Týden 5

Přednáška

Na rozšiřující přednášce minulý týden jsme se věnovali zejména

algoritmu, který k zadanému konečnému automatu sestrojí ekvivalentní regulární výraz

(tedy k automatu A sestrojí reg. výraz α tak, že $L(A) = [\alpha]$).

Nejprve jsme zavedli *generalizovaný konečný automat* (GKA), což je struktura $A = (Q, \Sigma, \delta, q_0, F)$, která se od (deterministického) KA liší *typem přechodové funkce*; máme zde $\delta : (Q \times Q) \to RV(\Sigma)$, což lze chápat tak, že pro každou dvojici stavů (q, q') (včetně případu $q = q'$) vedeme hranu z q do q' ohodnocenou regulárním výrazem $\delta(q, q')$ (což může být i regulární výraz $\emptyset$ či ε).

Ternární relaci $q \xrightarrow{w} q'$ pak definujeme induktivně takto:

1/ jestliže $w \in [\delta(q, q')]$, pak $q \xrightarrow{w} q'$;

2/ jestliže $q \xrightarrow{u} q'$ a $q' \xrightarrow{v} q''$, pak $q \xrightarrow{uv} q''$.

Stejně jako dříve definujeme $L(A) = \{w \in \Sigma^* \mid q_0 \xrightarrow{w} F\}$.

Každý konečný automat (ať už deterministický či nedeterministický) lze snadno převést na ekvivalentní GKA (promyslete si!); stačí nám tedy ukázat konstrukci, která k danému GKA sestrojí ekvivalentní regulární výraz.

V prvé řadě (snadno) upravíme daný GKA tak, že do jeho počátečního stavu nevedou žádné hrany, čímž rozumíme, že každá jeho vstupní hrana má přiřazen regulární výraz $\emptyset$; navíc automat upravíme tak, že má jen jeden přijímající stav a ten nemá výstupní hrany (resp. jeho výstupní hrany jsou označeny $\emptyset$).

Nyní postupně vypouštíme stavy, které nejsou počáteční ani přijímající. Když vypustíme stav q, změníme aktuální δ na δ' takto: pro každou dvojici (q', q'') zbylých stavů definujeme $\delta'(q', q'') = \delta(q', q'') + \delta(q', q)(\delta(q, q)^*)\delta(q, q'')$.

(Diskutovali jsme korektnost tohoto postupu a ilustrovali jej na příkladu.)

Jako další věc jsme si ukázali jednoduchý převod konečného automatu na bezkontextovou gramatiku, čímž jsme prokázali, že

každý regulární jazyk je bezkontextovým jazykem (ale ne naopak).

‘Překladač’ sestrojující k regulárnímu výrazu ekvivalentní konečný automat

Připomeňme si 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 pro zjednodušení vynechali symboly $\emptyset$ a ϵ. Naše G tedy generuje jazyk v abecedě $\Sigma = \{\, a, b, +, {}^*, (,) \,\}$, jehož prvky jsou příslušné regulární výrazy.
Pokusíme se navrhnout algoritmus, který k zadanému regulárnímu výrazu sestrojí derivační strom podle uvedené gramatiky. Ukáže se, že se velmi přirozeně nabízí využití dynamické datové struktury *zásobník*.
Jako příklad řetězce z $L(G)$ vezměme

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

Proveďme 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 není zkonstruovat tento konkrétní strom (který díky jednoduchosti vstupního řetězce jistě zkonstruujete téměř bez přemýšlení), ale ilustrovat algoritmickou metodu, která stojí v pozadí reálné syntaktické analýzy v překladačích. Po zkonstruování stromu si připomeneme, jak lze takovouto syntaktickou strukturu využít pro generování „cílového kódu“, čímž v našem případě rozumíme (program konstruující) konečný automat, který přijímá jazyk reprezentovaný 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 $X \longrightarrow Y_1 Y_2 \ldots Y_n$
(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 teď omezujeme na ono 'vytušení', později se na to podíváme pořádněji.)

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

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

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

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

Nyní se uplatní Instrukce 5 a dostáváme

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

Máme tedy na vrcholu zásobníku pravou stranu pravidla $T \longrightarrow FT$; opět lze vytušit (později dokážeme), 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

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

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. Na cvičení byste měli mj. zjistit, 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

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

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

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

Máme na vrcholu pravou stranu pravidla $R \longrightarrow T + R$; dá se (vytušit a) 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

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

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

(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

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

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

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

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

$p_{21} : [T, p_{20}]$
$p_{20} : [F, p_{18}, p_{19}]$
$p_{18} : [F, p_{15}, p_{16}, p_{17}]$
$p_{16} : [R, p_{12}, p_{13}, p_{14}]$
$p_{12} : [T, p_6, p_7] \quad p_{14} : [R, p_{11}]$
$p_6 : [F, p_2] \quad p_7 : [T, p_5] \quad p_{11} : [T, p_{10}]$
$\qquad\qquad p_5 : [F, p_4] \quad p_{10} : [F, p_9]$
$p_2 : [C, p_1] \quad p_4 : [C, p_3] \quad p_9 : [C, p_8]$
$p_{15} : [(] \quad p_1 : [a] \quad p_3 : [a] \quad p_{13} : [+] \quad p_8 : [b] \quad p_{17} : [)] \quad p_{19} : [^*]$

v níž je celé vstupní slovo přečteno 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.

Díky postupu naší konstrukce by mělo být 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; podívejme se na něj nyní podrobněji.

Funkce First a Follow; LR(1) gramatika

Uvažujme obecnou bezkontextovou gramatiku $G = (\Pi, \Sigma, S, P)$ a definujme funkci

$$First : (\Pi \cup \Sigma)^* \to \mathcal{P}(\Sigma \cup \{\varepsilon\})$$

následující induktivní definicí (která je, jako obvykle, i návodem k algoritmu). Není těžké ověřit, že $First(\alpha)$ obsahuje právě ty terminály a, pro něž platí $\alpha \Rightarrow^* a\beta$ pro nějaké β; $First(\alpha)$ navíc obsahuje ε, pokud $\alpha \Rightarrow^* \varepsilon$.

- $First(\varepsilon) = \{\varepsilon\}$
- $a \in \Sigma, \beta \in (\Pi \cup \Sigma)^* \implies First(a\beta) = \{a\}$
- $(X \to \beta) \in P, a \in First(\beta) \implies a \in First(X)$ (tedy $First(\beta) \subseteq First(X)$)
- $a \in \Sigma, a \in First(X), \beta \in (\Pi \cup \Sigma)^* \implies a \in First(X\beta)$
- $\varepsilon \in First(X), \beta \in (\Pi \cup \Sigma)^* \implies First(\beta) \subseteq First(X\beta)$

Funkci $First$ můžeme přirozeně rozšířit i na podmnožiny množiny $(\Pi \cup \Sigma)^*$:
$First(A) = \bigcup_{\alpha \in A} First(\alpha)$.

Dále definujeme funkci $Follow : \Pi \to \mathcal{P}(\Sigma \cup \{\varepsilon\})$, kde $Follow(X)$ obsahuje $a \in \Sigma$ právě tehdy, když $S \Rightarrow^* \alpha X a \beta$ pro nějaké α, β; $Follow(X)$ dále obsahuje ε právě tehdy, když $S \Rightarrow^* \alpha X$ pro nějaké α. Následuje induktivní definice (návod k algoritmu).

- $\varepsilon \in Follow(S)$
- $(X \to \alpha Y \beta) \in P \implies Follow(Y) \supseteq First(\beta \cdot Follow(X))$

Pozorování. Když $Follow(X) \cap First(u) = \emptyset$, tak neplatí $S \Rightarrow^* \alpha X u$.

Jinými slovy: když se v našem postupu na vrcholu zásobníku objeví pravá strana β pravidla $X \to \beta$, ale aktuální čtený vstupní symbol nepatří do $Follow(X)$, děláme jen dobře, že neredukujeme podle $X \to \beta$ (tato redukce by nemohla vést k úspěšné konstrukci derivačního stromu).

Když ovšem onen čtený symbol patří do $Follow(X)$, nemusí to ještě znamenat, že redukce podle $X \to \beta$ je v pořádku. (Proč?)

Před exaktním vyřešením konfliktů se znovu podívejme na náš postup konstrukce derivačního stromu zdola-nahoru. Jedno důležité pozorování: postupně jsme redukovali podle pravidel gramatiky, což dává posloupnost $rule_1, rule_2, \ldots, rule_k$ pravidel, v níž se každé pravidlo gramatiky může vyskytovat vícekrát (a také 0-krát). Když bychom nyní konstruovali *pravou* derivaci (z počátečního neterminálu) a používali postupně pravidla $rule_k, rule_{k-1}, \ldots, rule_1$ (obrátili jsme pořadí), tak odvodíme výchozí vstupní slovo. Konstrukci našeho stromu tedy můžeme chápat jako konstrukci pravé derivace vstupního slova pozpátku. Při úspěšné konstrukci je tedy v každém kroku řetězec αu, kde α je posloupnost neterminálů a terminálů uložená v zásobníku (pravý konec α je na vrcholu) a kde u je (dosud nepřesunutý) zbytek vstupního slova, příslušnou pravou větnou formou, tedy z počátečního S (v našem konkrétním případě R) se αu odvodí pravým odvozením.

Když jsme si takto náš postup důkladněji promysleli, všimneme si, že všechny případné konflikty v našem postupu by byly vyřešeny, kdyby platila např. následující (postačující) podmínka:

> Pro jakákoli dvě různá pravidla $X_1 \to \beta$, $X_2 \to \gamma\delta$, kde γ je neprázdným sufixem β nebo β je sufixem γ (může také být $\delta = \varepsilon$), platí:
>
> $Follow(X_1) \cap First(\, \delta \cdot Follow(X_2)\,) = \emptyset$

Když za platnosti této podmínky se při našem postupu na vrcholu zásobníku objeví pravá strana nějakého pravidla či dokonce najednou pravé strany více pravidel, pak aktuálně čtený symbol rozhodne, zda a podle jakého pravidla redukovat či zda přesunout další symbol do zásobníku (či ohlásit chybu).

Na cvičení máte zjistit, zda uvedená podmínka platí pro naši gramatiku generující regulární výrazy. Měli byste zjistit, že ji zcela nesplňuje. Zbývající konflikty ovšem lze vyřešit např. díky následujícímu pozorování.

S výjimkou počáteční situace (kdy je zásobník prázdný) a závěrečné úspěšné situace (kdy zásobník obsahuje pouze počáteční neterminál), musí být v úspěšném výpočtu na vrcholu zásobníku vždy neprázdný prefix pravé strany nějakého pravidla. (Umíte to dokázat?)

Užitím tohoto pozorování lze snadno ukázat správnost naší instrukce 7 (je-li na vrcholu zásobníku FT, zredukujeme podle $T \longrightarrow FT$). (Proč nemůže redukce podle $R \longrightarrow T$ ani přesun dalšího symbolu do zásobníku vést k úspěchu?) Tímto je exaktně ověřena správnost našich „vytušených“ instrukcí.

Poznámka. Naše gramatika pro regulární výrazy je speciálním případem tzv. LR(1) gramatiky.

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 již 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($v.succ_1$);
}

Cvičení

Příklad 5.1

Případně dokončete konstrukce bezkontextových gramatik z dřívějšího cvičení.
Promyšlením tvarů slov navíc zkonstruujte gramatiky pro

- jazyk palindromů $\{w \in \{a, b\}^* \mid w = w^R\}$,
- jazyk posloupností v abecedě $\{\,(,),[,]\,\}$, které odpovídají správnému uzávorkování.

Příklad 5.2

Uvažujme jazyk sestávající ze všech booleovských formulí s proměnnými $x1, x2, \ldots$ a logickými spojkami $\neg, \wedge, \vee$; mohou se v nich používat závorky $(,)$, ale není nutné plně závorkovat. Každá taková formule je tedy řetězcem v abecedě

$$\Sigma = \{\, x, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \neg, \wedge, \vee, (,)\,\};$$

jako příklad může sloužit řetězec $(\neg x15 \vee x2 \wedge x5) \wedge \neg x21 \vee \neg(x2 \vee x5)$, který do jazyka patří. (Samozřejmě zde můžeme preferovat přehlednější zápis $(\neg x_{15} \vee x_2 \wedge x_5) \wedge \neg x_{21} \vee \neg(x_2 \vee x_5)$, ale to není podstatné.)

Navrhněte co nejjednodušší bezkontextovou gramatiku generující uvedený jazyk.

Takto navržená (jednoduchá) gramatika asi není jednoznačná; ověřte.

Zkonstruujte pak pro stejný jazyk jednoznačnou gramatiku, u níž derivační stromy přirozeně odpovídají obvyklé prioritě operátorů: negace váže silněji než konjunkce a konjunkce váže silněji než disjunkce. (Zkuste postupovat podobně jako při konstrukci bezkontextové gramatiky generující regulární výrazy, tedy přidáváním vhodných neterminálů.)

Příklad 5.3

Připomeňme si gramatiku G

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

Zkonstruujte postupem zdola-nahoru (pravá derivace pozpátku) derivační strom pro slovo $ba(a+b)^*$.

Zkonstruujte množiny $First(X)$ a $Follow(X)$ pro všechny neterminály $X \in \{R, T, F, C\}$.

Ověřte, zda pro naši gramatiku platí následující podmínka:

> Pro jakákoli dvě různá pravidla $X_1 \to \beta$, $X_2 \to \gamma\delta$, kde γ je neprázdným sufixem β nebo β je sufixem γ (může také být $\delta = \varepsilon$), platí:
> $Follow(X_1) \cap First(\, \delta \cdot Follow(X_2)\,) = \emptyset\,.$

Tato podmínka má sloužit k rozhodnutí případných konfliktů, kdy není jasné, zda a podle jakého pravidla redukovat či zda přesunout další symbol do zásobníku. Pokud podmínka neplatí, vysvětlete, jak je možné dořešit zbývající konflikty, abychom docílili deterministického průběhu tzv. LR(1) analýzy (tedy konstrukce derivačního stromu ilustrované na přednášce).

Příklad 5.4

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

$$\begin{aligned} S &\longrightarrow aS \,|\, AB \,|\, CD \\ A &\longrightarrow aDb \,|\, AD \,|\, BC \\ B &\longrightarrow bSb \,|\, BB \\ C &\longrightarrow BA \,|\, ASb \\ D &\longrightarrow ABCD \,|\, \varepsilon \end{aligned}$$

Zobecněte svůj postup, tedy navrhněte algoritmus, který toto zjišťuje pro jakoukoli zadanou bezkontextovou gramatiku.

Přitom postupujte obecněji tak, že algoritmus pro danou $G = (\Pi, \Sigma, S, P)$ sestrojí množinu $\mathcal{T}_G = \{X \in \Pi \mid \exists u \in \Sigma^* : X \Rightarrow^* u\}$. Množinu $\mathcal{T}_G$ nejdříve zadejte jednoduchou induktivní definicí:

- $(X \to v) \in P, v \in \Sigma^* \Longrightarrow X \in \mathcal{T}_G$,
-

Příklad 5.5

Navrhněte bezkontextovou gramatiku G tak, že $L(G) = L_1 \cdot L_2$ kde
$L_1 = \{w \in \{a,b\}^* \mid w$ obsahuje podřetězec $bab\}$
$L_2 = \{a^n u \mid u \in \{a,b\}^*$ a $1 \leq n \leq$ délka$(u) \leq 2n\ \}$
Přitom použijte S jako počáteční neterminál, a pokud možno jen jedno pravidlo s S na levé straně. (Snažte se o přehledný návrh využívající co nejméně pravidel.)
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 5.6

(Nepovinně.)
Vraťme se ke gramatice G z příkladu 5.3.
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?
Jaké vidíte souvislosti s příkladem 5.3.?

Příklad 5.7

(Nepovinně.)
Navrhněte bezkontextovou gramatiku generující jazyk všech palindromů v abecedě $\{a,b\}$, jejichž délka je násobkem tří.
(Jedná se tedy o jazyk $L = \{\, w \in \{a,b\}^* \mid w = w^R$ 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.)