0	101/0
011	110/0	111/0
100	000/1	001/1
101	010/1	011/1
110	100/1	101/1
111	110/1	111/1



Vadí nám, když nevíme, kde automat startuje?
NE - po třech symbolech začne počítat správně

Výstup sekvenčních strojů

slovo ve vstupní abecedě → slovo ve výstupní abecedě

Mooreův stroj

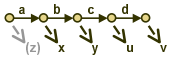
značkovací funkce $\mu: Q \to Y$

$\mu^*: Q \times X^* \to Y^*$

$\mu^*(q,\lambda) = \lambda$ (někdy $\mu^*(q,\lambda) = \mu(q)$)

$\mu^*(q,wx) = \mu^*(q,w) . \mu(\delta^*(q,wx))$

Příklad: $\mu^*(00{:}00, AABA) = (00{:}00\ .)\ 15{:}00\ .\ 30{:}00\ .\ 30{:}15\ .\ 40{:}15$



Mealyho stroj

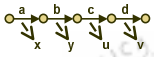
výstupní funkce $\lambda: Q \times X \to Y$

$\lambda^*: Q \times X^* \to Y^*$

$\lambda^*(q,\lambda) = \lambda$

$\lambda^*(q,wx) = \lambda^*(q,w) . \lambda(\delta^*(q,w),x)$

Příklad: $\mu^*(,000',1101010) = 0001101$



Převod Mooreova stroje na Mealyho

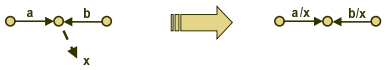
Nechť $A = (Q,X,Y,\delta,\mu,q_0)$ je Mooreův stroj.

Umíme najít Mealyho stroj B tak, že $\forall q,w\ \mu^*(q,w) = \lambda^*(q,w)$?

ANO!

položme $B = (Q,X,Y,\delta,\lambda,q_0)$, kde $\lambda(q,x) = \mu(\delta(q,x))$

tj. λ vrací značku stavu, do kterého přejdeme



Příklad:

stav	0	1	výstup
a	a	b	0
b	b	c	1
c	c	a	2

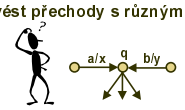
stav	0	1
a	a/0	b/1
b	b/1	c/2
c	c/2	a/0

Převod Mealyho stroje na Mooreův

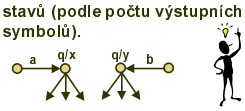
Nechť $A = (Q,X,Y,\delta,\lambda,q_0)$, je Mealyho stroj.

Sestrojme Mooreův stroj B tak, že $\forall q,w\ \lambda^*(q,w) = \mu^*(q,w)$.

Problém: do jednoho stavu mohou vést přechody s různým výstupem!



Řešení: stav rozdělíme na více stavů (podle počtu výstupních symbolů).



Teď už je to jednoduché! $B = (Q \times Y, X, Y, \delta', \mu', (q_0,\_))$, kde $\delta'((q,y),x) = (\delta(q,x), \lambda(q,x))$ a $\mu((q,y)) = y$

Příklad:

stav	0	1
a	a/0	b/0
b	a/1	b/1

stav	0	1	výstup
(a,0)	(a,0)	(b,0)	0
(a,1)	(a,0)	(b,0)	1
(b,0)	(a,1)	(b,1)	0
(b,1)	(a,1)	(b,1)	1

„Pravidelnost" regulárních jazyků

Konečný automat kóduje pouze konečnou informaci.

Přesto můžeme rozpoznávat nekonečné jazyky!

Můžeme přijímat libovolně (ale konečně) dlouhá slova!

Pozorování:

$L = \{w \mid w \in \{0,1\}^* \ \&\ |w|_1 = 3k+1\}$

$0100\underline{11010} \in L$ potom také:

$0100\underline{11010}\underline{11010} \in L$ řetězec 01101 jsme zdvojili

$0100\underline{11010}\underline{11010}\underline{11010} \in L$ řetězec 01101 jsme ztrojili

$0100 \in L$ řetězec 01101 jsme vypustili

Lze takovouto operaci provést s každým slovem libovolného regulárního jazyka?

ANO (pokud je slovo dostatečně dlouhé)

Iterační (pumping) lemma

Nechť L je regulární jazyk. Potom existuje přirozené číslo n takové, že libovolné slovo $z \in L$, $|z| \geq n$ lze psát ve tvaru uvw, kde:
$|uv| \leq n$, $1 \leq |v|$ a pro všechna $i \geq 0$ $uv^iw \in L$.

Důkaz:

za n vezměme počet stavů příslušného automatu

při zpracování slova délky ≥n se automat nutně musí dostat do jednoho stavu dvakrát (krabičkový princip)

vezměme takový stav p, který se opakuje jako první

potom $\delta(q_0,u)=p$ poprvé jsme se dostali do p

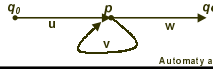
$\delta(q_0,uv)=p$ podruhé jsme se dostali do p

zřejmě $|uv| \leq n$ (pouze p se opakuje tj. maximálně n+1 stavů)

$|v| \geq 1$ (smyčka obsahuje alespoň jedno písmeno)

smyčku mezi prvním a druhým průchodem p (odpovídající slovo v) nyní můžeme vypustit nebo projít vícekrát

tj. $\forall i \geq 0$ $uv^iw \in L$



Iterační lemma a nekonečnost jazyků

Umíme algoritmicky rozhodnout, zda je regulární jazyk L nekonečný?

ANO!

jazyk L je nekonečný $\Leftrightarrow \exists u \in L$ tž. $n \leq |u| < 2n$
n je číslo z iteračního lemmatu

stačí prozkoumat všechna slova u taková, že $n \leq |u| < 2n$ (to je konečně mnoho slov)

PROČ?

- Pokud $\exists u \in L$ tž. $n \leq |u| < 2n$, potom lze slovo *u* pumpovat, čímž dostaneme nekonečně mnoho slov z jazyka L.
- Je-li jazyk L nekonečný, obsahuje slovo z takové, že $n \leq |z|$.
  - Pokud $|z| < 2n$ máme hledané slovo.
  - Jinak, podle iteračního lemmatu $z = uvw$ a $uw \in L$, tj. zkrácení
  - Platí-li stále $2n \leq |uw|$, opakuj zkracování se slovem *uw*.

  Poznámka: zkracujeme maximálně o *n* písmen, tedy interval [n,2n) nelze přeskočit!

Použití iteračního lemmatu

1) Důkaz neregulárnosti jazyka

$L = \{0^i 1^i \mid i \geq 0\}$ není regulární jazyk

sporem:

nechť L je regulární, potom lze pumpovat ($\exists n$ tž. ...)
vezměme slovo $0^n 1^n$ (to je určitě delší než n)
pumpovat můžeme pouze v části 0^n
potom, ale $0^{n+i} 1^n \notin L$ (i>0 je délka pumpované části), což je spor

2) POZOR! Iterační lemma představuje nutnou podmínku regulárnosti jazyka, nedává podmínku postačující.

Existuje jazyk, který není regulární a lze pumpovat.

$L = \{u \mid u=a^+b^ic^i \vee u=b^ic^j\}$
L není regulární (Nerodova věta)
vždy lze pumpovat první písmeno

Konečné automaty - shrnutí

Konečný automat

- jednoznačný redukovaný automat
- nedeterminismus (2^n), dvousměrný KA (n^n)
- Mealyho a Mooreův stroj

Automaty a jazyky

- regulární jazyky
- uzavřenost na množinové operace
- uzavřenost na řetězcové operace
- uzavřenost substituce

Charakteristika regulárních jazyků

- Nerodova věta (kongruence)
- Kleeneova věta (elementární jazyky a operace)
- Iterační lemma (iterace podslov, jen nutná podmínka)

Uvod 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

$V \rightarrow VV \mid (V) \mid ()$

Výraz (()())(()) je generován posloupností přepisů:
$V \rightarrow VV \rightarrow (V)V \rightarrow (VV)V \rightarrow (()V)V \rightarrow (()())V \rightarrow (()())(V) \rightarrow (()())(())$

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

$V \rightarrow T+V \mid T$
$T \rightarrow F^*T \mid F$
$F \rightarrow (V) \mid c$

Výraz c+c*c+c je generován posloupností přepisů:
$V \rightarrow T+V \rightarrow F+V \rightarrow c+V \rightarrow c+T+V \rightarrow c+F^*T+V \rightarrow c+c^*T+V \rightarrow c+c^*F+V \rightarrow c+c^*c+V \rightarrow c+c^*c+T \rightarrow c+c^*c+F \rightarrow c+c^*c+c$

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 $u,v \in V^*$ (zpravidla píšeme $u \rightarrow v$)

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 00110 ⇒ 00101 ⇒ 00011

00110 ⇒* 01100 dostaneme z 00110 ⇒ 01010 ⇒ 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
- 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:
$L(G) = \{w \mid w \in V_T^* \ \& \ S \Rightarrow^* w\}$.

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

Příklad:

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

Chomského hierarchie

Klasifikace gramatik podle tvaru přepisovacích pravidel.

gramatiky typu 0 (rekurzivně spočetné jazyky)

gramatiky v obecné formě (žádná omezení pravidel)

gramatiky typu 1 (kontextové jazyky)

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

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

gramatiky typu 2 (bezkontextové jazyky)

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)

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