Automaty a gramatiky

7

Roman Barták, KTIML

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

Co bylo minule

Chomského hierarchie a její uspořádání
- $\mathcal{L}_0 \supseteq \mathcal{L}_1 \supseteq \mathcal{L}_2 \supseteq \mathcal{L}_3$
- nevypouštějící bezkontextové gramatiky

Gramatiky typu 3 a regulární jazyky
- automat ↔ gramatika (standardní tvar)
- lineární gramatiky (zleva, zprava)

Bezkontextové gramatiky
- redukce bezkontextové gramatiky
  - generování terminálního slova
  - dosažitelnost
- není jednoznačnost

Bezkontextové gramatiky a derivace

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

Úmluva

neterminály	= velká písmena
terminály	= malá písmena
pravidla	= X→ u | v | w | … (pro stejnou levou stranu)

Derivace

$w \Rightarrow z$, jestliže: $\exists x,y,v \in (V_N \cup V_T)^*$ tž. $w=xAy$, $z=xvy$ a $(A \to v) \in P$

G: S → aSX | λ, X → XbSb | c

<u>S</u> ⇒ aS<u>X</u> ⇒ aSXb<u>S</u>b ⇒ a<u>S</u>Xbb ⇒ a<u>X</u>bb ⇒ acbb
<u>S</u> ⇒ a<u>S</u>X ⇒ a<u>X</u> ⇒ a<u>X</u>bSb ⇒ acb<u>S</u>b ⇒ acbb
<u>S</u> ⇒ aS<u>X</u> ⇒ aSXb<u>S</u>b ⇒ aS<u>X</u>bb ⇒ a<u>S</u>cbb ⇒ acbb

Pozorování
- stejná délka derivací (počet pravidel) + použita stejná pravidla
- liší se pořadím aplikace pravidel
- přepis neterminálu neovlivňuje derivaci ve zbytku slova

Kanonické derivace

Zdá se zbytečné zabývat se derivacemi s různým pořadím pravidel.

Definice:

Levé přepsání $w \Rightarrow z$, jestliže se přepisuje nejlevější neterminál:
$\exists v,y \in (V_N \cup V_T)^*$ $\exists x \in V_T^*$ $\exists A \in V_N^*$ tž. $w=xAy$, $z=xvy$ a $(A \to v) \in P$

Levá derivace vzniká použitím pouze levých přepsání.

Pravé přepsání a *pravá derivace* se definuje obdobně (vždy se přepisuje nejpravější neterminál)

Lemma: Pro bezkontextové gramatiky platí:
$X \Rightarrow^* w$ právě tehdy, když existuje levá (pravá) derivace w z X

Důkaz:

stačí ukázat, že existence derivace implikuje existenci levé derivace

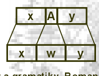
xAy ⇒ xwy, použitím $A \to w$

přepsání $A \to w$ neovlivní řetězce x a y ani jejich další přepisování

části x, A, y se přepisují nezávisle na sobě

můžeme preferovat aplikaci některých pravidel

formální důkaz (indukcí dle délky derivace)

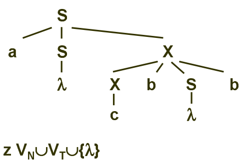


Derivační strom

Můžeme „výpočet" zachytit jinak než sekvencí pravidel?

Příklad:

G: S → aSX | λ, X → XbSb | c



Definice:

Derivační strom je takový strom, že:
- každý vrchol je ohodnocen prvkem z $V_N \cup V_T \cup \{\lambda\}$
- kořen je ohodnocen S (počáteční neterminál)
- vnitřní vrcholy jsou ohodnoceny prvkem z V_N
- je-li A ohodnocení vrcholu a $u_1,\ldots,u_n$ jsou ohodnocení jeho potomků (bráno zleva doprava), potom $(A \to u_1,\ldots,u_n) \in P$
- je-li vrchol ohodnocen λ, potom je to list, který je jediným potomkem svého rodiče.

V derivačním stromu hraje roli jak stromové uspořádání dané hranami, tak uspořádání zleva doprava.

Derivace a derivační stromy

Říkáme, že *derivační strom dává slovo w*, jestliže *w* je slovo složené z ohodnocení listů (bráno zleva doprava).

Několik zřejmých tvrzení:
- $S \Rightarrow^* w$ potom existuje derivační strom, který dává w (jednoznačně daný derivací)
- máme-li derivační strom, který dává w, potom $S \Rightarrow^* w$ (derivace ale není určena jednoznačně)
- každý derivační strom jednoznačně určuje levou (a pravou) derivaci

Derivační strom zastupuje derivace slova získané „stejným způsobem" (význam slova).

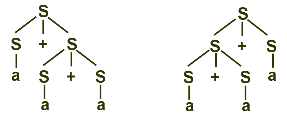
Je možné mít různé derivační stromy dávající stejné slovo (pro stejnou gramatiku)?

Jednoznačnost a víceznačnost BKG

Příklad:

$S \rightarrow S{+}S \mid a$

slovo a+a+a



Definice:

Bezkontextová gramatika G je víceznačná (nejednoznačná), jestliže $\exists w \in L(G)$, které má dvě různé levé derivace.

V ostatních případech *bezkontextová gramatika je jednoznačná*.

Bezkontextový jazyk L je jednoznačný, jestliže existuje jednoznačná bezkontextová gramatika G tak, že L=L(G).

Bezkontextový jazyk L je (podstatně) nejednoznačný, jestliže každá BKG G, taková že L=L(G) je nejednoznačná.

Příklad:

jazyk $\{a^i b^j c^k \mid i=j \vee j=k\}$ je podstatně nejednoznačný

pro slovo $a^i b^i c^i$ existují z principielních důvodů dva způsoby odvození

Od víceznačnosti k jednoznačnosti

Víceznačnost je potenciálním zdrojem potíží.

jedno slovo = více významů

U programovacích jazyků je víceznačnost nepřípustná!

Příklad:

$S \rightarrow S{+}S \mid a$ …víceznačná gramatika

$S \rightarrow a{+}S \mid a$ …ekvivalentní jednoznačná gramatika

Příklad 2:

$S \rightarrow$ if then S else S | if then S | λ

slovo „if then if then else“ má dva významy

„if then (if then else)“ nebo „if then (if then) else“

Řešení:

- syntaktická chyba (Algol 60)
- else patří k bližšímu if (preference pořadí pravidel)
- závorky begin-end (asi nejčistší řešení)

Jednoznačnost a kompilátory

$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid a$ … nejednoznačné

$E \rightarrow E{+}E \mid T, \quad T \rightarrow T{*}T \mid F, \quad F \rightarrow (E) \mid a$ … řeší prioritu operací

Kompilace výrazu (zásobník na mezivýsledky+dva registry):

(1) $E \rightarrow E{+}T$ … pop r1; pop r2; add r1,r2; push r2

(2) $E \rightarrow T$

(3) $T \rightarrow T{*}F$ … pop r1; pop r2; mul r1,r2; push r2

(4) $T \rightarrow F$

(5) $F \rightarrow (E)$

(6) $F \rightarrow a$ … push a

a+a*a, získáme postupnou aplikací pravidel 1,2,4,6,3,4,6,6

$\underline{E} \Rightarrow \underline{E}{+}T \Rightarrow \underline{T}{+}T \Rightarrow \underline{F}{+}T \Rightarrow a{+}\underline{T} \Rightarrow a{+}\underline{T}{*}F \Rightarrow a{+}\underline{F}{*}F \Rightarrow a{+}a{*}\underline{F} \Rightarrow a{+}a{*}a$

posloupnost obrátíme a vybere pouze pravidla generující kód 6,6,3,6,1

nyní pravidla nahradíme příslušným kódem

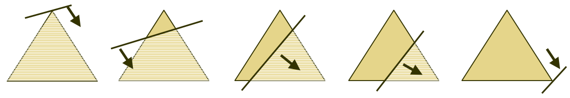
push a; push a; pop r1; pop r2; mul r1,r2; push r2; push a; pop r1; pop r2; add r1,r2; push r2

Analýza shora bezkontextových jazyků

Jak k danému slovu a zvolené bezkontextové gramatice najdeme odpovídající derivační strom?

Analýza shora

- konstruujeme levou derivaci
- dosud vygenerované slovo kontrolujeme se vstupem



V každém kroku derivace můžeme slovo psát ve tvaru *uv*, kde

- *u* obsahuje pouze terminály (již přečtená část slova)
- *v* začíná neterminálem (zatím nehotová část)

Postup hledání derivace:

1) vezmi první neterminál *A* z *v* a nahraď ho *w*, dle pravidla $A \rightarrow w$
2) vzniklé slovo *v* rozlož na *xy*, kde *x* obsahuje pouze terminály a *y* začíná neterminálem
3) zkontroluj *x* oproti vstupu a pokud je v pořádku, přidej *x* za *u*, polož *v* rovno *y* a opakuj od 1 dokud $v \neq \lambda$

Realizace analyzátoru

slovo generujeme na zásobník (LIFO fronta)

je-li vrchol zásobníku terminál, srovnáme ho se vstupem (čteme znak)

je-li vrchol zásobníku neterminál, nahradíme ho slovem dle pravidla

končíme, když je zásobník prázdný (musí být přečteno celé slovo)

Příklad:

(1) $E \rightarrow E{+}T$

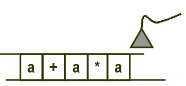
(2) $E \rightarrow T$

(3) $T \rightarrow T{*}F$

(4) $T \rightarrow F$

(5) $F \rightarrow (E)$

(6) $F \rightarrow a$



zásobník	zbytek vstupu	pravidlo
E	a+a*a	
E+T	a+a*a	(1)
T+T	a+a*a	(2)
F+T	a+a*a	(4)
a+T	a+a*a	(6)
+T	+a*a	krácení
T	a*a	krácení
T*F	a*a	(3)
F*F	a*a	(4)
a*F	a*a	(6)
*F	*a	krácení
F	a	krácení
a	a	(6)
λ	λ	krácení

Zásobníkový automat

Zásobníkovým automatem **nazýváme sedmici**

$$M = (Q, X, Y, \delta, q_0, Z_0, F),$$

kde

Q - neprázdná konečná množina stavů

X - neprázdná konečná vstupní abeceda

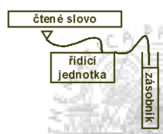
Y - neprázdná konečná zásobníková abeceda

δ - přechodová funkce $Q \times (X \cup \{\lambda\}) \times Y \rightarrow P_{FIN}(Q \times Y^*)$

$q_0 \in Q$ - počáteční stav

$Z_0 \in Y$ - počáteční zásobníkový symbol

F - množina koncových stavů



Poznámky:

- ZA je z principu nedeterministický
- vždy nahrazujeme vrchol zásobníku
- nemusíme číst vstupní symbol

Výpočet zásobníkového automatu

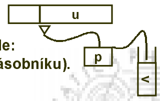
Instrukci $(p,a,Z) \to (q,w)$ ($(q,w) \in \delta(p,a,Z)$) lze vykonat, pokud:

- stav automatu je p
- na vstupu je symbol *a* (pouze pokud $a \neq \lambda$)
- na vrcholu zásobníku je symbol Z

Vykonání instrukce $(p,a,Z) \to (q,w)$ znamená:

- změnu stavu automatu z *p* na *q*
- je-li $a \neq \lambda$, posun čtecí hlavy (přečtení písmene *a*)
- smazání vrchního symbolu zásobníku (symbolu Z)
- přidání slova *w* na zásobník (nejvýše bude první písmeno z *w*)

Formalizace kroku zásobníkového automatu:



Situace zásobníkového automatu je trojice (p,u,v), kde:
$p \in Q$, $u \in X^*$ (zbytek čteného slova), $v \in Y^*$ (obsah zásobníku).

Situace $E_1=(p,au,Zv)$ *vede bezprostředně na situaci*
$E_2=(q,u,wv)$, když $p,q \in Q$, $u \in X^*$, $v,w \in Y^*$, $a \in X \cup \{\lambda\}$, $Z \in Y$, $(q,w) \in \delta(p,a,Z)$.
Píšeme $E_1 \vdash E_2$.

Situace E *vede na situaci* E' ($E \vdash^* E'$), právě když $E \vdash E_1$, $E_1 \vdash E_2 \dots E_n \vdash E'$

Zásobníkové automaty a jazyky

Kdy končí výpočet zásobníkového automatu:

- zásobník je prázdný
- není definována žádná instrukce

Přijímání koncovým stavem
slovo je celé přečteno a jsme v koncovém stavu

Přijímání prázdným zásobníkem
slovo je celé přečteno a zásobník je vyprázdněný

Nechť M je zásobníkový automat.

Jazyk rozpoznávaný automatem M *koncovým stavem* definujeme takto: $L(M) = \{w \mid w \in X^*, v \in Y^*, q \in F\ (q_0,w,Z_0) \vdash^* (q,\lambda,v)\}$.

Jazyk rozpoznávaný automatem M *prázdným zásobníkem* definujeme takto: $N(M) = \{w \mid w \in X^*, q \in Q\ (q_0,w,Z_0) \vdash^* (q,\lambda,\lambda)\}$.
Koncové stavy nás tady nezajímají, proto klademe $F=\emptyset$.

Zásobníkový automat v příkladě

$L = \{0^n1^n \mid n>0\}$

Přijímání prázdným zásobníkem

p-počáteční stav, Z-počáteční zásobníkový symbol

$\delta(p,0,Z) = \{(p,A)\}$ … čte první symbol 0
$\delta(p,0,A) = \{(p,AA)\}$ … čte další symboly 0
$\delta(p,1,A) = \{(q,\lambda)\}$ … čte první symbol 1
$\delta(q,1,A) = \{(q,\lambda)\}$ … čte další symboly 1

Přijímání koncovým stavem

p-počáteční stav, q_F-koncový stav, Z-počáteční zásobníkový symbol

$\delta(p,0,Z) = \{(p,AZ)\}$ … čte první symbol 0
$\delta(p,0,A) = \{(p,AA)\}$ … čte další symboly 0
$\delta(p,1,A) = \{(q,\lambda)\}$ … čte první symbol 1
$\delta(q,1,A) = \{(q,\lambda)\}$ … čte další symboly 1
$\delta(q,\lambda,Z) = \{(q_F,\lambda)\}$ … končí

Přijímání zásobníkem ⇒ přijímání stavem

Pro každý zásobníkový automat M_1 existuje ekvivalentní zásobníkový automat M_2 tak, že $N(M_1)=L(M_2)$ (prázdný zásobník → koncový stav).

Důkaz (konstruktivní):

idea:

- na zásobník přidáme speciální symbol,
- běžíme stejně jako M_1,
- je-li na zásobníku speciální symbol, končíme

formálně:

$M_1 = (Q_1,X,Y_1,\delta_1,q_1,Z_1,\{\}) \to M_2 = (Q_2,X,Y_2,\delta_2,q_2,Z_2,\{q_F\})$,
$q_2,q_F \notin Q_1$, $Q_2 = Q_1 \cup \{q_2,q_F\}$,
$Z_2 \notin Y_1$, $Y_2 = Y_1 \cup \{Z_2\}$,
$\delta_2 = \delta_1 + \delta_2(q_2,\lambda,Z_2)=\{(q_1,Z_1Z_2)\}$, $\forall q \in Q_1\ \delta_2(q,\lambda,Z_2)=\{(q_F,\lambda)\}$

$N(M_1)=L(M_2)$?

$w \in N(M_1) \Leftrightarrow (q_1,w,Z_1) \vdash^*_{M1} (q,\lambda,\lambda)$
$\Leftrightarrow (q_2,w,Z_2) \vdash_{M2} (q_1,w,Z_1Z_2) \vdash^*_{M2} (q,\lambda,Z_2) \vdash_{M2} (q_F,\lambda,\lambda)$
$\Leftrightarrow w \in L(M_2)$

Příklad převodu

$L = \{0^n1^n \mid n\geq 0\}$

$M_1 = (\{p,q\}, \{0,1\}, \{Z,A\}, \delta, p, Z,\{\})$, $L=N(M_1)$

$\delta(p,\lambda,Z) = \{(p,\lambda)\}$ … čte prázdné slovo
$\delta(p,0,Z) = \{(p,A)\}$
$\delta(p,0,A) = \{(p,AA)\}$
$\delta(p,1,A) = \{(q,\lambda)\}$
$\delta(q,1,A) = \{(q,\lambda)\}$

$M_2 = (\{p,q,q_2,q_F\}, \{0,1\}, \{Z,A,Z_2\}, \delta_2, q_2, Z_2, \{q_F\})$, $L=L(M_2)$

$\delta_2(q_2,\lambda,Z_2) = \{(p,ZZ_2)\}$ … nastartování výpočtu M_1
$\delta_2(p,\lambda,Z) = \{(p,\lambda)\}$
$\delta_2(p,0,Z) = \{(p,A)\}$
$\delta_2(p,0,A) = \{(p,AA)\}$
$\delta_2(p,1,A) = \{(q,\lambda)\}$
$\delta_2(q,1,A) = \{(q,\lambda)\}$
$\delta_2(p,\lambda,Z_2) = \{(q_F,\lambda)\}$ … ukončení výpočtu
$\delta_2(q,\lambda,Z_2) = \{(q_F,\lambda)\}$ … “

Přijímání stavem ⇒ přijímání zásobníkem

Pro každý zásobníkový automat M_1 existuje ekvivalentní zásobníkový automat M_2 tak, že $L(M_1)=N(M_2)$ (koncový stav → prázdný zásobník).

Důkaz (konstruktivní):

idea:

- na zásobník přidáme speciální symbol (proti vyprázdnění),
- běžíme stejně jako M_1,
- v koncovém stavu smažeme zásobník (nedeterminismus!)

formálně:

$M_1 = (Q_1,X,Y_1,\delta_1,q_1,Z_1,F) \to M_2 = (Q_2,X,Y_2,\delta_2,q_2,Z_2,\{\})$,
$q_2,q_M \notin Q_1$, $Q_2 = Q_1 \cup \{q_2,q_M\}$,
$Z_2 \notin Y_1$, $Y_2 = Y_1 \cup \{Z_2\}$,
δ_2 „=” $\delta_1 + \forall q_F \in F\ \forall Z \in Y_2\ \delta_2(q_F,\lambda,Z)=(\delta_1(q_F,\lambda,Z) \cup \{(q_M,\lambda)\})$,
$\delta_2(q_2,\lambda,Z_2)=\{(q_1,Z_1Z_2)\}$, $\forall Z \in Y_2\ \delta_2(q_M,\lambda,Z)=\{(q_M,\lambda)\}$

$L(M_1)=N(M_2)$?

$w \in L(M_1) \Leftrightarrow (q_1,w,Z_1) \vdash^*_{M1} (q_F,\lambda,v)$
$\Leftrightarrow (q_2,w,Z_2) \vdash_{M2} (q_1,w,Z_1Z_2) \vdash^*_{M2} (q_F,\lambda,vZ_2) \vdash^*_{M2} (q_M,\lambda,\lambda)$
$\Leftrightarrow w \in N(M_2)$

Příklad převodu

$L = \{ w \mid w \in \{0,1\}^*, |w|_0 = |w|_1\}$

$M_1 = (\{p,q\}, \{0,1\}, \{Z,A,B\}, \delta, p, Z, \{p\})$, $L=L(M_1)$

- $\delta(p,0,Z) = \{(q,AZ)\}$ … čte první nulu
- $\delta(p,1,Z) = \{(q,BZ)\}$ … čte první jedničku
- $\delta(q,0,A) = \{(q,AA)\}$ … přidává další nulu
- $\delta(q,0,B) = \{(q,\lambda)\}$ … krátí nulu oproti předchozí jedničce
- $\delta(q,1,A) = \{(q,\lambda)\}$ … krátí jedničku oproti předchozí nule
- $\delta(q,1,B) = \{(q,BB)\}$ … přidává další jedničku
- $\delta(q,\lambda,Z) = \{(p,Z)\}$ … počet nul a jedniček vyrovnán

$M_2 = (\{p,q,q_2,q_M\}, \{0,1\}, \{Z,A,B,Z_2\}, \delta_2, q_2, Z_2, \{\})$, $L=N(M_2)$

- $\delta_2(q_2,\lambda,Z_2) = \{(p,ZZ_2)\}$ … nastartování výpočtu M_1
- " … stejné instrukce jako u M_1
- $\delta_2(p,\lambda,X) = \{(q_M, \lambda)\}$ $\forall X \in \{Z,A,B,Z_2\}$ … přechod do mazacího stavu
- $\delta_2(q_M,\lambda,X) = \{(q_M, \lambda)\}$ $\forall X \in \{Z,A,B,Z_2\}$ … mazání zásobníku