Automaty a formální jazyky I.

podle přednášek z roku 1991 zpracoval Josef Pojsl

Obsah

1 Pojmy jazyka a gramatiky

V této kapitole budou uvedeny základní pojmy teorie automatů a formálních jazyků a způsob práce s nimi.

1.1 Jazyky

Definice 1.1
Libovolná neprázdná konečná množina V je *abecedou.* Její prvky nazýváme *znaky* nebo *symboly.*

Definice 1.2
Slovo (řetězec) nad abecedou V je libovolná posloupnost konečné délky tvořená symboly z V. Slova zapisujeme bez čárek, tedy posloupnost $w = \{a_i\}_{i=1}^{n}$, kde $a_i \in V$, zapíšeme jako slovo $w = a_1 \cdots a_n$.
Délka slova w je délka w jako posloupnosti, značíme $|w| = n$.
Prázdné slovo, tedy posloupnost délky nula, značíme ε (v jiné literatuře také λ nebo e), $|\varepsilon| = 0$.
Symbolem V^* značíme množinu všech slov nad abecedou V, symbolem V^+ označujeme množinu $V^* \setminus \{\varepsilon\}$.

Definice 1.3
Jazyk L nad abecedou V je libovolná množina slov, $L \subseteq V^*$.

Příklad 1.4
Nechť $V = \{a, b\}$ je abeceda. Jazyk L nad touto abecedou může být definovaný takto: $L = \{w : w = a^i b^i, i \geq 0\}$. Potom slova $\varepsilon, ab, aabb$ patří do jazyka L, ale slova b, ba nejsou z jazyka L.

Definice 1.5
Definujme operaci *zřetězení:* $V^* \times V^* \longrightarrow V^*$ takto:

$$\left.\begin{array}{l} w_1 = a_1 \cdots a_n \\ w_2 = b_1 \cdots b_m \end{array}\right\} \Rightarrow w_1 w_2 = a_1 \cdots a_n b_1 \cdots b_m$$

Příklad 1.6
(i) $\varepsilon a = a\varepsilon = a$ (ii) $a^i a^j = a^{i+j}$

Definice 1.7
Definujeme také operaci *zřetězení jazyků* $L_1, L_2 \subseteq V^*$: $2^{V^*} \times 2^{V^*} \longrightarrow 2^{V^*}$ takto:

$$L_1 L_2 = \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$$

Definice 1.8
Nechť S je jazyk. Pak jazyk S^* definovaný $S^* = \bigcup_{i=0}^{\infty} S^i$ nazýváme *uzávěr jazyka* S.

Máme tedy definovány jazyky, které však mohou být nekonečné. Budeme dále hledat nástroje, jak tyto jazyky konečně reprezentovat.
Za nástroje budeme používat *algoritmy* nebo *procedury.* Ty mohou rozpoznávat příslušnost slova do jazyka.

algoritmus — program, který vždy zastaví a pro slovo w na vstupu vrátí odpověď `ano`, jestliže $w \in L$, a odpověď `ne`, jestliže $w \notin L$. Řekneme, že tento *algoritmus rozpoznává jazyk* L, který pak nazveme *rekurzivní jazyk.*

procedura — program, který pro vstupní slovo $w \in L$ se zastaví a vrátí odpověď `ano` a pro vstup $w \notin L$ se buď zastaví a vrátí `ne` nebo se vůbec nezastaví. Řekneme, že tato *procedura částečně rozpoznává jazyk* L, který pak nazveme *rekurzivně vyčíslitelný jazyk.*

Lze také vyžadovat existenci procedury, která by všechna slova jazyka generovala. Tento požadavek je ekvivalentní s existencí rozpoznávací procedury. Pro jazyk L tedy existuje procedura jej generující právě tehdy, když je L rekurzivně vyčíslitelný jazyk.

Jazyky později rozdělíme do jistých tříd podle toho, zda a jakým způsobem se dají mechanicky rozpoznávat, příp. generovat. O těchto třídách se pak budeme snažit zjistit, zda jsou uzavřené na jisté operace na jazycích.

Definice 1.9

Nechť $\mathcal{L} \subseteq 2^{\Sigma^*}$ je třída jazyků. Řekneme, že tato třída je *uzavřená* vzhledem k nějaké operaci, jestliže platí: patří-li všechny operandy (jazyky) dané operace do třídy jazyků $\mathcal{L}$, pak do této třídy patří i výsledek operace (jazyk).

Těmito operacemi mohou být všechny množinové operace, protože jazyky jsou také množinami. Bude-li abeceda Σ pevná, můžeme za takovou operaci vzít i doplněk jazyka L, tedy jazyk $\Sigma^* \setminus L$. S vědomím, že se pohybujeme v universu slov Σ^*, budeme někdy značit doplněk jazyka L jako $\overline{L}$. Také operaci uzávěru jazyka je možno testovat na uzavřenost v jistých třídách.

Definujeme dále operaci *zřetězení* jazyků, o které budeme také někdy dokazovat uzavřenost jistých tříd jazyků.

Definice 1.10

Nechť U a V jsou dva jazyky. *Zřetězení* jazyků U, V značíme a definujeme takto: $UV = \{uv : u \in U \wedge v \in V\}$.

1.2 Gramatiky

Definice 1.11

Gramatika je čtveřice $G = (N, \Sigma, P, S)$ s následujícím významem:

N — množina *neterminálů,* které figurují jako tzv. *metasymboly*

Σ — množina *terminálů;* platí $N \cap \Sigma = \emptyset$ a označíme $V = N \cup \Sigma$, V nazýváme *celkovou abecedou gramatiky* G

P — množina *pravidel* $P \subseteq V^*NV^* \times V^*$; jde tedy o dvojice slov, přičemž první z nich obsahuje alespoň jeden neterminální symbol

S — *počáteční symbol (kořen)* gramatiky G, je to neterminál ($S \in N$)

Poznámka 1.12

Pravidla typu $[\alpha, \beta]$ gramatiky G budeme zapisovat ve tvaru $\alpha \to \beta$

Definice 1.13

Budeme nyní definovat *relaci odvození* v gramatice G, což bude relace na množině slov celkové abecedy G: $\Rightarrow_G \subseteq V^{*2}$. Řekneme, že *slovo* $y \in V^*$ *se dá (v jednom kroku) odvodit (resp. derivovat) ze slova* $x \in V^*$ v gramatice $G = (N, \Sigma, P, S)$, $x \Rightarrow_G y$, jestliže existuje pravidlo $(\alpha \to \beta) \in P$ a slova $\gamma \in V^*$ a $\delta \in V^*$ tak, že $x = \gamma\alpha\delta$ a $y = \gamma\beta\delta$.

Poznámka 1.14

Zavedeme následující konvenci na symbolická značení:

$$\begin{aligned} \text{neterminály} &: A, B, \ldots, Z \\ \text{terminály} &: a, b, c, \ldots \\ \text{řetězce terminálů} &: u, v, w, x, y, z \\ \text{obecné řetězce} &: \alpha, \beta, \gamma, \ldots \end{aligned}$$

Definice 1.15

Pomocí relace odvození v jednom kroku zavedeme relaci *odvození,* $\Rightarrow_G^*$, jako tranzitivní a reflexivní obal relace $\Rightarrow_G$. Podobně *netriviální odvození,* $\Rightarrow_G^+$, je pouze tranzitivním obalem relace $\Rightarrow_G$.

Příklad 1.16
Nechť $G_1 = (\{A, S\}, \{a, b\}, P, S)$, kde $P = \{S \to aAb, aA \to aaAb, A \to \varepsilon\}$, je gramatika. Pak v ní existují např. tato odvození:

$$\begin{aligned} aaAbb &\Rightarrow_{G_1} aabb \\ aaAbb &\Rightarrow_{G_1} aaaAbbb \end{aligned}$$

Definice 1.17
Všem slovům $\alpha \in V^*$ takovým, že $S \Rightarrow_G^* \alpha$, říkáme *větné formy* gramatiky G. Je-li navíc $\alpha \in \Sigma^*$, pak slovo α nazýváme *větou* gramatiky G.

Jazyk generovaný gramatikou G je množina všech vět gramatiky G, značíme $L(G) = \{w \in \Sigma^* : S \Rightarrow_G^* w\}$.

Gramatiky G_1 a G_2 nazýváme *ekvivalentní,* pokud $L(G_1) = L(G_2)$.

Příklad 1.18
V gramatice G_1 z příkladu 1.16 je

$$\begin{aligned} S &\Rightarrow_{G_1} aAb \Rightarrow_{G_1} ab \\ S &\Rightarrow_{G_1} aAb \Rightarrow_{G_1} aaAbb \Rightarrow_{G_1} aabb \end{aligned}$$

Jsou tedy aAb, $aaAbb$ její větné formy a ab a $aabb$ její věty. Celkem $L(G_1) = \{a^i b^i : i \in \mathbf{N}\}$.

Gramatika $G_2 = (\{S\}, \{a, b\}, \{S \to aSb, S \to ab\}, \{S\})$ je ekvivalentní gramatice G_1.

1.3 Chomského hierarchie

Některé speciální jazyky lze generovat speciálními gramatikami, tedy gramatikami s pravidly speciálního typu. Tyto gramatiky nepoužívají všechny možnosti nabízené obecnou definicí gramatiky 1.11. Je možno se setkat se širokou škálou takových typů gramatik, my však uvedeme pouze nejznámější klasifikaci, která se nazývá *Chomského*[1] *hierarchie.*

Definice 1.19

1. Gramatiku, která odpovídá obecné definici 1.11 bez dalších omezení, nazveme *gramatika typu 0.* Jazyk generovaný touto gramatikou nazveme *jazyk typu 0.*

2. Gramatiku G nazveme *gramatika typu 1 (kontextová, CS*[2]*),* když pro všechna její pravidla $\alpha \to \beta$ platí: $|\alpha| \leq |\beta|$. Ekvivalentní definice: všechna pravidla jsou tvaru $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$, kde $\alpha_1, \alpha_2 \in V^*$, $\beta \in V^+$ a A je neterminál.

   Jazyk generovaný takovou gramatikou se nazývá *jazyk typu 1 (kontextový, CS).*

3. Gramatiku G nazveme *gramatika typu 2 (bezkontextová, CF*[3]*),* když všechna její pravidla jsou tvaru $A \to \beta$, kde A je neterminál a $\beta \in V^+$.

   Jazyk generovaný takovou gramatikou se nazývá *jazyk typu 2 (bezkontextový, CF).*

4. Gramatiku G nazveme *gramatika typu 3 (regulární, levolineární),* když všechna její pravidla jsou tvaru $A \to \beta$, kde A je neterminál a $\beta \in V^+$ je řetězec tvaru a nebo aB.

   Jazyk generovaný takovou gramatikou nazýváme *jazyk typu 3 (regulární).*

Příklad 1.20
Gramatika G_3

$$\begin{aligned} G_3: \quad & S \to a \mid b \mid \cdots \mid z \\ & S \to aA \mid bA \mid \cdots \mid zA \\ & A \to 0 \mid 1 \mid \cdots \mid 9 \mid a \mid b \mid \cdots \mid z \mid 0A \mid 1A \mid \cdots \mid 9A \end{aligned}$$

[1] Čteme [čomského]
[2] angl. *context sensitive*
[3] angl. *context free*

je regulární a generuje stejný jazyk jako G_4

$$\begin{array}{ll} G_4: & S \to L \\ & S \to SL \mid SN \\ & L \to a \mid b \mid \cdots \mid z \\ & N \to 0 \mid 1 \mid \cdots \mid 9 \end{array}$$

Lemma 1.21
Nechť G je kontextová (resp. bezkonetxtová, regulární) gramatika. Pak existuje kontextová (resp. bezkontextová, regulární) gramatika G_1 taková, že $L(G) = L(G_1)$ a počáteční neterminál gramatiky G_1 se nevyskytuje na žádné pravé straně pravidla G_1.

Důkaz: Z původní gramatiky $G = (N, \Sigma, P, S)$ sestrojíme gramatiku G_1 takto: Zvolíme neterminál S_1 neobsažený v G, pak $G_1 = (N \cup \{S_1\}, \Sigma, P_1, S_1)$, kde $P_1 = P \cup \{S_1 \to \alpha : (S \to \alpha) \in P\}$. (Nemůžeme zavést pouze pravidlo $S_1 \to S$, protože by porušovalo regularitu.) □

Poznámka 1.22
Je-li L jazyk CS (resp. CF, reg.), pak také $L \cup \{\varepsilon\}$ a $L \setminus \{\varepsilon\}$ jsou jazyky CS (resp. CF, reg.).

Definice 1.23
Gramatika G je *rekurzivní,* pokud existuje algoritmus, který pro každé slovo w určuje, zda $w \in L(G)$.

Věta 1.24
Je-li gramatika G kontextová, pak je rekurzivní.

Důkaz: (Hezky proveden v [HopUll]—věta 9.7 na straně 227.) Příslušný algoritmus nejprve testuje, zda je vstupní slovo w prázdné ($w = \varepsilon$). Víme, že $\varepsilon \in L(G)$, právě když $(S \to \varepsilon)$ je pravidlo G (pro počáteční neterminál S).

Nechť je tedy $w \in \Sigma^+$. Označíme délku slova w jako n ($|w| = n \geq 1$). Budeme konstruovat množiny T_m^n obsahující všechny větné formy gramatiky G kratší nebo stejné délky jako n, které se dají odvodit nejvýše v m krocích. Tedy

$$T_m^n = \{\alpha : \alpha \in (N \cup \Sigma)^* \wedge S \Rightarrow_G^{m'} \alpha \wedge |\alpha| \leq n \wedge m' \leq m\}.$$

Takové množiny budeme konstruovat induktivně. Zřejmě $T_0 = \{S\}$ a

$$T_m^n = T_{m-1}^n \cup \{\alpha : \exists \beta \in T_{m-1}^n : \beta \Rightarrow_G \alpha \wedge |\alpha| \leq n\},$$

protože gramatika G je kontextová a jakmile je dosaženo v odvození jisté délky slova, výsledná věta už nemůže být kratší.

Zřejmě platí, že pokud $S \Rightarrow_G^* \alpha$ a $|\alpha| \leq n$, pak existuje m takové, že $\alpha \in T_m^n$.

Víme, že $T_{m-1}^n \subseteq T_m^n$. Dále, hodnota T_m^n je funkcí pouze T_{m-1}^n a množiny pravidel P gramatiky G: $T_m^n = f(T_{m-1}^n, P)$. P je při induktivní konstrukci vzhledem k m konstantní. Stane-li se tedy jednou, že $T_m^n = T_{m-1}^n$, pak již pro všechna $m'' \geq m$ platí: $T_{m''}^n = T_{m-1}^n$ a algoritmus tedy může skončit. To však někdy jistě nastane, protože slov délky nejvýše n tvořených z *konečné* abecedy $N \cup \Sigma$ je konečný počet. □

Příklad 1.25
Nechť gramatika G (kontextová) je zadána následujícími pravidly:

$$\begin{array}{lll} G: & \text{(i)} & S \to aSBC \\ & \text{(ii)} & S \to aBC \\ & \text{(iii)} & SB \to BC \\ & \text{(iv)} & aB \to ab \\ & \text{(v)} & bB \to bb \\ & \text{(vi)} & bC \to bc \\ & \text{(vii)} & cC \to cc \end{array}$$

Pokusíme se zjistit, zda slovo $w = abca$ je větou gramatiky G. Budeme tedy konstruovat množiny $T_i^{|w|} = T_i^4$:

$$
\begin{aligned}
T_0^4 &= \{S\} \\
T_1^4 &= \{S, aSBC, aBC\} \\
T_2^4 &= \{S, aSBC, aBC, abC\} \\
T_3^4 &= \{S, aSBC, aBC, abC, abc\} \\
T_4^4 &= \{S, aSBC, aBC, abC, abc\}
\end{aligned}
$$

Vidíme, že $T_3^4 = T_4^4$, tedy algoritmus končí. Slovo w se v množinách neobjevilo, proto dostáváme správný výsledek, že $abca = w \notin L(G) = \{a^n b^n c^n : n \in \mathbf{N}\}$.

2 Konečné automaty

V této kapitole zavedeme formální výpočetní prostředek zvaný konečný automat (KA). Uvedeme dva druhy KA: deterministické a nedeterministické. Později ukážeme souvislot mezi konečnými automaty a jazyky typu 3 (regulární).

2.1 Deterministické KA

Deterministický KA (zvaný též zkráceně pouze KA) definujeme následovně:

Definice 2.1
Konečný automat M nad abecedou Σ je pětice $(K, \Sigma, \delta, q_0, F)$, kde jednotlivé komponenty mají tento význam:

K — konečná neprázdná množina *stavů*

Σ — konečná neprázdná množina *symbolů;* abeceda

δ — *přechodová funkce;* $\delta : K \times \Sigma \to K$

$q_0 \in K$ — *počáteční stav*

$F \subseteq K$ — *množina koncových stavů*

Sémantiku konečných automatů lze definovat dvěma způsoby. První z nich používá tzv. *zobecněnou přechodovou funkci,* a druhý tzv. *konfigurace automatu.*

Definice 2.2
Z přechodové funkce δ lze induktivně zkonstruovat *zobecněnou přechodovou funkci* $\hat{\delta} : K \times \Sigma^* \to K$.

$$
\begin{aligned}
\hat{\delta}(q, a) &= \delta(q, a) && a \in \Sigma \\
\hat{\delta}(q, xa) &= \delta(\hat{\delta}(q, x), a) && a \in \Sigma \; x \in \Sigma^+
\end{aligned}
$$

Místo $\hat{\delta}$ lze také psát pouze δ. Na symbolech se totiž funkce δ i $\hat{\delta}$ rovnají, a u slov delších než 1 symbol půjde automaticky o zobecnění δ.

Definice 2.3 Sémantika KA I.
Řekneme, že konečný automat $M = (K, \Sigma, \delta, q_0, F)$ *akceptuje (rozpoznává, přijímá) slovo* w, jestliže existuje koncový stav $p \in F$ takový, že $\delta(q_0, w) = p$.
Dále $T(M)$ bude značení pro *jazyk akceptovaný automatem* M, tedy

$$T(M) = \{w \in \Sigma^* : M \text{ akceptuje } w\}. \tag{1}$$

Definice 2.4
Konfigurací automatu M nazveme libovolnou dvojici $[q, x]$, kde $q \in K$ je stav a $x \in \Sigma^*$ je slovo.
Relaci $\vdash$ na množině konfigurací automatu M *(krok výpočtu)* definujeme takto: $[q, ax] \vdash [p, x]$, pokud $\delta(q, a) = p$.

Definice 2.5 SÉMANTIKA KA II.

Řekneme, že konečný automat $M = (K, \Sigma, \delta, q_0, F)$ *akceptuje (rozpoznává, přijímá) slovo* w, jestliže existuje koncový stav $p \in F$ takový, že $[q_0, w] \vdash^* [p, \varepsilon]$.

Jazyk akceptovaný automatem M již definujeme stejně jako v definici 2.3.

Definice 2.6

Dva konečné automaty M a N nazveme *ekvivalentní,* jestliže $T(M) = T(N)$.

Poznámka 2.7

Funkce δ může být (a také často bývá) parciální.

Konečné automaty lze reprezentovat několika různými způsoby.

a) pětice uvedená v definici 2.1

b) tabulka přechodové funkce δ se specifikací počátečního a koncových stavů

c) stavový diagram

d) výpočetní strom

Tyto reprezentace bude nejlépe ukázat na příkladě.

Příklad 2.8

Nechť je dán konečný automat

$$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_0\}).$$

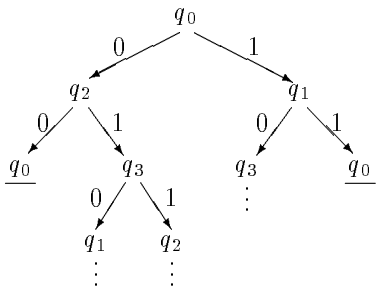
Přechodovou funkci δ specifikuje následující tabulka:

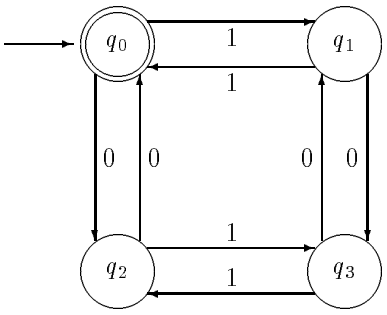
$\leftrightarrow$

	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Šipka směrem k q_0 znamená, že jde o počáteční stav, a šipka směrem od q_0 znamená, že je to zároveň jediný koncový stav.

Následuje ukázka stavového diagramu a výpočetního stromu pro tento automat.





Ve výpočetním stromu se snažíme zachytit všechny možné posloupnosti stavů. každá větev dává právě jedno odvození, a pokud končí, sestavíme odečtením symbolů na této větvi slovo příjimané automatem. Výpočetní strom bývá často nekonečný. V jeho kořeni je počáteční stav, v listech mohou být pouze koncové stavy.

Stavový diagram dostaneme z výpočetního stromu tak, že všechny výskyty téhož stavu sjednotíme do jediného uzlu. Dostáváme tak graf, ve kterém na počáteční stav ukazuje šipka zvenku a koncové stavy jsou označeny dvojitým kroužkem.

Budeme se nyní zabývat problémem vztahu automatu a jeho jazyka. Jazyky akceptované konečnými automaty jsou totiž speciální; nelze např. sestrojit KA rozpoznávající jazyk $\{0^n 1^n : n \geq 0\}$.

Definice 2.9

Nechť R je relace ekvivalence. Řekneme o ní, že je *zprava invariantní (pravá kongruence),* jestliže pro každou trojici x, y, z platí: je-li $x \,\mathrm{R}\, y$, pak také $xz \,\mathrm{R}\, yz$.

Index ekvivalence je počet tříd odpovídajícího rozkladu.

Příklad 2.10

Definujeme relaci R na slovech abecedy Σ pro nějaký konečný automat $M = (K, \Sigma, \delta, q_0, F)$ takto: $x \,\mathrm{R}\, y$, pokud $\delta(q_0, x) = \delta(q_0, y)$. Tato relace je zprava invariantní.

Věta 2.11 NERONOVA

Následující tři tvrzení jsou ekvivalentní:

1. Jazyk $L \subseteq \Sigma^*$ rozpoznává nějaký konečný automat.
2. Jazyk L je sjednocením některých tříd vytvořených zprava invariantní ekvivalencí s konečným indexem.
3. Nechť R je relace na slovech ze Σ^* definovaná: $x \,\mathrm{R}\, y$, pokud pro všechna slova $r \in \Sigma^*$ platí: $(xr \in L \Leftrightarrow yr \in L)$. Pak R má konečný index.

Důkaz: $1 \Rightarrow 2$: (viz také [Chytil], věta 2.18 na straně 57) Nechť L je rozpoznávaný konečným automatem $M = (K, \Sigma, \delta, q_0, F)$. Definujeme relaci R' jako v předcházejícím příkladě, tedy $x \,\mathrm{R'}\, y$, pokud $\delta(q_0, x) = \delta(q_0, y)$. Tato relace je zprava invariantní. Index R' je konečný, protože může být nejvýše roven počtu stavů automatu M. Potom jazyk L je sjednocením těch tříd ekvivalence podle R', které odpovídají koncovým stavům automatu M.

$2 \Rightarrow 3$: Dokážeme, že libovolná relace R' splňující podmínku části 2 věty je zjemnění relace R z části 3 věty. Musíme tedy ukázat, že pokud $x \,\mathrm{R'}\, y$, pak také $x \,\mathrm{R}\, y$:

Protože R' je zprava invariantní, pak pro každé slovo $r \in \Sigma^*$ platí: $xr \,\mathrm{R'}\, yr$, a tedy $\delta(q_0, xr) = \delta(q_0, yr)$. Je-li tedy $xr \in L$, pak $F \ni \delta(q_0, xr) = \delta(q_0, yr) \in F$, a proto i $yr \in L$. Analogicky dokážeme opak. Celkem $x \,\mathrm{R}\, y$.

Je tedy R' zjemnění R. Z toho plyne, že poněvadž R' má konečný index, má také R konečný index.

$3 \Rightarrow 1$: Z relace R zkonstruujeme konečný automat, který bude rozpoznávat jazyk L. Nejprve si však ověříme, že relace R je zprava invariantní:

Je-li $x \,\mathrm{R}\, y$, pak pro všechna slova $w, r \in \Sigma^*$ platí: $(xwr \in L \Leftrightarrow ywr \in L)$, a tedy $xw \,\mathrm{R}\, yw$.

Nechť konečný automat $M' = (K', \Sigma, \delta', q'_0, F')$ je dán takto:

$K' = \Sigma^*/\,\mathrm{R}$ — množina stavů odpovídá třídám rozkladu podle relace R, $[x] \in K'$ označuje třídu obsahující prvek x

$q'_0 = [\varepsilon]$ — počáteční stav je třída obsahující prázdné slovo

$F' = \{[x] : x \in L\}$ — množina koncových stavů je množina všech tříd, které obsahují slova jazyka L

$\delta'([x], a) = [xa]$ — z třídy $[x]$ a se symbolem a na vstupu se dostaneme do třídy $[xa]$; zde je potřeba pravá invariance relace R, aby hodnota přechodové funkce δ' byla dána jednoznačně

Z pravidla pro funkci δ' odvodíme induktivně, že $\delta'(q'_0, x) = [x]$. Proto $x \in T(M')$ právě tehdy, když $[x] \in F'$, a to je z definice F' ekvivalentní tomu, že $x \in L$. Celkem $T(M') = L$, a tedy automat M' rozpoznává jazyk L.

Konečný index relace R je zde třeba k zajištění konečného počtu stavů automatu M', což vyžaduje definice KA. □

Příklad 2.12

S pomocí Neronovy věty ukážeme, že jazyk $\{0^n 1^n\}$ není rozpoznatelný žádným konečným automatem.

Budeme postupovat sporem. Předpokládejme tedy, že jazyk L je rozpoznatelný konečným automatem M s konečným počtem stavů. Potom existuje zprava invariantní ekvivalence $\sim$ s konečným indexem taková, že jazyk L je sjednocením jistých tříd rozkladu podle $\sim$.

Nechť rozklad $\{0,1\}^*/\sim$ má n_0 tříd. Uvažme slova $0^{n_0+1}1^0, 0^{n_0+1}1^1, 0^{n_0+1}1^2, \ldots, 0^{n_0+1}1^{n_0}$. Těch je celkem $n_0 + 1$, proto dvě různá slova z nich vybraná patří do jedné třídy. Nechť jsou to $0^{n_0+1}1^i$ a $0^{n_0+1}1^j$ pro $i < j$. Tedy platí:

$$0^{n_0+1}1^i \sim 0^{n_0+1}1^j.$$

Protože $\sim$ je zprava invariantní, je také

$$0^{n_0+1}1^i1^{n_0-i+1} \sim 0^{n_0+1}1^j1^{n_0-i+1},$$

tedy

$$0^{n_0+1}1^{n_0+1} \sim 0^{n_0+1}1^{n_0+1+j-i}.$$

Na levé straně je slovo z jazyka L, na druhé straně slovo, které do L nepatří. Jazyk L tedy nemůže být sjednocením tříd rozkladu podle relace $\sim$, což je spor.

Lemma 2.13 PUMPING LEMMA

Nechť L je jazyk rozpoznatelný konečným automatem. Pak existuje číslo p takové, že každé slovo $w \in L$, pro něž $|w| \geq p$, se dá rozložit takto: $w = xyz$ a $|y| \leq p$, přičemž všechna slova $w_i = xy^iz$ jsou také z jazyka L.

Důkaz: Nechť $M = (K, \Sigma, \delta, q_0, F)$ je automat, který rozpoznává jazyk L. Označíme $p = |K|$ a ukážeme, že splňuje vlastnost požadovanou ve znění věty.

Nechť w je libovolné slovo jazyka L takové, že $|w| \geq p$. Víme, že $(q_0, w) \vdash^* (q_f, \varepsilon)$ je posloupnost $|w|$ konfigurací a $q_f \in F$. Podle Dirichletova principu musí tedy toto odvození slova w projít některým stavem alespoň dvakrát (nechť je to např. q). Tedy platí:

$$(q_0, w = xyz) \vdash^* (q, yz) \vdash^+ (q, z) \vdash^* (q_f, \varepsilon). \tag{2}$$

Zřejmě také $(q, y^iz) \vdash^+ (q, z)$, a tedy

$$(q_0, w_i = xy^iz) \vdash^* (q, y^iz) \vdash^+ (q, z) \vdash^* (q_f, \varepsilon). \tag{3}$$

□

Každý jazyk L je jistě rozpoznatelný více různými automaty. My budeme pro daný jazyk hledat jakýsi kanonický automat, který nazveme minimálním automatem. Nakonec najdeme algoritmus, který z daného automatu vytvoří minimální automat rozeznávající stejný jazyk, jako automat původní.

Definice 2.14

Automat s nejmenším počtem stavů, který rozpoznává daný jazyk L, se nazývá *minimální automat* pro L.

Definice 2.15

Nechť $M = (K, \Sigma, \delta, q_0, F)$ je konečný automat. Jeho *stav* q nazveme *nedosažitelný* právě tehdy, když neexistuje $x \in \Sigma^*$ takové, že $\delta(q, x) = q$.

Věta 2.16

Minimální automat pro L je dán jednoznačně až na izomorfismus a je to automat M' z důkazu Neronovy věty (2.11).

Důkaz: Každý automat $M = (K, \Sigma, \delta, q_0, F)$ definuje odpovídající relaci R' z bodu 2 Neronovy věty, která je zjemněním relace R z bodu 3 této věty. Automat M' má stejný počet stavů jako ekvivalence R tříd rozkladu, a automat M má tolik stavů jako ekvivalence R' tříd rozkladu. Protože je R' zjemněním R, platí: $|K'| \leq |K|$.

Platí-li rovnost, je třeba najít izomorfismus na stavech obou automatů. Každému stavu $q \in K$ přiřadíme stav $q' \in K'$ takto: Jestliže je stav q nedosažitelný, lze jej úplně odstranit a automat proto nebyl minimální. Tedy existuje $x \in \Sigma^*$ takové, že $\delta(q_0, x) = q$. Potom stav q ztotožníme se stavem $\delta'(q'_0, x) = [x]$. Takové přiřazení je korektní. Vezmeme-li totiž dvě různá slova x, y taková, že $\delta(q_0, x) = \delta(q_0, y) = q$, pak z definice relace R' přímo plyne, že x R' y. R' je však dle Neronovy věty zjemněním R, tedy také x R y. Potom $\delta'(q'_0, x) = \delta'(q'_0, y)$, a obraz stavu q je dán jednoznačně bez ohledu na výběr slova x.

Nyní již lze snadno ověřit, že takové zobrazení stavů je izomorfismus. □

Známe tedy automat, který je pro daný jazyk minimální. Naším požadavkem je však algoritmus, který pro daný automat vrátí jeho minimální podobu. Z předchozích úvah plyne, že minimální automat nemá žádné nedosažitelné stavy. Následující algoritmus je odstraňuje.

Algoritmus 2.1 ELIMINACE NEDOSAŽITELNÝCH STAVŮ KA

Vstup: Automat $M = (K, \Sigma, \delta, q_0, F)$

Výstup: Automat M' bez nedosažitelných stavů; $T(M) = T(M')$

1. $i := 0$
2. $s_i := \{q_0\}$
3. `repeat`
   3.1. $s_{i+1} := S_i \cup \{q : \exists p \in S_i, a \in \Sigma : \delta(p, a) = q\}$
   3.2. $i := i + 1$
4. `until` $S_i = S_{i-1}$
5. $M' := M/S_i$ (M' dostaneme z M tak, že vezmeme v úvahu pouze stavy S_i)

Je-li n počet stavů automatu M, pak algoritmus skončí nejpozději při $i = n$.

Odstraněním nedosažitelných stavů však ještě nedostaneme minimální automat. Mohou se totiž vyskytnout různé stavy automatu, které se chovají stejně, a mohou tedy být sjednoceny do jediného stavu.

Definice 2.17
Nechť $M = (K, \Sigma, \delta, q_0, F)$ je konečný automat. Jeho *stavy* p a q nazveme *ekvivalentní,* jestliže pro každé slovo $w \in \Sigma^*$ platí: $\delta(p, w) \in F \Leftrightarrow \delta(q, w) \in F$.

Definice 2.18
Nechť $M = (K, \Sigma, \delta, q_0, F)$ je konečný automat. Definujeme posloupnost relací na jeho stavech $\overset{0}{\sim}, \overset{1}{\sim}, \ldots$ takto:

$$p \overset{0}{\sim} q, \text{ pokud } p \in F \Leftrightarrow q \in F$$
$$P \overset{i}{\sim} q, \text{ pokud } p \overset{i-1}{\sim} q \wedge \forall a \in \Sigma : \delta(p, a) \overset{i-1}{\sim} \delta(q, a)$$

Lemma 2.19
Nechť $M = (K, \Sigma, \delta, q_0, F)$ je konečný automat. Pak platí následující tvrzení:

(i) Každá z relací $\overset{i}{\sim}$ je ekvivalence na K. (Dále budeme značit rozklad $K/\overset{i}{\sim}$ jako R_i)

(ii) Pro každé $i \in \mathbf{N}$ je R_{i+1} zjemnění R_i

(iii) Pro každé $i \in \mathbf{N}$platí: Je-li $R_i = R_{i+1}$, pak pro každé přirozené $t \geq 1$ je $R_i = R_{i+t}$

(iv) Označíme $n = |K|$ počet stavů automatu M. Pak existuje $k \leq n - 1$ takové, že $R_k = R_{k+1}$

(v) Pro každé k takové, že $R_k = R_{k+1}$, platí: $\overset{k}{\sim} = \sim$.

Důkaz:

(i),(ii) Přímo z definice.

(iii) Víme, že $R_i = R_{i+1}$, a tedy $\overset{i}{\sim} = \overset{i+1}{\sim}$. Z definice $\overset{i+1}{\sim}$ plyne, že $p \overset{i+1}{\sim} q \Leftrightarrow p \overset{i}{\sim} q \wedge \forall a \in \Sigma : \delta(p, a) \overset{i}{\sim} \delta(q, a)$. Protože $\overset{i}{\sim} = \overset{i+1}{\sim}$, je nyní $p \overset{i+1}{\sim} q \wedge \forall a \in \Sigma : \delta(p, a) \overset{i+1}{\sim} \delta(q, a)$. To však podle definice znamená, že $p \overset{i+2}{\sim} q$. Je tedy $p \overset{i+1}{\sim} q \Leftrightarrow p \overset{i+2}{\sim} q$, tedy $\overset{i+1}{\sim} = \overset{i+2}{\sim}$.

Předchozí poznatek můžeme induktivně aplikovat na jakékoli $i + t$, kde $t \geq 1$.

(iv) Označíme c_i index ekvivalence $\overset{i}{\sim}$. Předpokládejme, že $R_0, \ldots, R_k$ jsou navzájem různé a $R_k = R_{k+1}$ je první výskyt rovnosti sousedů v řadě R_i. Protože R_{i+1} je zjemněním R_i, platí vztah: $1 < c_0 < c_1 \cdots < c_k \leq n$. Proto $k + 1 \leq n$, tedy $k \leq n - 1$.

(v) Definice relace $\overset{i}{\sim}$ se dá interpretovat takto: $p\overset{i}{\sim}q$, jestliže pro každé slovo x o délce nejvýše i platí, že $\delta(p,x) \in F \Leftrightarrow \delta(q,x) \in F$.

Nechť $R_k = R_{k+1}$. Podle bodu (iii) je $R_k = R_{k+t}$ pro každé $t \geq 1$. Vezmeme-li tedy jakoukoli horní hranici d pro délku slova, potom $p\overset{i}{\sim}q$ je ekvivalentní s tím, že pro všechna slova x o délce nejvýše d platí: $\delta(p,x) \in F \Leftrightarrow \delta(q,x) \in F$. Tato vlastnost je tedy splněna pro všechna slova x, což je ekvivalentní definici relace $\sim$. Je tedy $p\overset{i}{\sim}q$, právě když $p \sim q$, tedy $\overset{i}{\sim} = \sim$.

□ Právě dokázané lemma napovídá, jak lze získat rozklad na stavech automatu podle relace $\sim$. Budeme konstruovat relace $\overset{i}{\sim}$ (resp. odpovídající rozklady) tak dlouho, dokud se bude následující relace (resp. rozklad) lišit od předchozí. Výsledek je relace $\sim$, resp. odpovídající rozklad, jehož třídy mohou být stavy automatu ekvivalentního původnímu.

Algoritmus 2.2 ELIMINACE VÝSKYTŮ EKVIVALENTNÍCH STAVŮ KA

Vstup: Automat $M = (K, \Sigma, \delta, q_0, F)$

Výstup: Rozklad $K/\sim$ stavů na vzájemně ekvivalentní

1. $i := 0$
2. $R_0 := \{F, K \setminus F\}$
3. `repeat`
   - 3.1. spočti R_{i+1} z R_i podle definice (2.18)
   - 3.2. $i := i + 1$
4. `until` $R_i = R_{i-1}$

K automatu M získáme minimální automat rozpoznávající stejný jazyk tak, že s pomocí algoritmu 2.1 odstraníme nedosažitelné stavy, a pak vytvoříme rozklad na množině stavů s pomocí algoritmu 2.2.

2.2 Nedeterministické KA

Nedeterministické KA (zkráceně NKA, deterministický KA bude dále DKA) poskytují volbu při přechodu stavů. Oborem hodnot přechodové funkce nebudou nyní pouhé stavy automatu, ale všechny podmnožiny množiny stavů.

Definice 2.20

Nedeterministický konečný automat M nad abecedou Σ je pětice $(K, \Sigma, \delta, q_0, F)$, kde jednotlivé komponenty mají tento význam:

K — konečná neprázdná množina *stavů*

Σ — konečná neprázdná množina *symbolů;* abeceda

δ — *přechodová funkce;* $\delta\colon K \times \Sigma \to 2^K$

$q_0 \in K$ — *počáteční stav*

$F \subseteq K$ — *množina koncových stavů*

Definice 2.21

Konfigurací automatu M nazveme libovolnou dvojici $[q, x]$, kde $q \in K$ je stav a $x \in \Sigma^*$ je slovo (žádná změna).

Relaci $\vdash$ na množině konfigurací automatu M *(krok výpočtu)* definujeme takto: $[q, ax] \vdash [p, x]$, pokud $p \in \delta(q, a)$.

Automat akceptuje slovo $w \in \Sigma^*$, pokud $(q_0, w) \vdash (q_f, \varepsilon)$ a $q_f \in F$ (žádná změna).

Poznámka 2.22

Přechodovou funkci δ můžeme opět rozšířit na definiční obor $K \times \Sigma^*$ takto:

$$\delta(q, \varepsilon) = \{q\}$$
$$\delta(q, xa) = \bigcup_{p \in \delta(q,x)} \delta(p, a)$$

U nedeterministických KA provádíme navíc rozšíření na definiční obor $2^K \times \Sigma^*$: $\delta(P, x) = \bigcup_{p \in P} \delta(p, x)$.

Definice 2.23

Řekneme, že *automat M akceptuje slovo w*, pokud $\delta(q_0, w) \cap F \neq \emptyset$.

Pokusíme se najít vztah mezi deterministickými a nedeterministickými KA.

Věta 2.24

Nechť L je jazyk rozpoznávaný nějakým NKA. Pak existuje DKA, který rozpoznává jazyk L.

Důkaz: Sestrojíme DKA $M' = (K', \Sigma, \delta', q_0', F')$ z NKA M tak, že stavy automatu M' ztotožníme se všemi podmnožinami stavů automatu M. Tedy $K' = 2^K$. Počáteční stav q_0' bude odpovídat množině $\{q_0\}$, a koncové stavy F' budou všechny podmnožiny K, které obsahují alespoň jeden koncový stav: $F' = \{P : P \subseteq K \wedge P \cap K \neq \emptyset\}$.

Pro zavedení funkce přechodové δ' použijeme rozšíření δ na definiční obor $2^K \times \Sigma$. Potom bude $\delta'(Q, a) = P$ jedině tehdy, když $\delta(Q, a) = P$.

Dokážeme nyní indukcí vzhledem k délce slova x, že

$$\delta'(q_0', x) = Q \Leftrightarrow \delta(q_0, x) = Q \tag{4}$$

Nechť $|x| = 0$, tedy $x = \varepsilon$. Potom triviálně $\delta'(\{q_0\}, \varepsilon) = \{q_0\}$, a $\delta(q_0, \varepsilon) = \{q_0\}$.

indukční krok: Předpokládejme, že výraz (4) platí pro všechna slova $|x| \leq l$. Zřejmě platí pro všechny symboly $a \in \Sigma$:

$$\delta'(q_0', xa) = \delta'(\delta'(q_0', x), a).$$

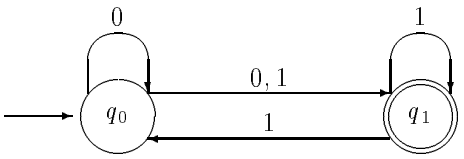
Podle předpokladu je $\delta'(q_0', x) = Q$, právě když $\delta(q_0, x) = Q$. Z definice funkce δ' víme, že $\delta'(Q, a) = P$, právě když $\delta(Q, a) = P$. Celkem dostáváme, že $\delta'(q_0', xa) = P$ nastane právě tehdy, když $\delta(q_0, xa) = P$.

Z výše uvedeného již přímo plyne, že $\delta'(q_0', x) \in F'$ je ekvivalentní s $\delta(q_0, x) \cap F \neq \emptyset$ pro každé slovo x, a tedy oba automaty přijímají stejný jazyk. □

Příklad 2.25

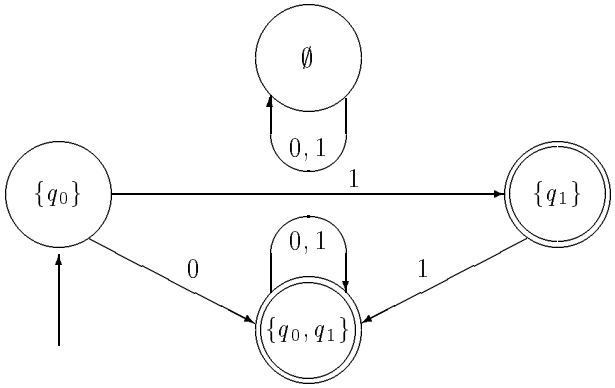
Nedeterministický konečný automat $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ má následující přechodovou funkci a diagram:

		0	1
→	q_0	$\{q_0, q_1\}$	$\{q_1\}$
←	q_1		$\{q_0, q_1\}$



Vytvoříme k němu odpovídající deterministický automat $M' = (K', \{0, 1\}, \delta', \{q_0\}, F')$, kde množina stavů je $K' = \{\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$ a konečné stavy jsou $F' = \{\{q_1\}, \{q_0, q_1\}\}$. Přechodová funkce δ' je pak dána následující tabulkou a diagramem:

		0	1
	$\emptyset$	$\emptyset$	$\emptyset$
→	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_1\}$
←	$\{q_1\}$		$\{q_0, q_1\}$
←	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$



2.3 Rozhodnutelné problémy z oblasti KA

Existuje algoritmus, který zjistí, zda jsou dva KA ekvivalentní. Stačí je totiž převést do minimálních tvarů a zjistit, zda jsou izomorfní.

Dá se také snadno zjisti, zda množina slov příjimaných konečným automatem je prázdná nebo neprázdná, konečná nebo nekonečná. Přímo z Pumping lemmatu (2.13) se totiž dají odvodit následující dvě lemmata:

Lemma 2.26
Množina slov příjimaných konečným automatem je neprázdná právě tehdy, když tento automat přijímá slovo délky menší než n, kde n je počet jeho stavů.

Lemma 2.27
Množina slov přijímaných konečným automatem je nekonečná právě tehdy, když tento automat přijímá slovo délky l: $n \leq l \leq 2n$, kde n je počet stavů automatu.

3 Regulární gramatiky a jazyky

3.1 Konečné automaty a regulární gramatiky

V této části ukážeme, že jazyky akceptované konečnými automaty jsou právě regulární jazyky, tedy odpovídají regulárním gramatikám.

Věta 3.1
Nechť $G = (N, \Sigma, P, S)$ je regulární gramatika. Pak existuje konečný automat M takový, že $T(M) = L(G)$.

Důkaz: Nechť $M = (K, \Sigma, \delta, q_0, F)$. Sestrojíme nedeterministický KA M, který bude přijímat právě jazyk $L(G)$. Stavy K automatu M ztotožníme s neterminály gramatiky G, a přidáme navíc jeden stav, který neodpovídá žádnému neterminálu v G: $K = N \cup \{A\}$, kde $A \notin N$. Počáteční stav q_0 bude odpovídat startovnímu neterminálu S gramatiky. Koncové stavy F automatu zadáme tímto předpisem:

$$F = \begin{cases} \{S, A\} & \text{, pokud } (S \to \varepsilon) \in P \\ \{A\} & \text{, pokud } (S \to \varepsilon) \notin P \end{cases}$$

Přechodovou funkci δ zvolíme takto:

$$\delta(B, a) = \begin{cases} A \in \delta(B, a) & \text{, pokud } (B \to a) \in P \\ C \in \delta(B, a) & \text{, pokud } (B \to aC) \in P \\ \delta(A, *) = \emptyset & , \end{cases}$$

Nyní dokážeme, že $T(M) = L(G)$.

$L(G) \subseteq T(M)$: Nechť $x \in L(G)$ je neprázdné slovo, tedy $x = a_1 a_2 \cdots a_n$ pro $n \geq 1$. Slovo x je tedy větou gramatiky G, proto $S \Rightarrow_G^* x$. Protože G je regulární, musí odvození proběhnout takto:

$$S \Rightarrow_G a_1 A_1 \Rightarrow_G a_1 a_2 A_2 \Rightarrow_G \cdots \Rightarrow_G a_1 \cdots a_{n-1} A_{n-1} \Rightarrow_G a_1 \cdots a_n$$

Odvodíme z toho následující vlastnosti automatu M:

$$\begin{array}{rcl} (S \to a_1 A_1) \in P & \Rightarrow & A_1 \in \delta(S, a_1) \\ (A_1 \to a_2 A_2) \in P & \Rightarrow & A_2 \in \delta(A_1, a_2) \\ \vdots & \vdots & \vdots \\ (A_{n-1} \to a_n) \in P & \Rightarrow & A \in \delta(A_{n-1}, a_n) \end{array}$$

Můžeme tedy dostat tuto posloupnost výpočetních kroků v našem automatu:

$$(S, a_1 \cdots a_n) \vdash (A_1, a_2 \cdots a_n) \vdash \cdots \vdash (A_{n-1}, a_n) \vdash (A, \varepsilon)$$

Celkem $x = a_1 \cdots a_n \in T(M)$.

$T(M) \subseteq L(G)$: důkaz lze provést zcela analogicky pro $x \neq \varepsilon$.

Pro úplnost musíme uvést ještě případ prázdného slova. Je-li $\varepsilon \in L(G)$, pak $S \in F$ a $\varepsilon \in T(M)$. Je-li naopak $\varepsilon \in T(M)$, pak musí být $S \in F$, a tedy $(S \to \varepsilon) \in P$. □

Věta 3.2
Nechť $M = (K, \Sigma, \delta, q_0, F)$ je konečný automat. Pak existuje regulární gramatika $G = (N, \Sigma, P, S)$ taková, že $L(G) = T(M)$.

Důkaz: Bez újmy na obecnosti lze předpokládat, že M je deterministický. Zvolíme gramatiku G tak, že její neterminály budou odpovídat stavům automatu ($N = K$), abecedy budou shodné, a startovní neterminál bude odpovídat počátečnímu stavu automatu ($S = q_0$). Pravidla P budeme potom konstruovat takto:

$$\begin{aligned} \text{je-li } \delta(B,a) = C, &\quad \text{pak} \quad (B \to aC) \in P \\ \text{je-li } \delta(B,a) = C \text{ a } C \in F, &\quad \text{pak} \quad (B \to a) \in P \end{aligned}$$

Důkaz shodnosti jazyků $L(G)$ a $T(M)$ lze vést zcela analogicky důkazu předchozí věty. □

3.2 Vlastnosti regulárních jazyků

V této části ověříme, že třída regulárních jazyků, kterou budeme dále značit $\mathcal{L}_3$, je uzavřená na všechny obvyklé operace.

Věta 3.3
Třída regulárních jazyků $\mathcal{L}_3$ je uzavřená vzhledem k operaci sjednocení.

Důkaz: Nechť L_1 a L_2 jsou dva regulární jazyky. Chceme dokázat, že jazyk $L_1 \cup L_2$ je také regulární.

Nechť jazyk L_1 je generovaný gramatikou G_1 a jazyk L_2 gramatikou G_2. Označíme $G_i = (N_i, \Sigma_i, P_i, S_i)$ pro $i = 1, 2$. Předpokládejme, že $N_1 \cap N_2 = \emptyset$ (jinak lze neterminály jedné z gramatik přejmenovat). Zvolíme neterminál $S \notin N_1 \cup N_2$ a definujeme gramatiku $G = (N_1 \cup N_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P, S)$. Množinu pravidel P pak sestrojíme takto: $P = \{S \to \alpha : (S_1 \to \alpha) \in P_1 \vee (S_2 \to \alpha) \in P_2\} \cup P_1 \cup P_2$. Nyní je zřejmé, že $(S \Rightarrow_G \alpha)$ právě tehdy, když $(S_1 \Rightarrow_{G_1} \alpha)$ nebo $(S_2 \Rightarrow_{G_2} \alpha)$. Gramatika G, která je jistě regulární, rozeznává jazyk $L_1 \cup L_2$. Tento jazyk je proto také regulární. □

Věta 3.4
Třída regulárních jazyků $\mathcal{L}_3$ je uzavřená vzhledem k operaci doplňku.

Důkaz: Nechť L_1 je regulární jazyk nad abecedou Σ_1. Chceme dokázat, že také jazyk $\Sigma^* \setminus L$, kde $\Sigma_1 \subseteq \Sigma$, je regulární.

Nechť $M_1 = (K_1, \Sigma_1, \delta_1, q_1, F_1)$ je automat, který rozpoznává jazyk L_1, přičemž přechodová funkce δ_1 je totální. Je-li $\Sigma = \Sigma_1$, pak automat $\overline{M_1} = (K_1, \Sigma, \delta_1, q_1, K_1 \setminus F_1)$ zřejmě rozpoznává jazyk $\Sigma^* \setminus L$.

Je-li $\Sigma \supset \Sigma_1$, je třeba vytvořit nový stav $\overline{q} \notin K_1$. Funkci δ_1 rozšíříme na $\overline{\delta_1}$ pro tento stav tak, že $\overline{\delta_1}(\overline{q}, *) = \overline{q}$. Dále přechodovou funkci $\overline{\delta_1}$ musíme rozšířit z abecedy Σ_1 na abecedu Σ, a to tak, že pro všechna $a \in \Sigma \setminus \Sigma_1$ bude $\overline{\delta_1}(*, a) = \overline{q}$. Ve všech ostatních případech bude $\overline{\delta_1} = \delta_1$. Automat $\overline{M_1}$ nyní zavedeme jako $(K_1 \cup \{\overline{q}\}, \Sigma, \overline{\delta_1}, q_1, K_1 \cup \{\overline{q}\} \setminus F_1)$. Zřejmě všechna slova $x \in \Sigma_1^*$ přijímá automat $\overline{M_1}$ právě tehdy, když je nepřijímá automat M_1. Slova $y \in \Sigma^* \setminus \Sigma_1^*$ jsou přijímána všechna. Automat $\overline{M_1}$ tedy přijímá právě jazyk $\Sigma^* \setminus L$. □

Důsledek 3.5
Třída regulárních jazyků $\mathcal{L}_3$ je uzavřena vzhledem k operaci průniku.

Důkaz: Mějme dva regulární jazyky L_1 a L_2. S pomocí de Morganova pravidla $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ a předchozích dvou vět je zřejmě vidět, že jejich průnik je také regulární jazyk. □

Poznámka 3.6
Toto je alternativní důkaz uzavřenosti regulárních jazyků na operaci průniku. Z automatů M_1 a M_2, kde $M_i = (K_i, \Sigma, \delta_i, q_i, F_i)$, rozeznávajících jazyky L_1, resp. L_2, sestrojíme automat M, který bude rozeznávat jazyk $L_1 \cap L_2$.

Nechť $M = (K_1 \times K_2, \Sigma, \delta, [q_1, q_2], F_1 \times F_2)$. Přechodovou funkci δ tohoto automatu definujeme takto:

$$\delta([p_1, p_2], a) = [r_1, r_2], \text{ pokud } \delta_i(p_i, a) = r_i \text{ pro } i = 1, 2$$

Poznámka 3.7
Třída regulárních jazyků $\mathcal{L}_3$ s částečným uspořádáním $\subseteq$ a operacemi $\cup$, $\cap$ a $\overline{\ }$ tvoří Booleovu algebru s nejmenším prvkem $\emptyset$ a největším prvkem Σ^*.

Věta 3.8
Třída regulárních jazyků $\mathcal{L}_3$ je uzavřena vzhledem k operaci zřetězení.

Důkaz: Nechť L_1 a L_2 jsou dva regulární jazyky. Chceme dokázat, že jazyk L_1L_2 je také regulární.

Nechť jazyk L_1 je rozpoznávaný automatem M_1 a jazyk L_2 automatem M_2. Označíme $M_i = (K_i, \Sigma, \delta_i, q_i, F_i)$ pro $i = 1, 2$. Zkonstruujeme nyní automat $M = (K_1 \cup K_2, \Sigma, \delta, q_1, F)$. Přechodovou funkci δ zavedeme podle pravidel:

$$\delta(q,a) = \begin{cases} \{\delta_1(q,a)\} & \text{pro } q \in K_1 \setminus F_1 \\ \{\delta_1(q,a), \delta_2(q_2,a)\} & \text{pro } q \in F_1 \\ \{\delta_2(q,a)\} & \text{pro } q \in K_2 \end{cases}$$

Množina koncových stavů F bude rovna

$$F = \begin{cases} F_1 \cup F_2 & \text{pro } \varepsilon \in L_2 \\ F_2 & \text{pro } \varepsilon \notin L_2 \end{cases}$$

Automat M pak zřejmě bude rozpoznávat právě jazyk L_1L_2, který je proto regulární. □

Věta 3.9
Třída regulárních jazyků $\mathcal{L}_3$ je uzavřena vzhledem k operaci uzávěru.

Důkaz: Nechť L je regulární jazyk. Chceme ukázat, že také L^* je regulární jazyk.

Nechť $M = (K, \Sigma, \delta, q_0, F)$ je konečný automat, který rozpoznává jazyk L. Nechť $M' = (K \cup \{q_0'\}, \Sigma, \delta', q_0', F \cup \{q_0'\})$ je automat, kde $q_0' \notin K$ je nový stav a přechodová funkce δ' nabývá těchto hodnot:

$$\begin{aligned} \delta'(q_0', a) &= \begin{cases} \{\delta(q_0,a), q_0\} & \text{pro } \delta(q_0,a) \in F \\ \{\delta(q_0,a)\} & \text{jinak} \end{cases} \\ \delta'(q, a) &= \begin{cases} \{\delta(q,a), q_0\} & \text{pro } \delta(q,a) \in F \\ \{\delta(q,a)\} & \text{jinak} \end{cases} \qquad \text{pro } q \neq q_0' \end{aligned}$$

Automat M' nyní rozpoznává jazyk L^*, který je proto regulární. □

Věta 3.10
Všechny konečné jazyky lze rozeznat konečným automatem.

Důkaz: Množinu $\emptyset$ rozpoznáme konečným automatem $M = (\{q_0\}, \Sigma, \delta, q_0, \emptyset)$, kde $\delta(q_0, *) = q_0$.

Množinu $\{\varepsilon\}$ konečným automatem $M = (\{q_0, p\}, \Sigma, \delta, q_0, \{q_0\})$, kde $\delta(q_0, *) = p$ a $\delta(p, *) = p$.

Množinu $\{x\}$, kde $x = a_1 \cdots a_n$ je neprázdné slovo rozpoznáme konečným automatem $M = (\{q_i : i = 0, \ldots, n\}, \Sigma, \delta, q_0, \{q_n\})$, kde $\delta(q_{i-1}, a_i) = q_i$.

Protože každou konečnou množinu můžeme složit jako sjednocení konečně mnoha jednoprvkových množin, můžeme konečněmnohokrát aplikovat větu 3.3 o uzavřenosti regulárních jazyků na sjednocení. □

Tvrzení 3.11
Třída jazyků $\mathcal{L}_3$ je nejmenší třída jazyků, která obsahuje všechny konečné množiny a je uzavřena vzhledem ke sjednocení, zřetězení a uzávěru.

3.3 Regulární výrazy

V této části uvedeme regulární výrazy (RV) a přechodové grafy (PG), které, jak ukážeme, jsou další alternativní reprezentací regulárních jazyků.

Definice 3.12
Třída *regulárních výrazů* nad abecedou Σ je definována induktivně takto:

1. ε a $\emptyset$ jsou regulární výrazy
2. každé písmeno $a \in \Sigma$ je regulární výraz nad Σ
3. jsou-li R, R_1, R_2 regulární výrazy nad Σ, pak také $(R_1 + R_2)$, $(R_1 \cdot R_2)$, $(R)^*$ jsou regulární výrazy nad Σ

Definice 3.13
Každý regulární výraz R nad Σ popisuje jistou množinu slov $\widetilde{R}$ nad Σ:

1. je-li $R = \varepsilon$, pak $\widetilde{R} = \{\varepsilon\}$
   je-li $R = \emptyset$, pak $\widetilde{R} = \emptyset$
2. je-li $R = a$, pak $\widetilde{R} = \{a\}$
3. je-li $R = (R_1 + R_2)$, pak $\widetilde{R} = \widetilde{R_1} \cup \widetilde{R_2}$
   je-li $R = (R_1 \cdot R_2)$, pak $\widetilde{R} = \widetilde{R_1}\widetilde{R_2}$
   je-li $R = (R_1)^*$, pak $\widetilde{R} = \widetilde{R_1}^*$

Příklad 3.14
Mějme abecedu $\Sigma = \{a, b\}$. Toto jsou příklady regulárních výrazů: $R_1 = ba^*$, $R_2 = a^*ba^*ba^*$, $R_3 = (a+b)^*$. Jejich odpovídající jazyky jsou:

$$\begin{aligned} \widetilde{R_1} &= \{ba^*\} \\ \widetilde{R_2} &= \{w \in \Sigma^* : w \text{ obsahuje právě dvě } b\}, \\ \widetilde{R_3} &= \Sigma^*. \end{aligned}$$

Definice 3.15
Dva regulární výrazy R_1 a R_2 nazveme *ekvivalentní,* pokud popisují stejné množiny ($\widetilde{R_1} = \widetilde{R_2}$).

Definice 3.16
Regulární množiny nad abecedou Σ definujeme induktivně takto:

1. každá konečná množina slov nad Σ (včetně $\emptyset$) je regulární množina nad Σ
2. jestliže U, V jsou regulární množiny nad Σ, pak také $U \cup V$, UV, U^* jsou regulární množiny nad Σ

Poznámka 3.17
Z výše uvedených definic je zřejmé, že množina je regulární, právě když může být popsána regulárním výrazem.

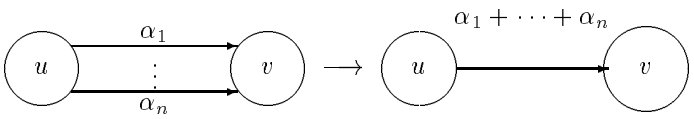
Definice 3.18
Přechodový graf T nad abecedou Σ je orientovaný multigraf, jehož každá hrana je označena návěštím, které je slovem nad Σ. Obyčejně je jeden z uzlů označen jako *vstupní uzel* (↘◯ nebo ⊖) a některé uzly jako *koncové uzly* (◯↙ nebo ⊕). Pro slovo $w \in \Sigma^*$ nazveme w-cestou z uzlu i do uzlu j v T cestu z i do j takovou, že zřetězením všech návěští na hranách po cestě získáme slovo w.

Řekneme, že slovo $w \in \Sigma^*$ je *akceptováno přechodovým grafem* T, jestliže existuje w-cesta z počátečního do některého z koncových uzlů grafu T. Slovo ε je navíc akceptováno v případě, že počáteční uzel je zároveň koncovým.

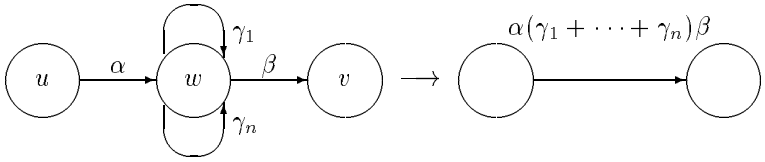
Množinu všech slov akceptovaných přechodovým grafem T nazveme *jazyk akceptovaný přechodovým grafem* T a označíme $\widetilde{T}$.

Věta 3.19 KLEENE
(i) Ke každému přechodovému grafu T nad Σ existuje regulární výraz R nad Σ takový, že $\widetilde{R} = \widetilde{T}$ a naopak.
(ii) Ke každému regulárnímu výrazu R nad Σ existuje konečný automat M s abecedou Σ takový, že $T(M) = \widetilde{R}$ a naopak.

Důkaz: (i) Označíme x počáteční uzel grafu T a vytvoříme nový koncový uzel y tak, že z každého původního koncového uzlu vede do y hrana označená ε. V tomto novém grafu budiž y jediný koncový uzel. Budeme dále konstruovat tzv. *zobecněný přechodový graf,* jehož hrany jsou označeny regulárními výrazy. Nejprve vyloučíme násobné hrany. Jestliže z uzlu u vede do v n hran označených $\alpha_1, \ldots, \alpha_n$, nahradíme je jedinou hranou označenou $\alpha_1 + \cdots + \alpha_n$:



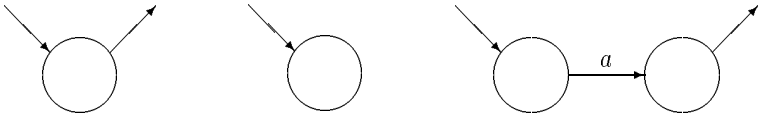
Dále se budeme postupně zbavovat všech uzlů na cestě z x do y. Nechť mezi uzly u a v leží uzel w. Hrana z u do w je označena α, hrana z w do v je označena β. Uzel w má n „smyček“ označených $\gamma_1, \ldots, \gamma_n$:



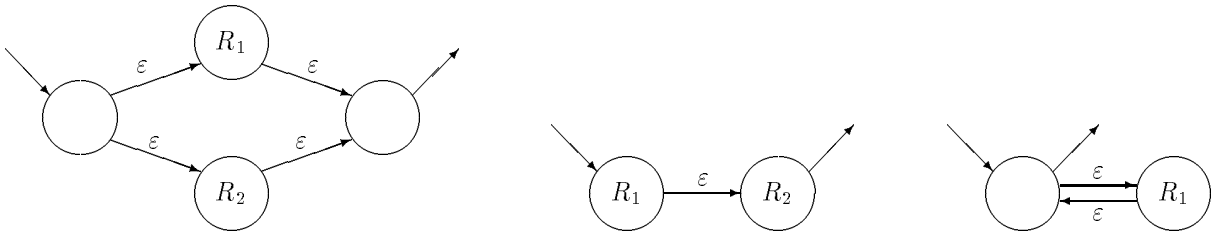
Uzel w odstraníme spolu se všemi zmíněnými hranami, a místo toho zavedeme hranu z u do v, označenou $\alpha(\gamma_1 + \cdots + \gamma_n)\beta$ (pokud taková hrana už existuje, přidáme k ní tento výraz jako další sčítanec).

Tímto způsobem dostaneme zobecněný přechodový graf, který bude mít pouze uzly x a y s jedinou hranou mezi nimi. Ta bude označena regulárním výrazem, který akceptuje stejný jazyk jako původní graf.

Naopak, máme-li dán regulární výraz R, zkonstruujeme odpovídající přechodový graf induktivně vzhledem ke konstrukci R podle těchto pravidel:



$R = \varepsilon$ $R = \emptyset$ $R = a$



$R = (R_1 + R_2)$ $R = (R_1 \cdot R_2)$ $R = (R_1)^*$

Výsledný přechodový graf bude zřejmě akceptovat ten jazyk, který popisuje původní regulární výraz.

(ii) Vezmeme v úvahu regulární výraz R. Budeme z něho konstruovat konečný automat, a to opět indukcí vzhledem ke konstrukci R.

Je-li $R = \emptyset$, $R = \varepsilon$ nebo $R = a$, pak odpovídající jazyky jsou $\emptyset$, resp. $\{\varepsilon\}$, $\{a\}$. Automaty, které rozeznávají tyto množiny, jsou uvedeny v důkazu věty 3.10.

Je-li $R = (R_1 + R_2)$, hledáme automat, který rozpoznává jazyk $\widetilde{R_1} \cup \widetilde{R_2}$. Protože podle indukčního předpokladu máme automaty pro množiny $\widetilde{R_1}$ i $\widetilde{R_2}$, sestrojíme z nich automat M podobně jako v poznámce 3.6. Nahradíme však množinu koncových stavů $F_1 \times F_2$ množinou $F_1 \times Q_2 \cup Q_1 \times F_2$, aby automat M rozeznával sjednocení obou původních jazyků místo průniku.

Je-li $R = (R_1 \cdot R_2)$, hledáme automat, který rozpoznává jazyk $\widetilde{R_1}\widetilde{R_2}$. Podle indukčního předpokladu opět známe automaty pro jazyky $\widetilde{R_1}$ i $\widetilde{R_2}$, proto můžeme hledaný automat sestrojit stejně, jako v důkazu věty 3.8.

Je-li $R = (R_1^*)$, známe už automat pro $\widetilde{R_1}$ a hledáme automat pro jazyk $(\widetilde{R_1})^*$. Ten lze vytvořit tak jako v důkazu věty 3.9. □

4 Zásobníkové automaty

V předchozím jsme uvedli konečné automaty a dokázali, že akceptují právě regulární jazyky. V této kapitole bude definován jiný formální výpočetní prostředek, zásobníkový automat. Ukážeme o něm, že má větší výpočetní sílu než konečný automat, protože bude umět rozpoznávat bezkontextové jazyky.

4.1 Zásobníkový automat a jeho rozšíření

Zásobníkový automat je vlastně konečný automat obohacený o potenciálně nekonečný zásobník.

Definice 4.1

Zásobníkový automat (ZA) nad abecedou Σ je systém $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde jednotlivé komponenty mají následující význam:

Q — konečná neprázdná množina *stavů*

Σ — *vstupní abeceda*

Γ — *zásobníková abeceda;* množina symbolů, které se mohou objevit v zásobníku

δ — *přechodová funkce;* $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to$ do množiny všech konečných podmnožin $Q \times \Gamma^*$

$q_0 \in Q$ — *počáteční stav*

$Z_0 \in \Gamma$ — *dno zásobníku*

$F \subseteq Q$ — *množina koncových stavů*

Definice 4.2

Konfigurací ZA je libovolná trojice $(q, x, \alpha) \in Q \times \Sigma^* \times \Gamma^*$.

Krok výpočtu ZA definujeme jako binární relaci $\vdash$ na množině konfigurací takto: jsou-li $q, q' \in Q$ stavy, $a \in \Sigma \cup \{\varepsilon\}$ vstupní symbol, $Z \in \Gamma$ zásobníkový symbol, $\alpha, \beta \in \Gamma^*$ zásobníková slova a $w \in \Sigma^*$ vstupní slovo, pak

$$(q, aw, Z\beta) \vdash (q', w, \alpha\beta), \text{ pokud } (q', \alpha) \in \delta(q, a, Z).$$

Jinými slovy, je-li (q', α) prvkem $\delta(q, a, Z)$, znamená to, že ve stavu q s vrcholem zásobníku Z může ZA přečíst symbol a a vrchol zásobníku Z nahradit slovem α, přičemž stav se změní na q'.

Značíme $\vdash^*$ reflexivní a tranzitivní uzávěr relace $\vdash$, $\vdash^+$ pouze tranzitivní uzávěr. Navíc $\vdash^k$ znamená složení k relací $\vdash$.

Řekneme, že ZA *rozpoznává slovo w koncovým stavem,* pokud $(q_0, w, Z_0) \vdash^* (q_f, \varepsilon, \alpha)$, kde $q_f \in F$ je koncový stav. ZA *rozeznává slovo w prázdným zásobníkem,* pokud $(q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)$, kde q je libovolný stav.

Značíme $T(M)$ *jazyk akceptovaný koncovým stavem ZA*, tedy množinu všech slov rozpoznávaných koncovým stavem. Dále značíme $N(M)$ *jazyk akceptovaný prázdným zásobníkem ZA*, tedy množinu všech slov rozpoznávaných prázdným zásobníkem.

Příklad 4.3

Podle příkladu 2.12 neexistuje konečný automat, který by rozeznával jazyk $\{0^n 1^n : n \geq 1\}$. Najdeme však zásobníkový automat, který tento jazyk akceptuje.

Nechť $M = (\{q_0, q_1\}, \{0, 1\}, \{Z_0, 0\}, \delta, q_0, Z_0, \emptyset)$ je zásobníkový automat. Přechodovou funkci δ definujeme takto:

$$\begin{aligned}
\delta(q_0, 0, Z_0) &= \{(q_0, 0Z_0)\} \\
\delta(q_0, 0, 0) &= \{(q_0, 00)\} \\
\delta(q_0, 1, 0) &= \{(q_1, \varepsilon)\} \\
\delta(q_1, 1, 0) &= \{(q_1, \varepsilon)\} \\
\delta(q_1, \varepsilon, Z_0) &= \{q_1, \varepsilon\}
\end{aligned}$$

Je vidět, že ve stavu q_0 může být na vstupu 0, která se vždy přidá do zásobníku. Při prvním výskytu symbolu 1 se stav změní na q_1, a jedna 0 je ze zásobníku odebrána. Odebrání 0 ze zásobníku provádí i stav q_1 při příchodu 1 na vstupu, a jiný symbol už nemůže přijít. Automat tedy akceptuje prázdným zásobníkem posloupnost nul a po nich jedniček, jichž je stejný počet. Proto $N(M) = L$.

Věta 4.4

Jazyk L je rozpoznatelný koncovým stavem nějakého ZA právě tehdy, když je rozpoznatelný prázdným zásobníkem nějakého ZA.

Důkaz: $\Rightarrow$: Nechť $L = T(M_1)$, kde $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Zkonstruujeme ZA $M_2 = (Q \cup \{q_\varepsilon, q_0'\}, \Sigma, \Gamma \cup \{Z_0'\}, \delta', q_0', Z_0', \emptyset)$. Přechodová funkce δ' přitom nabývá hodnot podle těchto pravidel:

$$\begin{array}{rl} \delta'(q_0', \varepsilon, Z_0') = \{(q_0, Z_0 Z_0')\} & \\ (p, \gamma) \in \delta'(q, a, Z), & \text{pokud } (p, \gamma) \in \delta(q, a, Z) \\ (q_\varepsilon, \varepsilon) \in \delta'(q, \varepsilon, Z) & \text{pro všechna } q \in F \text{ a } Z \in \Gamma \cup \{Z_0'\} \\ \delta'(q_\varepsilon, \varepsilon, Z) = \{(q_\varepsilon, \varepsilon)\} & \text{pro všechna } Z \in \Gamma \cup \{Z_0'\} \end{array}$$

Porovnáme-li výpočetní kroky v původním automatu M_1 a novém M_2, dojdeme k následující ekvivalenci:

$$\begin{array}{ccccccc} & & (q_0, w, Z_0) & \vdash_{M_1}^n & (q_f, \varepsilon, Y_1 \cdots Y_r) & & \\ & & & \Updownarrow & & & \\ (q_0', w, Z_0') & \vdash_{M_2} & (q_0, w, Z_0 Z_0') & \vdash_{M_2}^n & (q_f, \varepsilon, Y_1 \cdots Y_n Z_0') & \vdash_{M_2}^{r+1} & (q_\varepsilon, \varepsilon, \varepsilon), \end{array}$$

kde $q_f \in F$ je koncový stav původního automatu M_1. Je tedy zřejmé, že $N(M_2) = T(M_1)$.

$\Leftarrow$: Nechť automat $M_2 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$ je zásobníkový automat, který rozeznává jazyk L prázdným zásobníkem. Zkonstruujeme ZA $M_2 = (Q \cup \{q_0', q_f\}, \Sigma, \Gamma \cup \{Z_0'\}, \delta', q_0', Z_0', \{q_f\})$. Přechodovou funkci δ' definujeme takto:

$$\begin{array}{rl} \delta'(q_0', \varepsilon, Z_0') = \{(q_0, Z_0 Z_0')\} & \\ (p, \gamma) \in \delta'(q, a, Z), & \text{pokud } (p, \gamma) \in \delta(q, a, Z) \\ \delta'(q, \varepsilon, Z_0') = (\{q_f, \varepsilon\}) & \end{array}$$

Z těchto vztahů lze odvodit podobnou ekvivalenci jako výše při důkazu obrácení věty:

$$\begin{array}{ccccccc} & & (q_0, w, Z_0) & \vdash_{M_2}^n & (q, \varepsilon, \varepsilon) & & \\ & & & \Updownarrow & & & \\ (q_0', w, Z_0') & \vdash_{M_1} & (q_0, w, Z_0 Z_0') & \vdash_{M_1}^n & (q, \varepsilon, Z_0') & \vdash_{M_1} & (q_f, \varepsilon, \varepsilon), \end{array}$$

z čehož plyne, že $T(M_1) = N(M_2)$. □

Definice 4.5

Rozšířený zásobníkový automat je systém $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, ve kterém všechny symboly mají tentýž význam jako v zásobníkovém automatu, až na přechodovou funkci δ, jejímž definičním oborem je $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*$. (Rozšířený zásobníkový automat se tedy rozhoduje na základě řetězce na vrcholu zásobníku místo jediného symbolu.)

Věta 4.6

Nechť L je jazyk rozeznávaný koncovým stavem rozšířeného zásobníkového automatu M. Potom existuje (obyčejný) zásobníkový automat M_1, který rozpoznává koncovým stavem jazyk L.

Důkaz: Precizně provedený důkaz je zdlouhavý a technicky náročný. Uvedeme pouze princip, tedy popíšeme, jak z automatu M lze sestrojit automat M_1, aby byla věta splněna.

Nechť $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ a $T(M) = L$. Budeme konstruovat automat

$$M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_1, Z_1, F_1),$$

kde $Z_1 \notin \Gamma$ a $q_1 \notin Q$. Zvolíme nejprve číslo m jako maximální délku slova α, pro které je $\delta(q, a, \alpha) \neq \emptyset$. Jinými slovy, m je největší počet znaků, které může automat M odebrat z vrcholu zásobníku v jednom kroku.

Stavy automatu M_1 budou odpovídat dvojicím $[q, \alpha]$, kde $q \in Q$ je stav původního automatu a α je zásobníkové slovo automatu M_1 délky nejvýše m: $Q_1 = \{[q, \alpha] : q \in Q, \alpha \in \Gamma_1^*, 0 \leq |\alpha| \leq m\}$. Zásobníková abeceda Γ_1 je pouze rozšířením Γ o symbol Z_1: $\Gamma_1 = \Gamma \cup \{Z_1\}$. q_1—počáteční stav automatu M_1—bude odpovídat dvojici $[q_0, Z_0 Z_1^{m-1}]$. Množina koncových stavů automatu bude rovna: $F_1 = \{[q, \alpha] : [q, \alpha] \in Q_1, q \in F\}$. Jde tedy o ty stavy z Q_1, které vznikly z koncového stavu automatu M.

Přechodovou funkci δ_1 definujeme takto:

1. Nejprve uvedeme přechody δ_1, které vzniknou z přechodů původní funkce δ. Nechť tedy $\delta(q, a, X_1 \cdots X_k)$ obsahuje prvek $(r, Y_1 \cdots Y_l)$. Budeme rozlišovat dvě možnosti:
   (a) $l \geq k$: pro všechny zásobníkové symboly $Z \in \Gamma_1$ a všechna zásobníková slova α délky $m - k$ klademe: množina $\delta_1([q, X_1 \cdots X_k\alpha], a, Z)$ obsahuje prvek $([r, \beta], \gamma Z)$, kde $\beta\gamma = Y_1 \cdots Y_l\alpha$, přičemž slovo β má délku právě m.
   (b) $l < k$: pro všechny zásobníkové symboly $Z \in \Gamma_1$ a všechna zásobníková slova α délky $m - k$ klademe: množina $\delta_1([q, X_1 \cdots X_k\alpha], a, Z)$ obsahuje prvek $([r, Y_1 \cdots Y_l\alpha Z], \varepsilon)$.
2. Musíme umět doplnit „zásobníkovou komponentu“ nových stavů na délku m, aby mohly být prováděny přechody podle předchozího bodu. Definujeme tedy pro každý stav $q \in Q$ původního automatu, pro každý zásobníkový symbol $Z \in \Gamma_1$ a každé zásobníkové slovo α délky ostře menší než m tento přechod: $\delta_1([q, \alpha], \varepsilon, Z) = \{([q, \alpha Z], \varepsilon)\}$.

Takto definovaný automat M_1 splňuje podmínku věty. □

4.2 Stromy odvození pro CFG

V příští části odhalíme vztah bezkontextových gramatik a zásobníkových automatů. Stromy odvození bezkontextových gramatik jsou pěknou ilustrací tohoto vztahu, proto je uvedeme na tomto místě.

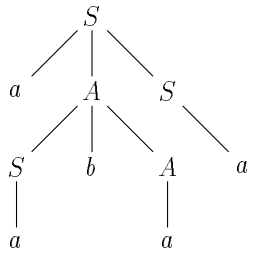
Definice 4.7
Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika. Strom T nazveme *strom odvození (derivační strom)* pro G, jestliže

- každý uzel je označen návěštím—symbolem z $N \cup \Sigma$
- návěští kořene je S
- má-li uzel u návěští A a má-li alespoň jednoho následníka, pak A je neterminál
- jsou-li uzly $u_1, \ldots, u_k$ po řadě následníky uzlu u s návěštími v pořadí $A_1, \ldots, A_k$ a návěští uzlu u je A, pak $A \to A_1 \cdots A_k$ je pravidlem gramatiky G

Příklad 4.8
Nechť G je gramatika:

$$\begin{aligned} S &\to aAS \mid a \\ A &\to SbA \mid SS \mid a \end{aligned}$$

Potom následující strom je derivačním stromem pro G:



Všímněme si, že když seřadíme listy tohoto stromu, dostaneme slovo *aabaa*, které je větou gramatiky G.

Definice 4.9
Výsledkem derivačního stromu nazveme řetěz vzniklý zřetězením jeho listů zleva doprava.

Věta 4.10
Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika. Pak pro každé slovo $\varepsilon \neq \alpha \in \Sigma^*$ platí: $S \Rightarrow^* \alpha$ právě tehdy, když existuje derivační strom pro G s výsledkem α.

Důkaz: $\Leftarrow$: indukcí vzhledem k počtu nelistových uzlů ve stromu. Nechť tedy α je výsledkem stromu.

1. Nechť existuje ve stromu T jediný nelistový uzel. Musí to být kořen s n následníky $A_1, \ldots, A_n$. Potom $\alpha = A_1 \cdots A_n$, a podle definice derivačního stromu je $S \to A_1 \cdots A_n$. Celkem $(S \to \alpha) \in P$.

2. Předpokládejme, že pro všechny stromy s méně než k uzly tvrzení platí. Mějme nyní strom T s k uzly, jehož výsledkem je α. Označíme následníky kořene jako $A_1, \ldots, A_n$. Opět podle definice derivačního stromu musí být $S \to A_1 \cdots A_n$ pravidlem gramatiky G. Pro každý symbol A_i mohou nastat tyto možnosti:

   (a) A_i je nelist: Pak A_i je neterminál a musí být kořenem podstromu s méně než k uzly a výsledkem α_i. Podle indukčního předpokladu platí: $A \Rightarrow^* \alpha_i$.

   (b) A_i je list: Položíme nyní $\alpha_i = A_i$ a platí: $A_i \Rightarrow^0 \alpha_i$.

   Protože zřejmě platí $\alpha = \alpha_1 \cdots \alpha_n$ a odvodili jsme, že pro každé i je $A_i \Rightarrow^* \alpha_i$, existuje v gramatice G následující odvození:

$$A \Rightarrow A_1 \cdots A_n \Rightarrow^* \alpha_1 A_2 \cdots A_n \Rightarrow^* \cdots \Rightarrow^* \alpha_1 \cdots \alpha_n = \alpha,$$

   tedy celkem $A \Rightarrow^* \alpha$.

$\Rightarrow$: důkaz lze vést analogicky. □

4.3 Zásobníkové automaty a bezkontextové gramatiky

Ukážeme nyní, že mexi bezkontextovými gramatikami a zásobníkovými automaty existuje stejný vztah jako mezi regulárními gramatikami a konečnými automaty.

Lemma 4.11 O SYNTAKTICKÉ ANALÝZE SHORA DOLŮ
Nechť $G = (N, \Sigma, P, S)$ je libovolná bezkontextová gramatika. Pak existuje zásobníkový automat R takový, že $N(R) = L(G)$.

Důkaz: Nechť automat $R = (\{q\}, \Sigma, N \cup \Sigma, \delta, S, \emptyset)$. Přechodovou funkci δ zavedeme takto: je-li $A \to \alpha$ pravidlo gramatiky G, pak $\delta(q, \varepsilon, A) \ni (q, \alpha)$. Navíc, pro každý vstupní symbol $a \in \Sigma$ je $\delta(q, a, a) = \{(q, \varepsilon)\}$.
Dokážeme nyní, že $N(R) = L(G)$. Chceme tedy odvodit tuto ekvivalenci:

$$S \Rightarrow^* w \Leftrightarrow (q, w, S) \vdash^* (q, \varepsilon, \varepsilon) \tag{5}$$

$\Rightarrow$: indukcí vzhledem k počtu odvození v gramatice G. Aby se dala indukce použít, budeme vztah (5) dokazovat pro všechny neterminály, a ne jen pro S.
Indukcí vzhledem k m dokazujeme, že pro každé m existuje n tak, že

$$\text{pokud } A \Rightarrow^m w, \text{ pak } (q, w, A) \vdash^n (q, \varepsilon, \varepsilon). \tag{6}$$

$m = 1$: Nechť $w = a_1 \cdots a_k$. Protože $A \Rightarrow^1 w$, musí existovat pravidlo $A \to w$ v gramatice G. Potom podle pravidel pro přechodovou funkci δ dostáváme: $\delta(q, \varepsilon, A) \ni (q, w)$. Je tedy v automatu R možný tento výpočet:

$$(q, a_1 \cdots a_k, A) \vdash (q, a_1 \cdots a_k, a_1 \cdots a_k) \vdash^k (q, \varepsilon, \varepsilon).$$

indukční krok: Předpokládejme, že vztah (6) platí pro všechna $l < m$. Mějme nyní odvození $A \Rightarrow^m w$. Musí tedy existovat pravidlo $A \to x_1 \cdots x_k$, které je v odvození $A \Rightarrow^m w$ použito jako první. Symboly x_i mohou být terminály i neterminály. Je-li x_i terminální symbol, pak položíme $w_i = x_i$. V tomto případě platí: $(q, w_i, x_i) \vdash (q, \varepsilon, \varepsilon)$. Je-li x_i neterminální symbol, pak existuje odvození $x_i \Rightarrow^{m_i} w_i$, kde $w_i \in \Sigma^*$ je součástí slova w. Celkem dostáváme, že $x_1 \cdots x_k \Rightarrow^* w_1 \cdots w_k = w$. Víme však, že $m_1 + \cdots + m_k < m$, proto na všechny odvození $x_i \Rightarrow^{m_i} w_i$ lze použít indukční předpoklad. Existuje tedy výpočet v R: $(q, w_i, x_i) \vdash^{n_i} (q, \varepsilon, \varepsilon)$. Kromě toho víme, že $A \to x_1 \cdots x_k$ je pravidlo v G, proto $\delta(q, \varepsilon, A) \ni (q, x_1 \cdots x_k)$. Tedy $(q, w, A) \vdash (q, w_1 \cdots w_k, x_1 \cdots x_k)$. Ukázali jsme, že $(q, w_i, x_i) \vdash (q, \varepsilon, \varepsilon)$ pro každé x_i, ať už je terminálem nebo neterminálem. Proto $(q, w_1 \cdots w_k, x_1 \cdots x_k) \vdash^* (q, \varepsilon, \varepsilon)$.
$\Leftarrow$: opět indukcí vzhledem k počtu odvození v gramatice G. Pomůžeme si podobně, jako v předchozím důkazu obrácené implikace.

Indukcí vzhledem k n dokazujeme, že pro každé n existuje m tak, že

$$\text{pokud } (q,w,A) \vdash^n (q,\varepsilon,\varepsilon), \text{ pak } A \Rightarrow^m w. \tag{7}$$

$n = 1$: Musí být $w = \varepsilon$, a tedy $A \to \varepsilon$ je pravidlem gramatiky G

indukční krok: Předpokládejme, že vztah (7) platí pro každé $l < n$. Mějme nyní výpočet $(q,w,A) \vdash^n (q,\varepsilon,\varepsilon)$. Vezměme jeho první krok: $(q,w,A) \vdash (q,w,x_1 \cdots x_k)$. Musí tedy být $A \to x_1 \cdots x_k$ pravidlem gramatiky G. Rozložme dále slovo w na části $w_1 \cdots w_k$ tak, že $(q,w_i,x_i) \vdash (q,\varepsilon,\varepsilon)$. Bude tak buď $x_i = w_i$ terminál, tedy $x_i \Rightarrow^0 w_i$, nebo $x_i \Rightarrow^{n_i} w_i$ podle indukčního předpokladu. Celkem dostáváme odvození

$$A \Rightarrow x_1 \cdots x_k \Rightarrow^{n_1} w_1 x_2 \cdots x_k \Rightarrow \cdots \Rightarrow w_1 \cdots w_k = w.$$

□

Příklad 4.12

Uvažujme gramatiku G se startovním neterminálem E a s následujícími pravidly:

$$\begin{aligned} E &\to E + T \mid T \\ T &\to T * F \mid F \\ F &\to i \mid (E) \end{aligned}$$

Podle předchozí věty se pokusíme sestrojit automat R, který bude rozeznávat jazyk $L(G)$ prázdným zásobníkem.

Nechť tedy $R = (\{q\}, \Sigma, \Gamma, \delta, S, \emptyset)$. V našem případě je vstupní abeceda rovna $\Sigma = \{+, *, i, (,)\}$ a zásobníková abeceda $\Gamma = \{E, T, F\} \cup \Sigma$. Funkce δ bude nabývat těchto hodnot:

$$\begin{aligned} \delta(q,\varepsilon,E) &= \{(q,E+T),(q,T)\} \\ \delta(q,\varepsilon,T) &= \{(q,T*F),(q,F)\} \\ \delta(q,\varepsilon,F) &= \{(q,i),(q,(E))\} \\ \delta(q,a,a) &= \{(q,\varepsilon)\} \qquad \text{pro } a \in \Sigma \end{aligned}$$

Obrázek 1 ukazuje, jak lze provést odvození slova $w = i+i*i$ v automatu R. Tomuto postupu se říká *syntaktická analýza shora dolů.* V konfiguracích je vynechán stav q, protože je vždy stejný.

Lemma 4.13 O SYNTAKTICKÉ ANALÝZE ZDOLA NAHORU

Nechť $G = (N, \Sigma, P, S)$ je libovolná bezkontextová gramatika. Pak existuje rozšířený zásobníkový automat R takový, že $T(R) = L(G)$.

Důkaz: Sestrojíme rozšířený zásobníkový automat $R = (\{q,r\}, \Sigma, \Sigma \cup N \cup \{\$\}, \delta, q, \$, \{r\})$, kde \$ je nový symbol, který není ani neterminál ani terminál gramatiky G. Přechodovou funkci automatu G nyní zvolíme takto:

$$\begin{aligned} \delta(q,a,\varepsilon) &= \{(q,a)\} \qquad \text{pro všechna } a \in \Sigma \\ \delta(q,\varepsilon,\alpha) &\ni (q,A) \qquad \text{pro } A \to \alpha \text{ pravidlo } G \\ \delta(q,\varepsilon,\$S) &= \{(r,\varepsilon)\} \end{aligned}$$

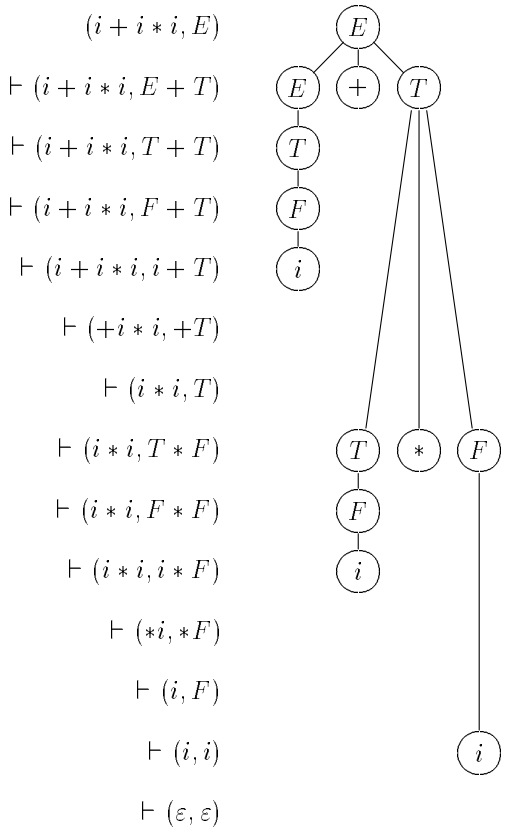
Dokažme nyní, že $L(G) = T(R)$.

$\subseteq$: Budeme dokazovat implikaci

$$(S \Rightarrow^* \alpha A y \Rightarrow^n) \Rightarrow ((q,xy,\$) \vdash^* (q,y,\$\alpha A))$$

indukcí vzhledem k n.

1. $n = 0$: triviálně
2. *indukční krok:* Nechť tvrzení platí pro všechna $k < n$. Potom $\alpha A y \Rightarrow \alpha\beta r \Rightarrow^{n-1} xy$. Rozlišíme dva případy:
   (a) $\alpha\beta \in \Sigma^*$: Musí být $\alpha\beta = x$, a tedy $(q,xy,\$) \vdash^* (q,y,\$\alpha\beta) \vdash (q,y,\$\alpha A)$ díky definici přechodové funkce δ.



Obr. 1: Příklad syntaktické analýzy shora dolů

(b) $\alpha\beta \notin \Sigma^*$: Označme $\alpha\beta = \gamma Br$, kde B je nepravější neterminál slova $\alpha\beta$. Protože platí

$$S \Rightarrow^* \gamma Bry \Rightarrow^{n-1} xy,$$

můžeme použít indukční předpoklad: $(q, xy, \$) \vdash^* (q, ry, \$\gamma B)$. Podle pravidla pro funkci δ dále platí:

$$(q, y, \$\underbrace{\gamma Br}_{=\alpha\beta}) \vdash (q, y, \$\alpha A).$$

Celkem dostáváme $L(G) \subseteq T(R)$.

$\supseteq$: Budeme nyní dokazovat tuto implikaci:

$$((q, xy, \$) \vdash^n (q, y, \$\alpha A)) \Rightarrow (\alpha Ay \Rightarrow^* xy)$$

Důkaz povedeme opět indukcí vzhledem k n.

1. $n = 0$: triviálně
2. *indukční krok:* Předpokládejme, že tvrzení platí pro všechna $k < n$. V konfiguraci $(q, y, \$\alpha A)$ je vrchol zásobníku neterminál A, tedy poslední krok musel být redukcí (použití pravidla $A \to \beta$) a ne čtením.

   $$(q, xy, \$) \vdash^{n-1} (q, y, \$\alpha\beta) \vdash (q, y, \$\alpha A)$$

   Podle indukčního předpokladu je $\alpha\beta y \Rightarrow^* xy$. Celkem tedy $\alpha Ay \Rightarrow \alpha\beta y \Rightarrow^* xy$.

Abychom mohli použít indukce, dokazovali jsme silnější tvrzení pro všechny neterminály. Jeho speciálním případem je ekvivalence

$$((q, w, \$) \vdash (q, \varepsilon, \$S)) \Leftrightarrow (S \Rightarrow^* w)$$

□

Příklad 4.14

Vezměme stejnou gramatiku G jako v příkladu 4.12. Pokusíme se k ní podle předcházející věty sestrojit rozšířený zásobníkový automat, který bude rozeznávat jazyk $L(G)$ koncovým stavem.

Nechť tedy $R = (\{q, r\}, \Sigma, \Sigma \cup N \cup \{\$\}, \delta, q, \$, \{r\})$. V našem případě je opět abeceda $\Sigma = \{+, *, i, (,)\}$, množina neterminálů rovna $N = \{E, T, F\}$. Pro přechodovou funkci dostáváme tato pravidla:

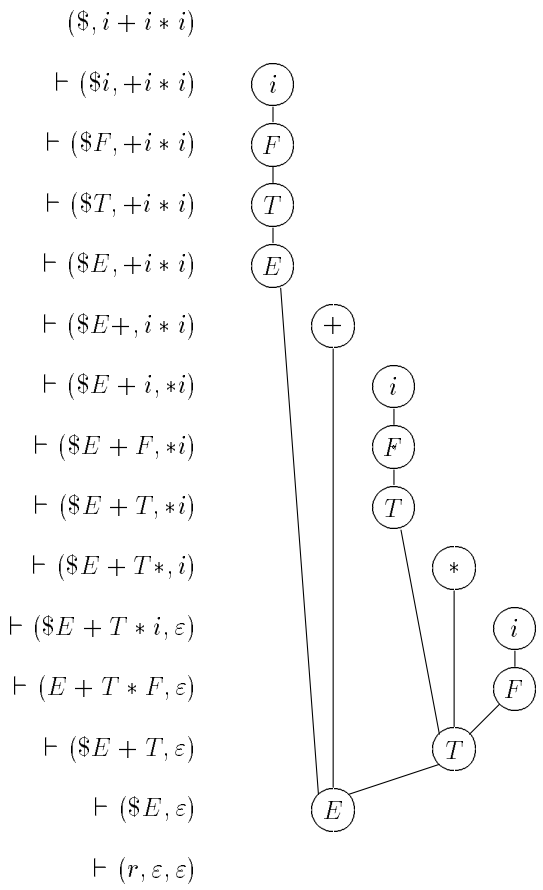
$$\begin{aligned}
\delta(q, a, \varepsilon) &= \{(q, a)\} \qquad \text{pro } a \in \Sigma \\
\delta(q, \varepsilon, E + T) &= \{(q, E)\} \\
\delta(q, \varepsilon, T) &= \{(q, E)\} \\
\delta(q, \varepsilon, T * F) &= \{(q, T)\} \\
\delta(q, \varepsilon, F) &= \{(q, T)\} \\
\delta(q, \varepsilon, i) &= \{(q, F)\} \\
\delta(q, \varepsilon, (E)) &= \{(q, F)\} \\
\delta(q, \varepsilon, \$E) &= \{(r, \varepsilon)\}
\end{aligned}$$

Obrázek 2 demonstruje odvození slova $w = i + i * i$ v automatu R. Takovému odvození se říká *syntaktická analýza zdola nahoru.* V konfiguracích je až na poslední vynechán stav, protože je stále q.

Poznámka 4.15

Syntaktická analýza zdola nahoru uvedená v předcházející větě je nedeterministická, jako ostatně zásobníkové automaty jako takové. Mohou nastat dva typy situací, ve kterých má automat více možností postupu; říká se jim *konflikty:*

- konflikt čtení×redukce: Na vrcholu zásobníku je slovo α a v původní gramatice byla dvě pravidla: $A \to \alpha$ a $B \to \alpha\beta$. Automat tedy neví, zda má redukovat podle prvního pravidla nebo číst dál.



Obr. 2: Příklad syntaktické analýzy zdola nahoru

- konflikt redukce×redukce: Na vrcholu zásobníku je slovo $\gamma\alpha$ a v původní gramatice byla tato pravidla: $A \to \alpha$ a $C \to \gamma\alpha$. Automat neví, zda má redukovat podle prvního nebo podle druhého pravidla.

Lemma 4.16
Nechť L je jazyk rozeznávaný prázdným zásobníkem automatu M. Pak M je bezkontextový jazyk.

Důkaz: viz [Chytil], věta 4.50 na straně 109. Ke zkoušce nebyl vyžadován. □

Důsledek 4.17
Pro libovolný jazyk L jsou následující tři tvrzení ekvivalentní.

1. L je bezkontextový jazyk, tedy existuje bezkontextová gramatika G tak, že $L(G) = L$.
2. Existuje zásobníkový automat M, který rozpoznává jazyk L koncovým stavem, tedy $T(M) = L$.
3. Existuje zásobníkový automat M. který rozpoznává jazyk L prázdným zásobníkem. tedy $N(M) = L$.

5 Bezkontextové gramatiky a jazyky

Definice 5.1
Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika a nechť $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_n$ je odvození v G. Řekneme, že toto odvození je *nejlevější* (resp. *nejpravější*), jestliže každé $\alpha_i = x_i A_i \beta_i$ (resp. $\alpha_i = \beta_i A_i x_i$), kde $x_i \in \Sigma^*$ je slovo složené pouze z terminálů, $A_i \in N$ je neterminál a $\beta_i \in (N \cup \Sigma)^*$ je libovolné slovo, a α_{i+1} vznikne z α_i nahrazením A_i slovem γ při použití pravidla $A_i \to \gamma$. Někdy také říkáme *levá* (resp. *pravá*) derivace.

Jestliže je tedy $\alpha_0 \Rightarrow \cdots \Rightarrow \alpha_n$ levá (resp. pravá) derivace, pak všechna α_i nazýváme *levá* (resp. *pravá*) *větná forma.*

Poznámka 5.2
Nechť $w \in L(G)$ je slovo pro nějakou bezkontextovou gramatiku G. Potom počet různých derivačních stromů s výsledkem w je roven počtu různých levých derivací $S \Rightarrow^* w$ slova w a to je rovno počtu různých pravých derivací slova w.

Protože pojmy levé a pravé derivace jsou zřejmě duální, budeme brát dále v úvahu pouze levé derivace.

5.1 Transformace CFG

Bezkontextové gramatiky hrají důležitou roli v syntaktické analýze. Práci s obecnou bezkontextovou gramatikou lze někdy usnadnit tak, že zavedeme na chování gramatik další podmínky, kterých lze po jistých transformacích dosáhnout u každé bezkontextové gramatiky. Těmito podmínkami a transformacemi se zabývá tato část.

Definice 5.3
Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika. Řekneme, že slovo $w \in L(G)$ je v gramatice G *víceznačné (nejednoznačné),* jestliže existují dvě různá levá odvození slova w v G.

Jestliže existuje alespoň jedno víceznačné slovo $w \in L(G)$, pak o gramatice G řekneme také, že je *víceznačná.*

Bezkontextový jazyk L nazveme *víceznačný,* jestliže je generován pouze víceznačnými bezkontextovými gramatikami.

Obyčejně je velmi komplikované dokázat, že jazyk je víceznačný. Musí se totiž dokázat o každé gramatice, která jazyk generuje, že je víceznačná.

Příklad 5.4
Příkladem jazyka, o kterém se to dokázat podařilo, je $L = \{a^i b^j c^k : i = j \vee j = k\}$. Jedna část gramatiky pro L musí generovat slova, pro která $i = j$, a druhá slova s $j = k$. Ukazuje se však, že slova jazyka L, pro které je $i = j = k$, musí mít vždy dvě různé levé derivace, a to v každé ze zmíněných částí gramatiky jednu.

Příklad 5.5
Zde je příklad nejednoznačné gramatiky generující výrazy s operacemi + a ∗:

$$E \to E + E \mid E * E \mid (E) \mid i$$

Tato gramatika je víceznačná, protože v ní existují tato dvě odvození slova $i + i + i$:

$$E \Rightarrow E + E \Rightarrow E + E + E \Rightarrow i + E + E \Rightarrow i + i + E \Rightarrow i + i + i$$
$$E \Rightarrow E + E \Rightarrow i + E \Rightarrow i + E + E \Rightarrow i + i + E \Rightarrow i + i + i$$

Nejednoznačnost lze však v tomto případě napravit. Gramatika

$$\begin{aligned} E &\to E + T \mid E * T \mid T \\ T &\to (E) \mid i \end{aligned}$$

generuje stejný jazyk a je přitom jednoznačná.

Definice 5.6
Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika. Řekneme, že symbol $X \in (N \cup \Sigma)$ je *nepoužitelný*, pokud neexistuje odvození $S \Rightarrow^* uXv \Rightarrow^* uwv$, kde $w \in \Sigma^*$.

Poznámka 5.7
Nepoužitelnost může být dvou druhů:

1. Neexistuje odvození $S \Rightarrow^* uXv$. Tomuto problému se říká *dostupnost symbolu* a zvažuje se u terminálů i u neterminálů.

2. Pro neterminál X se ověřuje, zda lze z tohoto neterminálu odvodit terminální slovo. Pokud $X = S$, jde o problém, zda jazyk $L(G)$ je prázdný či nikoli.

   Tento problém lze pro obecný neterminál X rozhodnout. Můžeme totiž induktivně konstruovat množinu „použitelných“ neterminálů tak, že v prvním kroku v ní budou pouze neterminály, z nichž lze pomocí nějakého pravidla přímo odvodit terminální slovo. V dalších krocích přidáme ty neterminály, z nichž lze odvodit slovo složené z terminálů a těch neterminálů, které byly přidány v předchozích krocích. Následující algoritmus je formálním zápisem tohoto postupu.

Algoritmus 5.1 Test, zda jazyk generovaný CFG je neprázdný

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: `ano`, pokud $L(G)$ je neprázdná množina, jinak `ne`

1. $N_0 := \emptyset$
2. $i := 0$
3. `repeat`
   3.1. $N_{i+1} := N_i \cup \{A : (A \to \alpha) \in P \wedge \alpha \in (N_i \cup \Sigma)^*\}$
   3.2. $i := i + 1$
4. `until` $N_i = N_{i-1}$
5. `if` $S \in N_i$ `then return ano`
6. `else return ne`

Věta 5.8
Algoritmus 5.1 je správný a vždy skončí.

Důkaz: $\Rightarrow$: Budeme dokazovat implikaci

$$A \in N_i \Rightarrow (\text{existuje } w \in \Sigma^* : A \Rightarrow^* w) \tag{8}$$

indukcí vzhledem k i.

1. $i = 0$: triviálně
2. *indukční krok:* Předpokládejme, že (8) platí pro i. Nechť $A \in N_{i+1}$. Potom buď $A \in N_i$, a tedy podle indukčního předpokladu existuje odvození požadovaných vlastností, nebo existuje pravidlo $A \to \alpha$ takové, že $\alpha = X_1 \cdots X_k$, kde buď X_j je neterminál z množiny N_i, nebo terminální řetězec. V případě terminálního řetězce zřejmě $X_j \Rightarrow^0 X_j$ a označíme $x_j = X_j$. Pokud je $X_j \in N_i$, potom podle indukčního předpokladu $X_j \Rightarrow^* x_j$, kde x_j je terminální řetězec. Je tedy možno v gramatice G derivovat

$$A \Rightarrow X_1 \cdots X_k \Rightarrow^* x_1 X_2 \cdots X_k \Rightarrow^* \cdots \Rightarrow^* x_1 \cdots x_k \in \Sigma^*$$

$\Leftarrow$: Budeme se snažit ukázat, že

$$(A \Rightarrow^* w \in \Sigma^*) \Rightarrow \text{existuje } i : A \in N_i, \tag{9}$$

a to indukcí vzhledem k délce odvození $A \Rightarrow^* w$ (označíme n).

1. $A \Rightarrow w$, pak musí být $A \to w$ pravidlo v G, a tedy $A \in N_1$.
2. *indukční krok:* Nechť vztah (9) platí pro všechna $l < n$. Jestliže $A \Rightarrow^n w$, můžeme tuto derivaci rozepsat na
$$A \Rightarrow X_1 \cdots X_k \Rightarrow^{n-1} x_1 \cdots x_k = w,$$
kde $X_j \Rightarrow^{n_j} x_j$. Víme, že $\sum n_j < n$, můžeme proto použít indukční předpoklad na všechny neterminály z X_j. Každý z nich je tedy jistě obsažen v množině N_{i_j}. Vezmeme nyní $i = \max_{j=1\cdots k} i_j + 1$. Potom všechny neterminály z X_j se objeví v množině N_{i-1}. Protože $A \to X_1 \cdots X_k$ je pravidlo, v němž na pravé straně vystupuje slovo tvořené abecedou $(N_{i-1} \cup \Sigma)$, patří neterminál A do množiny N_i.

Algoritmus 5.1 je tedy správný. Zbývá ukázat, že se zastaví. Předpokládejme, že se nezastaví. To se může stát jedině v případě, že generuje nekonečnou řadu množin N_i, pro něž $N_i \neq N_{i-1}$. Protože však $N_{i-1} \subseteq N_i$, je zřejmě $N_{i-1} \subset N_i$. Všechny množiny N_i jsou navíc vybrány z množiny neterminálů N. Celkem dostáváme řadu

$$N_0 \subset N_1 \subset \cdots \subset N_i \subset \cdots \subseteq N.$$

To je však spor s tím, že množina N je konečná. □

Důsledek 5.9
Pro každou bezkontextovou gramatiku je rozhodnutelné, zda jazyk, který generuje, je prázdný či nikoli.

Definice 5.10
Nechť $G = (N, \Sigma, P, S)$. Řekneme, že symbol $X \in (N \cup \Sigma)$ je *nedosažitelný* v G, jestliže neexistuje derivace $S \Rightarrow^* \alpha X \beta$.

Následující algoritmus odstraňuje nedosažitelné symboly bezkontextové gramatiky tak, že induktivně konstruuje množinu dosažitelných. Jakmile už není co přidat, algoritmus končí.

Algoritmus 5.2 Odstranění nedosažitelných symbolů CFG

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: bezkontextová gramatika $G' = (N', \Sigma', P', S)$ bez nedosažitelných symbolů taková, že $L(G') = L(G)$

1. $V_0 := \{S\}$
2. $i := 0$
3. `repeat`
   3.1. $V_{i+1} := V_i \cup \{X : \exists A \to \alpha X \beta \wedge A \in V_{i-1}\}$
   3.2. $i := i + 1$
4. `until` $V_i = V_{i-1}$
5. $N' := N \cap V_i$
6. $\Sigma' := \Sigma \cap V_i$
7. $P' := P \cap N' \times (N' \cup \Sigma')$

I pro algoritmus 5.2 by se dala formulovat a dokázat věta o správnosti a skončení. Správnost je však zřejmá a skončení plyne opět z toho, že konstruujeme ostře rostoucí podmnožiny konečné množiny $N \cup \Sigma$.

Oba poslední algoritmy se dají spojit v jeden, který bude umět odstranit nepoužitelné symboly obou typů.

Algoritmus 5.3 Odstranění nepoužitelných symbolů CFG

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: bezkontextová gramatika $G_1 = (N_1, \Sigma_1, P_1, S)$ bez nepoužitelných symbolů taková, že $L(G_1) = L(G)$

1. S použitím algoritmu 5.1 odstraníme nepoužitelné symboly druhého druhu, tedy ty neterminály, z nichž se nedá derivovat terminální slovo, a dále
2. s pomocí algoritmu 5.2 odstraníme nepoužitelné symboly prvního druhu, tedy ty nedosažitelné.

Definice 5.11

Řekneme, že bezkontextová gramatika $G = (N, \Sigma, P, S)$ je *bez ε-pravidel*, jestliže neobsahuje žádné pravidlo $A \to \varepsilon$ nebo obsahuje jediné ε-pravidlo $S \to \varepsilon$, přičemž S se nevyskytuje na pravé straně žádného pravidla.

Algoritmus 5.4 Odstranění ε-pravidel CFG

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup bezkontextová gramatika $G' = (N', \Sigma, P', S')$ bez ε-pravidel taková, že $L(G') = L(G)$

1. $N_1 := \{A : (A \to \varepsilon) \in P\}$
2. $i := 1$
3. `repeat`
   3.1. $N_{i+1} := N_i \cup \{B : (B \to A) \in P \wedge A \in N_i\}$
   3.2. $i := i + 1$
4. `until` $N_i = N_{i-1}$
5. Množina N_i nyní obsahuje neterminály, ze kterých je možno odvodit ε. Množinu nových pravidel P' nyní zkonstruujeme takto:

   Nechť $A \to \alpha_0 B_1 \alpha_1 \cdots B_k \alpha_k$ je pravidlo z P, přičemž všechny neterminály B_j patří do N_i a slova α_j naopak neobsahují žádný neterminál z N_i. Potom do P' vložíme všechna možná pravidla $A \to \alpha_0 x_1 \alpha_1 \cdots x_k \alpha_k$ taková, že $x_j = \varepsilon$ nebo $x_j = B_j$. Pravidla $A \to \varepsilon$ se v P' neobjeví.

6. Je-li $S \in N_i$, pak přidáme nový neterminál S' ($N' = N \cup \{S'\}$ pro $S' \notin N$) a pravidla $S' \to \varepsilon \mid S$ do P'. V opačném případě položíme $N' = N$ a $S' = S$.

Příklad 5.12
Nechť gramatika $G = (\{S\}, \{a, b\}, P, S)$ obsahuje následující pravidla:

$$S \to aSbS \mid bSaS \mid \varepsilon$$

S pomocí předchozího algoritmu dostaneme gramatiku $G' = (\{S', S\}, \{a, b\}, P', S')$ s pravidly:

$$\begin{aligned} S' &\to \varepsilon \mid S \\ S &\to aSbS \mid abS \mid aSb \mid ab \\ S &\to bSaS \mid baS \mid bSa \mid ba \end{aligned}$$

Definice 5.13
Pravidlo $A \to B$, kde A, B jsou neterminály, nazveme *jednoduché pravidlo.*

Algoritmus 5.5 Odstranění jednoduchých pravidel CFG

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$ bez ε-pravidel

Výstup: bezkontextová gramatika $G' = (N, \Sigma, P', S)$ bez jednoduchých pravidel taková, že $L(G') = L(G)$

1. Pro každý neterminál $A \in N$ zkonstruujeme množinu $N_A \subseteq N$ těch neterminálů B, pro které $A \Rightarrow^* B$.
2. Do P' vložíme všechna nejednoduchá pravidla z P. S každým takovým pravidlem $(B \to \alpha) \in P$ vložíme do P' navíc pravidla $A \to \alpha$ pro všechny neterminály A takové, že množina N_A obsahuje prvek B.

Definice 5.14
Bezkontextová gramatika G se nazývá *necyklická,* pokud v ní neexistuje odvození tvaru $A \Rightarrow^+ A$, kde A je neterminál.

Definice 5.15
Bezkontextová gramatika G se nazývá *vlastní,* pokud je necyklická, bez ε-pravidel a nemá nepoužitelné symboly.

Důsledek 5.16
Pro každou bezkontextovou gramatiku G existuje vlastní bezkontextová gramatika G' taková, že $L(G') = L(G)$.

5.2 Normální formy CFG

Vlastnosti CFG a příslušné transformace z předchozí části, spolu s některými dalšími, se kombinují do tzv. *normálních forem* bezkontextových gramatik.

Definice 5.17
Bezkontextová gramatika $G = (N, \Sigma, P, S)$ je v *Chomského*[4] *normální formě (CNF),* pokud všechna její pravidla jsou tvaru $X \to YZ$ nebo $X \to a$, kde X, Y, Z jsou neterminály a a je terminální symbol, a může případně obsahovat pravidlo $S \to \varepsilon$.

Ukážeme, že každou bezkontextovou gramatiku lze převést do Chomského normální formy při zachování generovaného jazyka. Uvedeme nejprve příslušný algoritmus.

Algoritmus 5.6 Transformace CFG do CNF

[4] Čteme opět [čomského]

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$ bez jednoduchých pravidel

Výstup: bezkontextová gramatika $G' = (N', \Sigma, P', S)$ v CNF taková, že $L(G') = L(G)$

1. $P := \emptyset$; $N' := N$
2. Vlož do P' všechna pravidla tvaru $A \to a$, $A \to BC$, $S \to \varepsilon$ z P
3. `for` všechna pravidla $(A \to X_1 \cdots X_k) \in P$, kde $X_i \in (N \cup \Sigma)$ jsou symboly a $k > 2$ `do`
   - 3.1. $N' := N' \cup \{X'_1, \ldots, X'_{k-2}, \langle X_2 \cdots X_k \rangle, \ldots, \langle X_{k-1} X_k \rangle\}$
   - 3.2. Vlož do P' pravidla

$$\begin{aligned} A &\to X'_1 \langle X_2 \cdots X_k \rangle \\ \langle X_2 \cdots X_k \rangle &\to X'_2 \langle X_3 \cdots X_k \rangle \\ &\vdots \\ \langle X_{k-1} X_k \rangle &\to X_{k-1} X_k , \end{aligned}$$

   kde $\langle X_i \cdots X_k \rangle$ jsou nové neterminály, a

$$X'_i = \begin{cases} X_i, & \text{pokud } X_i \in N \\ \text{nový neterminál}, & \text{pokud } X_i \in \Sigma \end{cases}$$

   - 3.3. Pro všechny nové neterminály X'_i, kdy X_i musí být terminální symbol, vlož do P' nová pravidla $X'_i \to X_i$.

Algoritmus se evidentně zastaví, protože provádíme cykly pouze přes prvky množiny pravidel P, která je konečná, a také všechny ostatní kroky jsou konečné.

Věta 5.18
Ke každému bezkontextovému jazyku L existuje gramatika G' v Chomského normální formě taková, že $L(G') = L$.

Důkaz: Nechť $L = L(G)$ pro nějakou bezkontextovou gramatiku. Bez újmy na obecnosti můžeme předpokládat, že G nemá ε-pravidla ani jednoduchá pravidla (lze odstranit algoritmy 5.4 a 5.5). Pošleme tuto gramatiku na vstup předchozího algoritmu 5.6 a dostaneme tak gramatiku G'. Ta je zřejmě v Chomského normální formě. Musíme ještě dokázat, že $L(G') = L(G)$.

Uvedeme tzv. lemma o substituci, které lze s výhodou uplatnit při důkazu.

Lemma 5.19 O SUBSTITUCI CFG
Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika a $A \to \alpha B \beta$ její pravidlo. Vezmeme všechna B-pravidla z P: $B \to \gamma_1 \mid \cdots \mid \gamma_n$ a vytvoříme novou množinu pravidel

$$P' = (P \setminus \{A \to \alpha B \beta\} \cup \{A \to \alpha\gamma_1\beta, \ldots, \alpha\gamma_n\beta\})$$

a sestavíme novou gramatiku $G' = (N, \Sigma, P', S)$. Potom $L(G') = L(G)$.

Důkaz: zřejmý. □

pokračování důkazu věty 5.18: Aplikujeme předchozí lemma o substituci na všechny nově zavedené neterminály a dostaneme původní gramatiku. □

Příklad 5.20
Nechť gramatika $G = (\{S, A, B\}, \{a, b\}, P, S)$ má následující pravidla:

$$\begin{aligned} S &\to aAB \mid BA \\ A &\to BBB \mid a \\ B &\to AS \mid b \mid bB \end{aligned}$$

S pomocí algoritmu 5.6 sestrojíme gramatiku $G' = (\{S, A, B, a', b', \langle AB \rangle, \langle BB \rangle\}, \{a, b\}, P', S)$, která má tato pravidla:

$$\begin{array}{lll} S \to BA & A \to a & B \to AS \\ S \to a'\langle AB\rangle & A \to B\langle BB\rangle & B \to b \\ a' \to a & \langle BB\rangle \to BB & B \to b'B \\ \langle AB\rangle \to AB & & b' \to b \end{array}$$

Poznámka 5.21

Každý derivační strom utvořený podle gramatiky v Chomského normální formě má tyto dvě vlastnosti:

1. z každého vnitřního uzlu vycházejí dvě nebo jedna hrana
2. z uzlu vychází jediná hrana právě tehdy, když tato hrana vchází do listu.

Tyto vlastnosti často umožňují zjednodušovat tvrzení týkající se bezkontextových jazyků.

Definice 5.22

Bezkontextová gramatika $G = (N, \Sigma, P, S)$ je v *Greibachové normální formě (GNF),* pokud je bez ε-pravidel a všechna pravidla mají tvar $A \to a\alpha$, kde A je neterminál, a terminál a α (případně prázdný) řetězec neterminálů.

Definice 5.23

Neterminál A bezkontextové gramatiky G se nazývá *rekurzivní,* pokud existuje odvození $A \Rightarrow^+ \alpha A\beta$ pro nějaké $\alpha, \beta \in (N \cup \Sigma)^*$. Je-li $\alpha = \varepsilon$ (resp. $\beta = \varepsilon$), pak A se nazývá *levorekurzivní* (resp. *pravorekurzivní*).

Lemma 5.24 Odstranění bezprostřední levé rekurze

Nechť

$$A \to A\alpha_1 \mid \cdots \mid A\alpha_m \mid \beta_1 \mid \cdots \mid \beta_n \tag{10}$$

jsou všechna A-pravidla gramatiky $G = (N, \Sigma, P, S)$. Zkonstruujeme gramatiku $G' = (N \cup \{A'\}, \Sigma, P', S)$, kde A' je nový neterminál a P' vznikne z P nahrazením všech A-pravidel těmito pravidly:

$$A \to \beta_1 \mid \cdots \mid \beta_n \mid \beta_1 A' \mid \cdots \mid \beta_n A' \tag{11}$$

$$A' \to \alpha_1 \mid \cdots \mid \alpha_m \mid \alpha_1 A' \mid \cdots \mid \alpha_m A' \tag{12}$$

Pak $L(G') = L(G)$.

Důkaz: nabíledni. □

Příklad 5.25

Podle předchozího lemmatu lze převést bezprostředně levorekurzivní gramatiku

$$\begin{array}{lll} E & \to & E + T \mid T \\ T & \to & T * F \mid F \\ F & \to & i \mid (E) \end{array}$$

do tvaru bez levé rekurze:

$$\begin{array}{ll} E \to T \mid TE' & E' \to +T \mid +TE' \\ T \to F \mid FT' & T' \to *F \mid *FT' \\ F \to i \mid (E) & \end{array}$$

Algoritmus 5.7 Eliminace levé rekurze CFG

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: bezkontextová gramatika $G' = (N', \Sigma, P', S)$ bez levé rekurze taková, že $L(G') = L(G)$

1. Zvolíme libovolné uspořádání množiny neterminálů a označíme $N = \{A_1, \ldots, A_n\}$

2. `for` $i := 1$ `to` n `do`

   2.1. `for` $j := 1$ `to` $i - 1$ `do`

      2.1.1. Všechna pravidla $A_i \to A_j\alpha$ nahradíme pravidly $A_i \to \beta_1\alpha \mid \cdots \mid \beta_k\alpha$, kde $A_j \to \beta_1 \mid \cdots \mid \beta_k$ jsou všechna momentálně platná A_j-pravidla

      2.1.2. Provedeme eliminaci bezprostřední levé rekurze podle lemmatu 5.24.

Věta 5.26
Předchozí algoritmus odstranění levé rekurze je správný.

Důkaz: Ekvivalence gramatik plyne z lemmatu o substituci 5.19 a lemmatu o odstranění bezprostřední levé rekurze 5.24.

Zbývá ukázat, že G' skutečně není levorekurzivní (tedy neobsahuje neterminál, který by byl levorekurzivní). Žádný neterminál však není bezprostředně levorekurzivní. Dále, pokud $A_k \to A_l\alpha$, potom musí být $k < l$. Dokážeme indukcí vzhledem ke k:

1. $k = 1$: A_1 není (jako všechny ostatní neterminály) bezprostředně levorekurzivní, proto musí být $1 < l$ pro pravidlo $A_1 \to A_l\alpha$.

2. *indukční krok:* Nechť $A_k \to A_l\alpha$ je pravidlo původní gramatiky G. Jestliže $k < l$, je vše v pořádku. Bylo-li $k > l$, je toto pravidlo nahrazeno pravidly $A_k \to \beta_1\alpha \mid \cdots \mid \beta_m\alpha$, kde β_j jsou pravé strany všech nových A_l-pravidel. Na A_l lze použít indukční předpoklad, tedy slova β_j neobsahují na prvním místě neterminál menší nebo roven l. Nechť tedy obsahuje některé β_j neterminál A_r na prvním místě. Protože $l < r$, pak buď A_r bude ještě nahrazeno svými pravými stranami (pokud $r < k$) nebo bude odstraněno jako bezprostřední levá rekurze (pokud $r = k$) nebo je v pořádku (pokud $r > k$).

Kdyby se dalo v G' odvodit $A_k \Rightarrow^* A_k\alpha$, muselo by odvození postupovat $A_k = A_{r_0} \Rightarrow A_{r_1}\alpha_1 \cdots \Rightarrow A_{r_{s-1}}\alpha_{s-1} \Rightarrow A_{r_s}\alpha_s$, kde $A_{r_s} = A_k$ a $\alpha_s = \alpha$. Je tedy $r_0, r_1, \ldots, r_s$ ostře rostoucí posloupnost, ve které první a poslední prvek jsou si rovny, což je spor. □

Poznámka 5.27
Nechť $G = (N, \Sigma, P, S)$ je bezkontextová nelevorekurzivní gramatika. Potom existuje uspořádání $(N, \leq)$ takové, že pokud $A \to B\alpha$ je pravidlo, pak $A < B$. Příslušnou relaci R lze definovat takto: $A \,\mathrm{R}\, B$, pokud $A = B$ nebo $A \Rightarrow^+ B\alpha$. Uspořádání lze pak libovolně zúplnit.

Algoritmus 5.8 Transformace CFG do GNF

Vstup: nelevorekurzivní bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: bezkontextová gramatika $G' = (N', \Sigma, P', S)$ v Greibachové normální formě taková, že $L(G') = L(G)$

1. Zvolíme nějaké uspořádání $\leq$ s vlastností podle předchozí poznámky a označíme $N = \{A_1, \ldots, A_n\}$; $A_{i-1} < A_i$.

2. `for` $i := n$ `to` 1 `do`

   2.1. `for` $j := n$ `to` $i + 1$ `do`

      2.1.1. Všechna pravidla $A_i \to A_j\alpha$ nahradíme pravidly $A_i \to \beta_1\alpha \mid \cdots \mid \beta_k\alpha$, kde $A_j \to \beta_1 \mid \cdots \mid \beta_k$ jsou všechna A_j-pravidla

   2.2. Všechna pravidla $A_i \to aX_1 \cdots X_m$ nahradíme $A_i \to aX'_1 \cdots X'_m$, kde $X'_l = X_l$ pro $X_l \in N$ a X'_l je nový neterminál pro $X_l \in \Sigma$, přičemž přidáme pravidlo $X'_l \to X_l$.

Věta 5.28
Předchozí algoritmus transformace do Greibachové normální formy pracuje správně.

Důkaz: Sestupnou indukcí lze ukázat, že všechna A_i-pravidla výsledné gramatiky mají na prvním místě pravé strany terminál.

1. $i = n$: Pokud $A_n \to A_k\alpha$ v původní gramatice, bylo by $n < k$. To však není možné, protože A_n je maximální neterminál. Je tedy na prvním místě pravé strany všech A_n-pravidel terminál.

2. *indukční krok:* Nechť $A_i \to A_k\alpha$ bylo pravidlo původní gramatiky G. Potom muselo být $i < k$, a tedy toto pravidlo je v průběhu algoritmu nahrazeno pravidly $A_i \to \beta_1\alpha \mid \cdots \mid \beta_m\alpha$, kde β_j jsou pravé strany všech A_k-pravidel. Na ně můžeme užít indukční předpoklad, tedy na prvním místě všech β_j je terminál. Proto i na první místě všech pravých stran A_i-pravidel je terminál.

Všechny terminály X_l uvnitř pravých stran pravidel byly nahrazeny neterminály, proto všechna tato pravidla splňují podmínku GNF. Také nově zavedená pravidla $X'_l \to X_l$ jsou v pořádku. □

Příklad 5.29

Výslednou gramatiku z příkladu 5.25 převedeme do GNF. Nejprve uspořádáme neterminály: $E < E' < T < T' < F$. Po uplatnění předchozího algoritmu dostaneme gramatiku

$$\begin{aligned}
F &\to i \mid (E)' \\
T' &\to *FT' \mid *F \\
T &\to i \mid (E)' \mid iT' \mid (E)'T' \\
E' &\to +TE' \mid +T \\
E &\to i \mid (E)' \mid iT' \mid (E)'T' \mid iE' \mid (E)'E' \mid iT'E' \mid (E)'T'E' \\
)' &\to)
\end{aligned}$$

Důsledek 5.30

Ke každé bezkontextové gramatice existuje k ní ekvivalentní gramatika, která je v Greibachové normální formě.

Důkaz: Algoritmus 5.7 odstraní z libovolné bezkontextové gramatiky levou rekurzi. Výslednou gramatiku pošleme na vstup algoritmu 5.8, který ji převede do Greibachové normální formy. □

Definice 5.31

Řekneme, že gramatika G je v *redukované Greibachové normální formě,* pokud neobsahuje ε-pravidla a všechna pravidla mají tvar $A \to a\alpha$, kde a je terminál a α je buď prázdné slovo, nebo složené z jednoho či dvou neterminálů ($\alpha = \varepsilon$ nebo $\alpha = B$ nebo $\alpha = BC$).

5.3 Uzávěrové vlastnosti CFL

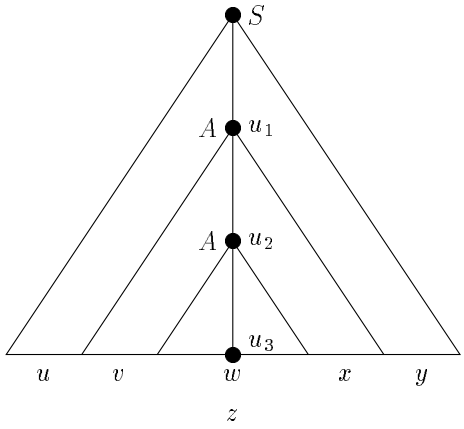
Věta 5.32 Pumping lemma pro CFL neboli „*uvwxy*“ teorém

Nechť L je bezkontextový jazyk. Potom existují přirozená čísla p, q taková, že pro každé slovo $z \in L$ jazyka L platí: pokud $|z| > p$, pak slovo z lze zapsat ve tvaru $z = uvwxy$, kde $|vwx| \leq q$, alespoň jedno ze slov v, x je různé od prázdného slova ε a všechna slova uv^iwx^iy patří do jazyka L pro každé přirozené $i \geq 0$.

Důkaz: Nechť $G = (N, \Sigma, P, S)$ je gramatika v Chomského normální formě, která generuje jazyk L. Taková gramatika podle věty 5.18 vždy existuje. Nechť k je počet neterminálů této gramatiky. Položíme $p = 2^{k-1}$ a $q = 2^k$.

V dalším využijeme vlastnosti derivačních stromů příslušných gramatikám v Chomského normální formě, které byly zmíněny v poznámce 5.21. Vnitřní uzly takového stromu mají vždy dva syny, kromě uzlů bezprostředně předcházejících listy stromu. Jestliže takový strom nemá žádnou cestu delší než j, pak terminální řetězec odvozený na jeho základě má délku nejvýše 2^{j-1}. Strom, z jehož každého uzlu vycházejí právě dvě hrany a jehož každá cesta má délku právě j, má 2^j listů. U studovaných derivačních stromů k tomu přistupuje navíc okolnost, že hrana vedoucí do libovolného listu odpovídá pravidlu typu $A \to a$. Je tedy počet listů roven počtu uzlů, které jim bezprostředně předcházejí. Těch je, jak již bylo uvedeno, nejvýše 2^{j-1}. I listů je proto nejvýše 2^{j-1}.

Jestliže je tedy slovo $z \in L$ delší než p, existuje v libovolném derivačním stromu příslušném k nějaké derivaci z cesta délky větší než k. Zvolíme nyní pevně jeden takový pevný derivační strom T a označme v něm C jeho nejdelší cestu. Protože ta má délku větší než k, vyskytuje se na ní jeden neterminál alespoň dvakrát. Označíme tento neterminál A a vybereme na cestě C uzly u_1, u_2, u_3 těchto vlastností (viz obrázek 3):



Obr. 3: Výběr uzlů u_1, u_2, u_3 v derivačním stromu z gramatiky v CNF

1. Oba uzly u_1, u_2 odpovídají neterminálu A
2. Uzel u_1 leží blíž kořenu stromu než u_2
3. u_3 je list
4. Cesta z u_1 do u_3 má délku nejvýše $k+1$

Uzel u_1 určuje ve stromu T podstrom T_1, který má kořen právě u_1, podobně u_2 určuje podstrom T_2, přičemž T_2 je podstromem T_1. Strom T_1 odpovídá levé derivaci nějakého slova z_1, které má délku nejvýše 2^k. To z toho důvodu, že každá cesta v T_1 má délku nejvýše $k+1$. Kdyby totiž existovala v T_1 cesta delší než $k+1$, dala by společně s cestou z kořene stromu T do u_1 cestu delší než C, což je spor s maximalitou cesty C. Proto $|z_1| \leq q$. Strom T_2 je podstromem T_1 a určuje derivaci nějakého slova, které označíme w. Slovo z_1 se pak dá složit takto: $z_1 = vwx$ pro nějaká slova v, x. Pravidlo odpovídající hranám vycházejícím z uzlu u_1 musí být tvaru $A \to BC$. Protože jen po jedné ze dvou hran, které vedou z uzlu u_1, může pokračovat cesta C, dá se z druhé větve odvodit neprázdné slovo ve stromu T. Proto je alespoň jedno ze slov v, x neprázdné. Celkem jsme ukázali, že

$$A \Rightarrow_G^* vAx \Rightarrow_G^* vwx, \tag{13}$$

kde $|vwx| \leq q$ a $vx \neq \varepsilon$. Z toho ovšem plyne, že také

$$A \Rightarrow_G^* vAx \Rightarrow_G^* vvAxx \Rightarrow_G^* \cdots \Rightarrow_G^* v^i A x^i \Rightarrow_G^* v^i w x^i \tag{14}$$

Kromě toho $A \Rightarrow_G^* w = v^0 w x^0$. Celé slovo z se zřejmě dá zapsat jako $z = uvwxy$ a všechny komponenty mají požadované vlastnosti. □

Právě dokázanou větu lze použít pro důkaz, že nějaký jazyk není bezkontextový, jak ukazuje následující příklad.

Příklad 5.33

Ukážeme, že jazyk $L = \{a^n b^n c^n : n \geq 0\}$ není bezkontextový.

Předpokládejme naopak, že L bezkontextový je. Tedy existují čísla p, q s vlastnostmi uvedenými v předcházejícím Pumping lemmatu. Zvolme číslo $k > \frac{p}{3}$. Potom slovo $w_1 = a^k b^k c^k$ je možno psát ve tvaru $uvwxy$, kde $vx \neq \varepsilon$ a každé $w_i = uv^i wx^i z$ leží v L. Protože ve všech slovech v L se symboly a, b, c vyskytují výhradně v abecedním pořádku, může každé ze slov v, x obsahovat pouze stejné symboly. Ve slovech w_i tedy s rostoucím i roste počet nejvýše dvou symbolů, zatímco třetího symbolu zůstává stále stejné množství. To ale není v jazyce L možné.

Tvrzení 5.34

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1$$

Důkaz: Tvrzení říká, že $\mathcal{L}_3$ je vlastní podtřídou $\mathcal{L}_2$ a $\mathcal{L}_2$ je vlastní podtřídou $\mathcal{L}_1$. Jinými slovy, existuje bezkontextový jazyk, který není regulární, a existuje kontextový jazyk, který není bezkontextový.

V příkladu 2.12 je uveden důkaz, že jazyk $\{0^n1^n\}$ není rozpoznatelný žádným konečným automatem, a není proto regulární. Příklad 4.3 ukazuje, že je tento jazyk rozpoznatelný zásobníkovým automatem, a je proto bezkontextový.

Podle předchozího příkladu není $\{a^nb^nc^n\}$ bezkontextový jazyk. Ukážeme však, že je kontextový. Je totiž generován následující kontextovou gramatikou:

$$\begin{array}{lll} X_0 \to X_0Z_1 & X_0 \to X_1Z_1 & X_1 \to aY_1 \\ X_1 \to aX_1Y_1 & Y_1Z_1 \to Y_1Z_3 & Y_1Z_3 \to YZ_3 \\ YZ_3 \to YZ & Y_1Y \to Y_1Y_2 & Y_1Y_2 \to YY_2 \\ YY_2 \to YY_1 & ZZ_1 \to Z_2Z_1 & Z_2Z_1 \to Z_2Z \\ Z_2Z \to Z_1Z & Y \to b & Z \to c \end{array}$$

kde X_0 je počáteční symbol. □

Věta 5.35

Existuje algoritmus, podle kterého lze ke každé bezkontextové gramatice G sestrojit čísla m, n taková, že $L(G)$ je nekonečný právě tehdy, když existuje číslo $z \in L(G)$ takové, že $m < |z| \leq n$.

Důkaz: Budeme opět předpokládat, že gramatika G je v Chomského normální formě, což můžeme podle věty 5.18. K této gramatice existují podle „*uvwxy*“ teorému čísla p, q příslušných vlastností. Položme $m = p$ a $n = p + q$.

Jestliže nyní $L(G)$ obsahuje slovo z takové, že $m < |z| \leq n$, lze je psát ve tvaru $z = uvwxy$, kde $vx \neq \varepsilon$ a uv^iwx^iy leží v $L(G)$ pro každé i. To ale znamená, že jazyk $L(G)$ je nekonečný.

Je-li naopak jazyk $L(G)$ nekonečný, obsahuje nekonečně mnoho slov délky větší než p. Z množiny těchto slov vybereme slovo z minimální délky. Kdyby $|z| > p + q$, aplikujeme na ně opět „*uvwxy*“ teorém a můžeme je psát ve tvaru $z = uvwxy$, kde opět v nebo x je neprázdné a $|vwx| \leq q$ a přitom $uv^iwx^iy \in L(G)$ pro každé i, tedy i pro $i = 0$. Z toho dostáváme $uwy \in L(G)$, ale $|uwy| < |uvwxy|$ a $|uwy| > (p + q) - q = p$, což je spor s minimalitou výběru slova z. □

Uvědomme si, že právě dokázaná věta umožňuje najít algoritmus, který pro každou bezkontextovou gramatiku rozhodne, zda generuje konečný nebo nekonečný jazyk. Je k tomu ovšem ještě třeba najít algoritmus, který by pro libovolné slovo w a bezkontextovou gramatiku rozhodoval, zda $w \in L(G)$. Pak stačí otestovat všechna slova délky mezi p a $p + q$. Existence tohoto algoritmu je dokázána obecněji pro kontextové gramatiky ve větě 7.4.

Věta 5.36

Třída bezkontextových jazyků $\mathcal{L}_2$ není uzavřena na operaci průniku.

Důkaz: Jazyky $L_1 = \{a^nb^nc^m : n, m \geq 1\}$ a $L_2 = \{a^mb^nc^n : m, n \geq 1\}$ jsou bezkontextové. Jazyk L_1 je například generován gramatikou

$$\begin{array}{rcl} S & \to & S_1C \\ S_1 & \to & aS_1b \mid ab \\ C & \to & Cc \mid c. \end{array}$$

Obdobně lze zkonstruovat bezkontextovou gramatiku generující jazyk L_2. Avšak jazyk

$$\{a^nb^nc^n : n \geq 1\} = L_1 \cap L_2$$

není podle příkladu 5.33 bezkontextový. □

Věta 5.37

Třída bezkontextových jazyků $\mathcal{L}_2$ je uzavřena na operaci sjednocení.

Důkaz: Nechť L_1 a L_2 jsou jazyky generované bezkontextovými gramatikami $G_1 = (N_1, \Sigma_1, P_1, S_1)$ a $G_2 = (N_2, \Sigma_2, P_2, S_2)$. Předpokládejme, že $N_1 \cap N_2 = \emptyset$, čehož lze vždy dosáhnout vhodným přejmenováním neterminálů. Jazyk $L_1 \cup L_2$ je pak generován bezkontextovou gramatikou

$$G = (N_1 \cup N_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}, S).$$

Derivace v gramatice G totiž budou vždy probíhat tak, že nejprve se vybere buď $S \to S_1$ nebo $S \to S_2$. Pak už bude jistě odvozování pokračovat pouze podle G_1, resp. G_2. □

Věta 5.38
Třída bezkontextových jazyků $\mathcal{L}_2$ není uzavřena na operaci doplňku.

Důkaz: Plyne bezprostředně z de Morganových pravidel a předchozích dvou vět. Je totiž $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. Kdyby byla třída $\mathcal{L}_2$ uzavřena na doplněk, protože je uzavřena na sjednocení, musela by také být uzavřena na průnik. To však není pravda. □

Věta 5.39
Třída bezkontextových jazyků $\mathcal{L}_2$ je uzavřená na operaci průniku s regulárním jazykem.

Důkaz: Chceme dokázat, že pokud $L \in \mathcal{L}_2$ je bezkontextový jazyk a $R \in \mathcal{L}_3$ je regulární jazyk, pak $L \cap R \in \mathcal{L}_2$ je bezkontextový.

Nechť $M = (Q_1, \Sigma, \Gamma, \delta_1, q_{01}, Z_0, F_1)$ je zásobníkový automat takový, že $T(M) = L$. Nechť dále $A = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ je konečný automat takový, že $T(A) = R$.

Sestrojíme zásobníkový automat $M' = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, přičemž stavy automatu M' odpovídají kartézskému součinu stavů M a A: $Q = Q_1 \times Q_2$, koncové stavy jsou koncové stavy obou automatů: $F = F_1 \times F_2$ a počáteční stav je dán počátečními stavy obou automatů: $q_0 = [q_{01}, q_{02}]$. Přechodová funkce δ je dána takto: $\delta([p,q], a, Z)$ obsahuje prvek $([r,s], \gamma)$, právě když $\delta_1(p, a, Z)$ obsahuje (r, γ) a $\delta_2(q, a) = s$. Potom zřejmě $T(M') = L \cap R$. □

Věta 5.40
Třída bezkontextových jazyků $\mathcal{L}_2$ je uzavřená na iteraci i na uzávěr.

Důkaz: Chceme ukázat, že pokud je jazyk $L \in \mathcal{L}_2$, pak také $L^i, L^*, L^+ \in \mathcal{L}_2$.

Vezměme bezkontextovou gramatiku $G = (N, \Sigma, P, S)$, která generuje jazyk L. Potom gramatika $G_i = (N \cup \{S'\}, \Sigma, P \cup \{S' \to S^i\}, S')$ generuje zřejmě jazyk L^i. Gramatika $G_+ = (N \cup \{S'\}, \Sigma, P \cup \{S' \to SS'\}, S')$ generuje jazyk L^+ a gramatika $G_* = (N \cup \{S'\}, \Sigma, P \cup \{S' \to SS', S' \to \varepsilon\}, S')$ generuje jazyk L^*. Ve všech případech je S' nový neterminál neobsažený v N. □

Věta 5.41
Třída bezkontextových jazyků $\mathcal{L}_2$ je uzavřená na operaci zřetězení.

Důkaz: Nechť L_1 je bezkontextový jazyk generovaný gramatikou $G_1 = (N_1, \Sigma_1, P_1, S_1)$ a L_2 buď bezkontextový jazyk generovaný gramatikou $G_2 = (N_2, \Sigma_2, P_2, S_2)$. Zkonstruujeme gramatiku

$$G = (N_1 \cup N_2 \cup \{S'\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S' \to S_1 S_2\}, S'),$$

která ze zřejmých důvodů generuje jazyk $L_1 L_2$ a zůstává přitom bezkontextová. □

Příklad 5.42
Jazyk

$$L = \{a^{m_1} b^{m_1} a^{m_2} b^{m_2} \cdots a^{m_n} b^{m_n} : m_i \geq 0, n \geq 0\}$$

je bezkontextový podle předchozí věty proto, že je zřetězením bezkontextových jazyků $L_1 L_2 \cdots L_n$, kde $L_i = \{a^{m_i} b^{m_i} : m_i \geq 0\}$.

Přejdeme nyní blíže ke vztahu bezkontextových a regulárních jazyků. Jak známo, regulární jazyky jsou podtřídou bezkontextových jazyků. Pokusíme se objevit, čím se vyznačují bezkontextové jazyky, které nejsou regulární.

Definice 5.43
O bezkontextové gramatice $G = (N, \Sigma, P, S)$ řekneme, že má vlastnost *sebevložení,* jestliže existuje neterminál $A \in N$ a slova $u, v \in (N \cup \Sigma)^+$ tak, že $A \Rightarrow^+ uAv$.

Bezkontextový jazyk L má vlastnost *sebevložení,* jestliže <u>každá</u> gramatika, která ho generuje, má vlastnost sebevložení.

Věta 5.44
Bezkontextový jazyk má vlastnost sebevložení právě tehdy, když není regulární

Důkaz: $\Rightarrow$: Je-li jazyk regulární, pak ho lze generovat regulární gramatikou. Žádná regulární gramatika však vlastnost sebevložení nemá, proto ani žádný regulární jazyk nemá vlastnost sebevložení.

$\Leftarrow$: Nechť naopak je jazyk L generovaný nějakou bezkontextovou gramatikou, která nemá vlastnost sebevložení. Tuto gramatiku můžeme transformovat do Greibachové normální formy. Při procedurách s tím spojených nemůžeme vlastnost sebevložení přidat, pokud v gramatice předtím nebyla obsažena. Označme nyní n počet neterminálů transformované gramatiky a d maximální délku pravé strany pravidla. Ukážeme, že při libovolné levé derivaci podle G se v žádném řetězci nemůže vyskytovat více než nd neterminálů.

Předpokládejme, že je to naopak možné. Tzn. existuje řetězec α, pro který $S \Rightarrow^* \alpha$ je levá derivace a α obsahuje více než nd neterminálů. Uvědomme si, že levá derivace podle gramatiky v Greibachové normální formě zaručuje, že řetězec α se dá rozepsat jako $\alpha = x\beta$, kde $x \in \Sigma^*$ je terminální řetězec a $\beta \in N^*$ je neterminální řetězec.

Protože při každém kroku derivace přibude ve slově α nejvýš $d-1$ nových neterminálů, má ve stromu popisujícím levou derivaci α cesta od kořene k tomu listu, který je nejlevějším neterminálem α, délku větší než n. Dokonce i vrcholů, ze kterých vycházejí alespoň tři hrany, je na této cestě více než n. To znamená, že alespoň dva z těchto vrcholů musí být ohodnoceny týmž neterminálem A. Z toho však odvodíme, že existuje derivace $A \Rightarrow^* uAv$, kde u i v jsou neprázdné řetězce, což je spor.

V každém kroku levé derivace se v řetězci α vyskytuje nejvýše nd neterminálů, které navíc tvoří souvislý řetěz na pravém kraji slova α. Můžeme proto simulovat všechny levé derivace v této gramatice jinou gramatikou, ve které budou všechny nd-tice původních neterminálů figurovat coby neterminály. Formálně gramatiku zapíšeme takto: $G' = (N', \Sigma, P', S')$, kde množina neterminálů N' je tvořena těmito prvky: $N' = \{(\alpha) : \alpha \in N^* \wedge |\alpha| \leq nd\}$. Startovní neterminál bude $S' = (S)$ a pravidla P' vytvoříme takto: pro každé pravidlo z P tvaru $A \to b\alpha$, kde $\alpha \in N^*$ (gramatika je v Greibachové normální formě), a pro každé slovo $\beta \in N^*$ takové, že $|\alpha\beta| \leq n$, bude P' obsahovat pravidlo $(A\beta) \to b(\alpha\beta)$. Pak zřejmě $L(G') = L(G)$, a přitom je G' regulární gramatika. Proto je jazyk L regulární. □

Příklad 5.45
Jazyk generovaný gramatikou

$$S \to aSb \mid aS \mid Sb \mid \varepsilon,$$

tedy $\{a^*b^*\}$, je regulární, protože výše uvedená gramatika nemá vlastnost sebevložení.

6 Deterministické ZA a CFL

Uvedeme nyní krátce problematiku deterministických zásobníkových automatů a deterministických bezkontextových jazyků. Jde o velmi důležitou třídu jazyků, která má své uplatnění při syntaktické analýze. Některé její speciální případy a vlastnosti jsou obsahem druhé části této přednášky, která se zabývá výhradně syntaktickou analýzou.

6.1 Deterministické ZA

Definice 6.1
Zásobníkový automat $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ nazveme *deterministický (DZA),* jestliže platí:
Je-li $\delta(q, \varepsilon, Z) \neq \emptyset$, tedy neprázdná množina, je pro každý terminální symbol $\delta(q, a, Z) = \emptyset$, tedy prázdné. Dále, libovolná množina, která je výsledkem funkce δ v jakémkoli bodě definičního oboru, má nejvýše 1 prvek.

Poznámka 6.2
Analogicky obyčejnému zásobníkovému automatu lze definovat u deterministického ZA akceptování slov koncovým stavem i prázdným zásobníkem. U DZA však nejsou jazyky akceptované koncovým stavem, právě když jsou akceptovány prázdným zásobníkem.

Příklad 6.3

Jazyk $L = \{w \in \{a,b\}^* : w$ obsahuje stejný počet symbolů a jako $b\}$ je rozpoznatelný DZA koncovým stavem, ale není rozpoznatelný prázdným zásobníkem žádného DZA.

Sestrojíme nejprve DZA, který přijímá jazyk L koncovým stavem. Nechť

$$M = (\{p,q\}, \{a,b\}, \{Z, A, A', B, B'\}, \delta, p, Z, \{p\})$$

a přechodovou funkci definujeme takto:

$$\begin{array}{ll}
\delta(p,a,Z) = (q, A'Z) & \\
\delta(p,b,Z) = (q, B'Z) & \\
\delta(q,a,B) = (q,\varepsilon) & \delta(q,b,A) = (q,\varepsilon) \\
\delta(q,a,B') = (p,\varepsilon) & \delta(q,b,A') = (p,\varepsilon) \\
\delta(q,a,A) = (q,AA) & \delta(q,b,B) = (q,BB) \\
\delta(q,a,A') = (q,AA') & \delta(q,b,B') = (q,BB')
\end{array}$$

M je zřejmě deterministický zásobníkový automat a během výpočtu vede evidenci o rozdílu počtu symbolů a a symbolů b, které doposud přečetl. Jestliže převládá symbol a o n výskytů, pak zásobník obsahuje počáteční symbol, nad ním A' a nad ním $n-1$ symbolů A. Podobně je tomu u b. Čárkované symboly signalizují, že počet výskytů se liší o jedničku, takže vždy v okamžiku, kdy se bilance vyrovnává, může automat přejít do stavu p.

Můžeme také dokázat, že jazyk L není rozpoznatelný žádným deterministickým zásobníkovým automatem prázdným zásobníkem. Deterministický zásobníkový automat totiž končí výpočet vždy ve chvíli, kdy je prázdný zásobník, a nemůže v tomto případě pokračovat ve výpočtu. Kdyby nějaký automat rozpoznával jazyk L prázdným zásobníkem, musel by mít prázdný zásobník po přečtení slova $abba$. Avšak slovo $abbaba$ je také slovem jazyka L, ale automat jej nemůže rozpoznat, protože po přečtení $abba$ nemůže pokračovat ve výpočtu.

Věta 6.4

Ke každému deterministickému zásobníkovému automatu M_1 existuje deterministický zásobníkový automat M_2 takový, že $N(M_1) = T(M_2)$.

Existuje regulární jazyk, který není rozpoznatelný deterministickým zásobníkovým automatem prázdným zásobníkem.

Důkaz: Automat M_2 dostaneme z M_1 stejným způsobem, jako tomu bylo v nedeterministickém případě—viz důkaz věty 4.4.

Příklad takového regulárního jazyka je $\{a, aa\}$, což lze zdůvodnit stejně jako v předchozím příkladu. □

Definice 6.5

Jazyk L nazveme *bezprefixový,* pokud existuje deterministický zásobníkový automat, který ho rozeznává prázdným zásobníkem.

Definice 6.6

Jazyk L nazveme *deterministický bezkontextový,* pokud existuje deterministický zásobníkový automat, který ho rozeznává koncovým stavem.

Důsledek 6.7

Třída bezprefixových jazyků je vlastní podtřídou deterministických bezkontextových jazyků, které jsou podtřídou (zatím nevíme, zda vlastní) bezkontextových jazyků.

6.2 Uzávěrové vlastnosti deterministických CFL

Věta 6.8

Třída deterministických bezkontextových jazyků je uzavřena na operaci průniku s regulárním jazykem.

Důkaz: Potřebujeme dokázat, že pokud je L deterministický bezkontextový jazyk a R regulární, pak $L \cap R$ je také deterministický bezkontextový jazyk. Důkaz lze vést obdobně jako v případě nedeterministickém, tedy zkonstruujeme zásobníkový automat M' ze zásobníkového automatu M a konečného automatu A takových, že $T(M) = L$ a $T(A) = R$. Konstrukce proběhne stejně jako v důkazu věty 5.39. Automaty M i R byly deterministické, proto také automat M' bude deterministický a bude rozeznávat jazyk $L \cap R$. □

Věta 6.9
Třída deterministických bezkontextových jazyků není uzavřena vzhledem k operaci průniku.

Důkaz: Za usvědčující příklad mohou sloužit stejné jazyky, které jsme použili v důkazu věty 5.36. Jde o jazyky $L_1 = \{a^n b^n c^m : n, m \geq 1\}$ a $L_2 = \{a^m b^n c^n : m, n \geq 1\}$, které jsou deterministické bezkontextové. Jejich průnik $\{a^n b^n c^n : n \geq 1\}$ však není ani bezkontextový, natož pak deterministický bezkontextový. □

Věta 6.10
Deterministické bezkontextové jazyky jsou uzavřeny na operaci doplňku.

Důkaz: Nechť $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je deterministický ZA a $T(M) = L$. Chceme získat deterministický ZA, který rozeznává doplněk jazyka L. K tomu však nestačí pouze změnit automat M tak, že se nahradí cílová množina F množinou $Q \setminus F$. Tento způsob jsme bez problému použili u konečných automatů—viz důkaz věty 3.4. Zde narážíme na potíže:

(1) M nemusí během výpočtu nad určitým vstupním slovem nikdy dojít až na konec slova, a to buď

(1a) proto, že se dostane do situace, kdy další krok není definován, tj. když se zcela vyprázdní zásobník, nebo

(1b) proto, že se dostane do cyklu, při kterém automat přepracovává obsah zásobníku, ale dělá jen ε-kroky, tj. nepožaduje další vstupní symbol.

Jestliže se automat nedostane na konec slova, slovo není přijato. Při pouhém prohození přijímajících a nepřijímajících stavů se nově vzniklý automat opět nedostane na konec slova a toto slovo opět nebude přijato.

(2) Po zpracování určitého slova může automat udělat několik ε-kroků, při kterých se dostane do koncových i nekoncových stavů. V tom případě je slovo přijato původním i transformovaným automatem.

Musíme tedy překonat obě obtíže. K překonání první z nich si pomůžeme následujícím lemmatem.

Lemma 6.11 O POKRAČUJÍCÍM AUTOMATU
Ke každému deterministickému zásobníkovému automatu M existuje deterministický zásobníkový automat M' takový, že $T(M') = T(M)$ a M' vždy projde celé vstupní slovo.

Důkaz: Nejprve pozměníme automat M tak, že do zásobníkové abecedy přidáme nový symbol Z_1, který bude počáteční. Na počátku výpočtu bude tento symbol vložen na dno zásobníku. Potom necháme na dno položit počáteční symbol původního automatu a automat se chová stejně jako původní automat M. Jestliže se původní automat M zastavil proto, že jeho zásobník byl prázdný, nový automat bude mít stále ještě symbol Z_1 na dně zásobníku. Vytvoříme tedy nový stav K, do kterého necháme automat přejít v případě, že zbyde na dně pouze Z_1. V tomto stavu bude automat zůstávat, v zásobníku bude neustále pouze symbol Z_1 a automat bude takto číst libovolný symbol vstupu. Dostane se tak do stavu K (někdy nazývaný *mrtvý stav*), kdy celé vstupní slovo je přečtené, a tedy toto slovo zůstává zcela správně nepřijato ani novým automatem, protože K není stav koncový. Nenastane tedy situace popsaná v bodě (1a).

Zbývá vyřešit případ (1b), tj. případ nekonečné posloupnosti ε-kroků. Takové situace je možno rozdělit na dva druhy: (i) zásobník během nekonečné posloupnosti ε-kroků neomezeně roste, nebo (ii) zásobník nikdy nepřekročí určitou maximální délku. Prozkoumáme tedy oba druhy situací.

Případ (i): Nechť se tedy automat dostane do nekonečného cyklu, ve kterém vykonává pouze ε-kroky, a nechť během tohoto cyklu délka zásobníku přeroste všechny meze. Označíme: $s = |Q|$ počet stavů, $t = |\Gamma|$ počet zásobníkových symbolů a

$$r = \max_{p \in Q, Z \in \Gamma} \{|\gamma| : \delta(p, \varepsilon, Z) = (p', \gamma)\} - 1$$

maximální počet symbolů, o něž se během jednoho ε-kroku může zvětšit zásobník. Ukážeme následující tvrzení: Zásobník neomezeně poroste a bude nadále vykonávat pouze ε-kroky, právě když se během nějaké nepřerušované posloupnosti ε-kroků zvětší o více než rst symbolů.

$\Rightarrow$: Jestliže roste zásobník nade všechny meze, pak pochopitelně překročí i číslo rst.

$\Leftarrow$: Pokud obráceně během nějaké nepřetržité posloupnosti ε-kroků se zásobník zvětší o více než rst symbolů, pak z této posloupnosti lze vybrat více než st konfigurací takových, že při přechodu od jedné k následující

zásobník nikdy neklesne zpět na úroveň počtu symbolů, kterou měl v první zmíněné konfiguraci. Mezi vybranými více než st konfiguracemi existují alespoň dvě, které se shodují vnitřním stavem i vrchním symbolem zásobníku. Označíme tyto konfigurace k_1 a k_2. Protože tyto konfigurace jsou vlastně identické a automat je deterministický, bude se po dosažení konfigurace k_2 chovat stejně, jako kdyby pokračoval z k_1. Automat tedy bude stále cyklicky vykonávat pouze ε-kroky, a jeho zásobník přitom poroste nade všechny meze.

Stačí proto modifikovat automat M tak, že do konečné paměti se zabuduje *čítač,* který může nabývat hodnot mezi 0 a rst. Tento čítač se vynuluje vždy po přečtení nového vstupního symbolu, zvětší či zmenší se po provedení ε-kroku o hodnotu, o kterou se změnila velikost zásobníku, a místo záporných čísel nabývá stále hodnoty 0. Jestliže by měl tento čítač přesáhnout hodnotu rst, přejde do mrtvého stavu, ve kterém přečte zbylé vstupní symboly a slovo nepřijme. Takto modifikovaný automat se nedostane do nekonečného ε-cyklu (1b), případ (i) zásobníku rostoucího nade všechny meze.

Případ (ii): Nechť M vykonává posloupnost ε-kroků, při které výška zásobníku nepřesáhne určitou hranici. Podle toho, co jsme dokázali, nikdy během této posloupnosti nevzroste zásobník o více než rst symbolů. Jestliže však automat vykonává nepřetržitou posloupnost ε-kroků, při které zásobník nikdy neklesne pod určitou hranici h a nikdy nad tuto hranici nevzroste o více než rst symbolů, potom nejpozději za

$$s(t+1)^{rst}$$

taktů se dostane do cyklu. Během té doby se totiž musí dostat alespoň do dvou konfigurací shodujících se ve vnitřním stavu automatu a v obsahu zásobníku nad hranicí h.

Potíž typu (1b) (ii) lze proto obejít další modifikací automatu M, která spočívá v zařazení nového konečného čítače, který může nabývat hodnot od 0 po $s(t+1)^{rst}$. Tento čítač nazveme pro rozlišení *čítač2.* Čítač2 se vynuluje, kdykoli se prvotní čítač vynuluje nebo klesne na hodnotu 0. Po dobu, kdy prvotní čítač nabývá nenulové hodnoty, se hodnota čítače2 v každém taktu zvětší o 1.

Takto modifikovaný automat opět simuluje automat M. Pouze v případě, že některý z čítačů „přeteče“, přejde automat do mrtvého stavu, ve kterém přečte celý zbytek slova, aniž by ho přijal. Čítače lze implementovat tak, že stavy Q nahradíme kartézským součinem $Q \times [0, \max]$, kde $[0, \max]$ je příslušný interval přirozených čísel, ve kterém se pohybují hodnoty čítače. Přechodovou funkci pak lze upravit tak, že zaznamenává do druhé komponenty stavu (čítače) příslušné hodnoty podle pravidel, která jsme pro čítače uvedli.

Popsaný automat je hledaným automatem M'. Tím je dokázáno lemma 6.11 o pokračujícím automatu. □

dokončení důkazu věty 6.10: Zbývá odstranit obtíž (2), při které automat přechází po přečtení celého vstupního slova pomocí ε-kroků mezi koncovými i nekoncovými stavy. Předpokládejme, že automat M má již odstraněnu kompletně obtíž (1) podle předchozího lemmatu. Vytvoříme automat $\overline{M} = (\overline{Q}, \Sigma, \Gamma, \overline{\delta}, \overline{q_0}, Z_0, \overline{F})$ takový, že $T(\overline{M}) = \Sigma^* \setminus L$. Definujeme stavy $\overline{Q} = Q \times \{p, n, f\}$, koncové stavy $\overline{F} = Q \times \{f\}$ a počáteční stav

$$\overline{q_0} = \begin{cases} [q_0, p], & \text{pokud } q_0 \in F \\ [q_0, n], & \text{pokud } q_0 \notin F \end{cases}$$

Automat $\overline{M}$ simuluje výpočet automatu M, přičemž první složka vždy odpovídá stavům M. Přitom automat $\overline{M}$ uchovává informaci o tom, zda od posledního přečtení vstupního symbolu prošel automat M koncovým stavem či nikoli. Pokud prošel, je druhá složka stavu rovna p, jestliže neprošel, je tato složka rovna n. Teprve v okamžiku, kdy M nemůže učinit další ε-krok, provede $\overline{M}$ jeden ε-krok navíc. Při tomto kroku nezmění první složku svého stavu, pouze může případně změnit druhou složku svého stavu na f. Tuto změnu může provést pouze u stavů, jejichž druhá složka je n. Formálně,

1. nechť $\delta(q, a, Z) = (q', \gamma)$ je přechod automatu M. Potom pro stavy typu f a p platí

$$\overline{\delta}([q, f], a, Z) = \overline{\delta}([q, p], a, Z) = \begin{cases} ([q', p], \gamma), & \text{pokud } q' \in F \\ ([q', n], \gamma), & \text{pokud } q' \notin F \end{cases}$$

2. nechť $\delta(q, \varepsilon, Z) = (q', \gamma)$ je přechod automatu M. Potom pro stavy všech typů platí

$$\begin{aligned} \overline{\delta}([q, p], \varepsilon, Z) &= ([q', p], \gamma) \\ \overline{\delta}([q, n], \varepsilon, Z) &= \begin{cases} ([q', p], \gamma), & \text{pokud } q' \in F \\ ([q', n], \gamma), & \text{pokud } q' \notin F \end{cases} \\ \overline{\delta}([q, f], \varepsilon, Z) &\text{ je definováno libovolně} \end{aligned}$$

3. nechť $\delta(q, \varepsilon, Z) = \emptyset$ v automatu M. Potom zavedeme

$$\overline{\delta}([q, n], \varepsilon, Z) = ([q, f], Z).$$

$\overline{M}$ tedy vstoupí po přečtení n symbolů do koncového stavu právě tehdy, když zjistí, že se M nedostal do koncového stavu ani při čtení n-tého symbolu ani v žádném dalším ε-kroku. $\overline{M}$ proto skutečně přijímá jazyk $\Sigma^* \setminus L$, a přitom je deterministický. □

Důsledek 6.12

Víme, že třída bezkontextových jazyků není uzavřená na doplněk, ale deterministické bezkontextové jazyky jsou uzavřené na doplněk. Z toho vyvozujeme, že deterministické bezkontextové jazyky jsou vlastní podtřídou bezkontextových jazyků.

Příklad 6.13

Jazyk $L = \{a^i b^j c^k : i \neq j \vee j \neq k \vee i \neq k\}$ je bezkontextový jazyk, který však není deterministický. Bezkontextovou gramatiku generující L lze sestrojit snadno. Kdyby však byl L deterministický, musel by také být $\overline{L}$ deterministický a tím spíše bezkontextový. Protože bezkontextové jazyky jsou uzavřené na průnik, byl by v tom případě také jazyk $\overline{L} \cap \{a^* b^* c^*\}$ bezkontextový. Jde však o jazyk $\{a^n b^n c^n\}$, o kterém již víme, že bezkontextový není.

Důsledek 6.14

Třída deterministických bezkontextových jazyků není uzavřena na operaci sjednocení.

Důkaz: Kdyby byla uzavřena na sjednocení, byla by na základě věty 6.10 a de Morganových vzorců uzavřena i na průnik, což není, jak víme z věty 6.9. □

7 Kontextové jazyky

V první kapitole byla uvedena Chomského hiearchie jazyků. U kontextových jazyků jsou uvedeny dvě definice s poznámkou, že jsou ekvivalentní. Gramatiky definované těmito dvěma definicemi se jistě nekryjí, ale generují stejné jazyky.

Definice 7.1

Gramatikám, jejichž všechna pravidla jsou tvaru $\alpha \to \beta$, kde $|\alpha| \leq |\beta|$, budeme říkat *monotónní.*

Gramatiky, jejichž všechna pravidla jsou tvaru $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$, budeme nazývat *kontextové.*

Z definice je zřejmě vidět, že všechny kontextové gramatiky jsou také monotónní. Následující věta uvádí, že naopak ke každé monotónní gramatice existuje kontextový ekvivalent.

Věta 7.2

Ke každé monotónní gramatice existuje s ní ekvivalentní kontextová gramatika.

Důkaz: Vezměme monotónní gramatiku G. Vytvoříme z ní gramatiku, jejíž pravidla budou buď obsahovat neprázdné neterminální řetězce na obou stranách, nebo jediný neterminál na levé straně a jediný terminál a případně ε na straně pravé. Takové gramatice se říká *separovaná.*

Vezměme tedy každé pravidlo $\alpha \to \beta$, které dané podmínky nesplňuje. Je-li $\beta \neq \epsilon$, pak výskyt terminálu x na obou stranách pravidla nahradíme novým neterminálem $\overline{x}$ a přidáme pravidla $\overline{x} \to x$ pro příslušné terminální symboly. Pokud $\beta = \varepsilon$, provedeme tutéž operaci s levou stranou α pravidla a ε na pravé straně nahradíme speciálním neterminálem $\overline{\varepsilon}$. Přidáme pak další pravidlo $\overline{\varepsilon} \to \varepsilon$. Tato gramatika je ekvivalentní původní, je separovaná a navíc jí zůstala zachována monotónnost.

Vezměme nyní každé pravidlo monotónní separované gramatiky $A_1 \cdots A_m \to B_1 \cdots B_n$ pro $m \leq n$, které nemá tvar požadovaný po kontextové gramatice. Odstraníme je tedy a nahradíme pravidly

$$\begin{aligned} A_1 A_2 \cdots A_m &\to C_1 A_2 \cdots A_m \\ C_1 A_2 A_3 \cdots A_m &\to C_1 C_2 A_3 \cdots A_m \end{aligned}$$

$$\begin{aligned}
&\vdots \\
C_1 \cdots C_{m-1} A_m &\to C_1 \cdots C_{m-1} C_m B_{m+1} \cdots B_n \\
C_1 \cdots C_{m-1} C_m B_{m+1} \cdots B_n &\to C_1 \cdots C_{m-1} B_m B_{m+1} \cdots B_n \\
&\vdots \\
C_1 B_2 \cdots B_n &\to B_1 B_2 \cdots B_n,
\end{aligned}$$

kde $C_1, \ldots C_m$ jsou nově přidané neterminály, v žádných jiných pravidlech se nevyskytující. Výsledná gramatika je kontextová při zachování generovaného jazyka. □

Poznámka 7.3

I pro kontextové jazyky byl nalezen výpočetní formalismus, který odpovídá právě jim. Jmenuje se *lineární omezený automat (LBA).*[5] Jazyky přijímané LBA jsou právě kontextové.

Kontextové jazyky jsou podtřídou třídy všech rekurzivních množin. Existuje totiž algoritmus, s pomocí kterého lze ověřit, zda dané slovo do jazyka generovaného nějakou kontextovou gramatikou patří či nikoli. U jazyků typu 0 to již není možné.

Věta 7.4

Existuje algoritmus, který pro každou monotónní gramatiku G a libovolné slovo w rozhoduje, zda $w \in L(G)$.

Důkaz: Nechť $G = (N, \Sigma, P, S)$ je monotónní gramatika. Pro libovolná dvě slova u, v lze po konečně mnoha krocích ověřit, zda $u \Rightarrow_G v$. Stačí totiž vyzkoušet všechny možné aplikace pravidel z P na u.

Pro libovolné slovo $w \in \Sigma^*$ existuje jen konečně mnoho slov $w_1, \ldots, w_n \in (N \cup \Sigma)^*$, která mají délku menší nebo rovnu slovu w: $|w_i| \leq |w|$. Protože G je monotónní, musí se každá derivace w v této gramatice skládat výhradně ze slov vybraných z $w_1, \ldots, w_n$. Z těchto slov lze utvořit jen konečně mnoho posloupností bez opakování prvků, a o každé z nich lze ověřit, zda je derivací slova w se začátkem v S. Na posloupnosti bez opakování se můžeme omezit proto, že v minimálních derivacích k opakování nedochází.

Stačí tedy prohledat všechny posloupnosti bez opakování vytvořené ze slov $w_1, \ldots, w_n$. Pokud některá z nich je derivací slova w, potom $w \in L(G)$, jinak $w \notin L(G)$. □

Věta 7.5

Existuje rekurzivní jazyk, který není kontextový. (Kontextové jazyky jsou vlastní podtřídou rekurzivních množin.)

Důkaz: Zkonstruujeme daný jazyk tzv. *Cantorovou diagonální metodou.* Princip konstrukce se opírá o fakt, že všech slov libovolné abecedy Σ^* je spočetně mnoho, stejně jako všech možných kontextových gramatik. Můžeme tedy uspořádat slova Σ^* do nekonečné posloupnosti $x_1, \ldots, x_i, \ldots$ a všechny kontextové gramatiky uspořádáme také do posloupnosti $G_1, \ldots, G_i, \ldots$ Pro každou gramatiku G_i existuje podle předchozí věty algoritmus, který rozhoduje o příslušnosti libovolného slova do jazyka této gramatiky. Definujeme nyní jazyk $L = \{x_i : x_i \notin L(G_i)\}$. Jde tedy o množinu takových slov x_i, která nejsou akceptována gramatikou G_i.

Můžeme jistě sestrojit algoritmus, který rozhodne, zda libovolné slovo w patří do jazyka L. Stačí totiž najít index i slova $w = x_i$ a podle předchozí věty zjistit, zda $x_i \in L(G_i)$. Avšak jazyk L není generován žádnou kontextovou gramatikou, protože od jazyka každé kontextové gramatiky G_i se liší v akceptování slova x_i. □

8 Turingovy stroje

Uvedeme nyní výpočetní formalismus zvaný *Turingův stroj.* Je to nejsilnější výpočetní mechanismus, jaký známe. A. Church např. tvrdí, že libovoný proces, který lze intuitivně nazvat algoritmem, se dá realizovat na Turingově stroji (tzv. *Churchova teze*). Turingův stroj má svoji základní podobu a dále několik modifikací.

[5] Nebyl na přednáškách detailně rozebírán a není ani součástí státnicových otázek.

8.1 Základní model TS

Turingův stroj je systém sestávající se z konečněstavové řídící jednotky, která může používat paměť ve formě *pásky* rozdělené na *políčka.* V každém políčku může být zapsán symbol z jisté konečné abecedy, nebo políčko může být prázdné. Páska je alespoň v jednom směru nekonečná. Turingův stroj se může v každém výpočetním kroku posunout po pásce o jedno políčko doprava nebo doleva a přitom měnit obsah políčka, na kterém se právě nachází, a vnitřní stav jednotky. Změny obsahu políček i pohyb po pásce závisí jednak na obsahu čteného políčka, jednak na vnitřním stavu řídící jednotky.

Definice 8.1
Turingův stroj (TS) je systém $T = (Q, \Sigma, \Gamma, B, \delta, q_0, F)$ s následujícím významem jednotlivých komponent:

Q — množina *stavů*

Σ — *vstupní abeceda*

Γ — *pásková abeceda*

$B \in \Gamma$ — interpretujeme jako *prázdný symbol,* tedy že na daném políčku pásky není žádný symbol; dále předpokládáme, že $\Sigma \subseteq \Gamma \setminus \{B\}$

δ — (parciální) *přechodová funkce*; $\delta\colon Q \times \Gamma \to Q \times (\Gamma \setminus \{B\}) \times \{L, R\}$

q_0 — *počáteční stav*

F — množina *koncových stavů*

Přechodovou funkci interpretujeme tak, že pokud $\delta(q, x) = (q', x', d)$, potom ve stavu q a se symbolem x na aktuálním políčku přejde Turingův stroj do stavu q', přepíše aktuální políčko symbolem x' a posune se doleva, pokud $d = L$, nebo doprava, pokud $d = R$.

Definice 8.2
Konfigurací Turingova stroje $T = (Q, \Sigma, \Gamma, B, \delta, q_0, F)$ budeme nazývat každou trojici (q, α, i) takovou, že $q \in Q$ je stav, $\alpha \in (\Gamma \setminus \{B\})^*$ je slovo na pásce a $i \in \mathbf{N}$ je pořadí symbolu ve slově α, které právě TS čte. Požadujeme, aby $1 \leq i \leq |\alpha| + 1$.

Definice 8.3
Na množině konfigurací Turingova stroje T definujeme relaci $\vdash$ *výpočetního kroku* takto:

$$\begin{aligned}
(q, A_1 \cdots A_n, i) &\vdash (p, A_1 \cdots A_{i-1} A A_{i+1} \cdots A_n, i+1) \quad \text{pokud} \quad \delta(q, A_i) = (p, A, R) \quad \text{pro } 1 \leq i \leq n \\
(q, A_1 \cdots A_n, i) &\vdash (p, A_1 \cdots A_{i-1} A A_{i+1} \cdots A_n, i-1) \quad \text{pokud} \quad \delta(q, A_i) = (p, A, L) \quad \text{pro } 2 \leq i \leq n \\
(q, A_1 \cdots A_n, n+1) &\vdash (p, A_1 \cdots A_n A, n+2) \quad \text{pokud} \quad \delta(q, B) = (p, A, R) \quad \text{pro } n \geq 1 \\
(q, A_1 \cdots A_n, n+1) &\vdash (p, A_1 \cdots A_n A, n) \quad \text{pokud} \quad \delta(q, B) = (p, A, L) \quad \text{pro } n \geq 2
\end{aligned}$$

Relace $\vdash^*$, resp. $\vdash^+$, jsou tranzitivním a reflexivní, resp. reflexivním, uzávěrem relace $\vdash$.

Počáteční konfigurací je každá konfigurace $(q_0, w, 1)$, kde $w \in \Sigma^*$ je vstupní slovo. *Koncovou konfigurací* je každá konfigurace (q, α, i), kde $q \in F$ je koncový stav.

Řekneme, že Turingův stroj T *přijímá* slovo w, jestliže $(q_0, w, 1) \vdash^* K$ pro nějakou koncovou konfiguraci K. Označujeme množinově

$$\text{přijímá}(T) = \{w\colon w \in \Sigma^*, (q_0, w, 1) \vdash^* (q, \alpha, i), q \in F\}.$$

Řekneme, že TS *zamítá* slovo w, pokud $(q_0, w, 1) \vdash^* K$ pro nějakou nekoncovou konfiguraci K a neexistuje konfigurace K' taková, že $K \vdash K'$. Množinový zápis:

$$\text{zamítá}(T) = \{w\colon w \in \Sigma^*, (q_0, w, 1) \vdash^* (q, \alpha, i), q \notin F, \neg\exists K'\colon (q, \alpha, i) \vdash K'\}$$

Řekneme, že TS *cyklí* na slově w, pokud ho nepřijímá ani nezamítá:

$$\text{cyklí}(T) = \Sigma^* \setminus (\text{přijímá}(T) \cup \text{zamítá}(T))$$

Jazyk rozpoznávaný (přijímaný) Turingovým strojem T je množina slov, které přijímá; značíme také $L(T) = \text{přijímá}(T)$. Pokud $\text{cyklí}(T) = \emptyset$, řekneme, že Turingův stroj T *rozhoduje jazyk* $L(T)$.

Věta 8.4 Turingova
Problém zastavení Turingova stroje je nerozhodnutelný.

Důkaz: Je třeba si uvědomit, že rozhodnutelnost v našem smyslu znamená existenci Turingova stroje řešícího daný problém.

Předpokládejme nejprve, že existuje Turingův stroj A, který umí rozhodovat, zda daný Turingův stroj se zastaví pro dané slovo. Je třeba zakódovat Turingovy stroje do nějaké podoby čitelné strojem A. Pro Turingův stroj M označíme tento kód $d(M)$.

Ze stroje A nyní zkonstruujeme Turingův stroj B tímto způsobem. Nechť vstupem stroje B je pouze zakódovaný Turingův stroj $d(M)$. Pak stroj B uvnitř simuluje práci stroje A tak, že mu pošle svůj vstup $d(M)$ zdvojený. Jestliže stroj A tento vstup akceptuje (to je právě tehdy, když Turingův stroj M necyklí pro vstup $d(M)$), nechť stroj B se zacyklí. Když naopak A zamítne tento vstup (tedy M cyklí pro vstup $d(M)$), nechť stroj B necyklí a skončí (nezáleží, zda přijetím nebo zamítnutím). Turingův stroj B lze jistě sestrojit za předpokladu existence stroje A.

Zkusme si nyní představit, jak zareaguje stroj B na vstup $d(B)$. Jestliže B necyklí pro vstup $d(B)$, pak A reaguje na vstup složený z dvojnásobného $d(B)$ přijetím. Potom se ale B zacyklí. Naopak, pokud B cyklí pro vstup $d(B)$, A zamítne vstup $d(B)d(B)$, a proto se B nezacyklí. Dostáváme tak spor. □

8.2 Některé modifikace základního modelu TS

Modifikace Turingova stroje se dají rozdělit na jeho rozšíření a zúžení. V prvním případě tedy dáváme stroji více možností a variant výpočtu, ve druhém jsou tyto možnosti omezovány.

Rozšíření Turingova stroje. Uvedeme Turingův stroj s oboustranně nekonečnou páskou, k-páskový Turingův stroj, Turingův stroj s dvourozměrnou páskou a nedeterministický Turingův stroj.

Definice 8.5
Turingův stroj s oboustraně nekonečnou páskou se liší od základního modelu tím, že poslední složka i konfigurací (q, α, i) může nabývat hodnot z oboru celých čísel. Přechodová funkce se této situaci přizpůsobí tak, že na levém kraji slova α se chová stejně jako na pravém kraji, tedy může se tímto směrem také rozšiřovat.

Věta 8.6
Libovolný výpočet každého Turingova stroje s oboustraně nekonečnou páskou lze simulovat na základním modelu Turingova stroje.

Důkaz: Za páskovou abecedu vezmeme místo Γ množinu $\Gamma' = \Gamma \times (\Gamma \cup \{\emptyset\})$, kde $\emptyset \notin \Gamma$. Z oboustraně nekonečné pásky

A_{-m}	A_{-m+1}	$\cdots$	A_{-1}	A_0	A_1	$\cdots$	A_{n-1}	A_n

sestavíme jednostraně nekonečnou pásku s abecedou Γ':

A_0	A_1	$\cdots$	$\cdots$	A_n
$\emptyset$	A_{-1}	$\cdots$	A_{-m}	B

□

Definice 8.7
k-páskový Turingův stroj je Turingův stroj, který je schopen číst k různých pásek na různých místech, tedy má celkem k tzv. *čtecích hlav*. Všechny tyto pásky jsou jednostraně nekonečné.

Věta 8.8
Libovolný výpočet k-páskového Turingova stroje lze simulovat základním modelem Turingova stroje.

Důkaz: Simulujeme tzv. širokou páskou

B'	$\cdots$	X	$\cdots$
A_{11}	$\cdots$	A_{1i}	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
B'	$\cdots$	$\cdots$	X
A_{k1}	$\cdots$	$\cdots$	A_{kj}

Speciální symboly B' a X mají ten význam, že na každé pásce v místě, na níž se nachází hlava, je symbol X a na ostatních místech této pásky je symbol B'. □

Poznámka 8.9

Existuje také rozšíření Turingova stroje na dvourozměrnou pásku. I tento model se dá simulovat základním modelem Turingova stroje.

Definice 8.10

Nedeterministický Turingův stroj se od základního modelu liší svou přechodovou funkcí. Její obor hodnot je množina všech podmnožin $Q \times (\Gamma \setminus \{B\}) \times \{L, R\}$.

Věta 8.11

Libovolný výpočet každého nedeterministického Turongova stroje se dá simulovat základním modelem Turingova stroje.

Důkaz: Použijeme tři pásky, které se podle předchozího dají simulovat základním modelem: (1) pro vstupní slovo, (2) pro prohledávání výpočetního stromu do šířky a (3) pracovní. □

Je tedy vidět, že kromě „syntaktického cukru" jsme žádným rozšířením Turingova stroje nedosáhli větších výpočetních možností.

Zúžení Turingova stroje. Existuje tzv. *konečný stroj se zásobníkem,* který se nemůže pohybovat po pásce libovolně jako Turingův stroj. Vezmeme-li však takový konečný stroj se dvěma zásobníky, lze jím simulovat Turingův stroj tak, že jeden zásobník použijeme na část pásky od hlavy včetně vpravo a druhý zásobník na část pásky od čtecí hlavy vlevo. Používá se také tzv. *konečný stroj s frontou,* u něhož však může dojít k zacyklení.

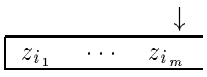
Definice 8.12

Počítadlo je stroj, který v každé chvíli nabývá hodnoty z množiny přirozených čísel a rozeznává symboly *inc* pro zvýšení hodnoty o 1 a *dec* pro snížení hodnoty o 1.

Lemma 8.13

Každý zásobník lze (realizovat) simulovat pomocí dvou počítadel.

Důkaz: Nechť Γ je zásobníková abeceda. Označíme $k = |\Gamma| + 1$ a očíslujeme všechny zásobníkové symboly takto: $\Gamma = \{z_1, z_2, \ldots, z_{k-1}\}$. Potom zásobník



s vrcholem na m-tém místě budeme reprezentovat číslem

$$j = k^0 i_m + k^1 i_{m-1} + \cdots + k^{m-1} i_1 .$$

Operací $j \bmod k$ zjistíme, co je na vrcholu zásobníku, operací $j \operatorname{div} k$ tento vrchol odstraníme a konečně operací $kj + r$ přidáme prvek z_r na vrchol zásobníku. Právě kvůli násobení je nutno přidat druhé počítadlo. □

Důsledek 8.14

Turingův stroj lze simulovat konečným strojem se čtyřmi počítadly.

Lemma 8.15

Čtyři počítadla lze simulovat s pomocí dvou počítadel.

Důkaz: Stav $\langle i,j,k,l\rangle$ čtyř počítadel převedeme na stav $2^i3^j5^k7^l$ jednoho počítadla. Kvůli operacím násobení a dělení je však potřeba dvou počítadel. □

Poznámka 8.16

Často je třeba kódovat konfigurace Turingova stroje do slov. Jestliže množiny stavů Q a páskové abecedy Γ jsou disjunktní, lze konfiguraci (q,α,i) kódovat do slova $\alpha_1 q\alpha_2$, kde $\alpha_1\alpha_2=\alpha$ a $|\alpha_1|=i-1$. Tedy první symbol slova α_2 je právě na pozici čtecí hlavy.

9 Turingovy stroje a jazyky typu 0

Ukážeme, že Turingovy stroje rozpoznávají právě jayzky typu 0.

Věta 9.1

Každý jazyk rozpoznatelný Turingovým strojem je typu 0.

Důkaz: Nechť $T=(Q,\Sigma,\Gamma,b,\delta,q_0,F)$ je Turingův stroj. Sestrojíme gramatiku $G=(N,\Sigma,P,S)$ takovou, že $L(G)=L(T)$.

K abecedě Γ sestrojíme nejprve abecedu dvojníků $\overline{\Gamma}=\{\overline{x}\colon x\in\Gamma\}$. Gramatiku G pak definujeme takto: $N=Q\cup\overline{\Gamma}\cup\{S,A,B,C\}$ pro nové neterminály S,A,B,C neobsažené v množině stavů Q ani páskové abecedě Γ Turingova stroje, a množinu pravidel sestrojíme takto:

$$\text{I}\quad\begin{cases} S \to Aq_0B\\ A \to x A\overline{x} & \text{pro všechna } x\in\Gamma\setminus\{b\}\\ A \to B\\ B \to \overline{b}B \mid \overline{b}\end{cases}$$

$$\text{II}\quad\begin{cases} \overline{x}q\overline{y} \to q'\overline{x'}\overline{y} & \text{jestliže } \delta(q,x)=(q',x',R)\\ \overline{x}q\overline{y} \to \overline{x'}\overline{y}q' & \text{jestliže } \delta(q,x)=(q',x',L)\end{cases}$$

$$\text{III}\quad\begin{cases} q \to C & \text{pro všechna } q\in F\\ Ca \to C & \text{pro všechna } a\in\overline{\Gamma}\\ aC \to C & \text{pro všechna } a\in\overline{\Gamma}\\ C \to \varepsilon\end{cases}$$

Levé strany pravidel II a III neobsahují žádný ze symbolů S,A,B a tyto symboly ani nemohou vzniknout aplikací pravidel těchto dvou skupin. Jestliže tedy v nějaké derivaci je použito nejprve pravidlo ze skupiny II nebo III před pravidlem ze skupiny I, může být toto pořadí obráceno, aniž by se změnilo derivované slovo. Budeme tedy dále předpokládat, že derivace podle skupiny I předcházejí derivacím podle II i III. Po první aplikaci pravidla ze skupiny III pak již není možné použít pravidlo ze skupiny I. Ukážeme nyní, že $L(G)=L(T)$.

$\subseteq$: Nechť $S\Rightarrow^* w$ je odvození slova w. Aplikací pravidel ze skupiny I vznikne slovo tvaru $w\overline{b}^i\overline{w}^Rq_0\overline{b}^j$, kde $\overline{w}^R=\overline{w_n}\cdots\overline{w_1}$, pokud $w=w_1\cdots w_n$. Pravidla typu II simulují výpočet Turingova stroje na slově w. Je však simulován jako zrcadlový obraz výpočtu na dvojnících původních symbolů. K pravidlům typu III se může přejít pouze v případě. když se v simulovaném výpočtu dostane stroj do koncového stavu, tedy pro $w\in L(T)$. V tom případě se ve slově objeví symbol C, celkem je větná forma ve tvaru $w\overline{u}C\overline{v}$. Symbol C zlikviduje postupně všechny symboly slov $\overline{u}$ i $\overline{v}$ a nakonec sám spáchá sebevraždu. Tak vznikne terminální řetězec w.

$\supseteq$: Pokud obráceně $w\in L(T)$, potom pravidly typu I lze derivovat slovo $w\overline{b}^i\overline{w}^R\overline{b}^j$, kde i a j jsou dostatečně velká čísla, aby na úseku pásky b^jwb^i proběhl celý výpočet stroje T nad w. Potom pravidla II nasimulují tento výpočet, opět zrcadlově obrácený na dvojnících symbolů. Jakmile se výpočet dostane do koncového stavu, pravidla III zruší všechny symboly až na w. □

Věta 9.2

Každý jazyk typu 0 je rozpoznatelný Turingovým strojem.

Důkaz: Detailní důkaz tohoto tvrzení je technicky náročný. Bude proto uvedena pouze hlavní idea, která je poměrně jednoduchá.

Pro danou gramatiku $G=(N,\Sigma,P,S)$ lze každou derivaci

$$S\Rightarrow_G w_1\Rightarrow_G w_2\Rightarrow_G\cdots\Rightarrow_G w_n=w\in\Sigma^*$$

zakódovat jako slovo $\#S\#w_1\#w_2\#\cdots\#w_n\#$. Každou derivaci podle G lze proto ztotožnit s jistým slovem v abecedě $N \cup \Sigma \cup \{\#\}$.

Je možné sestrojit Turingův stroj T_1 takový, že přijímá právě slova typu $\#u\#v\#$, kde $u \Rightarrow_G v$ je jeden krok odvození podle G. Tento stroj modifikujeme na stroj T_2 tak, že T_2 přijímá právě ta slova v abecedě $N \cup \Sigma \cup \{\#\}$, která jsou derivacemi podle G. Konečně sestrojíme Turingův stroj T, který, když dostane na pásce vstupní slovo w, generuje nalevo od něj postupně slova v abecedě $N \cup \Sigma \cup \{\#\}$, počínaje nejkratšími. Pokaždé, když vygeneruje nějaké slovo, zkontroluje jeho vnitřní část odpovídající stroji T_2, zda se jedná o derivaci podle G. □

Důsledek 9.3

Jazyk je typu 0 právě tehdy, když je rozpoznatelný Turingovým strojem.

10 Rozhodnutelnost v teorii jazyků

Příhodnější název pro tuto kapitolu by byl „Nerozhodnutelnost...“, protože tu budeme u většiny problémů dokazovat. Budeme si pomáhat redukcí tzv. Postova korespondenčního problému, který uvedeme nejprve.

10.1 Postův korespondenční problém

Definice 10.1

Postovým korespondenčním problémem (PKP) budeme nazývat každou dvojici $\langle A, B\rangle$, kde A a B jsou dvě konečné posloupnosti slov abecedy Σ se stejným množstvím prvků. Budeme značit $A = \{u_1, \ldots, u_n\}$ a $B = \{v_1, \ldots, v_n\}$.

Řekneme, že posloupnost r čísel $1 \leq i_1, \ldots, i_r \leq n$ je *řešením PKP* $\langle A, B\rangle$, pokud platí:

$$u_{i_1} u_{i_2} \cdots u_{i_r} = v_{i_1} v_{i_2} \cdots v_{i_r} \tag{15}$$

Příklad 10.2

Nechť abeceda $\Sigma = \{0, 1\}$ a mějme Postův korespondenční problém $\langle\{1, 10111, 10\}, \{111, 10, 0\}\rangle$. Ten má řešení 2113:

$$\begin{aligned} u_2u_1u_1u_3 &= \overbrace{10111}^{u_2}\overbrace{1}^{u_1}\overbrace{1}^{u_1}\overbrace{10}^{u_3} \\ v_2v_1v_1v_3 &= \underbrace{10}_{v_2}\underbrace{111}_{v_1}\underbrace{111}_{v_1}\underbrace{0}_{v_3} \end{aligned}$$

Příklad 10.3

Postův korespondenční problém $\langle\{10, 011, 101\}, \{101, 11, 011\}\rangle$ nemá řešení. Pokud by je totiž měl, muselo by začínat první dvojicí $10, 101$, aby se shodoval první symbol. Potom je nutno použít třetí dvojice, abychom dostali slova $10101, 101011$. Ze stejného důvodu jako v minulém kroku, musíme nyní použít třetí dvojici atd. Nikdy takto nedostaneme shodná slova.

Jsme tedy schopni v některých konkrétních případech rozhodnout, zdá má daný PKP řešení. Jak ovšem uvidíme, nelze napsat proceduru, která by obecně rozhodovala, zda má daný PKP vůbec řešení.

Definice 10.4

Řekneme, že PKP $\langle A, B\rangle$, kde $A = \{u_1, \ldots, u_n\}$ a $B = \{v_1, \ldots, v_n\}$, má *inicializované řešení,* pokud pro nějaké $1 \leq i_1, \ldots, i_s \leq n$ platí:

$$u_1 u_{i_1} \cdots u_{i_s} = v_1 v_{i_1} \cdots v_{i_s} \tag{16}$$

Inicializované řešení je tedy takové řešení $i_1, \ldots, i_r$, že $i_1 = 1$.

Příklad 10.5

PKP z příkladu 10.2 nemá inicializované řešení.

Lemma 10.6

Kdyby existoval algoritmus, který by pro libovolný PKP rozhodoval, zda má řešení, pak by také existoval algoritmus, který by pro libovolný PKP rozhodoval, zda má inicializované řešení.

Důkaz: Předpokládejme existenci algoritmu $\mathcal{A}$, který pro libovolný PKP rozhodne, zda má řešení.

Nechť $\langle A, B\rangle$ je nějaký PKP, $A = \{u_1, \ldots, u_k\}$ a $B = \{v_1, \ldots, v_k\}$. Definujeme nový PKP $\langle C, D\rangle$ takto: Nechť \$ a ϕ jsou nově přidané symboly. Zavedeme funkce h_L a h_R na slovech předpisem: $h_L(a) = \phi a$ a $h_R(a) = a\phi$ pro každý symbol a a na slova se obě funkce homomorfně rozšíří. Položme

$$\begin{array}{rlrll} x_1 &= \phi h_R(u_1) & y_1 &= h_L(v_1) & \\ x_{i+1} &= h_R(u_i) & y_{i+1} &= h_L(v_i) & \text{pro } 1 \le i \le k \\ x_{k+2} &= \$ & y_{k+2} &= \phi\$ & \end{array}$$

a pak $C = \{x_1, \ldots, x_{k+2}\}$ a $D = \{y_1, \ldots, y_{k+2}\}$.

Není těžké ověřit, že $\langle C, D\rangle$ má řešení právě tehdy, když $\langle A, B\rangle$ má inicializované řešení. Více k tomu viz následující příklad.

Za předpokladu existence algoritmu $\mathcal{A}$ by bylo možno pro libovolné $\langle A, B\rangle$ rozhodnout existenci inicializovaného řešení tak, že bychom nejprve provedli příslušnou transformaci na $\langle C, D\rangle$ a pomocí algoritmu $\mathcal{A}$ rozhodli, zda má $\langle C, D\rangle$ řešení. □

Příklad 10.7

Mějme PKP $\langle A, B\rangle$ zadaný takto:

	A	B
1	10111	10
2	10	0
3	1	111

Potom PKP $\langle A, B\rangle$ má inicializované řešení právě tehdy, když $\langle C, D\rangle$ má řešení, kde $\langle C, D\rangle$ je dán:

	C	D
1	$\phi 1\phi 0\phi 1\phi 1\phi 1\phi$	$\phi 1\phi 0$
2	$1\phi 0\phi 1\phi 1\phi 1\phi$	$\phi 1\phi 0$
3	$1\phi 0\phi$	$\phi 0$
4	1ϕ	$\phi 1\phi 1\phi 1$
5	\$	$\phi\$$

Jde právě o transformaci z důkazu předchozího lemmatu. Inicializovanému řešení 1332 $\langle A, B\rangle$ odpovídá řešení 14435 $\langle C, D\rangle$. Obecně, je-li $1, i_1, \ldots, i_s$ inicializované řešení $\langle A, B\rangle$, pak $1, (i_1 + 1), \ldots, (i_s + 1), n + 2$ je řešení $\langle C, D\rangle$. Toto řešení je sice také vždy inicializované, ale $\langle C, D\rangle$ jiné řešení nemá, proto $\langle A, B\rangle$ má inicializované řešení, právě když $\langle C, D\rangle$ má vůbec řešení.

Věta 10.8

Neexistuje algoritmus, který by pro libovolný PKP rozhodoval, zda má inicializované řešení.

Důkaz: Jeho celé znění je uvedeno v [Chytil], věta 8.20 na straně 153. Hlavní myšlenka spočívá v tom, že převedeme nějaký Turingův stroj na PKP. Tento PKP bude mít inicializované řešení právě tehdy, když příslušný stroj M bude přijímat slovo w. Kdyby existoval algoritmus, který by rozhodoval, zda PKP má inicializované řešení, bylo by jej možno v tomto případě použít na rozhodnutí o problému zastavení Turingova stroje. To však není možné, proto daný algoritmus neexistuje. □

Důsledek 10.9

Neexistuje algoritmus, který by pro libovolný PKP rozhodoval, zda má řešení.

10.2 Nerozhodnutelné problémy z teorie jazyků

Uvedeme nyní několik problémů z oblasti teorie jazyků, které jsou algoritmicky nerozhodnutelné. Využijeme k tomu převážně toho, že problém existence řešení PKP je nerozhodnutelný, a to s pomocí vhodných redukcí PKP na dané problémy z teorie jazyků.

Věta 10.10
Neexistuje algoritmus, který by pro libovolné dvě bezkontextové gramatiky G_1 a G_2 rozhodoval, zda $L(G_1) \cap L(G_2) = \emptyset$.

Důkaz: Předpokládejme, že takový algoritmus existuje. Ukážeme, že bychom s pomocí něho mohli rozhodovat nějaký PKP.

Nechť $\langle A, B\rangle$ je libovolný PKP, $A = u_1, \ldots, u_n$ a $B = v_1, \ldots, v_n$. Zvolme symboly $a_1, \ldots, a_n$, které se nevyskytují v žádném u_i ani v_i. Dále sestrojme bezkontextové gramatiky

$$\begin{aligned} G_A\colon &\quad S \to u_i S a_i \mid u_i a_i \quad 1 \le i \le n \\ G_B\colon &\quad S \to v_i S a_i \mid v_i a_i \quad 1 \le i \le n \end{aligned}$$

Potom zřejmě $\langle A, B\rangle$ má řešení, právě když $L(G_A) \cap L(G_B) \neq \emptyset$. □

V kapitole 5 je uveden poměrně jednoduchý algoritmus 5.1, který rozhoduje, zda jazyk generovaný bezkontextovou gramatikou je prázdný či nikoli. Z tohoto hlediska je překvapující tvrzení následující věty.

Věta 10.11
Neexistuje algoritmus, který by pro libovolnou bezkontextovou gramatiku $G = (N, \Sigma, P, S)$ rozhodoval, zda $L(G) = \Sigma^*$.

Důkaz: Nechť $\langle A, B\rangle$ je nějaký PKP, $A = u_1, \ldots, u_n$ a $B = v_1, \ldots, v_n$. Sestrojíme gramatiky G_A a G_B stejně, jako v důkazu předchozí věty, tedy

$$\begin{aligned} G_A\colon &\quad S \to u_i S a_i \mid u_i a_i \quad 1 \le i \le n \\ G_B\colon &\quad S \to v_i S a_i \mid v_i a_i \quad 1 \le i \le n. \end{aligned}$$

Protože jsme symboly $a_1, \ldots, a_n$ volili tak, že se nevyskytují v žádném u_i ani v_i, jsou jazyky $L(G_A)$ a $L(G_B)$ deterministické. Potom podle věty 6.10 jsou také $\overline{L(G_A)}$ a $\overline{L(G_B)}$ deterministické bezkontextové jazyky a podle věty 5.37 je jazyk $\overline{L(G_A)} \cup \overline{L(G_B)}$ bezkontextový.

Protože důkazy všech zmíněných vět jsou konstruktivní, lze od $\langle A, B\rangle$ přejít uniformním způsobem ke gramatice G_{AB} takové, že $L(G_{AB}) = \overline{L(G_A)} \cup \overline{L(G_B)}$. Podle důkazu předchozí věty má $\langle A, B\rangle$ řešení, právě když $L(G_A) \cap L(G_B) \neq \emptyset$. To je ekvivalentní s tím, že $\overline{L(G_A) \cap L(G_B)} = \Sigma^*$. Avšak $\overline{L(G_A) \cap L(G_B)} = \overline{L(G_A)} \cup \overline{L(G_B)} = L(G_{AB})$. Celkem platí, že $\langle A, B\rangle$ má řešení právě tehdy, když $L(G_{AB}) = \Sigma^*$. Kdybychom tedy daný fakt o gramatice G_{AB} uměli algoritmicky rozhodnout, umíme tak rozhodnout i PKP $\langle A, B\rangle$, což, jak víme, není možné. □

Poznámka 10.12
Předchozí věta může sloužit jako ověření, že bezkontextové jazyky nejsou uzavřeny na operaci doplňku. Kdyby totiž byly, vzali bychom bezkontextovou gramatiku G' generující doplněk jazyka $L(G)$, a tu bychom poslali na vstup algoritmu 5.1. Zřejmě by tak $L(G) = \Sigma^*$, právě když $L(G') = \emptyset$. Dostali bychom tímto způsobem algoritmus rozhodující, zda $L(G) = \Sigma^*$.

Věta 10.13
Neexistuje algoritmus, který by pro libovolnou bezkontextovou gramatiku G a regulární jazyk R rozhodoval, zda $L(G) = R$ ani zda $L(G) \supseteq R$.

Důkaz: Tvrzení je přímým důsledkem předcházející věty, protože množina Σ^* je regulární jazyk. □

Věta 10.14
Neexistuje algoritmus, který by pro libovolné bezkontextové gramatiky G_1 a G_2 rozhodoval, zda $L(G_1) = L(G_2)$ ani zda $L(G_1) \subseteq L(G_2)$.

Důkaz: Z existence takového algoritmu by vyplynula možnost rozhodovat, zda $L(G) = R$, resp. zda $L(G) \supseteq R$ z předchozí věty. □

Věta 10.15

Nechť G_1 a G_2 jsou libovolné bezkontextové gramatiky. Pro žádný z následujících problémů neexistuje rozhodující algoritmus.

1. Je $L(G_1) \cap L(G_2)$ bezkontextový jazyk?
2. Je $\overline{L(G_1)}$ bezkontextový jazyk?
3. Je $L(G_1)$ regulární jazyk?

Důkaz: K důkazu této věty jsou potřeba některé znalosti z teorie abstraktních tříd jazyků, proto jej vynecháme. Je však uveden v [Chytil], věta 8.26 na straně 157. □

Věta 10.16

Neexistuje algoritmus, který by pro každou bezkontextovou gramatiku rozhodoval, zda je víceznačná.

Důkaz: Budiž $\langle A, B \rangle$ libovolný PKP, $A = u_1, \ldots, u_n$ a $B = v_1, \ldots, v_n$. Definujeme bezkontextovou gramatiku G:

$$\begin{aligned} S &\rightarrow S_1 \mid S_2 \\ S_1 &\rightarrow u_i S_1 a_i \mid u_i a_i \qquad \text{pro } 1 \leq i \leq n \\ S_2 &\rightarrow v_i S_2 a_i \mid v_i a_i \qquad \text{pro } 1 \leq i \leq n \end{aligned}$$

Zřejmě $L(G) = L(G_A) \cup L(G_B)$ pro gramatiky G_A a G_B používané v důkazu věty 10.10. Je však zřejmé, že G je víceznačná, právě když $L(G_A) \cap L(G_B) \neq \emptyset$, tj. právě když $\langle A, B \rangle$ má řešení. □

Literatura

[HopUll] J. E. Hopcroft, J. D. Ullman: Formal languages and their relation to automata, *Adison-Wesley*, 1969 (slovenský překlad z roku 1978)

[Chytil] M. Chytil: Teorie automatů a formálních jazyků, *skriptum PřF UJEP Brno*, 1982