1 Úvod

Formální jazyk je množina posloupností symbolů v nějaké konečné abecedě. Posloupnost symbolů může vyjadřovat například aritmetický výraz, logickou formuli nebo program v nějakém programovacím jazyce. Ve všech těchto případech je pevně dána konečná neprázdná množina symbolů, ze kterých se uvažované posloupnosti vytváří a kterou budeme nazývat abeceda. Konečné posloupnosti symbolů z dané abecedy budeme nazývat slova. Obvykle jsou k vyjádření smysluplné informace použita jen některá ze slov v dané abecedě. Např. aritmetický výraz musí splňovat pravidla rozmístění závorek a symboly vyjadřující hodnoty a operace musí být uspořádány tak, aby vyjadřovaly postup výpočtu hodnoty výrazu. Podobné je to pro logické formule nebo formálně zapsaná matematická tvrzení. Zápis programu v programovacím jazyce musí splňovat syntaktická pravidla programovacího jazyka.

Množiny slov, která mají definován význam v některé z výše uvedených oblastí, jsou příklady formálních jazyků. Může to být množina všech správně vytvořených aritmetických výrazů, množina všech logických formulí určitého typu nebo množina všech syntakticky správných programů v daném programovacím jazyce.

Budeme se zabývat způsoby, jak jazyky popisovat pomocí konečného zápisu, který budeme nazývat konečná reprezentace nebo jen reprezentace. Budeme studovat reprezentaci jazyků pomocí gramatik a automatů. Množinu jazyků, které lze definovat určitým typem gramatiky nebo automatu budeme nazývat třída jazyků.

Formální zavedení pojmů jako abeceda a jazyk se poprvé objevilo v metamatematice v souvislosti s exaktní definicí logické dedukce a pojmu důkaz. V souvislosti s rozvojem počítačů pak byly formální jazyky studovány v souvislosti se syntaktickou analýzou programovacích jazyků a v souvislosti s počítačovou analýzou přirozeného jazyka.

Budeme studovat především dva typy jazyků: regulární jazyky a bezkontextové jazyky. Regulární jazyky jsou ty, které lze reprezentovat konečným automatem, regulárním výrazem nebo přechodovým diagramem. Bezkontextové jazyky jsou ty, které lze reprezentovat bezkontextovou gramatikou nebo nedeterministickým zásobníkovým automatem. Okrajově budou uvedeny také kontextové jazyky definované kontextovými gramatikami. Bude také popsán Turingův stroj, což je příklad automatu, který umožňuje realizovat libovolný algoritmus. Využijeme jej k důkazu algoritmické nerozhodnutelnosti některých problémů týkajících se bezkontextových jazyků a bude později využit jako základní výpočetní model v předmětu výpočetní složitost.

2 Základní pojmy a značení

Abecedou budeme rozumět libovolnou neprázdnou konečnou množinu symbolů. Abecedy budeme obvykle značit Σ, případně s indexy. Symboly budeme obvykle označovat počátečními písmeny z latinské abecedy, tj. $a, b, c, \ldots$, případně s indexy.

Definice 2.1 Slovem nad abecedou Σ budeme rozumět libovolnou konečnou posloupnost symbolů ze Σ včetně prázdné posloupnosti, kterou budeme nazývat prázdné slovo.

Slova budeme obvykle označovat písmeny $u, v, w, \ldots$, opět případně s indexy. Prázdné slovo budeme značit ε. Množinu všech slov v abecedě Σ budeme značit Σ^*.

Délkou slova w rozumíme počet jeho symbolů a značíme $|w|$. Délka prázdného slova je 0. Jestliže w je slovo v abecedě Σ a $a \in \Sigma$, pak počet výskytů a ve w budeme značit $|w|_a$.

Jsou-li u a v slova, pak zápisem uv rozumíme jejich zřetězení. Podslovem slova w budeme rozumět libovolný souvislý úsek ve slově w, tj. posloupnost po sobě jdoucích symbolů ve w. Slovo v je podslovem w právě tehdy, když existují slova x, y tak, že $w = xvy$. Slovo u je prefixem slova v, jestliže existuje slovo z tak, že $v = uz$.

Reverzí slova w rozumíme slovo w^R, které vznikne z w obrácením pořadí symbolů. Například, $(aab)^R = baa$.

Definice 2.2 Jazykem nad abecedou Σ budeme rozumět libovolnou podmnožinu Σ^*.

Jazyky budeme obvykle označovat velkými písmeny, např. L, K, případně s indexy. Jestliže L_1, L_2 jsou jazyky, pak jejich zřetězením nazveme jazyk

$$L_1 \cdot L_2 = \{uv; u \in L_1, v \in L_2\}.$$

Například, $\{a, ab\} \cdot \{a, ba\} = \{aa, aba, abba\}$. Snadno lze ověřit, že zřetězení jazyků je asociativní operace. Pro libovolný jazyk L platí $L \cdot \{\varepsilon\} = \{\varepsilon\} \cdot L = L$ a $L \cdot \emptyset = \emptyset \cdot L = \emptyset$. Opakovaným zřetězováním téhož jazyka dostaneme jeho mocniny.

$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^{i+1} &= L^i \cdot L \end{aligned}$$

Protože je zřetězení asociativní, platí pro libovolná $i \geq 0$ a $j \geq 0$

$$L^{i+j} = L^i \cdot L^j$$

Jestliže L je jazyk, pak iterací L nazveme jazyk

$$L^* = \bigcup_{i \geq 0} L^i$$

a pozitivní iterací L nazveme jazyk

$$L^+ = \bigcup_{i \geq 1} L^i$$

Pro libovolný jazyk L platí $L^* = L^+ \cup \{\varepsilon\}$. Pokud L obsahuje prázdné slovo, pak $L^* = L^+$.

Dříve zavedený význam Σ^* lze považovat za speciální případ iterace jazyka, pokud množinu symbolů Σ ztotožníme s množinou slov, z nichž každé je tvořeno jedním symbolem abecedy.

Reverzí jazyka L rozumíme jazyk $L^R = \{w^R \mid w \in L\}$.

2.1 Konečná reprezentace jazyků

Pokud je jazyk konečný, tj. obsahuje jen konečně mnoho slov, pak jej lze popsat výčtem všech slov, která do něj patří. Pokud je těchto slov příliš mnoho nebo pokud je jazyk nekonečný, nelze výčet použít a přijdou na řadu jiné způsoby popisu jazyka, ale budeme vždy požadovat, aby samotný popis jazyka byl konečný. Poznamenejme, že z hlediska teorie množin je všech jazyků nespočetně mnoho, zatímco konečných reprezentací je jen spočetně mnoho. Existují tedy jazyky, které nemají konečnou reprezentaci.

Reprezentace jazyků lze zhruba rozdělit do dvou typů: rozpoznávání (přijímání) jazyka a generování jazyka. Rozpoznávání jazyka znamená, že uvažujeme algoritmus, který přijme na vstupu libovolné slovo a jako výstup vydá rozhodnutí, zda slovo do jazyka patří nebo ne. Generování jazyka L znamená, že popíšeme strukturu, která umožňuje vytvářet slova, přičemž platí, že slovo může být daným způsobem vytvořeno právě tehdy, když patří do L.

Příkladem reprezentace typu rozpoznávání jazyka jsou konečné automaty a příkladem struktur pro generování jazyka jsou gramatiky. Mezi nedeterministickými algoritmy pro rozpoznávání jazyků a systémy pro generování jazyků není ostrá hranice. Například přechodové diagramy je možné použít jako nedeterministický algoritmus pro rozpoznávání jazyka i jako strukturu pro generování jazyka. Později si ukážeme, že ke každé bezkontextové gramatice G lze sestrojit nedeterministický zásobníkový automat, který rozpoznává jazyk, který gramatika G generuje, a navíc, výpočty automatu jsou ve vzájemně jednoznačné korespondenci s podmnožinou odvození v gramatice, která je postačující pro vygenerování libovolného slova generovaného jazyka.

2.2 Regulární výrazy

Regulární výrazy jsou jednoduchý způsob zápisu regulárních jazyků, který je často poměrně úsporný, a proto jej budeme používat i při výkladu jiných modelů. Regulární výraz je zápis vyjádření jazyka z konečných jazyků pomocí operací sjednocení, zřetězení a iterace.

Definice 2.3 Regulární výrazy nad abecedou Σ jsou definovány následujícími pravidly.

- Každý symbol $a \in \Sigma$ a symbol ε je regulární výraz.
- Jestliže α, β jsou regulární výrazy, pak také $(\alpha + \beta)$, $(\alpha \cdot \beta)$ a $(\alpha)^*$ jsou regulární výrazy.

Pro regulární výraz α budeme jako $L(\alpha)$ označovat jazyk, který je výrazem α reprezentován. Výrazy popsané v prvním pravidle reprezentují jazyk, který obsahuje právě jedno slovo, které je výrazem určeno. Přesněji, pro libovolný symbol $a \in \Sigma$ je $L(a) = \{a\}$ a $L(\varepsilon) = \{\varepsilon\}$. Dále, operace $+$ označuje množinové sjednocení, operace $\cdot$ označuje zřetězení a * označuje iteraci. Přesněji, pro libovolné regulární výrazy α, β je $L((\alpha + \beta)) = L(\alpha) \cup L(\beta)$, $L((\alpha \cdot \beta)) = L(\alpha) \cdot L(\beta)$ a $L((\alpha)^*) = L(\alpha)^*$.

Operaci $\cdot$ pro zřetězení je obvyklé v zápisu regulárního výrazu vypouštět. Pokud jsou vynechány závorky, pak operace $\cdot$ má vyšší precedenci než $+$.

3 Gramatiky

Gramatika je určena abecedou jazyka, který generuje, množinou pomocných symbolů, počátečním symbolem a odvozovacími pravidly. Abecedu generovaného jazyka budeme značit Σ a její symboly budeme nazývat terminály. Pomocné symboly budeme nazývat neterminály a jejich množinu budeme značit V. Počáteční symbol budeme značit S a je prvkem V. Odvozovací pravidlo je zápis tvaru $\alpha \to \beta$, kde $\alpha, \beta \in (\Sigma \cup V)^*$ a α obsahuje alespoň jeden neterminál.

Odvozovací pravidla dovolují z počátečního symbolu nebo z dříve odvozených slov v abecedě $\Sigma \cup V$ odvodit slova další. Použití pravidla $\alpha \to \beta$ znamená, že ze slova tvaru $\gamma_1\alpha\gamma_2$ odvodíme slovo $\gamma_1\beta\gamma_2$. Jestliže slovo δ_2 získáme ze slova δ_1 jedním krokem přepsání, píšeme $\delta_1 \Rightarrow \delta_2$. Pokud je δ_2 získáno libovolným konečným počtem přepsání z δ_1, píšeme $\delta_1 \stackrel{*}{\Rightarrow} \delta_2$. Toto značení zahrnuje i případ nulového počtu kroků přepsání, kdy je $\delta_1 = \delta_2$.

Definice 3.1 *Jazyk slov generovaných gramatikou G je při výše uvedeném značení*

$$L(G) = \{w \in \Sigma^*; S \stackrel{*}{\Rightarrow} w\}$$

Pro gramatiku G je tedy $L(G)$ jazyk všech slov ze Σ^*, která jsou odvoditelná z počátečního symbolu S pomocí odvozovacích pravidel G. Větnou formou budeme rozumět libovolné slovo v abecedě $\Sigma \cup V$, které je odvoditelné z počátečního symbolu S pomocí odvozovacích pravidel.

Odvozením slova $w \in L(G)$ budeme rozumět posloupnost větných forem, která začíná S, končí w a každý prvek posloupnosti kromě prvního je odvoditelný z předchozího pomocí některého pravidla gramatiky. Proces odvozování tedy končí, pokud je odvozeno slovo v abecedě Σ, proto se symboly této abecedy nazývají terminály. Pomocné symboly se nazývají neterminály, protože slovo, ve kterém se některý pomocný symbol vyskytne, není konečnou podobou generovaného slova.

Definice 3.2 *Gramatika se nazývá bezkontextová, pokud jsou levé strany všech pravidel tvořeny právě jedním neterminálem.*

Speciálním případem bezkontextové gramatiky je lineární gramatika, která navíc splňuje, že pravá strana každého pravidla obsahuje nejvýše jeden neterminál.

Pokud neklademe omezení na délku levé strany pravidel, ale pravá strana každého pravidla má délku alespoň takovou jako je délka levé strany stejného pravidla, tedy všechna pravidla splňují $|\alpha| \leq |\beta|$, nazývá se gramatika kontextová.

Ke každé kontextové gramatice existuje ekvivalentní gramatika, jejíž pravidla jsou tvaru $\gamma_1 A \gamma_2 \to \gamma_1 \beta \gamma_2$, kde A je neterminál a β je neprázdné slovo. Provedení tohoto pravidla lze interpretovat jako provedení pravidla $A \to \beta$ za podmínky, že symbol A se nachází v kontextu $\gamma_1 A \gamma_2$, což je důvod pro název kontextová gramatika. Vzhledem k tomu, že při odvozování v kontextové gramatice se v žádném kroku odvozená větná forma nezkracuje, mohou se v odvození vyskytnout pouze větné formy, jejichž délka je omezena délkou výsledného slova. Protože těchto větných forem je konečně mnoho, je odvoditelnost slova v kontextové gramatice algoritmicky rozhodnutelná úloha.

Pokud neklademe na gramatiky žádná omezení, lze pomocí gramatik popsat libovolný částečně rekurzivní jazyk.

3.1 Příklad kontextové gramatiky

Na Obrázku 1 je příklad množiny pravidel kontextové gramatiky s abecedami $\Sigma = \{a, b, c\}$, $V = \{S, A, B, C\}$ a s počátečním symbolem S.

Věta 3.3 *Jestliže G je gramatika s počátečním symbolem S, abecedami Σ, V uvedenými výše a s množinou pravidel z Obrázku 1, pak $L(G)$ je množina slov tvaru $a^i b^i c^i$ pro $i \geq 1$.*

Důkaz. Není obtížné nahlédnout, že každé slovo tvaru $a^i b^i c^i$ pro $i \geq 1$ je v G odvoditelné. Nejprve pomocí opakování prvního pravidla odvodíme větnou formu

$$S \to ABCS$$
$$S \to ABc$$
$$BA \to AB$$
$$CA \to AC$$
$$CB \to BC$$
$$Cc \to cc$$
$$Bc \to bc$$
$$Bb \to bb$$
$$Ab \to ab$$
$$Aa \to aa$$

Obrázek 1: Příklad množiny pravidel kontextové gramatiky.

$(ABC)^{i-1}S$ a pomocí druhého pravidla z ní odvodíme $(ABC)^{i-1}ABc$. Pomocí pravidel, která mění pořadí neterminálů, setřídíme neterminály ve větné formě tak, že získáme $A^i B^i C^{i-1} c$. V posledním kroku budeme opakovaně používat pravidla, jejichž levá strana obsahuje neterminál a terminál, tak, že v každém kroku se vždy poslední výskyt neterminálu nahradí odpovídajícím terminálem, tedy C je nahrazeno c, B je nahrazeno b a A je nahrazeno a. Například, pro $i = 2$ takto získáme odvození

$$S \Rightarrow ABCS \Rightarrow ABCABc \Rightarrow ABACBc \Rightarrow AABCBc \Rightarrow AABBCc$$

$$\Rightarrow AABBcc \Rightarrow AABbcc \Rightarrow AAbbcc \Rightarrow Aabbcc \Rightarrow aabbcc$$

Pro opačný směr dokážeme nejprve dvě jednodušší tvrzení. V prvním použijeme značení $|\alpha|_{\{a,A\}}$ pro počet symbolů a a A ve slově α, tedy $|\alpha|_{\{a,A\}} = |\alpha|_a + |\alpha|_A$, a analogicky pro $\{b, B\}$ a $\{c, C\}$.

Lemma 3.4 *Jestliže $S \stackrel{*}{\Rightarrow} \alpha$, pak platí*

$$|\alpha|_{\{a,A\}} = |\alpha|_{\{b,B\}} = |\alpha|_{\{c,C\}}$$

Důkaz. Indukcí podle délky odvození. Pro nulový počet kroků odvození je $\alpha = S$ a tvrzení lemmatu je splněno, protože všechny počty symbolů v rovnosti jsou nulové. Pak je potřeba ověřit rozborem případů. že použití libovolného pravidla G na větnou formu splňující tvrzení lemmatu vede opět na větnou formu splňující tvrzení lemmatu. QED.

V následujícím lemmatu budeme k popisu jazyků používat regulární výrazy a pro jednoduchost budeme regulární výraz ztotožňovat s jazykem, který generuje. Budeme tedy psát například $u \in (a + b + c)^*$ místo $u \in L((a + b + c)^*)$.

Lemma 3.5 *Jestliže $S \stackrel{*}{\Rightarrow} \alpha$, pak platí některá z podmínek*

$$\begin{aligned} \alpha &\in (A+B+C)^*S && (1)\\ \alpha &\in (A+B+C)^*c^+ && (2)\\ \alpha &\in (A+B+C)^*b^+c^+ && (3)\\ \alpha &\in (A+B+C)^*a^+b^+c^+ && (4) \end{aligned}$$

Důkaz. Pokud $\alpha = S$, je splněna podmínka (1). Dále ukážeme, že všechna pravidla z Obrázku 1 zachovávají množinu větných forem, které splňují alespoň jednu z podmínek v tvrzení lemmatu.

- Pravidlo $S \to ABCS$ lze použít pouze na slova α splňující (1) a vede opět na slovo splňující (1).
- Pravidlo $S \to ABc$ lze použít pouze na slova α splňující (1) a vede na slovo splňující (2).
- Pravidla $BA \to AB$, $CA \to AC$, $CB \to BC$ lze použít na slova α splňující kteroukoli z podmínek v tvrzení lemmatu a vedou na slovo splňující stejnou podmínku jako slovo výchozí.
- Pravidla $Cc \to cc$, $Bc \to bc$ lze použít pouze na slova α splňující (2) a vedou na slovo splňující (2) nebo (3).
- Pravidla $Bb \to bb$, $Ab \to ab$ lze použít pouze na slova α splňující (3) a vedou na slovo splňující (3) nebo (4).
- Pravidlo $Aa \to aa$ lze použít pouze na slova α splňující (4) a vede na slovo splňující (4).

Všechny větné formy gramatiky G tedy splňují některou z podmínek v tvrzení lemmatu. QED.

Jestliže $w \in \Sigma^*$ a platí $S \stackrel{*}{\Rightarrow} w$, pak z Lemmatu 3.5 plyne, že $w \in a^*b^*c^+$. Pro slovo tohoto tvaru pak z Lemmatu 3.4 plyne, že $w = a^ib^ic^i$ pro $i \geq 1$. QED.

3.2 Příklady bezkontextových gramatik

Slovo w nazveme symetrické (palindrom), pokud je rovno své reverzi, tedy $w = w^R$. Uvažme lineární bezkontextovou gramatiku, jejíž množina neterminálů obsahuje pouze počáteční symbol S, množina terminálů je $\{a, b\}$ a pravidla jsou na Obrázku 2.

Lemma 3.6 *Gramatika z Obrázku 2 generuje právě všechna symetrická slova sudé délky v abecedě $\{a, b\}$.*

$$\begin{aligned} S &\to aSa\\ S &\to bSb\\ S &\to \varepsilon \end{aligned}$$

Obrázek 2: Bezkontextová gramatika pro symetrická slova sudé délky.

Důkaz. Indukcí podle délky slova ukážeme, že pro každé symetrické slovo w sudé délky v abecedě $\{a, b\}$ platí $S \stackrel{*}{\Rightarrow} w$. Pro prázdné slovo tvrzení platí. Pokud je w neprázdné, má některý z tvarů aua nebo bub, kde u je kratší symetrické slovo sudé délky. Z indukčního předpokladu tedy plyne $S \stackrel{*}{\Rightarrow} u$, a tedy platí $S \Rightarrow aSa \stackrel{*}{\Rightarrow} aua$ a také $S \Rightarrow aSa \stackrel{*}{\Rightarrow} bub$.

Pro opačnou implikaci stačí ověřit indukcí podle délky odvození, že všechny větné formy v uvedené gramatice jsou symetrická slova a obsahují sudý počet terminálních symbolů. QED.

Jako další příklad uvedeme gramatiku pro Dyckův jazyk. Budeme uvažovat posloupnosti závorek, ve kterých tvoří levé a pravé závorky vzájemně si odpovídající dvojice tak, jako v aritmetických výrazech. Příklady takových slov jsou (), (()), ()(), (()()). Jazyk slov tohoto tvaru se nazývá Dyckův jazyk a slova z tohoto jazyka lze generovat gramatikou z Obrázku 3. Počet slov délky $2n$ v tomto jazyce je roven Catalanovu číslu

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

Důkaz tohoto faktu lze najít například na Wikipedii
`http://en.wikipedia.org/wiki/Dyck_language`,
`http://en.wikipedia.org/wiki/Catalan_number`.

$$\begin{aligned} S &\to SS\\ S &\to (S)\\ S &\to \varepsilon \end{aligned}$$

Obrázek 3: Gramatika pro Dyckův jazyk.

Lemma 3.7 *Gramatika z Obrázku 3 s počátečním symbolem S generuje právě všechna slova z Dyckova jazyka v abecedě $\{(,)\}$.*

Důkaz. Nechť $\Sigma = \{(,)\}$. Gramatiku z Obrázku 3 označme jako G a Dyckův jazyk v abecedě Σ označme D. Pro libovolné slovo $w \in \Sigma^*$ označme $h(w) = |w|_(- |w|_)$, tedy $h(w)$ je rozdíl počtu levých a pravých závorek ve w, tedy hloubka vnoření závorek na konci slova w. Jestliže $w = x_1x_2\ldots x_n$, kde $x_1, \ldots, x_n \in \Sigma$, a $0 \leq i \leq |w| = n$, označme $h_i(w) = h(x_1x_2\ldots x_i)$. Pro každé slovo je $h_0(w) = 0$.

$$
\begin{array}{l}
S \to T \\
T \to (T \vee T) \mid (F \vee T) \mid (T \vee F) \\
T \to (T \wedge T) \mid (\neg F) \\
T \to 1 \\
F \to (F \wedge F) \mid (T \wedge F) \mid (F \wedge T) \\
F \to (F \vee F) \mid (\neg T) \\
F \to 0
\end{array}
$$

Obrázek 4: Gramatika pro konstantní výrokové formule s hodnotou 1.

Každé slovo $w \in D$ obsahuje stejný počet levých a pravých závorek a tedy pro něj platí $h_{|w|}(w) = h(w) = 0$. Dokážeme, že pro každé slovo $w \in \Sigma^*$ jsou následující tvrzení ekvivalentní

(1) $w \in L(G)$,

(2) $w \in D$,

(3) $h(w) = 0$ a pro každé $0 \leq i \leq |w|$ je $h_i(w) \geq 0$.

Implikaci (1) $\Rightarrow$ (2) lze dokázat následovně. Jestliže $w \in L(G)$, uvažme libovolné odvození $S = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \Rightarrow \alpha_n = w$. Indukcí podle $i \in \{0, \ldots, n\}$ dokážeme, že ve větné formě α_i lze závorky (tedy terminální symboly) vzájemně přiřadit tak, že se dvojice vzájemně přiřazených závorek nekříží. Počáteční větná forma $\alpha_0 = S$ toto tvrzení splňuje triviálně. Nechť $1 \leq i \leq n$ a α_{i-1} toto tvrzení splňuje. Pokud větná forma α_i vznikne z α_{i-1} pomocí pravidla $S \to SS$ nebo $S \to \varepsilon$, vzájemné přiřazení závorek zachováme. Pokud α_i vznikne pomocí pravidla $S \to (S)$, přiřadíme navzájem nově přidané závorky a přiřazení ostatních závorek zachováme. V obou případech vznikne větná forma, která tvrzení splňuje.

Implikaci (2) $\Rightarrow$ (3) lze dokázat následovně. Nechť $w \in D$. Jak již bylo zmíněno výše, platí $h(w) = 0$. Jestliže $u = x_1 x_2 \ldots x_i$ je libovolný prefix w, pak každé pravé závorce v u lze přiřadit jinou levou závorku v u, tedy $h_i(w) = h(u) \geq 0$.

Implikaci (3) $\Rightarrow$ (1) dokážeme indukcí podle délky slova w. Prázdné slovo $w = \varepsilon$ splňuje (3) i (1). Nechť w je neprázdné a všechna slova Σ^* délky menší než $|w|$ dokazovanou implikaci splňují. Pokud existuje $1 \leq i \leq |w| - 1$, pro které je $h_i(w) = 0$, pak označme jako u prefix w délky i a jako v zbytek slova w, jehož délka je $|w| - i$. Platí tedy $w = uv$, $|u| < |w|$, $|v| < |w|$, $h(u) = h_i(w) = 0$. Protože $h(u) + h(v) = h(w)$, platí také $h(v) = h(w) - h(u) = 0$. Navíc, pro každé $0 \leq j \leq |u|$ platí $h_j(u) = h_j(w) \geq 0$. Dále, pro každé $0 \leq j \leq |v|$ platí $h(u) + h_j(v) = h_{|u|+j}(w) \geq 0$. Protože $h(u) = 0$, platí také $h_j(v) \geq 0$. Slova u, v tedy splňují (3). Z indukčního předpokladu plyne, že existují odvození $S \stackrel{*}{\Rightarrow} u$ a $S \stackrel{*}{\Rightarrow} v$. S využitím těchto odvození lze sestrojit odvození $S \Rightarrow SS \stackrel{*}{\Rightarrow} uS \stackrel{*}{\Rightarrow} uv = w$. QED.

Pravidla se stejnou levou stranou lze při zápisu bezkontextové gramatiky spojit do jednoho řádku, v němž jsou jednotlivé pravé strany odděleny znakem |. Gramatiku pro symetrická slova sudé délky tedy lze zapsat také ve tvaru

$$S \to aSa \mid bSb \mid \varepsilon \ .$$

Toto zkrácené značení je použito také v Obrázku 4, kde je uvedena gramatika s neterminály S, T, F a terminály $0, 1, \vee, \wedge, \neg, (,)$.

Lemma 3.8 *Gramatika z Obrázku 4 s počátečním symbolem S generuje právě všechny plně uzávorkované konstantní booleovské výrazy s hodnotou* 1.

Důkaz. Pro $a \in \{0, 1\}$ nechť L_a je množina všech plně uzávorkovaných konstantních booleovských výrazů s výše uvedenými spojkami. Nejprve budeme uvažovat odvození $w \in \Sigma^*$ z neterminálů T a F a indukcí podle k dokážeme, že pro každé $k \geq 1$ platí následující implikace

- jestliže $T \stackrel{*}{\Rightarrow} w$ v nejvýše k krocích, pak $w \in L_1$
- jestliže $F \stackrel{*}{\Rightarrow} w$ v nejvýše k krocích, pak $w \in L_0$

Pokud $k = 1$, je odvození v předpokladu první implikace použitím pravidla $T \to 1$ a implikace tedy platí. Podobně, odvození v předpokladu druhé implikace je $F \to 0$ a implikace také platí. Pokud odvození w z T nebo F obsahuje více kroků a pro všechna kratší odvození platí indukční předpoklad, pak ověříme rozborem případů podle výchozího neterminálu a prvního použitého pravidla, že i celé odvození splňuje dokazované implikace. Při tomto postupu využijeme, že první pravidlo gramatiky, které může být v tomto případě použito, zaručuje buď $w \in L_0$ nebo $w \in L_1$ podle tabulky odpovídající spojky $\vee$, $\wedge$ nebo $\neg$. Obě implikace tedy platí pro libovolné $k \geq 1$.

Pro opačný směr dokážeme následující implikace indukcí podle počtu spojek $n \geq 0$ použitých ve výrazu

- jestliže $w \in L_1$ a w obsahuje n spojek, pak $T \stackrel{*}{\Rightarrow} w$
- jestliže $w \in L_0$ a w obsahuje n spojek, pak $F \stackrel{*}{\Rightarrow} w$

Pokud je $n = 0$, je $w = 0$ nebo $w = 1$ a implikace platí. Předpokládejme, že $n \geq 1$ a pro výrazy s méně než n spojkami implikace platí. Slovo w má některý z tvarů $(u_1 \vee u_2)$, $(u_1 \wedge u_2)$, nebo $(\neg u_1)$ pro vhodné výrazy $u_1, u_2 \in L_0 \cup L_1$. Podle indukčního předpokladu lze tyto výrazy odvodit z T nebo F podle jejich hodnoty a tedy i w lze odvodit z T nebo F v závislosti na jeho hodnotě.

Celkem tedy platí

- $T \stackrel{*}{\Rightarrow} w$ právě tehdy, když $w \in L_1$
- $F \stackrel{*}{\Rightarrow} w$ právě tehdy, když $w \in L_0$

Protože z počátečního symbolu S je odvoditelná stejná množina slov jako z T, je tvrzení věty dokázáno. QED.

3.3 Levá derivace, levá větná forma

Odvození v bezkontextové gramatice, ve kterém se v každém kroku nahrazuje nejlevější neterminál, se nazývá levá derivace. Větná forma, kterou lze odvodit levou derivací, se nazývá levá větná forma.

Lemma 3.9 *Jestliže G je bezkontextová gramatika, pak každé slovo z $L(G)$ lze odvodit levou derivací.*

Důkaz. Nechť Σ je terminální abeceda, V je neterminální abeceda a S je počáteční symbol G. Nechť $w \in L(G)$ a uvažme libovolnou derivaci $S \stackrel{*}{\Rightarrow} w$. Nechť $uA\alpha$, $u \in \Sigma^*$, $A \in V$, $\alpha \in (\Sigma \cup V)^*$ je první větná forma v této derivaci, ve které není v dalším kroku nahrazován nejlevější neterminál. Derivaci $S \stackrel{*}{\Rightarrow} uA\alpha$, která tvoří počáteční úsek původní derivace, označme jako d_0. Protože je gramatika bezkontextová, lze pokračování původní derivace od větné formy $uA\alpha$ rozdělit na kroky, které rozvíjí větnou formu uA a kroky, které rozvíjí větnou formu α. Tímto způsobem získáme derivaci $uA \stackrel{*}{\Rightarrow} v_1 \in \Sigma^*$, kterou označíme d_1, a derivaci $\alpha \stackrel{*}{\Rightarrow} v_2 \in \Sigma^*$, kterou označíme d_2, tak, že platí $v_1 v_2 = w$. Derivace d_0, d_1 a d_2 lze spojit tak, že vytvoří derivaci $S \stackrel{*}{\Rightarrow} uA\alpha \stackrel{*}{\Rightarrow} v_1\alpha \stackrel{*}{\Rightarrow} v_1 v_2 = w$, jejíž délka je stejná jako délka původní derivace, a počáteční úsek této derivace, kde jsou nahrazovány vždy nejlevější neterminály, je alespoň o jeden krok delší než v původní derivaci. Opakováním tohoto postupu získáme derivaci slova w, která je levou derivací. QED.

4 Automaty

Pojem automatu obecně nebudeme definovat, ale ukážeme si některé typy automatů, které budou využity v dalším výkladu. Automat se obvykle skládá z řídící jednotky, která nabývá konečně mnoha stavů, a z paměťových zařízení, která jsou řízena konečným počtem možných instrukcí. Množina stavů obsahuje alespoň stavy $\{q_0, q_+, q_-\}$, které se v uvedeném pořadí nazývají počáteční stav, přijímající stav a zamítající stav. Paměťová zařízení mohou být například

- Páska tvořená jednostranně nekonečnou (směrem vpravo) posloupností polí, z nichž každé může obsahovat jeden symbol z abecedy, kterou budeme obvykle značit Γ, a čtecí hlavou, která je v každém kroku výpočtu umístěna na některém poli pásky. Předpokládáme, že Γ obsahuje symbol $\square$ pro prázdné pole pásky. Instrukce pro pásku jsou: zápis symbolu, přečtení symbolu a přesun čtecí hlavy o jedno pole vlevo nebo vpravo. Množina instrukcí pro pásku může být ve speciálních případech omezena tak, že páska umožňuje pohyb hlavy pouze jedním směrem nebo může být určena pouze pro čtení symbolů a nikoli zápis.

- Zásobník, který je znám také jako paměť typu LIFO (last in first out), viz `https://en.wikipedia.org/wiki/Stack_(abstract_data_type)`. Instrukce pro zásobník jsou: čtení symbolu na vrcholu zásobníku, odebrání jednoho symbolu z vrcholu zásobníku a vložení slova na vrchol zásobníku po jednotlivých symbolech.

- Čítač, který obsahuje přirozené číslo, a je řízen instrukcemi přičíst 1, odečíst 1, pokud je číslo kladné, a test na rovnost 0.

Na začátku výpočtu automatu je řídící jednotka v počátečním stavu q_0 a vstupní slovo je zapsáno na pásce, která se nazývá vstupní. Pásky, jejichž čtecí hlava se může pohybovat oběma směry, mají v prvním poli zapsán znak $\triangleright$ označující začátek pásky a na začátku výpočtu je na těchto páskách čtecí hlava na druhém poli, tedy za symbolem $\triangleright$. Pásky, jejichž čtecí hlava se pohybuje pouze vpravo, symbol $\triangleright$ v prvním poli neobsahují a na začátku výpočtu je čtecí hlava na prvním poli. Podle typu automatu je vstupní slovo zapsáno od prvního pole vstupní pásky (toto platí pro konečný automat a zásobníkový automat) nebo je první symbol na pásce $\triangleright$ a vstupní slovo je zapsáno od druhého pole pásky (toto platí pro Turingův stroj). Všechna ostatní pole vstupní pásky jsou prázdná, tedy obsahují symbol $\square$ pro prázdné pole pásky. Pracovní pásky obvykle umožňují pohyb čtecí hlavy oběma směry, a tedy na začátku výpočtu obsahují v prvním poli symbol $\triangleright$, a všechna ostatní pole obsahují symbol $\square$. Zásobníky obsahují na začátku výpočtu slovo $S\square$, kde S je počáteční symbol zásobníku, a čítače mají hodnotu 0. Symbol $\square$ nelze ze zásobníku odebrat a pokud je tento symbol na vrcholu zásobníku, znamená to, že je zásobník prázdný.

Vstupní páska může být pouze pro čtení nebo může připouštět zápis symbolů. Pracovní pásky umožňují čtení i zápis symbolů. Způsob využití vstupní pásky závisí na typu složitosti, kterou pro daný TS měříme.

- Vstupní páska, která připouští zápis symbolů a je využívána také jako pracovní, se používá, pokud měříme čas výpočtu TS nebo prostor potřebný k výpočtu, který je alespoň tak velký jako délka vstupního slova.

- Vstupní páska, která je určena pouze pro čtení, se používá, pokud měříme prostor potřebný k výpočtu, který je menší než délka vstupního slova.

Výpočet deterministického automatu je po výše popsané inicializaci určen přechodovou funkcí. Argumenty této funkce jsou stav řídící jednotky, symboly čtené na páskách nebo na vrcholech zásobníků, případně test na 0 u čítačů. Hodnotou přechodové funkce pak je akce, která má být v daném případě provedena. Akcí se rozumí seznam instrukcí, které mají být provedeny na paměťových zařízeních, a stav, do kterého přejde řídící jednotka po jejich provedení. Výpočet končí, pokud řídící jednotka přejde do přijímajícího nebo zamítajícího stavu. Slovo je přijato deterministickým automatem podle toho, v kterém z těchto stavů výpočet skončil.

Pokud je automat nedeterministický, je výpočet popsán přechodovou relací, která každé kombinaci vstupů určí množinu možných akcí, která může být prázdná, jednoprvková nebo víceprvková. Každá z možných akcí je stejně jako v deterministickém případě určena seznamem instrukcí, které mají být provedeny na paměťových zařízeních, a stavem, do kterého řídící jednotka přejde po jejich provedení. Nedeterministický automat může mít pro tentýž vstup více možných výpočtů. Ukončení výpočtu a rozhodnutí o přijetí slova daným výpočtem je stejné jako u deterministického automatu. Přijetí slova nedeterministickým automatem je definováno tak, že slovo je přijato, pokud pro dané slovo existuje přijímající výpočet a je zamítnuto v opačném případě. Protože možných výpočtů nedeterministického automatu může být exponenciálně mnoho v délce výpočtu, nedeterministický automat typicky nereprezentuje efektivní postup rozhodování o náležení slova do jazyka, ale může být vhodný jako abstraktní popis jazyka.

4.1 Turingův stroj

Budeme nejprve uvažovat Turingův stroj (TS) s jednou páskou, která je vstupní a umožňuje čtení i zápis symbolů a pohyb hlavy oběma směry. Množinu stavů řídící jednotky budeme značit Q a budeme předpokládat, že obsahuje počáteční stav q_0, přijímající stav q_+ a zamítající stav q_-. Abecedu symbolů na pásce budeme značit Γ a budeme předpokládat, že obsahuje symbol pro začátek pásky $\triangleright$ a symbol pro prázdné pole pásky $\square$. Vstupní abecedu, tedy abecedu vstupních slov, budeme značit $\Sigma \subseteq \Gamma \setminus \{\triangleright, \square\}$. Kromě symbolů ze $\Sigma \cup \{\triangleright, \square\}$ může abeceda Γ obsahovat libovolný konečný počet pomocných symbolů, které jsou využívány při výpočtu. Předpokládáme, že Q a Γ jsou disjunktní. Výpočet TS je řízen přechodovou funkcí, což je funkce tvaru

$$f : (Q \setminus \{q_+, q_-\}) \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\}$$

a její význam je popsán níže.

Na začátku výpočtu je obsah pásky tvořen v prvním poli pásky symbolem $\triangleright$, pak následuje vstupní slovo $w \in \Sigma^*$ rozložené po symbolech do jednotlivých polí pásky a všechna zbylá pole obsahují symbol $\square$. Čtecí hlava je na prvním symbolu vstupního slova a řídící jednotka je ve stavu q_0.

Výpočet probíhá následovně. Jestliže je řídící jednotka ve stavu q, čtený symbol na pásce je a a $f(q, a) = (q', a', s)$, pak je provedena následující akce. Stav řídící jednotky se změní na q', čtený symbol je přepsán na a' a pokud pozici hlavy označíme i, přesune se hlava na pozici $i + s$. Výpočet končí, pokud řídící jednotka přejde do stavu q_+ nebo q_-. Vstupní slovo je přijato, pokud výpočet skončil ve stavu q_+, a zamítnuto, pokud skončil v q_-.

Výpočet TS lze simulovat pomocí booleovských obvodů, výrokových formulí ve tvaru CNF, případně kvantifikovaných výrokových formulí (QBF). Pro konstrukci těchto simulací je potřeba zvolit způsob popisu aktuálního stavu výpočtu, který budeme nazývat konfigurace a který určuje stav řídící jednotky, obsahy pásek a polohu čtecích hlav na páskách. Pro jednopáskový TS lze konfiguraci popsat slovem pevné délky, pokud je předem znám odhad prostoru, ve kterém výpočet probíhá. Tento způsob využijeme například pro simulaci TS pomocí booleovských obvodů a některých typů formulí. Pokud není prostor omezen a může během výpočtu narůstat, budeme konfiguraci zapisovat ve tvaru slova $uqv\#$, kde $uv \in \Gamma^*$, $q \in Q$ a symbol $\#$ je přidaný symbol, který není prvkem $\Gamma \cup Q$. Slovo uv je tvořeno obsahem všech neprázdných polí pásky a všech polí s prázdným symbolem, která TS navštívil při některém z předchozích kroků a do kterých zapsal prázdný symbol podle přechodové funkce. Symbol q určuje stav řídící jednotky a umístění tohoto symbolu určuje polohu čtecí hlavy tak, že je čten první symbol slova v, pokud je toto slovo neprázdné, nebo je čten první prázdný symbol $\square$ na pásce za slovem u, pokud je v prázdné slovo. Slovo u je prázdné, pokud je čtecí hlava na nejlevějším poli pásky.

Budeme uvažovat také vícepáskový TS, protože je v některých případech efektivnější. Například rozpoznávání symetrických slov má na jednopáskovém TS časovou složitost $\Theta(n^2)$ a na dvoupáskovém $\Theta(n)$, kde n je délka vstupního slova. Pro jednoduchost budeme předpokládat, že abeceda symbolů na všech páskách je Γ. Přechodová funkce pro TS s k páskami je pak tvaru

$$f : (Q \setminus \{q_+, q_-\}) \times \Gamma^k \to Q \times \Gamma^k \times \{-1, 0, 1\}^k$$

a pokud je $f(q, x_1, ..., x_k) = (q', x'_1, ..., x'_k, s_1, ..., s_k)$, pak v případě, že je řídící jednotka ve stavu q a na páskách jsou čteny symboly $x_1, ..., x_k$, pak se stav řídící jednotky změní na q', symboly na páskách jsou přepsány na symboly $x'_1, ..., x'_k$ a pokud jsou čtecí hlavy na polích s indexy $i_1, ..., i_k$, pak se čtecí hlavy posunou na pole s indexy $i_1 + s_1, ..., i_k + s_k$.

Pro úvahy o jazycích, které lze rozpoznávat s prostorem $O(\log n)$, kde n je délka vstupního slova, budeme používat TS se vstupní páskou pouze pro čtení, která umožňuje pohyb hlavy oběma směry, a s k páskami, kde $k \geq 1$, tedy s $k-1$ pracovními páskami. Pro jednoduchost budeme předpokládat, že všechny pásky mají stejnou abecedu, i když na vstupní pásce budou využity pouze symboly z množiny $\Sigma \cup \{\triangleright, \square\}$. Přechodová funkce je v tomto případě tvaru

$$f : (Q \setminus \{q_+, q_-\}) \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{-1, 0, 1\}^k$$

protože součástí vstupu jsou symboly čtené na všech páskách a výstup obsahuje symboly zapsané na pracovní pásky a směr pohybu hlavy na všech páskách.

4.2 Měření složitosti

Pro libovolnou nezápornou funkci f znamená

- $O(f)$ libovolnou nezápornou funkci g, pro kterou existuje n_0 a konstanta c tak, že $(\forall n \geq n_0)\ g(n) \leq cf(n)$.

- $\Omega(f)$ libovolnou nezápornou funkci g, pro kterou existuje n_0 a konstanta $\varepsilon > 0$ tak, že $(\forall n \geq n_0)\ g(n) \geq \varepsilon f(n)$.
- $\Theta(f)$ libovolnou nezápornou funkci g, která je současně $O(f)$ i $\Omega(f)$, tedy existují konstanty $0 < c_1 \leq c_2$ tak, že $(\forall n \geq n_0)\ c_1 f(n) \leq g(n) \leq c_2 f(n)$.

Speciálními případy tohoto značení je $O(1)$, $\Omega(1)$ a $\Theta(1)$, které v uvedeném pořadí znamenají funkce, které jsou shora, zdola a z obou stran omezené kladnou konstantou.

Časová složitost výpočtu TS je počet kroků, které výpočet provedl. Prostorová složitost výpočtu TS je maximální počet polí, které TS navštívil na některé pracovní pásce. Prostor by bylo možné definovat jako součet počtů polí navštívených na jednotlivých pracovních páskách, ale odhady pro maximum jsou jednodušší. Protože počet pásek pro daný TS je konstanta, liší se součet a maximum pouze o multiplikativní konstantu.

Nechť $t : \mathrm{N} \to \mathrm{N}$ a $s : \mathrm{N} \to \mathrm{N}$ jsou libovolné funkce. Pak DTIME(t(n)) označuje třídu jazyků, které lze rozpoznávat pomocí TS v čase $O(t(n))$ a DSPACE($s(n)$) označuje třídu jazyků, které lze rozpoznávat pomocí TS v prostoru $O(s(n))$. Analogicky se definují třídy NTIME($t(n)$) a NSPACE($s(n)$) pro nedeterministický TS, kdy se požaduje, aby omezení časové složitosti $O(t(n))$ nebo omezení prostorové složitosti $O(s(n))$ splnily všechny výpočty pro slovo délky n, přijímající i zamítající.

V některých výpočtech složitosti je použito následující lemma.

Lemma 4.1 *Délka binárního zápisu čísla $n \geq 1$ je $\lfloor \log_2 n \rfloor + 1 = \lceil \log_2(n+1) \rceil$.*

Důkaz. Nechť $n \geq 1$ a nechť l je délka binárního zápisu n. Pak platí

$$2^{l-1} \leq n \leq 2^l - 1 \tag{5}$$

a tedy

$$2^{l-1} \leq n < 2^l$$

Logaritmováním všech čísel v nerovnosti dostaneme

$$l - 1 \leq \log_2 n < l$$

což implikuje $l - 1 = \lfloor \log_2 n \rfloor$ a tedy první formu vyjádření. Druhou formu vyjádření odvodíme následovně. Z (5) plyne

$$2^{l-1} < n + 1 \leq 2^l$$

Logaritmováním všech čísel v nerovnosti dostaneme

$$l - 1 < \log_2(n+1) \leq l$$

což implikuje $\lceil \log_2(n+1) \rceil = l$. QED.

4.3 Turingův stroj pro rozpoznávání symetrických slov

Ukážeme příklad jednopáskového Turingova stroje, který rozpoznává symetrická slova sudé délky v abecedě $\{a, b\}$ v čase $O(n^2)$, kde n je délka vstupního slova. Pro jednopáskový stroj nelze tento odhad zlepšit, ale existuje TS se dvěma páskami, který pracuje v čase $O(n)$. Vstupní páska uvažovaného TS umožňuje čtení i zápis symbolů. Množinu stavů řídící jednotky ani přechodovou funkci nebudeme explicitně uvádět. TS bude popsán programem, ze kterého lze jeho formální popis pomocí přechodové funkce odvodit, a tento postup přibližně popíšeme.

V pseudokódu na Obrázku 5 je použita proměnná H, která v každém kroku výpočtu reprezentuje čtený symbol na pásce a proměnné

- $Y \in \{\Box, a, b\} = D(Y)$,
- $R \in \{\Box, 0, 1\} = D(R)$,

které jsou inicializovány $R = \Box$, $Y = \Box$ a jejichž obory hodnot budeme značit $D(Y)$ a $D(R)$. Podmínka na řádku 2 je nesplněna pouze v případě, že vstupní slovo je prázdné. Ve všech ostatních případech je výpočet ukončen provedením příkazu halt za řádkem 12 nebo za řádkem 15.

Pro vyjádření TS pomocí přechodové funkce převedeme nejprve program na vývojový diagram, jehož množinu vrcholů označme V. Uzly diagramu budou odpovídat očíslovaným řádkům programu a řádku i bude odpovídat uzel v_i. Očíslovány jsou řádky, které obsahují příkaz nebo podmínku. Všem řádkům s příkazem halt odpovídá jeden koncový uzel diagramu, proto je očíslován pouze poslední z nich. Hrany v diagramu jsou určeny možnými pořadími, v jakých mohou být příkazy programu prováděny. Například, while cyklus zahájený na řádku 5 odpovídá v diagramu hranám $v_5 \to v_6$, $v_5 \to v_7$ a $v_6 \to v_5$ a hrana z uzlu v_5 se při výpočtu vybírá podle podmínky v tomto uzlu. Podobně, podmíněný příkaz zahájený na řádku 8 odpovídá hranám $v_8 \to v_9$, $v_8 \to v_{11}$, $v_9 \to v_{10}$, $v_{10} \to v_{13}$, $v_{11} \to v_{12}$, $v_{12} \to v_{20}$ a hrana z uzlu v_9 se při výpočtu vybírá podle podmínky v tomto uzlu.

Jako množinu stavů TS použijeme $Q = V \times D(Y) \times D(R)$. Počáteční stav bude $q_0 = (v_1, \Box, \Box)$, přijímající stav bude $q_+ = (v_{20}, \Box, 1)$ a zamítající stav bude $q_- = (v_{20}, \Box, 0)$. Přechodovou funkci f popíšeme pro každý uzel v_i zvlášť. Uzel diagramu v_i a hrany, které z něj vedou, určují hodnotu přechodové funkce $f(q, x) = (q', x', s)$ pro všechny stavy $q = (v_i, Y, R)$, tedy pro $|D(Y)| \cdot |D(R)|$ stavů, a pro všechny hodnoty čteného symbolu $H = x \in \Gamma$, i v případě, že příkaz nebo podmínka ve v_i na čteném symbolu nezávisí. Příkazy a podmínky v uzlech diagramu neurčují oba parametry x' a s v přechodové funkci a chybějící údaje je potřeba doplnit. Například příkaz vpravo(H) určuje $s = 1$ a je potřeba doplnit $x' = x$. Příkaz $Y := \Box$ je v přechodové funkci vyjádřen akcí, která mění pouze stav řídící jednotky, a je potřeba doplnit $x' = x$ a $s = 0$.

```
1   while H ∈ Σ do
2       Y := H
3       H := □
4       vpravo(H)
5       while H ≠ □ do
6           vpravo(H)
–       od
7       vlevo(H)
8       if H = Y then
9           Y := □
10          H := □
–       else  [platí H = □ nebo H ∈ Σ \ {Y}]
11          Y := □
12          R := 0
–           halt
–       fi
13      vlevo(H)
14      if H = □ then
15          R := 1
–           halt
–       fi
16      while H ∈ Σ do
17          vlevo(H)
–       od
18      vpravo(H)
–   od
19  R := 1
20  halt
```

Obrázek 5: Pseudokód pro Turingův stroj rozpoznávající symetrická slova sudé délky. Výstup 0 (zamítnout) nebo 1 (přijmout) je v proměnné R.

Hodnoty přechodové funkce nebudeme uvádět pro všechny řádky programu na Obrázku 5, ale pouze pro několik vybraných řádků z tohoto programu jako příklady. Uvedené hodnoty přechodové funkce platí pro všechny kombinace hodnot proměnných $Y \in D(Y)$, $R \in D(R)$ a $x \in \Gamma$. Příkaz $Y := H$ na řádku 2 určuje hodnoty přechodové funkce

$$f((v_2, Y, R), x) \;=\; ((v_3, x, R), x, 0)$$

příkaz $R := 0$ na řádku 12 určuje hodnoty

$$f((v_{12}, Y, R), x) \;=\; ((v_{13}, Y, 0), x, 0)$$

řádek 5 obsahující podmínku $H \neq \square$ určuje hodnoty

$$\begin{aligned} f((v_5, Y, R), a) &= ((v_6, Y, R), a, 0) \\ f((v_5, Y, R), b) &= ((v_6, Y, R), b, 0) \\ f((v_5, Y, R), \square) &= ((v_7, Y, R), \square, 0) \end{aligned}$$

a příkaz vpravo(H) na řádku 6 určuje hodnoty

$$f((v_6, Y, R), x) \;=\; ((v_5, Y, R), x, 1)$$

4.4 Turingův stroj s logaritmickým prostorem

Ukážeme, že jazyky $\{a^i b^i \mid i \geq 0\}$ a $\{a^i b^i c^i \mid i \geq 0\}$ lze rozpoznávat v čase $O(n)$ a v prostoru $O(\log n)$. Ukážeme také, že uvedený prostor je nejmenší možný pro rozpoznávání obou uvedených jazyků.

Věta 4.2 *Jazyky $\{a^i b^i \mid i \geq 0\}$ a $\{a^i b^i c^i \mid i \geq 0\}$ lze rozpoznávat TS se vstupní páskou pouze pro čtení současně v čase $O(n)$ a v prostoru $O(\log n)$, kde n je délka vstupního slova.*

Důkaz. TS pro jazyk $L_1 = \{a^i b^i \mid i \geq 0\}$ bude mít dvě pracovní pásky, jednu pro počítání počtu výskytů symbolu a, druhou pro počítání výskytů b. TS pro jazyk $L_2 = \{a^i b^i c^i \mid i \geq 0\}$ bude mít tři pracovní pásky, jednu pro počítání počtu výskytů každého ze symbolů abecedy $\{a, b, c\}$. Počet symbolů bude v binárním zápisu, nejnižší řád bude na druhém poli pásky za symbolem $\triangleright$ a další řády budou postupně na dalších polích, zápis končí prvním výskytem symbolu $\square$. TS pro L_2 při výpočtu kontroluje, že vstupní slovo je tvaru $a^* b^* c^*$, a počítá délku bloku symbolů a, b, resp. c s využitím pracovních pásek, jak je uvedeno výše. Hlavní část programu je na Obrázku 6, kde jsou čtecí hlavy pracovních pásek pro symboly a, b, resp. c označeny W_A, W_B, resp. W_C. Funkce ZvetsiA() zvětšuje počet symbolů a zapsaný na odpovídající pracovní pásku a řídí se programem na Obrázku 7. Funkce ZvetsiB() a ZvetsiC() zvětšují počty symbolů b a c na odpovídajících páskách a lze je popsat analogickým programem, který využívá W_B nebo W_C místo W_A. Funkce PorovnejAB() porovná počty symbolů a, b a pokud jsou tyto počty různé, provede dosazení $R := 0$ a ukončí výpočet příkazem halt. Analogicky, PorovnejBC() porovná počty symbolů b, c a v případě neshody ukončí výpočet stejným způsobem. Programy pro tyto funkce lze snadno sestavit. Program pro jazyk L_1 je podobný jako program pro L_2, ale počítá a porovnává pouze počty symbolů a, b.

Odhad prostorové složitosti $O(\log n)$ pro oba jazyky plyne z toho, že při vstupu délky n může být na pracovních páskách zapsáno nejvýše číslo n, jehož binární zápis má délku $\lfloor \log_2 n \rfloor + 1$.

Odhad časové složitosti $O(n)$ získáme jako součet počtu provedení příkazů z hlavní části programu a z jednotlivých volaných funkcí. Každý příkaz hlavní

```
while H = a do
    ZvetsiA()
    vpravo(H)
od
while H = b do
    ZvetsiB()
    vpravo(H)
od
PorovnejAB()
while H = c do
    ZvetsiC()
    vpravo(H)
od
PorovnejBC()
if H = □ then
    R := 1
else
    R := 0
fi
halt
```

Obrázek 6: Program pro rozpoznávání L_2.

části programu je proveden nejvýše n krát a počet kroků ve funkcích pro porovnání počtů je $O(\log_2 n)$. Tyto části programu tedy přispívají do časové složitosti nejvýše $O(n)$ kroků. Odhad počtu kroků pro funkce, které zvyšují počet výskytů některého symbolu o jedna, je složitější, protože počet kroků, které provede jedno volání některé z těchto funkcí, je různý a závisí na délce úseku číslic 1 v nejnižších řádech zápisu čísla, které se zvětšuje. Tato délka může mít libovolnou hodnotu až do $O(\log_2 n)$. Označme jako p výslednou hodnotu počtu symbolů, která je zapsána na pracovní pásce pro počet symbolů a, což je současně počet volání funkce ZvetsiA(). Zřejmě platí $p \leq n$. Číslice řádu 0 v zápisu čísla se mění při každém volání funkce, tedy při p voláních. Číslice řádu 1 se mění při každém druhém volání funkce, tedy nejvýše při $p/2$ voláních. Obecně, číslice řádu i se mění nejvýše při $p/2^i$ voláních funkce. Řád nejvyšší číslice v p je $\lfloor \log_2 p \rfloor$. Celkový počet změn binárních číslic, které provede funkce ZvetsiA(), je tedy nejvýše

$$\sum_{i=0}^{\log_2 p} \frac{p}{2^i} \leq \sum_{i=0}^{\infty} \frac{p}{2^i} = 2p \leq 2n$$

Jedna změna číslice vyžaduje nejvýše jedno opakování každého z cyklů ve funkci ZvetsiA(). Tedy celkový počet kroků, které provede tato funkce, je $O(n)$. Stejný


```
while (W_A = 1) do
    W_A := 0
    vpravo(W_A)
od
W_A := 1
while (W_A ≠ ▷) do
    vlevo(W_A)
od
vpravo(W_A)
```

Obrázek 7: Program pro funkci ZvetsiA().

odhad platí i pro funkce ZvetsiB() a ZvetsiC(). Součet počtu kroků provedených v těchto funkcích je tedy nejvýše $O(n)$ a celkový odhad času je tedy také $O(n)$. QED.

Věta 4.3 *Každý TS, který rozpoznává některý z jazyků* $\{a^i b^i \mid i \geq 0\}$ *a* $\{a^i b^i c^i \mid i \geq 0\}$*, má prostorovou složitost* $\Omega(\log n)$*, kde* n *je délka vstupního slova.*

Důkaz. Nechť TS M rozpoznává jazyk $\{a^l b^l \mid l \geq 0\}$ a má k pracovních pásek, jejichž abeceda je Γ. Uvažujme výpočet pro slovo $a^l b^l$, kde $l \geq 1$, který využívá prostor s. Ukážeme sporem, že platí $l \leq |Q| \cdot |\Gamma|^{ks}$.

Předpokládejme, že $l > |Q| \cdot |\Gamma|^{ks}$. Nechť t_1 je poslední krok výpočtu, kdy se čtecí hlava nachází na symbolu x_1 vstupního slova. Pro $2 \leq i \leq l$, nechť t_i je první krok následující za krokem t_1, kdy je čtecí hlava na symbolu x_i vstupního slova. Pro $i = 1, \ldots, l$, nechť r_i je dvojice tvořená stavem řídící jednotky a obsahem prvních s polí na všech pracovních páskách v kroku t_i. Počet všech takových dvojic je $|Q| \cdot |\Gamma|^{ks}$. Protože $l > |Q| \cdot |\Gamma|^{ks}$, existují $i < j$ tak, že $r_i = r_j$. Výpočet pro slovo $a^l b^l$ je přijímající. Sestrojíme přijímající výpočet pro slovo $a^{l+j-i} b^l$ tak, že až do kroku t_j postupuje stejně jako původní výpočet a další část výpočtu probíhá stejně jako původní výpočet od kroku t_i do konce. Tento výpočet je také přijímající, což je spor, protože $l + j - i \neq l$.

Platí tedy $|Q| \cdot |\Gamma|^{ks} \geq l$. Protože $l = n/2$, platí

$$ks \log |\Gamma| + \log |Q| \geq \log(n/2)$$

a tedy

$$s \geq \frac{\log(n/2) - \log |Q|}{k \log |\Gamma|} = \Omega(\log n)$$

QED.

```
while R = □ do
    if Z ∈ V then
        vybrat libovolně α tak, že Z → α je pravidlo G
        pop()
        push(α)
    elif Z ∈ Σ then
        if H = Z then
            pop()
            vpravo(H)
        else [platí H ∈ Σ \ {Z} nebo H = □]
            R := 0
        fi
    else [platí Z = □]
        if H = □ then
            R := 1
        else [platí H ∈ Σ]
            R := 0
    fi
od
```

Obrázek 8: Pseudokód pro nedeterministický zásobníkový automat rozpoznávající jazyk generovaný bezkontextovou gramatikou. Proměnná R určuje po skončení výpočtu přijetí ($R = 1$) nebo zamítnutí ($R = 0$) slova daným výpočtem.

4.5 Nedeterministický zásobníkový automat

Popíšeme nedeterministický zásobníkový automat, který má řídící jednotku se stavy $\{q_0, q_+, q_-\}$, jeden zásobník a vstupní pásku, kde je vstupní slovo zapsáno počínaje prvním polem pásky a hlava na vstupní pásce se může v jednom kroku posunout vpravo nebo zůstat na místě. Tento typ automatu je postačující pro rozpoznávání slov generovaných libovolnou bezkontextovou gramatikou. Obecný nedeterministický zásobníkový automat může mít libovolně velký konečný počet stavů, které spoluurčují posloupnost akcí automatu. Třída rozpoznávaných jazyků se tím nezmění, ale rozpoznávání slov může být v některých případech efektivnější díky menšímu nedeterminismu.

Postup rozpoznávání jazyka generovaného libovolnou bezkontextovou gramatikou G (při obvyklém významu Σ, V, S) lze popsat pomocí pseudokódu v Obrázku 8, kde je použito následující značení

- Proměnná R je inicializovaná jako $\square$ a po skončení výpočtu obsahuje 1, pokud je slovo daným výpočtem přijato, nebo 0, pokud je slovo daným výpočtem zamítnuto.
- Proměnná H reprezentuje čtený symbol na vstupní pásce,

symbol Z	množina akcí v závislosti na symbolu H
$Z \in V$	akce tvaru $[\text{pop}(), \text{push}(\alpha), q_0]$ pro všechna α taková, že $Z \to \alpha$ je pravidlem G
$Z \in \Sigma$	pokud $H = Z$, pak $[\text{pop}(), \text{vpravo}(H), q_0]$, a v opačném případě $[q_-]$
$Z = \square$	pokud $H = \square$, pak $[q_+]$, a v opačném případě $[q_-]$

Obrázek 9: Popis přechodové funkce nedeterministického zásobníkového automatu rozpoznávající jazyk generovaný bezkontextovou gramatikou G.

- Proměnná Z reprezentuje symbol na vrcholu zásobníku,
- Funkce pop() odstraní symbol z vrcholu zásobníku,
- Funkce push(α) vloží do zásobníku slovo α,
- Funkce vpravo(H) posune čtecí hlavu na vstupní pásce o jeden symbol vpravo.

Výběr pravidla $Z \to \alpha$, které určí, kterým slovem α je nahrazen vrchol zásobníku, pokud $Z \in V$, je nedeterministický. To znamená, že výpočet tohoto automatu pro totéž vstupní slovo může probíhat více různými způsoby. Slovo je přijato, pokud existuje výpočet, který jej přijme.

Program na Obrázku 8 lze převést na popis automatu s přechodovou funkcí. Protože každé opakování cyklu v programu lze reprezentovat jedním krokem podle přechodové funkce, není nutné přidávat stavy, které kódují, který příkaz je právě prováděn, a množinu stavů výsledného automatu lze zvolit $Q = \{q_0, q_+, q_-\}$. Stav řídící jednotky automatu v každém okamžiku výpočtu odpovídá hodnotě proměnné R v programu ve stejném okamžiku výpočtu podle vzájemného přiřazení $q_0 \leftrightarrow \square$, $q_+ \leftrightarrow 1$, $q_- \leftrightarrow 0$. Přechodová funkce automatu je popsána na Obrázku 9, kde je pro každý možný symbol Z na vrcholu zásobníku a každý možný symbol H čtený na vstupní pásce, tedy pro $H \in \Sigma \cup \{\square\}$, popsána množina akcí $f(Z, H)$. Každá akce je zapsána jako seznam instrukcí v hranatých závorkách. Pokud je $Z \in V$, závisí počet možných akcí na gramatice a může být větší než jedna. V ostatních případech je akce určena jednoznačně.

Věta 4.4 *Pro libovolnou bezkontextovou gramatiku G rozpoznává nedeterministický zásobníkový automat popsaný v Obrázku 8 a, ekvivalentně, v Obrázku 9, jazyk $L(G)$.*

Důkaz. Výpočet automatu pro w simuluje některou levou derivaci slova w v G. Pro důkaz tohoto tvrzení doplníme větné formy v levé derivaci o znak $|$, který odděluje prefix složený z terminálních symbolů od zbytku větné formy. Prefix bude obsahovat terminální symboly vlevo od všech neterminálů, kromě některých terminálů, které byly vygenerovány naposledy použitým pravidlem gramatiky. Přesná definice je následující. Výchozí doplněná větná forma je $|S$. Doplněná levá

derivace může neterminál nahradit podle pravidla gramatiky pouze v případě, že bezprostředně vlevo od něj je symbol |. Pokud původní derivace obsahuje krok $uA\beta \Rightarrow u\alpha\beta$, pak doplněná derivace obsahuje krok $u|A\beta \Rightarrow u|\alpha\beta$. Kromě toho, doplněná levá derivace obsahuje kroky, které přesunou symbol | přes jeden terminální symbol vpravo, tedy kroky tvaru $u|x\alpha \Rightarrow ux|\alpha$ pro $x \in \Sigma$. Doplněná levá derivace odvozuje w, pokud její poslední větná forma je slovo $w|$.

Dále je potřeba ukázat, že pro každé $w \in \Sigma^*$ platí

- $w \in L(G)$ právě tehdy, když je odvozeno některou doplněnou levou derivací.
- Přijímající výpočet automatu pro w existuje právě tehdy, když je w odvozeno některou doplněnou levou derivací.

K důkazu prvního tvrzení je potřeba popsat konstrukce, které převádí levou derivaci na doplněnou levou derivaci pro stejné slovo a naopak. Pokud vycházíme z levé derivace, doplníme symbol | v každé větné formě před první neterminál nebo jako poslední symbol, pokud větná forma neterminál neobsahuje. Pak je potřeba přidat kroky, které přesouvají symbol | přes terminály. Opačným směrem stačí vypustit symbol | z každé doplněné větné formy a vypustit kroky, kdy se takto získané větné formy opakují.

Pro důkaz druhého tvrzení je potřeba ukázat konstrukce, které přijímající výpočet automatu převedou na doplněnou levou derivaci pro stejné slovo a naopak. Kroky automatu, kdy je na vrcholu zásobníku neterminál, odpovídají krokům derivace, které rozvíjí neterminál bezprostředně za symbolem |. Kroky, kdy je na vrcholu zásobníku terminál, odpovídají krokům derivace, které přesouvají symbol | přes jeden terminál. Přijímající výpočet končí s prázdným zásobníkem a po přečtení celého slova, tedy získáme doplněnou větnou formu tvaru $w|$. Konstrukce přijímajícího výpočtu na základě doplněné levé derivace se odvodí podobně, protože výsledek konstrukce je v obou směrech jednoznačně určen jejím vstupem. QED.

5 Regulární jazyky

Regulární jazyky byly původně definovány jako nejmenší třída jazyků, která obsahuje konečné jazyky a je uzavřena na operace zřetězení a iterace. Tomuto způsobu odvození jazyka odpovídají regulární výrazy. Stejnou třídu jazyků lze popsat také pomocí konečných automatů a nedeterministických konečných automatů s ε-přechody, které budeme nazývat přechodový diagram.

Konečný automat lze popsat jako jednopáskový Turingův stroj, jehož vstupní páska je pouze pro čtení a v každém kroku se čtecí hlava posune o jedno pole vpravo. Lze ukázat, že jednopáskový TS, jehož páska je pouze pro čtení, ale čtecí hlava se může pohybovat oběma směry, rozpoznává také pouze regulární jazyky. Tyto jazyky lze tedy také charakterizovat jako jazyky rozpoznatelné TS s prostorem $O(1)$, tedy s konstantním prostorem, který nezávisí na délce vstupního slova.

5.1 Konečné automaty

Konečný automat postupně čte ze vstupu jednotlivé symboly vstupního slova a po přečtení každého symbolu provede určitý výpočet s omezenou pamětí. Obsah této konečné paměti budeme nazývat stav. Požadavek, že výpočet probíhá s omezenou pamětí, budeme formalizovat tím, že množina stavů, které mohou při výpočtu nastat, je konečná. Výstup automatu závisí na stavu, do kterého automat přejde po přečtení celého slova. V základní formě konečného automatu je výstupem pouze informace, zda je slovo přijato. Lze uvažovat variantu, že výsledný stav poskytuje i další informace.

Definice 5.1 Konečný automat je určen následujícími parametry:

Σ	vstupní abeceda
Q	konečná množina stavů
$\delta : Q \times \Sigma \to Q$	přechodová funkce
$q_0 \in Q$	počáteční stav
$F \subseteq Q$	množina přijímajících stavů .

Výpočet konečného automatu probíhá následovně. Na začátku výpočtu se automat nachází ve stavu q_0 a postupně přijímá jednotlivé symboly vstupního slova. Po přijetí každého z nich určí svůj následující stav pomocí přechodové funkce. Pokud je na vstupu slovo $w = a_1 a_2 \dots a_n \in \Sigma^*$, pak automat postupně prochází stavy

$$q_0, q_1 = \delta(q_0, a_1), q_2 = \delta(q_1, a_2), \dots, q_n = \delta(q_{n-1}, a_n)$$

a slovo w je přijato, pokud $q_n \in F$. Pro formalizaci tohoto postupu použijeme zobecněnou přechodovou funkci $\delta^* : Q \times \Sigma^* \to Q$, která pro libovolný stav $q \in Q$, slovo $w \in \Sigma^*$ a symbol $a \in \Sigma$ splňuje

- $\delta^*(q, \varepsilon) = q$
- $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$

Definice 5.2 Konečný automat přijímá slovo $w \in \Sigma^*$, pokud $\delta^*(q_0, w) \in F$. Jazyk všech slov, která přijímá konečný automat M, budeme značit $L(M)$.

5.2 Dělitelnost čísel v obecné číselné soustavě

Uvažujme číselnou soustavu se základem $b \geq 2$ a číslicemi $\{0, \ldots, b-1\}$. Množinu číslic budeme pro jednoduchost ztotožňovat s množinou Z_b. Ukážeme konečné automaty pro rozpoznávání čísel v abecedě Z_b, která jsou dělitelná daným číslem $m \geq 1$, při čtení číslic zleva doprava a zprava doleva. Vstupní číslo budeme v obou případech zapisovat jako $x_n x_{n-1} \ldots x_0$ a jeho hodnota je

$$\mathrm{val}(x_n x_{n-1} \ldots x_0) = \sum_{i=0}^{n} x_i b^i$$

Při čtení zápisu čísla zleva doprava, tedy od nejvyšších řádů, lze použít automat s množinou stavů $Q = Z_m$, počátečním stavem $q_0 = 0$, množinou přijímajících stavů $F = \{0\}$ a s přechodovou funkcí $\delta : Z_m \times Z_b \to Z_m$ definovanou následovně

$$\delta(r, x) = (br + x) \bmod m$$

Po přečtení počátečního úseku čísla $x_n \ldots x_{i+1}$, tedy před čtením číslice x_i, je stav roven zbytku modulo m čísla tvořeného přečtenými číslicemi, tedy $r = \mathrm{val}(x_n \ldots x_{i+1}) \bmod m$.

Při čtení zápisu čísla zprava doleva, tedy od nejnižších řádů, lze použít automat s množinou stavů $Q = Z_m \times Z_m$, počátečním stavem $q_0 = (1, 0)$, množinou přijímajících stavů $F = Z_m \times \{0\}$ a s přechodovou funkcí $\delta : (Z_m \times Z_m) \times Z_b \to (Z_m \times Z_m)$ definovanou následovně

$$\delta((r_1, r_2), x) = ((br_1) \bmod m,\ (r_1 x + r_2) \bmod m)$$

Stav (r_1, r_2) po přečtení číslic $x_{i-1}, \ldots, x_0$, tedy před čtením číslice x_i, je

$$(r_1, r_2) = (b^i \bmod m,\ \mathrm{val}(x_{i-1} \ldots x_0) \bmod m)$$

5.3 Rozpoznávání čísel v programu

Uvedeme příklad zobecněného konečného automatu, který rozlišuje čísla ve zdrojovém textu programu. Zobecnění spočívá v tom, že nerozlišujeme pouze přijímající a zamítající stavy, ale přijímající stavy jsou ještě rozlišeny podle typu čísla, které odpovídá dosud přečteným symbolům ze vstupu. Budeme uvažovat čísla typu integer, čísla typu real bez exponentu a s exponentem. Pro jednoduchost

budeme požadovat, aby číslo typu real buď neobsahovalo desetinnou tečku nebo aby obsahovalo alespoň jednu číslici před desetinnou tečkou a alespoň jednu číslici za desetinnou tečkou.

Stavy automatu jsou $q_0, \ldots, q_7, q_-$ a přechodová funkce je popsána v Tabulce 10. V tabulce není třeba rozlišovat mezi znaménkem + a − a obě jsou shrnuta pod označení "znaménko". Podobně, označení "číslice" v tabulce znamená libovolnou číslici.

stav	přechod číslice	přechod znaménko	přechod "."	přechod "e"	označení stavu
q_0	q_2	q_1	q_-	q_-	neúplné
q_1	q_2	q_-	q_-	q_-	neúplné
q_2	q_2	q_-	q_3	q_5	integer
q_3	q_4	q_-	q_-	q_-	neúplné
q_4	q_4	q_-	q_-	q_5	real bez exponentu
q_5	q_7	q_6	q_-	q_-	neúplné
q_6	q_7	q_-	q_-	q_-	neúplné
q_7	q_7	q_-	q_-	q_-	real s exponentem
q_-	q_-	q_-	q_-	q_-	chyba

Obrázek 10: Konečný automat rozlišující čísla v textu programu

Poslední sloupec tabulky přiřazuje každému stavu automatu informaci, která se vztahuje k přečtené části vstupní posloupnosti. Stavy označené jako "neúplné" odpovídají vstupním posloupnostem, které nejsou číslo, ale mohou být na číslo doplněny dalšími znaky. Stavy z množiny $\{q_2, q_4, q_7\}$ odpovídají jednotlivým typům čísel. Pokud za množinu přijímajících stavů zvolíme všechny tyto stavy, bude automat přijímat všechny uvedené typy čísel. Stav q_- označený jako "chyba" znamená, že přijatá část vstupního slova není číslem a nemůže být na číslo doplněna dalšími znaky. Pokud do tohoto stavu automat přejde z některého ze stavů $\{q_2, q_4, q_7\}$, pak je přijatá posloupnost znaků bez naposledy čteného znaku považována za číslo, pokud je automat použit k rozčlenění vstupního programu na úseky, které mají přiřazen smysl pouze jako celek.

5.4 Přechodové diagramy

Přechodové diagramy jsou zobecněním konečných automatů a lze na ně snadno převést libovolný regulární výraz v podstatě při zachování jeho struktury. Platí také, že ke každému přechodovému diagramu lze sestrojit ekvivalentní konečný automat i regulární výraz, ale tyto konstrukce mohou exponenciálně zvětšit velikost reprezentace.

Definice 5.3 Přechodový diagram nad abecedou Σ je orientovaný graf, jehož

množinu vrcholů budeme označovat V. Graf přechodového diagramu může obsahovat smyčky a vícenásobné hrany a každá hrana je ohodnocena symbolem z množiny $\Sigma \cup \{\varepsilon\}$. Kromě toho je určena množina počátečních vrcholů $S \subseteq V$ a množina přijímajících vrcholů $F \subseteq V$.

V obrázcích budeme místo několika hran mezi dvěma vrcholy kreslit jednu hranu ohodnocenou odpovídající množinou symbolů. Na druhé straně, pro jednoduchost formálního popisu budeme podle definice výše předpokládat, že každé hraně je přiřazen právě jeden symbol. V tomto případě odpovídá každé cestě v diagramu jednoznačně určené slovo.

Definice 5.4 Cestu v diagramu nazveme w-cestou, jestliže slovo $w \in \Sigma^*$ vznikne zřetězením ohodnocení jednotlivých hran cesty. Hrany ohodnocené symbolem ε přispívají do tohoto zřetězení prázdným slovem a výsledné zřetězení je tedy stejné, jako kdyby tyto hrany byly vynechány.

Definice 5.5 Řekneme, že diagram přijímá slovo $w \in \Sigma^*$, jestliže v něm existuje w-cesta začínající ve vrcholu, který patří do S, a končí ve vrcholu, který patří do F. Jazyk všech slov přijímaných diagramem D budeme značit $L(D)$.

Věta 5.6 *Jestliže jsou L_1 a L_2 reprezentovány přechodovým diagramem, pak lze sestrojit přechodové diagramy pro jazyky $L_1 \cup L_2$, $L_1 \cdot L_2$ a L_1^*.*

Důkaz. Nechť pro $i = 1, 2$ je D_i diagram pro L_i a nechť S_i, F_i jsou odpovídající množiny počátečních a přijímajících vrcholů. Budeme předpokládat, že množiny vrcholů diagramů D_1 a D_2 jsou disjunktní.

1. $L_1 \cup L_2$: Spojíme diagramy D_1 a D_2 do jednoho diagramu, ve kterém tvoří D_1 a D_2 disjunktní části a definujeme $S = S_1 \cup S_2$ a $F = F_1 \cup F_2$.

2. $L_1 \cdot L_2$: Spojíme diagramy D_1 a D_2 jako v předchozím případě, z každého vrcholu F_1 přidáme ε-hranu do každého vrcholu S_2 a definujeme $S = S_1$, $F = F_2$.

3. L_1^*: K diagramu D_1 přidáme nový vrchol v, z každého vrcholu F_1 přidáme ε-hranu do každého vrcholu S_1 a zvolíme $S = S_1 \cup \{v\}$ a $F = F_1 \cup \{v\}$.

Dokažme, že diagram z bodu 3 přijímá jazyk L_1^*. Cesta délky 0, která začíná a končí ve vrcholu v, přijímá prázdné slovo. Každou další cestu z S do F lze rozdělit na úseky které vedou z S_1 do F_1 a jsou odděleny přechody přes přidané ε-hrany. Každý z těchto úseků je cestou v D_1. Je tedy $L(D) \subseteq L_1^*$. Naopak, ke každé posloupnosti slov $w_1, w_2, \ldots, w_k \in L_1$ pro $k \geq 1$ existuje cesta z S do F, která vznikne z cest pro slova w_i, $i = 1, 2, \ldots, k$ v D_1 tak, že tyto cesty spojíme přechodem přes přidané ε-hrany. Protože je tato cesta $w_1 w_2 \ldots w_k$-cestou, je $w_1 w_2 \ldots w_k \in L(D)$. Cesta délky 0 přes vrchol v zaručuje $\varepsilon \in L(D)$. Je tedy $L_1^* \subseteq L(D)$. QED.

5.5 Konečný automat je speciální přechodový diagram

Konstrukce v důkazu následující věty převede konečný automat na přechodový diagram pro stejný jazyk tak, že hrany a jejich ohodnocení v diagramu reprezentují tabulku přechodové funkce výchozího konečného automatu. V tomto smyslu je konečný automat speciálním případem přechodového diagramu. Naopak, přechodový diagram lze definovat také jako nedeterministický konečný automat s ε-přechody, což jsou kroky výpočtu, kdy není čten symbol ze vstupu a pouze se změní stav řídící jednotky.

Věta 5.7 *Ke každému konečnému automatu M existuje přechodový diagram D tak, že $L(D) = L(M)$.*

Důkaz. Nechť M je konečný automat s výše uvedeným významem Σ, Q, q_0, F a δ. Jazyk $L(M)$ lze vyjádřit přechodovým diagramem, jehož vrcholy jsou stavy z množiny Q, množina počátečních uzlů je $S = \{q_0\}$, množina přijímajících uzlů F je rovna množině přijímajících stavů výchozího konečného automatu a pro každý uzel q a každý symbol $x \in \Sigma$ vede z uzlu q do uzlu $\delta(q, x)$ hrana ohodnocená x.

Indukcí podle délky slova w lze dokázat, že pro každé slovo $w \in \Sigma^*$ existuje právě jedna w-cesta v diagramu, která začíná v q_0. Navíc, tato cesta končí ve vrcholu $\delta^*(q_0, w)$. Platí tedy $w \in L(D) \Leftrightarrow \delta^*(q_0, w) \in F \Leftrightarrow w \in L(M)$. QED.

5.6 Reverze jazyka

Pokud je jazyk vyjádřený regulárním výrazem nebo přechodovým diagramem, lze reverzi jazyka vyjádřit stejným typem reprezentace bez nárůstu velikosti. Regulární výraz se obrátí přirozeným způsobem. V přechodovém diagramu je potřeba obrátit směr orientace hran a vyměnit množiny S a F.

Pokud je jazyk L vyjádřen konečným automatem, lze jej převést na přechodový diagram a v této reprezentaci provést reverzi výše uvedeným způsobem. Tím vznikne přechodový diagram pro L^R, který obecně není konečným automatem a nejmenší konečný automat pro jazyk L^R může mít počet stavů exponenciálně větší než konečný automat pro jazyk L. Příkladem takových jazyků jsou $L = L_k^R$ a $L^R = (L_k^R)^R = L_k$, kde L_k je jazyk popsaný v následujícím odstavci.

Pro každé přirozené číslo $k \geq 1$, nechť L_k je jazyk určený regulárním výrazem $(a + b)^* a (a + b)^{k-1}$. Každý konečný automat pro jazyk L_k obsahuje alespoň 2^k stavů, protože libovolná dvě slova délky alespoň k, která se liší v některém z posledních k symbolů, musí automat převést do různých stavů. Na druhé straně, pro jazyk L_k^R existuje konečný automat s $k + 2$ stavy.

5.7 Ekvivalence konečných automatů a přechodových diagramů

Věta 5.8 *Ke každému přechodovému diagramu D existuje konečný automat M tak, že $L(M) = L(D)$.*

Důkaz. Nechť D je diagram s množinou vrcholů V. Nechť S je množina počátečních vrcholů a F množina přijímajících vrcholů D. Množina stavů Q konstruovaného automatu M bude tvořena systémem všech podmnožin V. Počáteční stav bude $q_0 = S'$, kde S' je množina vrcholů dosažitelných ε-cestou z některého vrcholu S. Zřejmě platí $S \subseteq S'$. Přechodová funkce $\delta : Q \times \Sigma \to Q$ a množina přijímajících stavů automatu F' budou určeny následovně:

δ Pro libovolný stav $K \in Q$, tedy $K \subseteq V$, a $a \in \Sigma$ je stav $\delta(K, a)$ roven množině vrcholů dosažitelných a-cestou z některého vrcholu K.

$F' = \{K \subseteq V; K \cap F \neq \emptyset\}$.

Dokážeme, že $L(M) = L(D)$.

Lemma 5.9 *Pro každé w je $\delta^*(q_0, w)$ právě množina vrcholů dosažitelných z S nějakou w-cestou.*

Důkaz. Indukcí podle délky w. Podle definice zobecněné přechodové funkce platí

$$\delta^*(q_0, \varepsilon) = S' = \text{vrcholy dosažitelné z } S \text{ } \varepsilon\text{-cestou.}$$

Dále, nechť tvrzení platí pro w. Dokažme jej pro wa, kde $a \in \Sigma$.

Nechť $v \in \delta^*(q_0, wa)$. Protože $\delta^*(q_0, wa) = \delta(\delta^*(q_0, w), a)$, je v dosažitelný a-cestou z některého vrcholu $\delta^*(q_0, w)$, tedy z vrcholu, který je podle indukčního předpokladu dosažitelný w-cestou z S. Z toho plyne, že v je dosažitelný wa-cestou z S.

Nechť v je vrchol dosažitelný wa-cestou z S. Na této cestě existuje vrchol v' tak, že

1. v' je dosažitelný w-cestou z S
2. v je dosažitelný a-cestou z vrcholu v'

Z indukčního předpokladu plyne $v' \in \delta^*(q_0, w)$ a tedy $v \in \delta(\delta^*(q_0, w), a) = \delta^*(q_0, wa)$. Tím je lemma dokázáno. QED.

Z dokázaného lemmatu plyne $w \in L(M) \Leftrightarrow \delta^*(q_0, w) \in F' \Leftrightarrow \delta^*(q_0, w) \cap F \neq \emptyset$ $\Leftrightarrow$ v F existuje vrchol dosažitelný w-cestou z $S \Leftrightarrow w \in L(D)$. Tím je věta dokázána. QED.

5.8 Ekvivalence regulárních výrazů a přechodových diagramů

Ukážeme, že jazyk lze zapsat regulárním výrazem právě tehdy, když lze reprezentovat pomocí přechodového diagramu a tedy i konečného automatu.

Věta 5.10 *Ke každému regulárnímu výrazu existuje ekvivalentní přechodový diagram.*

Důkaz. Indukcí podle složitosti regulárního výrazu. Pokud regulární výraz neobsahuje operace, reprezentuje jazyk obsahující jedno slovo délky nejvýše 1. Tento jazyk lze vyjádřit přechodovým diagramem. Uvažme regulární výraz některého z tvarů $(\alpha + \beta)$, $(\alpha \cdot \beta)$ nebo $(\alpha)^*$, kde α a β jsou jednodušší výrazy. Z indukčního předpokladu plyne, že jazyky reprezentované α a β lze vyjádřit přechodovým diagramem. Protože množina jazyků reprezentovaných přechodovými diagramy je uzavřená na sjednocení, zřetězení a iteraci podle Věty 5.6, lze také jazyk reprezentovaný celým výrazem vyjádřit pomocí přechodového diagramu. QED.

Věta 5.11 (Kleene) *Ke každému přechodovému diagramu existuje ekvivalentní regulární výraz.*

Důkaz. Nechť L je přijímán diagramem D. Důkaz provedeme indukcí přes zvětšující se množiny cest. Nechť množina vrcholů diagramu je $V = \{v_1, \ldots, v_n\}$. Řekneme, že cesta v diagramu prochází vrcholem u, pokud obsahuje hranu z nějakého vrcholu do u a navazující hranu z u do dalšího vrcholu. Nechť $R^k_{i,j}$ je jazyk slov, která jsou ohodnocením nějaké cesty, která vede z v_i do v_j a prochází pouze vrcholy z množiny $\{v_1, \ldots, v_k\}$.

Speciálně, $R^0_{i,j}$ je jazyk slov přijímaných cestami, které jsou tvořeny hranou z v_i do v_j, případně také cestou délky 0, pokud $i = j$. Pro každé i, j je $R^0_{i,j}$ vyjádřitelný regulárním výrazem, protože je konečný.

Dokážeme, že pro každé i, j, k, $1 \leq i, j, k \leq n$, platí

$$R^k_{i,j} = R^{k-1}_{i,j} \cup R^{k-1}_{i,k}(R^{k-1}_{k,k})^* R^{k-1}_{k,j} \ . \tag{6}$$

Zřejmě $R^{k-1}_{i,j} \subseteq R^k_{i,j}$. Jestliže slovo patří do jazyka $R^{k-1}_{i,k}(R^{k-1}_{k,k})^* R^{k-1}_{k,j}$, pak je ohodnocením cesty, která vede z v_i do v_k, pak případně jednou nebo vícekrát z v_k do v_k a pak vede z vrcholu v_k do v_j. Všechny úseky této cesty mezi uvedenými vrcholy prochází pouze vrcholy z množiny $\{v_1, \ldots, v_{k-1}\}$ a celá cesta prochází pouze vrcholy z množiny $\{v_1, \ldots, v_k\}$. Tím je dokázána inkluze

$$R^k_{i,j} \supseteq R^{k-1}_{i,j} \cup R^{k-1}_{i,k}(R^{k-1}_{k,k})^* R^{k-1}_{k,j} \ .$$

Nechť naopak $w \in R^k_{i,j}$. Slovo w je ohodnocením nějaké cesty z v_i do v_j, která může procházet vrcholy $v_1, \ldots, v_k$. Pokud tato cesta neprochází vrcholem v_k, pak

$w \in R^{k-1}_{i,j}$. Pokud prochází vrcholem v_k, pak ji rozdělíme na úseky podle průchodů tímto vrcholem. První úsek vede z vrcholu v_i do v_k, pak je nějaký (případně nulový) počet úseků z v_k do v_k a poslední úsek vede z v_k do v_j. Protože jednotlivé uvedené úseky vrcholem v_k neprochází, patří slovo w do jazyka $R^{k-1}_{i,k}(R^{k-1}_{k,k})^* R^{k-1}_{k,j}$ a platí tedy

$$R^k_{i,j} \subseteq R^{k-1}_{i,j} \cup R^{k-1}_{i,k}(R^{k-1}_{k,k})^* R^{k-1}_{k,j} \ .$$

Vyjádření jazyka $R^k_{i,j}$ ve tvaru (6) pomocí jazyků $R^{k-1}_{i',j'}$ pro vhodná i', j' obsahuje pouze operace sjednocení, zřetězení a iterace. Všechny jazyky $R^0_{i,j}$ jsou vyjádřitelné regulárním výrazem. Jestliže pro nějaké k jsou všechny jazyky $R^{k-1}_{i,j}$ vyjádřitelné regulárním výrazem, pak také jazyk $R^k_{i,j}$ je vyjádřitelný regulárním výrazem. Protože toto platí pro každé i, j, dostaneme indukcí pro $k = 1, \ldots, n$, že $R^k_{i,j}$ je vyjádřitelný regulárním výrazem pro všechny uvažované hodnoty i, j, k. Protože

$$L(D) = \bigcup_{i \in S} \bigcup_{j \in F} R^n_{i,j} \ ,$$

dokázali jsme, že $L(D)$ lze vyjádřit regulárním výrazem. QED.

Důkaz předchozí věty zároveň popisuje konstrukci regulárního výrazu pro $L(D)$. Označme jako d_k maximální délku regulárního výrazu pro jazyky $R^k_{i,j}$ pro všechna i, j. Do délky počítáme pouze výskyty symbolů abecedy a symbolu ε. Protože popsaná konstrukce vede na výraz, který neobsahuje iteraci podvýrazu, který je sám iterací, pak počet výskytů uvedených symbolů umožňuje odhadnout shora celkovou délku výrazu. Z konstrukce výrazu pro jazyk $R^k_{i,j}$ plyne

$$\begin{aligned} d_0 &\leq |\Sigma| + 1 \\ d_{k+1} &\leq 4 d_k \ , \end{aligned}$$

což implikuje $d_n \leq 4^n(|\Sigma| + 1)$. Délka výsledného výrazu je tedy nejvýše konstantní násobek $4^n(|\Sigma| + 1)$, což lze zapsat jako $O(4^n(|\Sigma| + 1))$.

5.9 Operace s jazyky vyjádřenými konečným automatem

V důkazu Věty 5.6 jsme ukázali, jak z přechodového diagramu pro jazyky L_1, L_2 získat přechodový diagram pro $L_1 \cup L_2$, $L_1 \cdot L_2$ a L_1^*. Nyní doplníme algoritmy pro další operace s regulárními jazyky, pro které je výhodnější reprezentace pomocí konečných automatů.

Nechť $L_i = L(M_i)$ pro $i = 1, 2$, přičemž automat M_i je určen parametry $\Sigma, Q_i, \delta_i, q_{0,i}, F_i$. Automaty pro jazyky $L_1 \cap L_2$, $L_1 \cup L_2$, $L_1 \setminus L_2$ a symetrickou diferenci $L_1 \triangle L_2 = (L_1 \setminus L_2) \cup (L_2 \setminus L_1)$ zkonstruujeme pomocí simulace paralelního běhu M_1 a M_2. Tomu odpovídá automat určený parametry:

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ \delta((q_1, q_2), a) &= (\delta_1(q_1, a), \delta_2(q_2, a)) \\ q_0 &= (q_{0,1}, q_{0,2}) \end{aligned}$$

Množinu přijímajících stavů F zvolíme různou podle toho, který jazyk bude automat přijímat.

$F_\cap = F_1 \times F_2$	pro $L_1 \cap L_2$
$F_\cup = F_1 \times Q_2 \cup Q_1 \times F_2$	pro $L_1 \cup L_2$
$F_\setminus = F_1 \times (Q_2 \setminus F_2)$	pro $L_1 \setminus L_2$
$F_\triangle = F_1 \times (Q_2 \setminus F_2) \cup (Q_1 \setminus F_1) \times F_2$	pro $L_1 \triangle L_2$

Jednotlivé přijímající množiny jsou zvoleny tak, že platí:

$$\begin{aligned} (q_1, q_2) \in F_\cap &\Leftrightarrow q_1 \in F_1 \wedge q_2 \in F_2 \\ (q_1, q_2) \in F_\cup &\Leftrightarrow q_1 \in F_1 \vee q_2 \in F_2 \\ (q_1, q_2) \in F_\setminus &\Leftrightarrow q_1 \in F_1 \wedge q_2 \notin F_2 \\ (q_1, q_2) \in F_\triangle &\Leftrightarrow q_1 \in F_1 \not\equiv q_2 \in F_2 \ . \end{aligned}$$

Dále platí, že pro každé vstupní slovo $w \in \Sigma^*$, přejde zkonstruovaný automat M do stavu (q_1, q_2) právě tehdy, když automat M_1 přejde do q_1 a M_2 přejde do q_2. Z toho plyne

$$\begin{aligned} \delta^*(q_0, w) \in F_\cap &\Leftrightarrow w \in L_1 \cap L_2 \\ \delta^*(q_0, w) \in F_\cup &\Leftrightarrow w \in L_1 \cup L_2 \\ \delta^*(q_0, w) \in F_\setminus &\Leftrightarrow w \in L_1 \setminus L_2 \\ \delta^*(q_0, w) \in F_\triangle &\Leftrightarrow w \in L_1 \triangle L_2 \ . \end{aligned}$$

Tím je konstrukce uvedených automatů dokázána.

5.10 Test ekvivalence konečných automatů

Každá cesta v přechodovém diagramu je w-cestou pro nějaké slovo w. Jestliže D je přechodový diagram, pak z toho plyne, že $L(D)$ je neprázdný právě tehdy, když v grafu diagramu existuje cesta z některého počátečního vrcholu do některého přijímajícího vrcholu. Existenci takové cesty lze zjistit efektivně například pomocí Dijkstrova algoritmu.

Test neprázdnosti jazyka definovaného pomocí přechodového diagramu lze použít i na jazyk definovaný konečným automatem. Z toho lze odvodit test ekvivalence pro konečné automaty. Nechť M_1 a M_2 jsou konečné automaty. Zřejmě platí

$$L(M_1) = L(M_2) \Leftrightarrow L(M_1) \triangle L(M_2) = \emptyset \ .$$

Platnost podmínky na pravé straně lze zjistit výpočtem konečného automatu pro $L(M_1) \triangle L(M_2)$ a testem neprázdnosti jazyka, který je reprezentován výsledným automatem.

Test ekvivalence konečných automatů lze získat také pomocí algoritmu pro redukci konečného automatu. Redukovaný automat je automat s minimálním počtem stavů pro daný jazyk a lze jej z libovolného konečného automatu získat efektivně. Navíc platí, že dva redukované automaty jsou ekvivalentní právě tehdy, když jsou izomorfní. Rozhodnout existenci izomorfizmu $f : Q_1 \to Q_2$ pro redukované automaty M_1 a M_2 s množinami stavů Q_1 a Q_2 a přechodovými funkcemi δ_1 a δ_2 lze efektivně. Izomorfizmus musí převádět počáteční stav M_1 na počáteční stav M_2 a kdykoli $f(q_1) = q_2$ pro $q_1 \in Q_1$ a $q_2 \in Q_2$, pak pro každý symbol $x \in \Sigma$ musí platit také $f(\delta_1(q_1, x)) = \delta_2(q_2, x)$.

6 Bezkontextové jazyky

Bezkontextová gramatika je popsána v Definici 3.2 a v Sekci 3.2 jsou uvedeny příklady jednoduchých bezkontextových gramatik. Jedná se o gramatiku, jejíž pravidla mají tvar $A \to \alpha$, kde $A \in V$ je neterminál a $\alpha \in (V \cup \Sigma)^*$. Jazyk, který je generován bezkontextovou gramatikou budeme nazývat bezkontextový.

Kromě odvození slova v podobě posloupnosti větných forem lze odvození v bezkontextové gramatice znázornit také ve formě derivačního stromu. V kořeni tohoto stromu je počáteční symbol S. Pokud je neterminál A v odvození přepsán pomocí pravidla $A \to b_1 b_2 \dots b_k$, kde $k \geq 0$ a pro $i = 1, \dots, k$ je $b_i \in \Sigma \cup V$, pak uzel derivačního stromu, který odpovídá danému výskytu neterminálu A, má k následníků, které odpovídají jednotlivým symbolům $b_1, \dots, b_k$ v tomto pořadí. V listech derivačního stromu jsou terminály, jejichž zřetězením při zachování pořadí vznikne odvozené slovo.

6.1 Zjednodušené aritmetické výrazy

Uvažme gramatiku G s množinou neterminálů $V = \{E, T, F\}$, množinou terminálů $\Sigma = \{0, 1, \dots, 9, +, -, *, /, (,)\}$, počátečním symbolem $S = E$ a s množinou pravidel

$$E \to E + T \mid E - T \mid +T \mid -T \mid T$$
$$T \to T * F \mid T/F \mid F$$
$$F \to (E) \mid 0 \mid 1 \mid \dots \mid 9.$$

Gramatika slouží k odvozování určitého omezeného typu výrazů bez proměnných, ale je možné ji rozšířit tak, že generuje právě všechny správně utvořené aritmetické výrazy. Označení jednotlivých neterminálů odpovídá tomu, jaké části výrazů jsou z nich odvoditelné, následovně:

E	výraz (expression)
T	člen součtu (term)
F	člen součinu (faktor) .

Například slovo $5-3$ je v této gramatice odvoditelné posloupností větných forem

$$E \Rightarrow E - T \Rightarrow T - T \Rightarrow F - T$$
$$\Rightarrow 5 - T \Rightarrow 5 - F \Rightarrow 5 - 3\ .$$

6.2 Vlastnosti bezkontextových jazyků

Bezkontextové jazyky jsou vlastním zobecněním regulárních jazyků. Věta 4.2 ukazuje příklad jazyka, který je bezkontextový, ale není regulární, a platí následující inkluze.

Věta 6.1 *Každý regulární jazyk je bezkontextový.*

Důkaz. Uvažme automat M určený parametry $Q, \Sigma, \delta, q_0, F$ a zkonstruujme bezkontextovou gramatiku G, která je s M ekvivalentní. Množina terminálů G je Σ a množina neterminálů G je množina stavů Q, přičemž předpokládáme, že $S = q_0$. Množinu pravidel gramatiky tvoří pravidla tvaru

$$q \to aq'$$

pro každé $q, q' \in Q$, $a \in \Sigma$ splňující $\delta(q, a) = q'$ a pravidla

$$q \to \varepsilon$$

pro všechna $q \in F$.

Lemma 6.2 *Jestliže $w \in \Sigma^*$ a $q, q' \in Q$, pak v gramatice G existuje odvození $q \stackrel{*}{\Rightarrow} wq'$ právě tehdy, když $q' = \delta^*(q, w)$.*

Důkaz. Indukcí podle délky w. Pro $w = \epsilon$ tvrzení platí, protože pro gramatiku G existuje odvození $q \stackrel{*}{\Rightarrow} q'$ právě tehdy, když $q = q'$, a této podmínce je ekvivalentní také $q' = \delta^*(q, \epsilon)$.

Nechť tvrzení platí pro slovo w. Dokažme je pro wa. Jestliže $q \stackrel{*}{\Rightarrow} waq'$, pak existuje q'' tak, že $q \stackrel{*}{\Rightarrow} wq'' \Rightarrow waq'$. Podle indukčního předpokladu je $q'' = \delta^*(q, w)$. Navíc, gramatika obsahuje pravidlo $q'' \to aq'$, a tedy podle její konstrukce platí $q' = \delta(q'', a) = \delta(\delta^*(q, w), a) = \delta^*(q, wa)$.

Nechť $q' = \delta^*(q, wa)$. Označme $q'' = \delta^*(q, w)$. Pak platí $q' = \delta(q'', a)$. Podle indukčního předpokladu je $q \stackrel{*}{\Rightarrow} wq''$. Z konstrukce gramatiky plyne, že $q'' \to aq'$ je jejím pravidlem, a tedy lze odvození wq'' prodloužit na $q \stackrel{*}{\Rightarrow} wq'' \Rightarrow waq'$. QED.

Nyní můžeme dokončit důkaz věty. Nechť $w \in L(M)$. Pak existuje $q \in F$ tak, že $q = \delta^*(q_0, w)$. Podle Lemmatu 6.2 tedy platí $S = q_0 \stackrel{*}{\Rightarrow} wq$. Protože $q \in F$, lze navíc k tomuto odvození doplnit krok $wq \Rightarrow w$. Je tedy $w \in L(G)$.

Nechť naopak $w \in L(G)$. Odvození w v G má nutně tvar $S = q_0 \stackrel{*}{\Rightarrow} wq \Rightarrow w$ pro vhodné q. Z toho plyne, že $q = \delta^*(q_0, w)$ a $q \in F$. Je tedy $w \in L(M)$. QED.

Definice 6.3 Gramatiku nazveme nevypouštějící, pokud pravá strana každého jejího pravidla je neprázdné slovo.

Věta 6.4 *Ke každé bezkontextové gramatice G_1 existuje nevypuštějící bezkontextová gramatika G_2 tak, že $L(G_2) = L(G_1) \setminus \{\varepsilon\}$.*

Důkaz. Nechť V_0 je množina neterminálů A takových, že $A \stackrel{*}{\Rightarrow} \varepsilon$. Pro každé pravidlo, které v pravé straně obsahuje k neterminálů z V_0, přidáme $2^k - 1$ pravidel, která vzniknou vypuštěním některé neprázdné podmnožiny těchto neterminálů

z pravé strany. Vzniklou gramatiku označme G'. Platí, že $L(G') = L(G_1)$, protože každé nové pravidlo lze nahradit posloupností pravidel původní gramatiky a žádná pravidla neubyla.

V druhém kroku odstraníme z G' pravidla tvaru $A \to \varepsilon$ a vzniklou gramatiku označíme G_2. Dokažme, že každé neprázdné slovo w, které je odvoditelné v gramatice G', je odvoditelné také v G_2. Pokud je v odvození w v G' použito pravidlo $A \to \varepsilon$, není neterminál A počáteční symbol gramatiky, protože w je neprázdné. Tento neterminál tedy byl do odvození zařazen jako symbol v pravé straně některého pravidla tvaru $B \to \alpha_1 A \alpha_2$. Protože $A \in V_0$, obsahuje gramatika G' také pravidlo $B \to \alpha_1\alpha_2$. Z odvození slova w odstraníme pravidla $B \to \alpha_1 A \alpha_2$ a $A \to \varepsilon$ a nahradíme je pravidlem $B \to \alpha_1\alpha_2$. Získáme odvození slova w v gramatice G', které obsahuje méně použití pravidel s prázdnou pravou stranou než původní odvození. Opakováním tohoto postupu získáme odvození w v G_2. Z toho plyne $L(G_2) \supseteq L(G') \setminus \{\varepsilon\}$. Protože G_2 negeneruje prázdné slovo a obsahuje pouze pravidla, která jsou pravidly G', je $L(G_2) \subseteq L(G') \setminus \{\varepsilon\}$. Celkem tedy dostáváme $L(G_2) = L(G') \setminus \{\varepsilon\} = L(G_1) \setminus \{\varepsilon\}$. QED.

Stejně jako regulární jazyky, jsou i bezkontextové jazyky uzavřeny na operace sjednocení, zřetězení a iterace.

Věta 6.5 *Jsou-li L_1, L_2 bezkontextové jazyky, pak také jazyky $L_1 \cup L_2$, $L_1 \cdot L_2$ a L_1^* jsou bezkontextové.*

Důkaz. Nechť G_i je gramatika pro jazyk L_i při obvyklém významu V_i, Σ, S_i a předpokládejme $V_1 \cap V_2 = \emptyset$. Pak gramatiku pro jazyk $L_1 \cup L_2$ lze získat následovně. Množina neterminálů je $\{S\} \cup V_1 \cup V_2$, kde S je nový počáteční symbol, a množina pravidel je sjednocení množin pravidel G_1 a G_2 s přidanými pravidly $S \to S_1$ a $S \to S_2$.

Jestliže místo posledních dvou pravidel gramatiky pro $L_1 \cup L_2$ přidáme pravidlo $S \to S_1S_2$, vznikne gramatika pro jazyk $L_1 \cdot L_2$.

Gramatika pro L_1^* vznikne z gramatiky G_1 přidáním pravidel $S \to S_1S$ a $S \to \epsilon$. Vzniklá gramatika generuje jazyk L_1^*, protože odvození využívající pouze přidaná pravidla umožňují odvodit právě všechny větné formy tvaru S_1^* a z nich lze odvodit právě všechna slova jazyka L_1^*. QED.

Věta 6.6 (rozšiřující tvrzení) *Je-li L bezkontextový jazyk a R je regulární, pak $L \cap R$ je bezkontextový jazyk.*

Důkaz. Nechť G je bezkontextová gramatika pro $L = L(G)$ a M je konečný automat pro $R = L(M)$. Zkonstruujeme bezkontextovou gramatiku G' pro $L \cap R$. Neterminály této gramatiky budou trojice $[q_1, A, q_2]$, kde $A \in V \cup \Sigma$, tedy A je neterminál nebo terminál G a q_1, q_2 jsou stavy M. Stav q_1 budeme nazývat

vstupním stavem dané trojice a q_2 výstupním stavem. Do G' zařadíme pravidla tří typů. Pravidla typu 1 vzniknou tak, že ke každému pravidlu

$$A \to \alpha_1 \alpha_2 \dots \alpha_k$$

gramatiky G vezmeme všechna pravidla tvaru

$$[q_1, A, q_2] \to [q_1, \alpha_1, r_1][r_1, \alpha_2, r_2] \dots [r_{k-2}, \alpha_{k-1}, r_{k-1}][r_{k-1}, \alpha_k, q_2]$$

kde r_i, $i = 1, \dots, k-1$ jsou libovolné stavy automatu M. Pravidla typu 2 jsou všechna pravidla tvaru

$$S \to [q_0, S, q] \ ,$$

kde $q \in F$. Pravidla typu 3 jsou všechna pravidla tvaru

$$[q_1, a, q_2] \to a \ ,$$

kde $q_1, q_2 \in Q$, $a \in \Sigma$ a platí $\delta(q_1, a) = q_2$.

Nechť $w \in L \cap R$, $w = a_1 a_2 \dots a_m$, a nechť $q_0, q_1, \dots, q_m$ je posloupnost stavů, kterými prochází M při přijímání slova w. Dále uvažme derivační strom pro w v gramatice G. Pro každé $i = 1, 2, \dots, m$ nahradíme v tomto stromu list a_i trojicí $[q_{i-1}, a_i, q_i]$ a doplníme jej odvozením $[q_{i-1}, a_i, q_i] \to a_i$, které patří do G' vzhledem k volbě stavů q_i. Všechny neterminály v tomto stromu postupně zdola nahoru nahradíme trojicemi $[p_1, A, p_2]$ následovně. Jestliže je uzel ohodnocen neterminálem A a všechny uzly, které jsou jeho následníci, jsou již ohodnoceny trojicí, pak daný uzel bude ohodnocen trojicí $[p_1, A, p_2]$, kde p_1 je vstupní stav nejlevějšího následníka a p_2 výstupní stav nejpravějšího následníka uvažovaného uzlu. Tím je zaručeno, že strom je po rozšíření derivačním stromem podle pravidel G'. V kořeni stromu je této úpravě trojice tvaru $[q_0, S, q_m]$. Protože $q_m \in F$, lze ke stromu přidat nový kořen ohodnocený S a odvodit z něj trojici $[q_0, S, q_m]$ pravidlem $S \to [q_0, S, q_m]$. Tím jsme zkonstruovali derivační strom w v G'.

Nechť naopak w je odvozeno nějakým derivačním stromem v G'. Terminály v G' lze odvodit pouze pravidly typu 3. Odstraňme ze stromu jeho listy. Novými listy se tak stanou trojice, ze kterých byly původní listy odvozeny pravidly typu 3.

Řekneme, že dvě trojice na sebe navazují, jestliže výstupní stav první trojice je roven vstupnímu stavu druhé trojice.

Lemma 6.7 *Jestliže derivační strom podle G' vznikne použitím některého pravidla typu 2 a libovolným počtem použití pravidel typu 1, pak trojice, které tvoří jeho listy, na sebe navazují.*

Důkaz. Indukcí podle počtu použití pravidel v odvození. Pokud strom vznikne pravidlem typu 2, je tvrzení pravdivé. Dále platí, že v pravidlech typu 1 jsou vstupní a výstupní stavy stanoveny tak, že rozšířením libovolného stromu, v němž

listy na sebe navazují, použitím jednoho pravidla typu 1 vznikne opět strom, v němž na sebe listy navazují. QED.

Podle lemmatu na sebe trojice v listech uvažovaného derivačního stromu navazují. Nechť jsou to trojice $[q_0, a_1, q_1], [q_1, a_2, q_2] \dots [q_{m-1}, a_m, q_m]$. Protože strom začíná některým pravidlem typu 2, je q_0 počáteční stav a q_m některý přijímající stav. Protože každá z uvažovaných trojic je levou stranou některého pravidla typu 3, je $\delta(q_{i-1}, a_i) = q_i$. Jinak řečeno, při čtení slova w prochází automat stavy $q_0, q_1, \dots, q_m$. Protože q_m je přijímající stav, je $w \in R$.

Dokažme ještě, že $w \in L$. V derivačním stromu pro w podle G' vypustíme kořen se symbolem S a listy s terminály. Zbyde strom složený z pravidel typu 1. Ve všech trojicích vypustíme vstupní a výstupní stav. Zbyde tedy pouze terminální nebo neterminální symbol a všechna pravidla, která ve stromu zbydou, jsou pravidla podle G. Získaný strom je derivační strom podle G, který odvozuje slovo w. Je tedy $w \in L$. QED.

6.3 Chomského normální forma

Definice 6.8 Řekneme, že bezkontextová gramatika je v *Chomského normální formě*, jestliže pravá strana každého jejího pravidla obsahuje buď dva neterminály nebo jeden terminál.

Gramatika v Chomského normální formě je nevypouštějící a tedy negeneruje prázdné slovo. K důkazu, že ke každé bezkontextové gramatice existuje gramatika v Chomského normální formě, která generuje stejný jazyk až na prázdné slovo, ukážeme nejprve dvě pomocné transformace gramatik.

Věta 6.9 *Ke každé bezkontextové gramatice existuje ekvivalentní gramatika, která neobsahuje pravidla typu $A \to B$, kde A, B jsou neterminály.*

Důkaz. Původní gramatiku označme G_1. Ke každému jejímu pravidlu typu $A \to B$, kde $A, B \in V$, přidejme do gramatiky všechna pravidla tvaru $A \to \alpha$, kde α není jeden neterminál a existují neterminály $B_1, B_2, \dots, B_k$ tak, že posloupnost

$$B \Rightarrow B_1 \Rightarrow \dots \Rightarrow B_k \Rightarrow \alpha$$

je odvozením v G_1. Přidaných pravidel je konečně mnoho a jejich nalezení lze provést efektivně. Vzniklou gramatiku označme G'. Gramatiku, která vznikne z G' vypuštěním pravidel typu $A \to B$, označme G_2.

Zřejmě platí $L(G') \supseteq L(G_1)$, protože G' vznikla z G_1 přidáním pravidel. Platí také $L(G') \subseteq L(G_1)$, protože použití každého nového pravidla lze nahradit posloupností původních pravidel. Je tedy $L(G') = L(G_1)$.

Protože G_2 vznikla z G' ubráním pravidel, je $L(G_2) \subseteq L(G')$. K důkazu $L(G_2) \supseteq L(G')$ ukážeme, že každý derivační strom v G' lze upravit na derivační strom v G_2.

Nechť strom podle G' obsahuje pravidlo $A \to B$. Jestliže pravidlo, které dále rozvíjí B, jej opět nahrazuje jedním neterminálem, sledujeme tuto posloupnost nahrazení až k neterminálu, který je nahrazen α, kde α není jeden neterminál. Podstrom mezi A a α odstraníme a nahradíme pravidlem $A \to \alpha$, které je pravidlem G_2. Touto operací jsme ve stromu snížili počet použití pravidel typu $A \to B$ při zachování odvozeného slova. Konečným počtem opakování této operace odstraníme všechna tato pravidla a zbývající strom bude stromem podle G_2.

Platí tedy $L(G_2) = L(G') = L(G_1)$ a gramatika G_2 splňuje podmínku z tvrzení věty. QED.

Věta 6.10 *Ke každé bezkontextové gramatice G_1 existuje bezkontextová gramatika G_2, která je v Chomského normální formě a splňuje $L(G_2) = L(G_1) \setminus \{\varepsilon\}$.*

Důkaz. Vezměme libovolnou bezkontextovou gramatiku G_1. Podle Věty 6.4 najdeme nevypouštějící gramatiku G' tak, že $L(G') = L(G_1) \setminus \{\varepsilon\}$. Bez újmy na obecnosti můžeme předpokládat, že G' splňuje ještě následující podmínky:

- Gramatika G' neobsahuje pravidla typu $A \to B$, $B \in V$.
- Pokud $A \to \alpha$ je pravidlem G' a $|\alpha| \geq 2$, pak $\alpha \in V^+$.

Pokud G' nesplňuje první podmínku, nahradíme ji ekvivalentní gramatikou podle Věty 6.9. Pokud nesplňuje druhou podmínku, přidáme do gramatiky ke každému terminálu a nový neterminál X_a a pravidlo $X_a \to a$ a ve všech pravidlech, jejichž pravá strana má délku alespoň 2, nahradíme každý terminální symbol a jemu příslušným neterminálem X_a. Získaná gramatika zřejmě generuje stejný jazyk jako gramatika G' a touto gramatikou nahradíme G'. Dále tedy můžeme předpokládat, že G' výše uvedené podmínky splňuje. Z toho plyne, že pro každé pravidlo $A \to \alpha$ v gramatice G' platí, že buď α je jeden terminál nebo $|\alpha| \geq 2$ a α obsahuje pouze neterminály.

Každé pravidlo G', které nesplňuje podmínky Chomského normální formy, je tvaru $A \to B_1 B_2 \dots B_k$, kde B_i jsou neterminály a $k \geq 3$. Ke každému takovému pravidlu doplníme skupinu pravidel

$$\begin{aligned} A &\to B_1 D_1 \\ D_1 &\to B_2 D_2 \\ &\vdots \\ D_{k-2} &\to B_{k-1} B_k \ , \end{aligned} \tag{7}$$

kde $D_1, D_2, \dots, D_{k-1}$ jsou nově přidané neterminály. Vzniklou gramatiku označme G''. Zřejmě platí $L(G') \subseteq L(G'')$. Jestliže se v derivačním stromu podle

G'' vyskytuje některé z pravidel některé skupiny přidaných pravidel (7), pak na toto pravidlo ve stromu bezprostředně navazují zbývající pravidla této skupiny, protože každý z neterminálů D_i je v levé straně právě jednoho pravidla a také v pravé straně právě jednoho pravidla. Tuto skupinu pravidel jako celek nahradíme pravidlem $A \to B_1 B_2 \dots B_k$. Opakováním tohoto postupu získáme z libovolného derivačního stromu v gramatice G'' derivační strom v gramatice G'. Je tedy $L(G') \supseteq L(G'')$, a tedy $L(G') = L(G'')$.

V dalším kroku vypustíme z G'' pravidla tvaru $A \to B_1 B_2 \dots B_k$ pro $k \geq 3$ a vzniklou gramatiku označme G_2. Zřejmě platí $L(G'') \supseteq L(G_2)$. Protože každé použití pravidla $A \to B_1 B_2 \dots B_k$, $k \geq 3$ lze nahradit použitím pravidel (7), je $L(G'') \subseteq L(G_2)$, a tedy $L(G'') = L(G_2)$.

Tím je věta dokázána, protože platí $L(G_2) = L(G'') = L(G') = L(G_1) \setminus \{\varepsilon\}$ a gramatika G_2 je v Chomského normální formě. QED.

6.4 Věta o nahrazení

Věta 6.11 *Pro každý bezkontextový jazyk L existuje přirozené číslo p tak, že pro každé slovo $z \in L$ délky alespoň p existují slova u, v, w, x, y tak, že platí*

(i) $z = uvwxy$

(ii) *délka vwx je nejvýše p*

(iii) *alespoň jedno ze slov v, x je neprázdné*

(iv) *pro každé $i \geq 0$ je $uv^i wx^i y \in L$.*

Důkaz. Nechť G je gramatika pro $L \setminus \{\varepsilon\}$ v Chomského normální formě a nechť k je počet jejích neterminálů. Zvolme $p = 2^k$.

Nechť $z \in L$ a $|z| \geq p$. Zvolme libovolný derivační strom pro z. Kdyby každá cesta v tomto stromu od kořene do některého listu obsahovala nejvýše k neterminálů, pak by strom obsahoval nejvýše 2^{k-1} listů, protože poslední neterminál v každé cestě má pouze jednoho následníka. To by byl spor s předpokladem, že délka z je alespoň p. Tedy existuje cesta s více než k neterminály. Vyberme nejdelší takovou cestu a uvažme na ní posledních $k+1$ neterminálů. Alespoň dva z nich jsou stejné. Nechť to jsou výskyty neterminálu A a odlišme je indexy. Blíže ke kořeni bude výskyt A_1, dále výskyt A_2. Pro vhodná slova u, v, w, x, y platí

$$\begin{aligned} S &\stackrel{*}{\Rightarrow} uA_1y \\ A_1 &\stackrel{*}{\Rightarrow} vA_2x \\ A_2 &\stackrel{*}{\Rightarrow} w \end{aligned} \tag{8}$$

Zřejmě platí $z = uvwxy$. Platí také $A_1 \stackrel{*}{\Rightarrow} vwx$. Protože žádná cesta z A_1 do listu neobsahuje více než $k+1$ neterminálů, je délka vwx nejvýše $p = 2^k$.

Předpokládejmě, že neterminál A_1 je ve stromu rozvinut pravidlem $A \to BC$. Pak uvažovaný strom obsahuje odvození

$$A_1 \Rightarrow BC \stackrel{*}{\Rightarrow} vA_2x$$

Jestliže neterminál A_2 je v podstromu rozvíjejícím C, obsahuje slovo v všechny symboly odvozené z neterminálu B a je tedy neprázdné. Analogicky, pokud je A_2 v podstromu rozvíjejícím B, je slovo x neprázdné.

Pro důkaz toho, že $uv^iwx^iy \in L$ pro $i = 0$, vyjdeme z odvození $S \stackrel{*}{\Rightarrow} uAy$ v (8). Neterminál A v pravé straně bude dále rozvinut podle odvození $A \stackrel{*}{\Rightarrow} w$, kterým bylo původně rozvíjeno A_2. Celkem získáme odvození

$$S \stackrel{*}{\Rightarrow} uAy \stackrel{*}{\Rightarrow} uwy$$

Pro důkaz toho, že $uv^iwx^iy \in L$ pro $i \geq 1$, vyjdeme opět z odvození (8). Pokud je $i = 1$, není třeba toto odvození upravovat. Pro $i > 1$ rozšíříme odvození tak, že odvození $A \stackrel{*}{\Rightarrow} vAx$ je použito několikrát po sobě. Například, pro $i = 2$ vznikne odvození

$$S \stackrel{*}{\Rightarrow} uAy \stackrel{*}{\Rightarrow} uvAxy \stackrel{*}{\Rightarrow} uvvAxxy \stackrel{*}{\Rightarrow} uvvwxxy$$

Tím je požadované tvrzení dokázáno. QED.

Důsledek 6.12 *Jazyk* $\{a^ib^ic^i \mid i \geq 1\}$ *není bezkontextový.*

Důkaz. Sporem. Předpokládejme, že $L = \{a^ib^ic^i \mid i \geq 1\}$ je bezkontextový a nechť p je číslo splňující podmínky věty o nahrazení. Vezměme slovo $z = a^pb^pc^p$. Protože délka z je alespoň p, víme z věty o nahrazení, že existují slova u, v, w, x, y, která splňují podmínky věty. Z podmínky (ii) plyne, že slova v a x mohou obsahovat nejvýše dva ze tří symbolů a, b, c. Z podmínky (iv) pro $i = 0$ plyne, že $uwy \in L$. To je spor, protože alespoň jeden ze symbolů a, b, c má v uwy méně výskytů než v z a alespoň jeden z těchto symbolů má stejný počet výskytů jako v z. QED.

Důsledek 6.13 *Jazyk* $\{ww \mid w \in \{a, b\}^*\}$ *není bezkontextový.*

Důkaz. Sporem. Kdyby jazyk z tvrzení věty byl bezkontextový, je podle Věty 6.6 bezkontextový i jazyk $L = \{ww \mid w \in \{a, b\}^*\} \cap (a^*b^*a^*b^*)$. Nechť p je číslo splňující podmínky věty o nahrazení pro jazyk L. Uvažme slovo $z = a^pb^pa^pb^p \in L$. Protože $|z| \geq p$, existuje rozklad $z = uvwxy$ podle věty o nahrazení. Protože pro každé $i \geq 1$ je $uv^iwx^iy \in L$, je každé ze slov v a x obsaženo v některém ze čtyř úseků slova $a^pb^pa^pb^p$ tvaru a^p nebo b^p. Protože platí $|vwx| \leq p$, jsou slova v a x obsažena buď ve stejném úseku nebo v sousedních úsecích. Slovo uv^2wx^2y je pak některého z tvarů $a^jb^pa^pb^p$, $a^pb^ja^pb^p$, $a^pb^pa^jb^p$, $a^pb^pa^pb^j$ kde $j > p$ nebo $a^{j_1}b^{j_2}a^pb^p$, $a^pb^{j_1}a^{j_2}b^p$, $a^pb^pa^{j_1}b^{j_2}$, kde $j_1, j_2 > p$. Protože slova žádného z těchto tvarů nepatří do L, je tento závěr ve sporu s podmínkou (iv) věty o nahrazení. QED.