Týden 6

Přednáška-pondělí

‘Překladač’ sestrojující k regulárnímu výrazu ekvivalentní konečný automat

Nejdříve jsme si z minula připomněli jednoznačnou gramatiku G pro jazyk $RV(\{a, b\})$

$$
\begin{aligned}
R &\longrightarrow T + R \mid T \\
T &\longrightarrow FT \mid F \\
F &\longrightarrow F^* \mid (R) \mid C \\
C &\longrightarrow a \mid b
\end{aligned}
$$

v níž jsme ale pro zjednodušení vynechali symboly $\emptyset$ a ε. Naše G tedy generuje jazyk v abecedě $\Sigma = \{a, b, +, ^*, (,)\}$.
Jako příklad řetězce z $L(G)$ jsme opět vzali

$$(aa + b)^* .$$

Poté jsme provedli dále popsanou konstrukci derivačního stromu pro slovo $(aa + b)^*$; jedná se o konstrukci typu zdola-nahoru (strom konstruujeme postupně od listů ke kořeni).
Účelem samozřejmě nebylo jen zkonstruovat tento konkrétní strom (který díky jednoduchosti vstupního řetězce jistě zkonstruujete téměř bez přemýšlení), ale alespoň naznačit algoritmickou metodu, která stojí v pozadí reálné syntaktické analýzy v překladačích. Po zkonstruování stromu si ukážeme, jak lze takovouto syntaktickou strukturu využít pro generování „cílového kódu“, čímž v našem případě rozumíme konečný automat přijímající jazyk, který je reprezentován vstupním regulárním výrazem.

Konstrukce derivačního stromu zdola-nahoru

Na vstupním řetězci se pohybuje čtecí hlava (jen doprava). Na začátku stojí na nejlevějším symbolu, její pozici znázorněme podtržením:

$$\underline{(}aa + b)^*$$

Budeme využívat datovou strukturu zásobník; jednotlivá položka ukládaná na zásobník (odpovídá vrcholu derivačního stromu a) obsahuje jeden terminál či neterminál gramatiky G. Položky s neterminály budou navíc obsahovat ukazatele (pointers) do paměti (typu halda, v níž dynamicky alokujeme potřebné ‘buňky’). Na začátku jsou zásobník i paměť prázdné.
Jedna z ‘instrukcí’, kterou aplikujeme, pokud jsou splněny její předpoklady, zní takto:

> **Instrukce 1**
>
> Jestliže zásobník neobsahuje na vrcholu pravou stranu nějakého pravidla gramatiky G a nebylo-li přečteno celé vstupní slovo, přesune se do zásobníku aktuálně čtený symbol ze vstupu a čtecí hlava se posune (o jedno pole doprava).

Na začátku tedy je Instrukce 1 aplikovatelná a po jejím provedení dostaneme následující konfiguraci; zde druhý řádek ukazuje obsah zásobníku (vlevo je vždy dno zásobníku, vpravo jeho vrchol). Jednu položku na zásobníku vždy ohraničujeme závorkami [ ]; ty nejsou symboly naší gramatiky G, takže si je s nimi nespleteme.

$(\underline{a}a + b)^*$
$[(]$

Znovu je aplikovatelná Instrukce 1, dojde tedy k dalšímu přesunu a následující konfigurace je

$(a\underline{a} + b)^*$
$[(][a]$

Nyní vrchní symboly zásobníku, v našem případě jeden, tvoří pravou stranu pravidla gramatiky, konkrétně pravidla $C \longrightarrow a$. Jedná se o terminální symbol, který se pochopitelně musí vyskytovat jako list v konstruovaném derivačním stromě. Je snadné nahlédnout, že předchůdce tohoto listu musí být v každém případě označen C a musí mít onen list označený a jako jediného následníka. Proto nemůžeme nic pokazit provedením tzv. redukce podle pravidla $C \longrightarrow a$. Obecně vypadá (procedura) redukce následovně.

Redukce podle pravidla $X \longrightarrow Y_1Y_2 \ldots Y_n$
(Bude se používat jen v případě, že zásobník má na vrcholu položky $pol_1, pol_2, \ldots, pol_n$ (kde pol_n je ta nejvrchnější) obsahující postupně symboly $Y_1, Y_2, \ldots, Y_n$; zde Y_i mohou být neterminály i terminály. K redukci ale nedojde v takovém případě automaticky, budou muset být případně splněny další podmínky.)

Nejprve se alokuje nová paměť: n buněk, do nichž jsou uloženy (kompletní) položky $pol_1, pol_2, \ldots, pol_n$; tyto položky se ze zásobníku odstraní a na vrchol zásobníku se vloží nová položka se symbolem X a n ukazateli (pointery) $p_1, p_2, \ldots, p_n$, které ukazují na nově alokované buňky: ukazatel p_i ukazuje na buňku (tedy je adresou buňky), do které byla uložena položka pol_i.

V našem případě tedy zformulujeme tuto instrukci:

Instrukce 2

Jestliže je na vrcholu zásobníku a, provedeme redukci podle pravidla $C \longrightarrow a$.

Její provedení v aktuální konfiguraci vyústí v novou konfiguraci

$(a\underline{a} + b)^*$
$[(][C, p_1]$
$p_1 : [a]$

Kromě vstupu a zásobníku teď už konfigurace zahrnuje i kousek haldy. Tam žádnou (viditelnou) strukturu nepředpokládáme, byť se pro názornost budeme snažit obsah zapisovat tak, ať je v zápisu vznikající derivační strom trochu 'vidět'. Můžete si samozřejmě dokreslovat příslušné šipky; zde by šlo o šipku, která znázorní, že položka $[C, p_1]$ ukazuje na paměťovou buňku (s adresou) p_1.

Vidíme teď, že opět máme pravou stranu nějakého pravidla na vrcholu zásobníku, konkrétně jde o pravidlo $F \longrightarrow C$. Jelikož C se jinde na pravé straně nevyskytuje, můžeme bezpečně používat následující instrukci:

Instrukce 3

Jestliže je na vrcholu zásobníku C, provedeme redukci podle pravidla $F \longrightarrow C$.

Její aplikací dostaneme konfiguraci

$(a\underline{a} + b)^*$
$[(][F, p_2]$
$p_2 : [C, p_1]$
$p_1 : [a]$

(Alokovali jsme samozřejmě novou buňku paměti, což je znázorněno hodnotou p_2, která dosud nebyla použita.)

Teď musíme být opatrní. Na vrcholu zásobníku je sice pravá strana nějakého pravidla, konkrétně $T \longrightarrow F$, ale F se vyskytuje i na pravých stranách jiných pravidel, konkrétně $T \longrightarrow FT$ a $F \longrightarrow F^*$. Není těžké vytušit, že by měly fungovat následující instrukce. ('Fungovat' znamená, že aplikace instrukce nemůže způsobit to, že konstrukce derivačního stromu zhavaruje, ač vstupní slovo je v $L(G)$. My se zde v naší ilustraci omezujeme na ono 'vytušení', ve skutečnosti bychom příslušné tvrzení samozřejmě museli dokázat.)

Instrukce 4

Jestliže je na vrcholu zásobníku F a aktuální čtený symbol je *, provedeme přesun čteného symbolu (* jde do zásobníku a čtecí hlava se posune) a pak redukci podle pravidla $F \longrightarrow F^*$.

Instrukce 5

Jestliže je na vrcholu zásobníku F a aktuální čtený symbol je + nebo), nebo je vstupní slovo již přečteno (čtecí hlava stojí za ním), pak provedeme redukci podle pravidla $T \longrightarrow F$.

Instrukce 6

Jestliže je na vrcholu zásobníku F a aktuální čtený symbol je a, b, nebo (, pak provedeme přesun (vstupního symbolu do zásobníku). (Položka s F bude 'čekat', až se v budoucnu objeví napravo od ní T jako vrchol zásobníku.)

V našem příkladu je tedy aplikována Instrukce 6, což vede ke konfiguraci

$$
\begin{array}{l}
(aa\underline{+}b)^* \\
[(][F, p_2][a] \\
p_2 : [C, p_1] \\
p_1 : [a]
\end{array}
$$

Na vrcholu se objevilo a, takže aplikacemi Instrukcí 2 a 3 se dostaneme do konfigurace

$$
\begin{array}{l}
(aa\underline{+}b)^* \\
[(][F, p_2][F, p_4] \\
p_2 : [C, p_1] \quad p_4 : [C, p_3] \\
p_1 : [a] \quad p_3 : [a]
\end{array}
$$

Nyní se uplatní Instrukce 5 a dostáváme

$$
\begin{array}{l}
(aa\underline{+}b)^* \\
[(][F, p_2][T, p_5] \\
\qquad\qquad p_5 : [F, p_4] \\
p_2 : [C, p_1] \quad p_4 : [C, p_3] \\
p_1 : [a] \quad p_3 : [a]
\end{array}
$$

Máme tedy na vrcholu zásobníku pravou stranu pravidla $T \longrightarrow FT$; opět lze vytušit (a lze pečlivě dokázat), že můžeme bezpečně používat následující instrukci:

Instrukce 7

Jestliže je na vrcholu zásobníku FT, zredukujeme podle pravidla $T \longrightarrow FT$.

Při její aplikaci teď poprvé redukujeme podle pravidla, u nějž je délka pravé strany větší než 1. Výsledkem bude konfigurace

$$
\begin{array}{l}
(aa\underline{+}b)^* \\
[(][T, p_6, p_7] \\
p_6 : [F, p_2] \quad p_7 : [T, p_5] \\
\qquad\qquad p_5 : [F, p_4] \\
p_2 : [C, p_1] \quad p_4 : [C, p_3] \\
p_1 : [a] \quad p_3 : [a]
\end{array}
$$

Nyní je na vrcholu zásobníku T, ale není tam FT. Pro budoucí redukci zahrnující ono T, které je momentálně na vrcholu, připadají tedy v úvahu pravidla $R \longrightarrow T + R$, $R \longrightarrow T$. Zase se (dá ukázat, že se) tento konflikt dá vyřešit pomocí aktuálně čteného symbolu:

Instrukce 8

Jestliže je na vrcholu zásobníku T (ale ne FT) a aktuální čtený symbol je $+$, provedeme přesun čteného symbolu ($+$ jde do zásobníku a čtecí hlava se posune).

Instrukce 9

Jestliže je na vrcholu zásobníku T (ale ne FT) a aktuální čtený symbol není $+$, provedeme redukci podle pravidla $R \longrightarrow T$.

Poznámka. Jeden příklad na cvičení žádá, ať zjistíte, jaké situace mohou nastat, když je na vstupu správně utvořený regulární výraz (tedy slovo z $L(G)$), na vrcholu zásobníku je T a aktuální čtený symbol není $+$. Toho lze využít tak, že při zjištění 'nelegální' situace může algoritmus ihned skončit zahlášením chyby (což znamená, že slovo na vstupu nepatří do $L(G)$).

V našem konkrétním případě je tedy aplikována Instrukce 8. Po ní je aplikovatelná Instrukce 1, takže dojde k dalšímu přesunu. Na vrcholu se ocitne b, pro které pochopitelně použijeme analogickou instrukci k Instrukci 2, což je

Instrukce 10

Jestliže je na vrcholu zásobníku b, provedeme redukci podle pravidla $C \longrightarrow b$.

Pak se uplatní instrukce 3, takže po této sérii skončíme v konfiguraci

$$
\begin{array}{l}
(aa + b)^* \\
[(][T, p_6, p_7][+][F, p_9] \\
p_6 : [F, p_2] \quad p_7 : [T, p_5] \\
\qquad\qquad p_5 : [F, p_4] \\
p_2 : [C, p_1] \quad p_4 : [C, p_3] \quad p_9 : [C, p_8] \\
p_1 : [a] \quad p_3 : [a] \qquad\qquad p_8 : [b]
\end{array}
$$

Jelikož další čtený symbol je pravá závorka, uplatní se Instrukce 5 a následně Instrukce 9. Dostaneme tedy

$$
\begin{array}{l}
(aa + b)^* \\
[(][T, p_6, p_7][+][R, p_{11}] \\
p_6 : [F, p_2] \quad p_7 : [T, p_5] \quad p_{11} : [T, p_{10}] \\
\qquad\qquad p_5 : [F, p_4] \quad p_{10} : [F, p_9] \\
p_2 : [C, p_1] \quad p_4 : [C, p_3] \quad p_9 : [C, p_8] \\
p_1 : [a] \quad p_3 : [a] \qquad\qquad p_8 : [b]
\end{array}
$$

Máme na vrcholu pravou stranu pravidla $R \longrightarrow T + R$; opět by šlo ověřit bezpečnost následující instrukce:

Instrukce 11

Jestliže je na vrcholu zásobníku $T+R$, provedeme redukci podle pravidla $R \longrightarrow T + R$.

Po její aplikaci dostáváme

$(aa + b)^*$
$[(][[R, p_{12}, p_{13}, p_{14}]$

$p_{12} : [T, p_6, p_7] \quad p_{14} : [R, p_{11}]$
$p_6 : [F, p_2] \quad p_7 : [T, p_5] \quad p_{11} : [T, p_{10}]$
$p_5 : [F, p_4] \quad p_{10} : [F, p_9]$
$p_2 : [C, p_1] \quad p_4 : [C, p_3] \quad p_9 : [C, p_8]$
$p_1 : [a] \quad p_3 : [a] \quad p_{13} : [+] \quad p_8 : [b]$

(Samozřejmě je jedno, kam např. buňku s + v znázornění haldy zakreslíme. Pro přehlednost obrázku je samozřejmě užitečné ji nakreslit na příslušné místo do "dolní (terminální) řady".)
Nyní se opět uplatní Instrukce 1 a potom následující zřejmá instrukce:

Instrukce 12

Jestliže je na vrcholu zásobníku (R), provedeme redukci podle pravidla $F \longrightarrow (R)$.

Výsledek bude

$(aa + b)^*$
$[F, p_{15}, p_{16}, p_{17}]$

$p_{16} : [R, p_{12}, p_{13}, p_{14}]$
$p_{12} : [T, p_6, p_7] \quad p_{14} : [R, p_{11}]$
$p_6 : [F, p_2] \quad p_7 : [T, p_5] \quad p_{11} : [T, p_{10}]$
$p_5 : [F, p_4] \quad p_{10} : [F, p_9]$
$p_2 : [C, p_1] \quad p_4 : [C, p_3] \quad p_9 : [C, p_8]$
$p_{15} : [(] \quad p_1 : [a] \quad p_3 : [a] \quad p_{13} : [+] \quad p_8 : [b] \quad p_{17} : [)]$

Nyní se uplatní Instrukce 4 a po ní Instrukce 5 a pak Instrukce 9. Výsledek bude konfigurace

$(aa + b)^*$
$[R, p_{21}]$

$p_{21} : [T, p_{20}]$
$p_{20} : [F, p_{18}, p_{19}]$
$p_{18} : [F, p_{15}, p_{16}, p_{17}]$

$p_{16} : [R, p_{12}, p_{13}, p_{14}]$
$p_{12} : [T, p_6, p_7] \quad p_{14} : [R, p_{11}]$
$p_6 : [F, p_2] \quad p_7 : [T, p_5] \quad p_{11} : [T, p_{10}]$
$p_5 : [F, p_4] \quad p_{10} : [F, p_9]$
$p_2 : [C, p_1] \quad p_4 : [C, p_3] \quad p_9 : [C, p_8]$
$p_{15} : [(] \quad p_1 : [a] \quad p_3 : [a] \quad p_{13} : [+] \quad p_8 : [b] \quad p_{17} : [)] \quad p_{19} : [^*]$

v níž je celé vstupní slovo přečtené a v zásobníku je jen položka s počátečním neterminálem R. Proběhl takto úspěšný výpočet, který sestrojil derivační strom pro slovo $(aa+b)^*$. Kořen stromu je ona položka $[R, p_{21}]$ v zásobníku.

Mělo by být všem zřejmé, že pokud výpočet pro nějaké vstupní slovo w takto úspěšně skončí (celé slovo přečteno a v zásobníku je jen položka obsahující počáteční neterminál), tak opravdu sestrojil derivační strom pro slovo w, podle gramatiky G, a tedy nutně $w \in L(G)$. Opačný směr, že totiž naše instrukce jsou bezpečné a nemohou zapříčinit neúspěšný výpočet pro nějaké $w \in L(G)$, už tak zřejmý není a museli bychom jej pečlivě dokázat, jak jsme zmiňovali v průběhu. I na takové ověření, a celou konstrukci našich instrukcí, ovšem existují algoritmy, ale tak daleko zde už nepůjdeme.

Sestrojení konečného automatu na základě derivačního stromu

V této části jen stručně dotáhneme, co jsme avízovali. Konstrukce derivačního stromu nebyla samoúčelná a neslouží tak jen ke zjištění, zda zadané slovo je generováno příslušnou gramatikou. Kořen zkonstruovaného stromu (položku s R v zásobníku po skončení úspěšného výpočtu) totiž můžeme předložit funkci NFA, která sestrojí odpovídající konečný automat – stačí nám nedeterministický s případnými ε-šipkami. Níže uvedená definice funkce snad nevyžaduje bližší komentář.

automat NFA(node v)
(* procedura vracející k zadanému vrcholu v derivačního stromu automat, např. ve formě tabulky, který odpovídá podvýrazu určenému podstromem s kořenem v *)
{

if ($v.symb =$ 'a') return

	a	b	ε
$\rightarrow q_1$	q_2	-	-
(q_2)	-	-	-

;

if ($v.symb =$ 'b') return

	a	b	ε
$\rightarrow q_1$	-	q_2	-
(q_2)	-	-	-

;

if ($v.symb =$ 'R' and $v.rule =$ "$R \longrightarrow R + T$")
return UNION(NFA($v.succ_1$), NFA($v.succ_3$));
if ($v.symb =$ 'T' and $v.rule =$ "$T \longrightarrow FT$") return CONC(NFA($v.succ_1$), NFA($v.succ_2$));
if ($v.symb =$ 'F' and $v.rule =$ "$F \longrightarrow F^*$") return ITER(NFA($v.succ_1$));
if ($v.symb =$ 'F' and $v.rule =$ "$F \longrightarrow (R)$") return NFA($v.succ_2$));
if ($v.succ_1 \neq nil$ and $v.succ_2 = nil$) (* je jen jeden následník v *) return NFA($v.succ_1$);
}

Přednáška-čtvrtek

Existence vnitřně nejednoznačných bezkontextových jazyků
Uvažujme jazyk

$$L = \{a^k b^k c^m d^m \mid k, m \geq 1\} \cup \{a^k b^m c^m d^k \mid k, m \geq 1\}.$$

Ten je sice bezkontextový, ale dokážeme, že jej negeneruje žádná jenoznačná bezkontextová gramatika.
Pro odvození sporu předpokládejme, že L je generován jednoznačnou bezkontextovou gramatikou $G = (\Pi, \Sigma, S, P)$.
G můžeme zredukovat (odstranit zbytečné neterminály), přičemž zůstává jednoznačná. (Množina derivačních stromů pro slova z $L = L(G)$ se nezměnila.) Lze rovněž snadno ověřit, že standardní konstrukce odstranění ε-pravidel a odstranění pravidel typu $X \longrightarrow Y$ také nemohou způsobit nejednoznačnost. Takže rovnou předpokládáme, že G nemá taková pravidla.
Nechť nyní pro neterminál $A \neq S$ neplatí $A \Rightarrow^+ u_1 A u_2$ (pro terminální slova u_1, u_2, kde tedy alespoň jedno neprázdné). Tedy na pravých stranách pravidel $A \longrightarrow \alpha_1 \mid \cdots \mid \alpha_m$ se nevyskytuje A. Nahradíme-li nyní pravé strany (jiných pravidel) obsahující A všemi možnosti, při nichž výskyty A nahrazujeme řetězci $\alpha_1, \ldots, \alpha_m$, stane se A nedosažitelným z S a je možné jej z gramatiky odstranit. Jednoznačnost opět nemohla být porušena.
Takže dále rovnou předpokládáme, že v G pro každý neterminál $A \neq S$ platí $A \Rightarrow^+ u_1 A u_2$.
Pro libovolné $A \Rightarrow^+ u_1 A u_2$ snadno vypozorujeme následující:

- u_1 obsahuje nanejvýš jeden ze symbolů a, b, c, d; podobně u_2.
- u_1 obsahuje jeden symbol a u_2 jiný.
- $|u_1| = |u_2|$
- jestliže $A \Rightarrow^+ u_3 A u_4$, pak u_1 a u_3 obsahují tentýž symbol, podobně u_2 a u_4.
- jsou jen následující možnosti:
  - $u_1 \in \{a\}^+$ a $u_2 \in \{b\}^+$,
  - $u_1 \in \{a\}^+$ a $u_2 \in \{d\}^+$,
  - $u_1 \in \{b\}^+$ a $u_2 \in \{c\}^+$,
  - $u_1 \in \{c\}^+$ a $u_2 \in \{d\}^+$,
- V množině neterminálů Π tedy lze přirozeně definovat příslušné třídy $\mathcal{C}_{ab}$, $\mathcal{C}_{ad}$, $\mathcal{C}_{bc}$, $\mathcal{C}_{cd}$. Tyto třídy jsou očividně disjunktní a S nepatří do žádné z nich. Neterminál z jedné třídy se může přepsat na neterminál z jiné třídy jedině v případě pravidla typu $X \longrightarrow v_1 Y v_2$, kde $X \in \mathcal{C}_{ad}$ a $Y \in \mathcal{C}_{bc}$.

- Pravidla $S \longrightarrow \alpha$ jsou jedině v některém z tvarů
  - α je terminální řetězec ($a^k b^k c^m d^m$ nebo $a^k b^m c^m d^k$),
  - α obsahuje jen jeden neterminál a ten je z třídy $\mathcal{C}_{ab}$ či $\mathcal{C}_{cd}$, nebo dva neterminály, kde levější je z třídy $\mathcal{C}_{ab}$ a ten druhý z $\mathcal{C}_{cd}$; v tom případě každé slovo generované z α musí být z jazyka $\{a^k b^k c^m d^m \mid k, m \geq 1\}$.
  - α obsahuje jen jeden neterminál a ten je z třídy $\mathcal{C}_{bc}$ či $\mathcal{C}_{ad}$; v druhém případě pak později může dojít k přepsání neterminálu z třídy $\mathcal{C}_{ad}$ na neterminál z třídy $\mathcal{C}_{bc}$. Zde každé slovo generované z α musí být z jazyka $\{a^k b^m c^m d^k \mid k, m \geq 1\}$.

Vypusťme teď z G pravidla typu $S \longrightarrow a^n b^n c^n d^n$ a rozdělme pravidla (disjunktně) do dvou gramatik G_1, G_2 tak, že G_1 generuje všechna $a^k b^k c^m d^m$ pro $k \neq m$, plus případně $a^n b^n c^n d^n$ pro nějaká n

a G_2 generuje všechna $a^k b^m c^m d^k$ pro $k \neq m$, plus případně $a^n b^n c^n d^n$ pro nějaká n.

Ukážeme, že ve skutečnosti G_1 generuje $a^n b^n c^n d^n$ pro skoro všechna n (tedy až na konečně mnoho výjimek). Podobně ukážeme, že G_2 generuje $a^n b^n c^n d^n$ pro skoro všechna n. To ovšem bude spor s jednoznačností G.

Předpokládejme tedy, že $J = \{n \mid G_1 \text{ negeneruje } a^n b^n c^n d^n\}$ je nekonečná. Očíslujme si pravidla $S \longrightarrow \alpha$ z G_1 jako $r_1, r_2, \ldots, r_s$. Pro každé $n_0 \in J$ odvození v G_1 podle schématu

$S \Rightarrow \alpha \Rightarrow^* a^k b^k c^m d^m$, kde $S \longrightarrow \alpha$ je pravidlo r_1

neodvodí buď slovo typu $a^{n_0} b^{n_0} c^m d^m$ nebo slovo typu $a^k b^k c^{n_0} d^{n_0}$ (jinak by se dalo odvodit i $a^{n_0} b^{n_0} c^{n_0} d^{n_0}$, jak plyne z možných tvarů pravidla r_1, které jsme zmínili výše).

Existuje tedy nekonečná množina $J_1 \subseteq J$ taková, že pro každé $k, m \in J_1$ neexistuje odvození $S \Rightarrow^* a^k b^k c^m d^m$ začínající pravidlem r_1 (tedy odvození podle schématu 1). Dále ovšem pro každé $n_0 \in J_1$ odvození v G_1 začínající pravidlem r_2 neodvodí buď slovo typu $a^{n_0} b^{n_0} c^m d^m$ nebo slovo typu $a^k b^k c^{n_0} d^{n_0}$ (jinak by se dalo odvodit i $a^{n_0} b^{n_0} c^{n_0} d^{n_0}$).

Tedy existuje nekonečná množina $J_2 \subseteq J_1$ taková, že pro každé $k, m \in J_2$ neexistuje odvození $S \Rightarrow^* a^k b^k c^m d^m$ začínající (pravidlem r_1 nebo) pravidlem r_2.

Když takto konstruujeme $J \supseteq J_1 \supseteq J_2 \supseteq \cdots \supseteq J_s$, zjistíme, že pro každé $k \neq m$, kde $k, m \in J_s$, platí, že G_1 nevygeneruje $a^k b^k c^m d^m$, což je spor.

Pro G_2 postupujeme obdobně, ale číslujeme jinak. Nejprve očíslujeme pravidla typu $S \longrightarrow \alpha$, kde α je terminální řetězec nebo obsahuje neterminál z $\mathcal{C}_{bc}$. Pravidlu r_i tohoto typu odpovídá schéma

$S \Rightarrow \alpha \Rightarrow^* a^k b^m c^m d^k$.

Další pořadová čísla přiřadíme (konečně mnoha) schématům

$S \Rightarrow \alpha \Rightarrow^* u_1 A u_2 \Rightarrow u_1 v_1 B v_2 u_2 \Rightarrow a^k b^m c^m d^k$

kde $A \in \mathcal{C}_{ad}$ a $B \in \mathcal{C}_{bc}$

(toto schéma je určeno dvěma pravidly, tedy $S \longrightarrow \alpha$ a $A \longrightarrow v_1 B v_2$).

Opět předpokládejme, že $J = \{n \mid G_2 \text{ negeneruje } a^n b^n c^n d^n\}$ je nekonečná. Pak pro každé $n_0 \in J$ odvození v G_1 podle prvního schématu neodvodí buď slovo typu $a^{n_0} b^{n_0} c^m d^m$ nebo slovo typu $a^k b^k c^{n_0} d^{n_0}$ (jinak by se očividně dalo odvodit i $a^{n_0} b^{n_0} c^{n_0} d^{n_0}$).

Existuje tedy nekonečná množina $J_1 \subseteq J$ taková, že pro každé $k, m \in J_1$ neexistuje odvození $S \Rightarrow^* a^k b^m c^m d^k$ podle prvního schématu. Atd.

Cvičení

Prezentace referátů

Referát č. 9 (Dynamické programování)
Algoritmus (Cocke-Younger-Kasami) pro rozpoznávání bezkontextových jazyků (aplikace metody dynamického programování):

Mějme dánu bezkontextovou gramatiku G v tzv. Chomského normální formě, tedy s pravidly pouze typu $\boxed{X \longrightarrow YZ}$ a $\boxed{X \longrightarrow a}$. Algoritmus pro zadané (terminální) slovo w zjistí, zda $w \in L(G)$.

Nástin: Označme $w = a_1 a_2 \dots a_n$. Systematicky vyplňujeme (dvourozměrné) pole D tak, že na závěr bude $D[i, j]$ $(1 \leq i \leq n,\ 0 \leq j \leq n-i)$ obsahovat množinu právě těch neterminálů X, z nichž lze odvodit $a_i a_{i+1} \dots a_{i+j}$.

Vysvětlete tento algoritmus ilustrací na vhodném příkladu.
(Další podklady je možno najít např. na
`http://www.cs.vsb.cz/jancar/VYCSLOZ/vycsloz.htm`, referát 4.)

Referát č. 10
Vysvětlete pojem regulárních gramatik (RG) a jejich vztah ke konečným automatům. Na vhodných příkladech ilustrujte převody mezi (N)KA a RG a naopak. (Můžete vyjít z materiálu
`http://www.cs.vsb.cz/jancar/TEORET-INF/teoret-inf.pdf`
a/nebo z jiných zdrojů.)

Příklady

Příklad 6.1

Připomeňme si gramatiku G

$$\begin{aligned} R &\longrightarrow T + R \mid T \\ T &\longrightarrow FT \mid F \\ F &\longrightarrow F^* \mid (R) \mid C \\ C &\longrightarrow a \mid b \end{aligned}$$

Ke každému slovu $w \in L(G)$ pochopitelně existuje příslušný derivační strom podle G (kde kořen je ohodnocen R a ohodnocení listů zleva doprava dává slovo w). Připomínáme, že strom je uspořádaný; každý vrchol má své následníky uspořádány „zleva doprava“. Tím je také indukováno uspořádání listů.

Snažte se co nejsrozumitelněji vyjádřit, co to vlastně znamená, že list ℓ_2 je vpravo od listu ℓ_1.

Řekneme, že (obecný) vrchol v *má napravo* list ℓ, jestliže ℓ je nejlevější z listů, které jsou vpravo od všech listů podstromu s kořenem v.

Nakreslete derivační strom pro nějaké (krátké) $w \in L(G)$, kde má nějaký vrchol ohodnocený T napravo list ohodnocený $+$.

Může mít v derivačním stromě pro (nějaké) $w \in L(G)$ vrchol ohodnocený T napravo list ohodnocený jinak než $+$? Ukažte všechny možnosti. Může se také stát, že takový vrchol napravo žádný list nemá?

(Poznámka. Toto je příklad avízovaný v zápisu přednášky u Instrukce 9.)

Příklad 6.2

Navrhněte gramatiku, která řeší Příklad 1 z ukázkové druhé zápočtové písemky. Přitom nejprve navrhněte gramatiky G_1, G_2 pro jazyky L_1, L_2 a z nich jsme pak jednoduše získejte gramatiku pro $L_1 \cdot L_2$ (jak je naznačeno v části 5.2., s. 166). Uvědomte si, proč je obecně důležité zajistit, aby množiny neterminálů gramatik G_1 a G_2 byly disjunktní.

Příklad 6.3

Zkuste zjistit, zda pro následující gramatiku G je $L(G) \neq \emptyset$, čili zda lze z neterminálu S vygenerovat alespoň jedno terminální slovo.

$S \longrightarrow aS \,|\, AB \,|\, CD$
$A \longrightarrow aDb \,|\, AD \,|\, BC$
$B \longrightarrow bSb \,|\, BB$
$C \longrightarrow BA \,|\, ASb$
$D \longrightarrow ABCD \,|\, \varepsilon$

Uměli byste svůj postup zobecnit, tedy navrhnout algoritmus, který toto zjišťuje pro jakoukoli zadanou bezkontextovou gramatiku?

Příklad 6.4

Navrhněte bezkontextovou gramatiku generující jazyk všech palindromů v abecedě $\{a, b\}$, jejichž délka je násobkem tří.

(Jedná se tedy o jazyk $L = \{\, w \in \{a,b\}^* \mid w = w^R \text{ a } (|w| \bmod 3) = 0 \,\}$.)

Zkuste nejprve najít gramatiku používající jediný neterminál. Poté popřemýšlejte, jestli použitím více neterminálů dokážete počet potřebných pravidel zmenšit.

(Nakonec porovnejte s řešením Cvičení 4.13. na s. 137.)