Automaty a gramatiky

11

Roman Barták, KTIML

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

Co bylo minule

Pumping lemma pro bezkontextové jazyky
- $|uvwxy|>p$, $|vwx|\leq q$, $vx\neq\lambda \Rightarrow \forall i\geq 0\ uv^iwx^iy\in L$
- nutná podmínka bezkontextovosti
- nekonečnost bezkontextových jazyků

Uzávěrové vlastnosti bezkontextových jazyků
- nejsou uzavřené na průnik
- jsou uzavřené na průnik s regulárním jazykem
- jsou uzavřené na sjednocení
- nejsou uzavřené na doplněk
- jsou uzavřené na substituci a homomorfismus
- jsou uzavřené na inverzní homomorfismus
- jsou uzavřené na kvocienty s regulárním jazykem

Zrcadlový obraz, zřetězení a iterace

Bezkontextové jazyky jsou uzavřené na zrcadlový obraz, zřetězení, iteraci a pozitivní iteraci.

1) zrcadlový obraz $L^R = \{w^R \mid w\in L\}$
- G: $X \rightarrow w^R$ (obrátíme pravou stranu pravidel)
- ZA: slova do zásobníku dáváme „opačně“

2) zřetězení $L_1 . L_2$
- G: $S \rightarrow S_1S_2$ (nejprve generujeme první slovo, potom druhé)
- ZA: nejprve běží první automat, potom druhý

3) iterace $L^* = \cup_{i\geq 0} L^i$
- G: $S' \rightarrow SS' \mid \lambda$ (opakovaně spouštíme generování slov z jazyka)
- ZA: na konci výpočtu můžeme restartovat + prázdné slovo

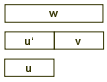
4) pozitivní iterace $L^+ = \cup_{i\geq 1} L^i$
- G: $S' \rightarrow SS' \mid S$ (opakovaně spouštíme generování slov z jazyka)
- ZA: na konci výpočtu můžeme restartovat

Uzávěrové vlastnosti obecně

Uzavřenost na substituci zajišťuje uzavřenost na sjednocení, zřetězení i iteraci.
- položme $\sigma(a) = L_1$, $\sigma(b)=L_2$
- potom $\sigma(\{a,b\}) = L_1 \cup L_2$, $\sigma(\{ab\}) = L_1 . L_2$, $\sigma(a^*) = L_1^*$

Uzavřenost na homomorfismus, inverzní homomorfismus a průnik s regulárním jazykem zajišťuje uzavřenost na kvocienty s regulárním jazykem.
- uděláme abecedu dvojníků X' pro X (původní abeceda)
- definujeme homomorfismus $h_1: (X \cup X') \rightarrow X$ takto $h_1(a) = h_1(a') = a$
  - $h_1^{-1}(w)$ - libovolné symboly jsou nahrazeny dvojníky
  - $h_1^{-1}(L) \cap X'^*R$ - dvojníci pouze na začátku následováni slovem z R
- definujme homomorfismus $h_2: (X \cup X') \rightarrow X$ takto $h_2(a) = \lambda$, $h_2(a') = a$
  - $h_2(w)$ - ze slova vyhodí původní symboly a nechá jen dvojníky

Potom:
- $L/R = h_2(h_1^{-1}(L) \cap X'^*R)$
- $R\backslash L = h_2(h_1^{-1}(L) \cap RX'^*)$



L

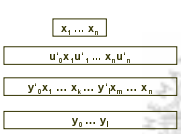
$h_1^{-1}(L) \cap X'^*R$

$h_2(h_1^{-1}(L) \cap X'^*R)$

Uzávěrové vlastnosti obecně - 2

Uzavřenost na substituci, homomorfismus a průnik s regulárním jazykem zajišťuje uzavřenost na inverzní homomorfismus.
- Je dán homomorfismus $h: X \rightarrow A^*$
- uděláme abecedu dvojníků X' pro X
- definujeme $\sigma : A^* \rightarrow P((X'\cup A)^*)$
  - $\sigma(a) = X'^*aX'^*$ (písmeno *a* „obalíme“ slovy z dvojníků)
  - $\sigma(L)$ - mezi symboly vloží slova z dvojníkových symbolů
- definujeme $L_1 = \{ a'w \mid a'\in X' \ \&\ h(a)=w\}$ (jazyk popisující h)
  - L_1 - slova poskládaná ze symbolu a jeho homomorfního obrazu
  - L_1 je konečný jazyk, a tedy L_1^* je regulární jazyk
- definujme $h_1: (X' \cup A)^* \rightarrow X^*$ takto
  - $h_1(a') = a$, pro $a'\in X'$
  - $h_1(c) = \lambda$, pro $c\in A$
  - maže původní symboly

Potom: $h^{-1}(L) = h_1(\sigma(L) \cap L_1^*)$



L

$\sigma(L)$

$\sigma(L) \cap L_1^*$

$h_1(\sigma(L) \cap L_1)$

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

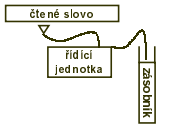
$0L_1 \cup 1L_2$ je DBKJ, $1L_1 \cup 1L_2$ není DBKJ

položme $h(0) = 1$

$h(x) = x$ pro ostatní symboly

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

Jak charakterizovat bezkontextové jazyky?



Pokud do zásobníku pouze přidáváme potom si stačí pamatovat poslední symbol.

Stačí konečná paměť → konečný automat.

Potřebujeme ze zásobníku také odebírat (čtení symbolu)!

Takový proces nelze zaznamenat v konečné struktuře.

Přidávání a odebírání ale není zcela libovolné

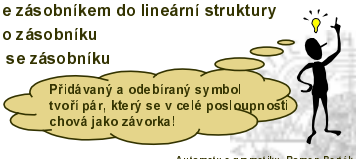
jedná se o zásobník tj. LIFO (last-in first-out) strukturu

Roztáhněme si výpočet se zásobníkem do lineární struktury

X - symbol přidán do zásobníku

X^{-1} - symbol odebrán se zásobníku

$Z\ Z^{-1}\ B\ A\ A^{-1}\ C\ C^{-1}\ B^{-1}$



Dyckovy jazyky

Dyckův jazyk D_n je definován nad abecedou $Z_n = \{a_1, a'_1, \ldots, a_n, a'_n\}$ následující gramatikou:

$S \rightarrow \lambda \mid SS \mid a_1 S a'_1 \mid \ldots \mid a_n S a'_n$.

Úvodní poznámky:

- Jedná se zřejmě o jazyk bezkontextový.
- Dyckův jazyk D_n popisuje správně uzávorkované výrazy s *n* druhy závorek.
- Tímto jazykem lze popisovat výpočty libovolného zásobníkového automatu.
- Pomocí Dyckova jazyka lze popsat libovolný bezkontextový jazyk.

$L = h(D \cap R)$

Regulární jazyk popisuje všechny kroky výpočtu

Dyckův jazyk vybírá pouze korektní výpočty

Homomorfismus čistí pomocné symboly

Dyckovy jazyky a bezkontextové jazyky

Pro každý bezkontextový jazyk L existuje regulární jazyk R tak, že $L=h(D \cap R)$ pro vhodný Dyckův jazyk D a homomorfismus h.

Důkaz:

máme zásobníkový automat přijímající L prázdným zásobníkem

BÚNO instrukce tvaru $\delta(q,a,X) \ni (p,w)$, $|w| \leq 2$

nechť R' obsahuje všechny výrazy

$q^{-1}aa \cdot {}^{-1}X^{-1}BAp$ pro instrukci $\delta(q,a,X) \ni (p,AB)$

podobně pro instrukce $\delta(q,a,X) \ni (p,A)$ a $\delta(q,a,X) \ni (p,\lambda)$

je-li $a=\lambda$, potom dvojici aa^{-1} nezařazujeme

definujme R takto $Z_0 q_0 R'^* Q^{-1}$

Dyckův jazyk je definován nad abecedou $X \cup X^{-1} \cup Q \cup Q^{-1} \cup Y \cup Y^{-1}$

$D \cap Z_0 q_0 R'^* Q^{-1}$ popisuje korektní výpočty

$Z_0\ q_0\ q_0^{-1}\ a\ a^{-1}\ Z_0^{-1}\ B\ A\ p\ p^{-1}\ b\ b^{-1}\ A^{-1}\ q\ q^{-1}\ c\ c^{-1}\ B^{-1}\ r\ r^{-1}$

homomorfismus h vydělí přečtené slovo, tj.

$h(a) = a$ pro vstupní (čtené) symboly

$h(y) = \lambda$ pro ostatní symboly

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!

$bB \rightarrow bb$

$bC \rightarrow cc$

$cC \rightarrow cc$

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 $|u| \leq |v|$.

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 $X \to x$ jsou kontextová)

zbývají pravidla $A_1 \ldots A_m \to B_1 \ldots B_n$ (kde $m \leq n$)

převedeme na kontextová pravidla s novými neterminály C

$A_1 \ldots A_m \to C_1 A_2 \ldots A_m$
$C_1 A_2 \ldots A_m \to C_1 C_2 \ldots A_m$ ...
$C_1 \ldots C_{m-1} A_m \to C_1 \ldots C_{m-1} C_m B_{m+1} \ldots B_n$
$C_1 \ldots C_m B_{m+1} \ldots B_n \to C_1 \ldots C_{m-1} B_m B_{m+1} \ldots B_n$ ...
$C_1 B_2 \ldots B_n \to B_1 \ldots B_n$

Příklad kontextového jazyka

$L = \{a^i b^j c^k \mid 1 \leq i \leq j \leq k\}$ 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:
$CB \to XB$, $XB \to XC$, $XC \to BC$

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

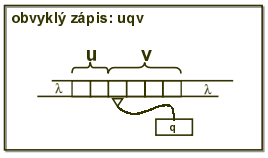
Definice Turingova stroje

Turingovým strojem nazýváme pětici $T=(Q,X,\delta,q_0,F)$, kde
- Q - neprázdná konečná množina stavů
- X - neprázdná konečná množina symbolů
  obsahuje symbol λ pro prázdné políčko
- δ - přechodová funkce $\delta : (Q-F) \times X \to Q \times X \times \{-1,0,1\}$
  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 $q_0 w$ na $q_F w^R$, kde $w \in \{a_1, \ldots, a_n\}^*$ (tj. obrácení slova).

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