FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO
Katedra informatiky

Juraj Minárik

Lindenmayerove systémy s interakciou

Diplomová práca

Školiteľ: Dr. Mária Pastorová

Bratislava 2006

Zadanie:

Naštudovať problematiku IL systémov, prehľadne ju spracovať, zosumarizovať, vyriešiť dielčie problémy, doplniť dôkazmi, predstaviť návrh IL generátora, pokúsiť sa ho vylepšiť a implementovať.

Poďakovanie:

Chcel by som poďakovať pani Dr. Márii Pastorovej za poskytnutie študijnych materiálov, za podporu pri vypracovávaní diplomovej práce a za pomoc pri riešení problémov.

Prehlasujem na svoju česť, že predkladanú diplomovú prácu som vypracoval samostatne a všetku použitú literatúru uvádzam v zozname.

Bratislava, marec 2006 _____

Abstrakt:

Práca skúma problematiku kontextových Lindenmayerových systémov ($(k,l)L$ Systémov). Porovnáva silu tried týchto systémov vzhľadom na rozloženie kontextu na pravý a ľavý a vzhľadom na dĺžku kontextu. Porovnáva triedy $(k,l)L$ Systémov s triedami Chomského hierarchie. Prezentuje vlastné dôkazy a vlastné algoritmy. V práci je predstavený návrh a optimalizácia generátora $(k,l)L$ systémov. Ako príloha k práci je implementácia navrhnutého $(k,l)L$ generátora.

Abstract:

Central objects of this thesis are the context Lindenmayer systems ($(k,l)L$ Systems). We comapare the generating power of these systems depending on partitioning of contexts to the left and right; and depending on the length of the context. We compare the classes of $(k,l)L$ systems with the Chomsky hierarchy. We present own proofs and algorithms. We introduce the design and optimalization of the generator of $(k,l)L$ systems. The implementation of the generator is presented in the addendum of the thesis.

Obsah

Úvod

V 60-tych rokoch minulého tisícročia sa pán A.Lindenmayer venoval modelovaniu vývoja bunkových organizmov a baktérií. V roku 1968 zadefinoval paralelné prepisovacie systémy, ktoré dostali po ňom meno Lindenmayerove systémy (skrátene L-systémy). Lindenmayerove systémy sa rýchlo ujali na modelovanie vývoja rastlín, organizmov, fraktálov, ale i zložitejších procesov.

0L-systémy odrazu paralelne prepisujú všetky jednotlivé znaky vetnej formy. V skutočných dejoch však často odvodenie elementu závisí aj od niekoľkých okolitých elementov (napríklad prítomnosť hormónu v okolí bunky). Preto boli definované IL-systémy, ktoré pri prepisovaní znaku berú do úvahy aj niekoľko okolitých znakov. Je veľa grafických systémov, ktoré pracujú na báze 0L a IL systémov s pridaním ďalších interpretačných značiek. Pomocou takýchto značiek môže mať jednoduchý L-systém široké použitie. Veľa príkladov možno nájsť v [7] a [8]. Preto je dôležité skúmať aj základné L-systémy.

Pre ľahšiu orientáciu čitateľa podrobnejšie rozoberieme obsah kapitol.

V kapitole 1 uvedieme niektoré označenia a definície, ktoré budeme v ďalších častiach používať.

V kapitole 2 zavedieme triedy Lindenmayerových systémov ako sú : $0L$,$(k,0)L$ a $(k,l)L$ systémy. Pri každej z týchto tried uvedieme príklady. V tejto časti sú obsiahnuté aj charakteristiky a tvrdenia o triedach jazykov $0L$ a $(k,0)L$ systémov. Keďže $(k,l)L$ systémy tvoria jednu z hlavných častí tejto práce, vety, algoritmy a charakteristiky sú prezentované v zvláštnej kapitole.

V kapitole 3 najskôr začneme jednoduchšími tvrdeniami, napríklad o uzavretosti tried $(k,l)L$ jazykov (podkapitola 3.1). Ďalej sa zameráme na porovnanie sily tried vzhľadom na dĺžku a rozloženie kontextu na ľavý a pravý (podkapitola 3.2). V závere podkapitoly zhrnieme prezentované výsledky do prehľadného diagramu. Poslednou časťou kapitoly sú vzťahy medzi triedami jazykov Chomského hierarchie a triedami jazykov $(k,l)L$ systémov. Zistené vzťahy sú takisto zakreslené v prehľadnom diagrame.

Kapitola 4 sa zameriava na návrh a implementáciu $(k,l)L$ generátora. Po úvode a prezentácii jednoduchšieho generátora pre $0L$ systémy v podkapitole 4.1 uvedieme formálnu definíciu generátora $(k,l)L$ systémov v kapitole 4.2. Táto podkapitola z veľkej časti čerpala hlavne z práce [4]. Po zadefinovaní generátora a prezentovaní niektorých príkladov uvedený návrh vylepšíme. V podkapitole 4.3 uvedieme možnosti optimalizácie tohto návrhu. Záver kapitoly je venovaný $(k,l)L$ generátoru, ktorý je implementovaný a optimalizovaný podľa predchádzajúceho návrhu. Samotný program sa nachádza na pribalenom CD-ROM.

Kapitola 1

Použité označenia

1.1 Front

Front znakov je postupnosť znakov, s ktorou môžeme vykonávať len tieto operácie:

- pridať znak na koniec
- odobrať znak zo začiatku (ak sa dá)

$F = \varepsilon$ je prázdny front. Neprázdny front F môžeme písať v tvare $F = aT$, kde a je znak, ktorý nazývame vrcholom frontu.

1.2 Text, vzorka

Ak budeme v nejakej postupnosti znakov hľadať určitú podpostupnosť, budeme túto postupnosť nazývať text a podpostupnosť vzorka.

1.3 Automat na rozpoznávanie vzorky v texte

Pre danú vzorku P skonštruujeme konečný automat s týmito vlastnosťami:

- automat rozpoznáva všetky výskyty vzorky P v zadanom texte
- automat pracuje efektívne, prezrie každý znak textu práve raz.

Najskôr zadefinujeme pomocnú funkciu σ nazvanú funkcia sufixu.

$\sigma : \Sigma^* \to \{0, 1, \ldots, |P|\}$ pričom $\sigma(x)$ je dĺžka najdlhšieho prefixu P, ktorý je sufixom x.

Teraz uvedieme konštrukciu automatu na rozpoznávanie danej vzorky P v texte.

$$A_p = (\{q_0, q_1, \ldots, q_{|P|}\}, \Sigma, \delta, q_0, \{q_{|P|}\}$$

pričom vzorka a text sú nad abecedou Σ a pre každý stav q_i a znak a

$$\delta(q_i, a) = q_{\sigma(P^{1,i}a)}$$

pričom $P^{1,i}$ označuje prvých i znakov vzorky P.

Táto konštrukcia je podrobnejšie opísaná a zdôvodnená v [1].

1.4 Automat rozpoznávajúci kontext

Automat rozpoznávajúci kontext $v_1v_2\ldots v_k$ je deterministický konečný automat

$$M_{v_1v_2\ldots v_k} = \left(K^{M_{v_1v_2\ldots v_k}}, \Sigma^{M_{v_1v_2\ldots v_k}}, \Delta^{M_{v_1v_2\ldots v_k}}, q_0^{M_{v_1v_2\ldots v_k}}, F^{M_{v_1v_2\ldots v_k}}\right)$$

pre ktorý platí:

$$\mathcal{L}(M_{v_1v_2\ldots v_k}) = \{wv_1v_2\ldots v_k \mid w \in (\Sigma^{M_{v_1v_2\ldots v_k}})^*\}$$

Pri konštrukcii takéhoto automatu postupujeme rovnako ako pri konštrukcii automatu na rozpoznávanie vzorky $v_1v_2\ldots v_k$ v texte.

1.5 Matica stavov

Nech M je konečná množina, ktorej prvkami sú automaty rozpoznávajúce kontext, nech $t : M \to \{1, 2, \ldots, |M|\}$ je bijektívna funkcia, ktorá čísluje prvky tejto množiny a nech $n \in \mathbb{N}$. Maticou stavov typu $n \times |M|$ nazývame maticu

$$\mathcal{A}_{M,t} = \begin{pmatrix} q_{0,1} & q_{0,2} & \cdots & q_{0,|M|} \\ q_{1,1} & q_{1,2} & \cdots & q_{1,|M|} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n-1,1} & q_{n-1,2} & \cdots & q_{n-1,|M|} \end{pmatrix}$$

pričom $q_{r,s} \in K^{t^{-1}(s)}$ pre $0 \leq r \leq n-1, 1 \leq s \leq |M|$.

1.6 Relácia $\sqsubseteq$

Nech $\mathcal{A}, \overline{\mathcal{A}}$ sú matice typu $n \times |M|$ nad tou istou množinou M, ktoré majú rovnakú číslovaciu funkciu t. Hovoríme, že matica $\overline{\mathcal{A}}$ vznikla z matice $\mathcal{A}$ podľa i-teho riadku a znaku a (píšeme $\mathcal{A} \overset{i,a}{\sqsubseteq} \overline{\mathcal{A}}$), ak platí

$$\mathcal{A} = \begin{pmatrix} q_{0,1} & \cdots & q_{0,|M|} \\ \vdots & \ddots & \vdots \\ q_{n-1,1} & \cdots & q_{n-1,|M|} \end{pmatrix} \quad \text{a} \quad \mathcal{B} = \begin{pmatrix} Q_{0,1} & \cdots & Q_{0,|M|} \\ \vdots & \ddots & \vdots \\ Q_{n-1,1} & \cdots & Q_{n-1,|M|} \end{pmatrix}$$

a pre všetky r, s $0 \leq r \leq n-1, 1 \leq s \leq |M|$ platí:

$$Q_{r,s} = \begin{cases} q_{r,s} & \text{ak } r \neq i \\ \Delta^{t^{-1}(s)}(q_{r,s}, a) & \text{ak } r = i \end{cases}$$

Hovoríme, že matica $\overline{\mathcal{A}}$ vznikla z matice $\mathcal{A}$ podľa i-teho riadku a slova $w = a_1 a_2 \ldots a_z$ (píšeme $\mathcal{A} \overset{i,w}{\sqsubseteq} \overline{\mathcal{A}}$), ak existujú matice $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_z$ také, že

$$\mathcal{A} \overset{i,a_1}{\sqsubseteq} \mathcal{A}_1 \overset{i,a_2}{\sqsubseteq} \mathcal{A}_2 \ldots \mathcal{A}_{z-1} \overset{i,a_z}{\sqsubseteq} \mathcal{A}_z = \overline{\mathcal{A}}$$

Ďalej hovoríme, že matica $\overline{\mathcal{A}}$ vznikla z matice $\mathcal{A}$ podľa slova w (píšeme $\mathcal{A} \overset{w}{\sqsubseteq} \overline{\mathcal{A}}$), ak existujú matice $\mathcal{A}_0, \mathcal{A}_2, \ldots, \mathcal{A}_{n-1}$ také, že

$$\mathcal{A} \overset{i,w}{\sqsubseteq} \mathcal{A}_0 \overset{i,w}{\sqsubseteq} \mathcal{A}_1 \ldots \mathcal{A}_{n-2} \overset{i,w}{\sqsubseteq} \mathcal{A}_{n-1} = \overline{\mathcal{A}}$$

1.7 Matica počiatočných stavov

Nech $\mathcal{A}_{M,t}$ je matica stavov typu $n \times |M|$.

$$\mathcal{A}_{M,t} = \begin{pmatrix} q_0^{t^{-1}(1)} & q_0^{t^{-1}(2)} & \ldots & q_0^{t^{-1}(M)} \\ q_0^{t^{-1}(1)} & q_0^{t^{-1}(2)} & \ldots & q_0^{t^{-1}(M)} \\ \vdots & \vdots & \ddots & \vdots \\ q_0^{t^{-1}(1)} & q_0^{t^{-1}(2)} & \ldots & q_0^{t^{-1}(M)} \end{pmatrix}$$

1.8 Iniciálna matica stavov

Nech $\mathcal{A}_{M,t}, \overline{\mathcal{A}}_{M,t}$ sú matice stavov typu $n \times |M|$ a nech $\mathcal{A}_{M,t}$ je matica počiatočných stavov. Iniciálnou maticou stavov nazývame maticu $\overline{\mathcal{A}}_{M,t}$ takú, že $\mathcal{A}_{M,t} \overset{g^k}{\sqsubseteq} \overline{\mathcal{A}}_{M,t}$.

1.9 Asymptotické horné ohraničenie

Pre funkciu $g(n)$ definujeme nasledujúcu množinu:

$$O(g(n)) = \{f(n) \mid \exists c, n_0 \in \mathbb{N} : \forall n \in \mathbb{N}, n > n_0 : 0 \leq f(n) \leq cg(n)\}$$

O notácia dáva horné ohraničenie funkcie až na konštantný faktor. Zápis $f(n) = O(g(n))$ znamená, že funkcia $f(n)$ je prvkom množiny $O(g(n))$.

Kapitola 2

Lindenmayerove systémy

Lindenmayerove systémy (L systémy) sa od frázových gramatík líšia v nasledovných aspektoch:

- všetky znaky sú terminálne, tj. každá vygenerovaná vetná forma patrí do jazyka
- prepisovacie pravidlá sa v jednom kroku odvodenia aplikujú paralelne na všetky znaky prepisovaného slova
- namiesto počiatočného symbolu je axioma, ktorá môže mať viac znakov

Ako jedny z prvých paralelných prepisovacích systémov sa pomerne dobre usadili. Používajú sa dodnes pri simuláciách fraktálov, simuláciách rastu rastlín, buniek a iných prosecov.

2.1 0L systémy

OL systémy sú základné systémy z rodiny Lindenmayerových systémov. Číslica 0 znamená, že pri prepisovaní neberieme do úvahy kontext prepisovaného znaku, ako je to napríklad pri $(k,l)L$ systémoch, ktoré sú popísané neskôr.

Definícia 2.1.1. $0L$ systém S je trojica $S = (\Sigma, P, w_0)$, pričom

Σ je konečná neprázdna abeceda systému
w_0 $\in \Sigma^+$ je axióma
P $\subset \Sigma \times \Sigma^*$ je konečná úplná množina pravidiel
(Úplná znamená, že pre každé $a \in \Sigma$ existuje pravidlo $a \to w \in P$)

Označenie: Pravidlo $(a, \alpha) \in P$ píšeme v tvare $a \to \alpha$.

Definícia 2.1.2. Nech $w = a_1 a_2 \dots a_m \in \Sigma^m$. Slovo $\overline{w} = \alpha_1 \alpha_2 \dots \alpha_m \in \Sigma^*$ nazývame odvodenie zo slova w (píšeme $w \Rightarrow \overline{w}$) práve vtedy, keď pre všetky $i = 1, 2, \dots, m$
$a_i \to \alpha_i \in P$.

Definícia 2.1.3. Slovo $\overline{w}$ je odvodené zo slova w na n krokov (píšeme $w \Rightarrow^n \overline{w}$), ak existuje postupnosť slov $u_0, u_1, \ldots, u_n$ taká, že $u_0 = w, u_n = \overline{w}$ a $u_0 \Rightarrow u_1, u_1 \Rightarrow u_2, \ldots, u_{n-1} \Rightarrow u_n$.

Definícia 2.1.4. Pre $n \in \mathbb{N}$ $L_n(S) = \{w \mid w_0 \Rightarrow^n w\}$ je množina slov, ktoré sú odvodené z axiómy práve na n krokov.

Poznámka: $L_0(S) = \{w_0\}$, reflexívny a tranzitívny uzáver relácie $\Rightarrow$ označíme $\Rightarrow^*$.

Definícia 2.1.5. Jazyk generovaný $0L$ systémom S je množina

$$L(S) = \{w | w_0 \Rightarrow^* w\}$$

Označenie: Ak pre všetky $a \in \Sigma$ existuje práve jedno pravidlo $a \to \alpha$, tak hovoríme o *deterministickom*, inak o *nedeterministickom* 0L systéme

Uvedieme príklady deterministického aj nedeterministického $(k, l)L$ systému.

Príklad 2.1.1. Majme deterministický 0L systém $S = (\{a, b, c\}, P, bc)$, kde $P = \{a \to aa, b \to cab, c \to \varepsilon\}$

Prvých niekoľko slov tohto systému vyzerá nasledovne:

$$\begin{aligned} w_0 &= bc \\ w_1 &= cab \\ w_2 &= aacab \\ w_3 &= aaaaaacab \\ &\ldots \end{aligned}$$

takže

$$\begin{aligned} L_0(S) &= \{bc\} \\ L_n(S) &= \{a^{2^n-2}cab | n \geq 1\} \\ \text{a teda} & \\ L(S) &= \{bc\} \bigcup \{a^{2^n-2}cab \mid n \geq 1\} \end{aligned}$$

Príklad 2.1.2. Majme nedeterministický 0L systém $S = (\{a\}, P, a)$, kde $P = \{a \to aa, a \to aaa\}$

Ľahko zistíme, že

$$\begin{aligned} L_0(S) &= \{a\} \\ L_1(S) &= \{aa, aaa\} \\ L_2(S) &= \{a^4, a^5, a^6, a^7, a^8, a^9\} \\ L_3(S) &= \{a^i \mid 8 \leq i \leq 27\} \\ &\ldots \\ L_n(S) &= \{a^i \mid 2^n \leq i \leq 3^n\} \\ L(S) &= \{a^i \mid i \geq 1\} \end{aligned}$$

Najdlhšie možné slovo, ktoré možno po i krokoch odvodenia vygenerovať, je slovo $w_{max} = 3^i$. (Na odvodenia použijeme iba pravidlo $a \rightarrow aaa$.) Naopak najkratšie možné slovo, ktoré možno po i krokoch odvodenia vygenerovať, je $w_{min} = 2^i$. (Na odvodenia použijeme iba pravidlo $a \rightarrow aa$.) Ak po i krokoch odvodenia vzniklo slovo $w = a^j$ a $j > 2^i$, potom mohlo po i krokoch odvodenia vzniknúť aj slovo $\overline{w} = a^j - 1$. Nastane to tak, že v jednom prípade nepoužijeme pravidlo $a \rightarrow aaa$, ale $a \rightarrow aa$. Teda w je ľubovolne dlhé slovo dlhšie ako 2^i a kratšie ako 3^i, teda $L(S) = \{a\} \cup \bigcup_{n>0} \{a^i | 2^n \leq i \leq 3^n\} = \{a^i | i \geq 0\}$

Definícia 2.1.6. $\mathcal{AFL}$ (Abstract Family of Languages) je každá trieda jazykov obsahujúca nejaký neprázdny jazyk a uzavretá na $\cup, \cdot, +, h_\varepsilon, h^{-1}, \cap R$.

Veta 2.1.1. $\mathcal{L}_{0L}$ je anti-$\mathcal{AFL}$ (tj. nie je uzavretá na žiadnu $\mathcal{AFL}$ operáciu).

Dôsledok 2.1.1. $\mathcal{L}_{0L}$ nie je uzavretá na substitúciu, $\cap$, zobrazenie a-prekladačom ani na ^{C} (komplement)

Veta 2.1.2. $\mathcal{L}_{0L}$ je uzavretá na zrkadlový obraz.

Trieda jazykov $\mathcal{L}_{0L}$ ako anti-$\mathcal{AFL}$ v abstraktnej teórií jazykov znamená veľmi zlé vlastnosti. Trochu lepšia je situácia pri $0L$-jazykoch s jednopísmenkovou abecedou. Vyslovíme niekoľko známych tvrdení.

Veta 2.1.3. Nech $L \in \mathcal{L}(0L)$ je jazyk $L \subseteq a^*$, potom $L^* \in \mathcal{L}(0L)$.

Poznámka: Keď $L \subseteq a^*$ je konečný jazyk, tak $L^* \in \mathcal{L}(0L)$.

Teraz si ukážeme niekoľko viet hovoriacich o vzťahu triedy $0L$ jazykov s triedami jazykov Chomského hierarchie.

Veta 2.1.4. Každá z tried $\mathcal{R}$, $\mathcal{L}_{CF} - \mathcal{R}$, $\mathcal{L}_{CS} - \mathcal{L}_{CF}$ obsahuje jazyky, ktoré sú $\mathcal{L}_{OL}$, aj jazyky, ktoré nie sú $\mathcal{L}_{OL}$.

Dôkaz: $0L$-jazyky v jednotlivých triedach sú:

1. $\{a\} \in \mathcal{R}$
2. $\{a^i b a^i | i \geq 1\} \in \mathcal{L}_{CF} - \mathcal{R}$
3. $\{a^{2^n} | n \geq 0\} \in \mathcal{L}_{CS} - \mathcal{L}_{CF}$

Jazyky v príslušných triedach, ktoré nie sú $0L$:

1. $\{a, a^2\} \in \mathcal{R}$
2. $\{a^i b^i | i \geq 1\} \in \mathcal{L}_{CF} - \mathcal{R}$
3. $\{a^{2^n} | n \geq 0\} \cup \{a^3\} \in \mathcal{L}_{CS} - \mathcal{L}_{CF}$

□

Veta 2.1.5. Každý jednopísmenkový $0L$-jazyk, ktorý obsahuje slovo ε, je regulárny.

Lema 2.1.1. Každý jazyk generovaný $0L$ systémom, v ktorom $a \to a \in P$ pre každé $a \in \Sigma$, je bezkontextový.

Lema 2.1.2. Každý jazyk $L \in \mathcal{L}_{CF}$ je tvaru $L' \cap R$, kde $L' \in \mathcal{L}_{0L}$ a $R \in \mathcal{R}$.

Lema 2.1.3. Nech $S = (\Sigma, P, w_0)$ je $0L$ systém, potom pre všetky slová $w \in L(S)$ existuje odvodenie, ktoré používa maximálne $k|w|$ priestoru, kde k je konštanta závislá od S.

Veta 2.1.6 (o lineárnom priestore). Nech A je turingov stroj taký, že existuje k také, že pre každé slovo w tento TS použije pri práci na w najviac $k|w|$ políčok. Potom $L(A) \in \mathcal{L}_{ECS}$.

Veta 2.1.7. Platí nasledujúci vyťah tried: $\mathcal{L}_{0L} \subseteq \mathcal{L}_{ECS}$

Uvedené tvrdenia s dôkazmi je možné nájsť v [3].

2.2 (k,0)L systémy (s jednostrannou interakciou)

$(k,0)L$ systémy na rozdiel od $0L$ systémov berú pri prepisovaní znaku do úvahy aj ľavý kontext dĺžky k, tj. k znakov pred prepisovaným znakom. Podobne pri prepisovaní znaku v $(0,l)L$ systéme berú do úvahy pravý kontext dĺžky l, teda l znakov po prepisovanom znaku.

Definícia 2.2.1. $(k,0)L$ systém S je štvorica $S = (\Sigma, g, P, w_0)$ s vlastnosťami:

Σ	je abeceda systému
g	$\notin \Sigma$ je špeciálny symbol
w_0	$\in \Sigma^+$ je axiôma
P	$\subset \left(\bigcup_{i=0}^{k} \{g\}^{k-i}\Sigma^i\right) \times \Sigma \times \Sigma^*$ je konečná množina pravidiel.

Pravidlo $(v,a,\alpha) \in P$ píšeme v tvare $v<a> \to \alpha$, $v<a>$ nazývame ľavá a α pravá strana pravidla. Špeciálny symbol g používame na rozpoznanie začiatku a na konca slova. Znaky na začiatku a konci slova sa prepisujú, akoby bolo pred a za slovom dostatočne veľa symbolov g.

Analogicky sa definuje aj $(0,l)L$ systém.

Definícia 2.2.2. Nech $w = a_1a_2 \dots a_i \dots a_m$ je slovo nad Σ. Slovo $\overline{w} = \alpha_1\alpha_2 \dots \alpha_i \dots \alpha_m$ je odvodené (resp. generované) z w, píšeme $w \Rightarrow \overline{w}$ práve vtedy, keď pre všetky $i = 1,2,\dots,m$ $a_{i-k}a_{i-k+1} \dots a_{i-1}<a_i> \to \alpha_i \in P$. V tomto pravidle položíme $a_j = g$ pre $j \leq 0$ alebo $j \geq m$.

Definícia 2.2.3. Slovo $\overline{w}$ je odvodené zo slova w na n krokov, píšeme $w \Rightarrow^n \overline{w}$, ak existuje postupnosť slov $u_0, u_1, \dots, u_n$ taká, že $u_0 = w, u_n = \overline{w}$ a $u_0 \Rightarrow u_1, u_1 \Rightarrow u_2, \dots, u_{n-1} \Rightarrow u_n$.

Definícia 2.2.4. Reflexívny a tranzitívny uzáver relácie $\Rightarrow$ označíme $\Rightarrow^*$.

Definícia 2.2.5. Pre $n \in \mathbb{N}$ $L_n(S) = \{w \mid w_0 \Rightarrow^n w\}$ je množina slov, ktoré sú odvodené z axiómy práve na n krokov.

Poznámka: $L_0(S) = \{w_0\}$.

Definícia 2.2.6. Jazyk generovaný $(k,0)L$ systémom S je množina

$$L(S) = \{w | w_0 \Rightarrow^* w\}$$

Označenie: Ak pre všetky $v<a>$ existuje práve jedno α také, že $v<a> \to \alpha \in P$, tak hovoríme o *deterministickom*, inak o *nedeterministickom* (k,0)L systéme

Uvedieme príklad nedeterministického $(k,l)L$ systému.

Príklad 2.2.1. Majme nedeterministický $(1,0)L$ systém $S = (\{a,b,c\}, g, P, c)$, kde tabuľka pravidiel P vyzerá nasledovne:

$g<a>\to a^2$	$g<b>\to b$	$g<c>\to ba \mid a^2$
$a<a>\to a^2$	$a<b>\to b$	$a<c>\to ba \mid a^2$
$b<a>\to a$	$b<b>\to b$	$b<c>\to ba \mid a^2$
$c<a>\to a^2$	$c<b>\to b$	$c<c>\to ba \mid a^2$

Prvých niekoľko odvodených slov tohto systému vyzerá nasledovne:

$$\begin{aligned} L_0 &= \{c\} \\ L_1 &= \{ba\} \cup \{a^2\} \\ L_2 &= \{ba\} \cup \{a^4\} \\ L_3 &= \{ba\} \cup \{a^8\} \\ &\dots \end{aligned}$$

ľahko zistíme, že systém generuje jazyk

$$L(S) = \{c, ba\} \bigcup \{a^{2^n} \mid n > 0\}$$

Ukážeme si základné tvrdenia platiace pre triedu $(k,0)L$ jazykov a porovnáme ju s triedami jazykov chomského hierarchie.

Lema 2.2.1. $\forall k > 0 \mathcal{L}(0L) \subsetneq \mathcal{L}((k,0)L)$

Dôkaz: Ukážeme, že $0L$ systém vieme jednoducho prepísať na $(k,0)L$ systém pre ľubovolné $k > 0$. Nech je pôvodný $0L$ systém $S = (\Sigma, w_0, P)$, potom $(k,0)L$ systém $\overline{S} = (\Sigma, g, w_0', P')$ bude definovaný nasledovne:

$$\Sigma = \Sigma$$

$$w_0' = w_0$$

$$P' = \{w<a> \to \alpha \mid a \to \alpha \in P, w \in \bigcup_{i=0}^{k} \{g\}^{k-i}\Sigma^i\}$$

Teda máme k dispozícií kontext, ale pri prepisovaní symbolu ho ignorujeme.

□

Príklad 2.2.2. Majme daný $0L$ systém $S = (\{a,b\}, \{a \to b, b \to ab\}, a)$. Zostrojíme k nemu $(2,0)L$ systém $S' = (\{a,b\}, \{gg<a> \to b, ga<a> \to b, aa<a> \to b, gb<a> \to b, bb<a> \to b, gg<b> \to ab, ga<b> \to ab, aa<b> \to ab, gb<b> \to ab, bb<b> \to ab\}, a)$. Ľahko sa presvedčíme, že oba systémy generujú ten istý jazyk.

Lema 2.2.2. Pre všetky $k \geq 0$ platí $\mathcal{L}((k,0)L) \subsetneq \mathcal{L}((k+1,0)L)$

Dôkaz: Dôkaz lemy je obdobný ako dôkaz silnejšej lemy 3.2.1 pre $l = 0$ v nasledujúcej kapitole.

□

Príklad 2.2.3. Ukážeme systém S generujúci jazyk $L \in \mathcal{L}((2,0)L) - \mathcal{L}((1,0)L))$.

$$S = (\{a\}, g, \{gg < a > \rightarrow a, ga < a > \rightarrow a, aa < a > \rightarrow aa\}, aaa)$$

Systém je deterministický a prvé 3 kroky odvodenia vyzerajú nasledovne:

$$aaa \Rightarrow aaa^2 \Rightarrow aaa^2a^2 \Rightarrow aaa^2a^2a^2a^2$$

Ľahko sa presvedčíme, že vygenerovaný jazyk bude $\{a^{2^n+2} \mid n \geq 0\}$

Dôsledok 2.2.1. Pre všetky $k, m \geq 0$ platí $\mathcal{L}((k,0)L) \subsetneq \mathcal{L}((m,0)L) \longleftrightarrow k < m$

Veta 2.2.1. Trieda jazykov $\mathcal{L}((k,0)L)$ nie je uzavretá na zjednotenie pre konkrétne k. Navyše existujú $k_1, k_2 > 0$ a jazyky $L_1 \in \mathcal{L}((k_1,0)L)$ a $L_2 \in \mathcal{L}((k_2,0)L)$, že neexistuje $k > 0$ pre ktoré $L_1 \cup L_2 \in \mathcal{L}((k,0)L)$.
Dôkaz: V ďalšej kapitole si dokážeme silnejšie tvrdenie - veta 3.1.1.

□

Veta 2.2.2. Každý konečný netriviálny jazyk patrí do triedy $\mathcal{L}((1,0)L)$. (netriviálny v zmysle má aspoň jedno slovo dĺžky aspoň 1)

Dôkaz: Nech $L = \{w_1, w_2, \ldots, w_n\}$ je konečný jazyk obsahujúci aspoň jedno slovo dĺžky aspoň 1. Zostrojíme k nemu $(1,0)L$ systém $S = (\Sigma, P, w_0)$, pričom:
Bez ujmy na všeobecnosti nech $|w_1| \geq 1$ a nech a je prvý symbol slova w_1.

$\Sigma = \{$ všetky symboly slov w_1 až $w_n\}$

$w_0 = w_1$

množinu P zostrojíme nasledovne:
Pre všetky $i = 1, \ldots, n$ vytvoríme pravidlá $g<a> \rightarrow w_i \in P$.
Pre všetky $b, c \in \Sigma$ vytvoríme pravidlá $b<c> \rightarrow \varepsilon$.

To znamená, že prvý znak exiomy prepíšeme na ľubovolné zo slov jazyka L a zvyšné znaky zmažeme.

□

2.3 (k,l)L systémy (s obojstrannou interakciou)

$(k,l)L$ systémy sa od $0L$ a $(k,0)L$ systémov líšia v tom, že pri prepisovaní znaku berú do úvahy ľavý aj pravý kontext prepisovaného znaku. Trieda jazykov $(k,l)L$ sa často nazýva aj ako trieda IL jazykov, teda jazykov s interakciou. $(k,l)L$ systém je tiež často označovaný ako IL systém - Lindenmayerov systém s interakciou. Trieda jazykov $(k,l)L$ je mohutnejšia ako triedy jazykov $0L$ a $(k,0)L$, ako si ukážeme v ďalšej kapitole. Tak isto si ukážeme, že rozdelenie na ľavý a pravý kontext nie je (až na okrajové prípady) podstatné.

Definícia 2.3.1. $(k,l)L$ systém S je štvorica $S = (\Sigma, g, P, w_0)$ s vlastnosťami:

Σ je abeceda systému
$g \notin \Sigma$ je špeciálny symbol
$w_0 \in \Sigma^+$ je axiôma
$P \subset \left(\bigcup_{i=0}^{k} \{g\}^{k-i}\Sigma^i\right) \times \Sigma \times \left(\bigcup_{j=0}^{l} \Sigma^j\{g\}^{l-j}\right) \times \Sigma^*$ je konečná množina pravidiel.

Pravidlo $(v_1, a, v_2, \alpha) \in P$ píšeme v tvare $v_1<a>v_2 \to \alpha$, $v_1<a>v_2$ nazývame ľavá a α pravá strana pravidla.

Definícia 2.3.2. Nech $w = a_1a_2 \ldots a_i \ldots a_m$ je slovo nad Σ. Slovo $\overline{w} = \alpha_1\alpha_2 \ldots \alpha_i \ldots \alpha_m$ je odvodené (resp. generované) z w, píšeme $w \Rightarrow \overline{w}$ práve vtedy, keď pre všetky $i = 1, 2, \ldots, m$ $a_{i-k}a_{i-k+1} \ldots a_{i-1}<a_i>a_{i+1} \ldots a_{i+l} \to \alpha_i \in P$. V tomto pravidle položíme $a_j = g$ pre $j \leq 0$ alebo $j \geq m$.

Definícia 2.3.3. Slovo $\overline{w}$ je odvodené zo slova w na n krokov, píšeme $w \Rightarrow^n \overline{w}$ ak existuje postupnosť slov $u_0, u_1, \ldots, u_n$ taká, že $u_0 = w, u_n = \overline{w}$ a $u_0 \Rightarrow u_1, u_1 \Rightarrow u_2, \ \ldots, u_{n-1} \Rightarrow u_n$.

Definícia 2.3.4. Reflexívny a tranzitívny uzáver relácie $\Rightarrow$ označíme $\Rightarrow^*$.

Definícia 2.3.5. Pre $n \in \mathbb{N}$ $L_n(S) = \{w \mid w_0 \Rightarrow^n w\}$ je množina slov, ktoré sú odvodené z axiómy práve na n krokov.

Poznámka: $L_0(S) = \{w_0\}$.

Definícia 2.3.6. Jazyk generovaný $(k,l)L$ systémom S je množina

$$L(S) = \{w | w_0 \Rightarrow^* w\}$$

Označenie: Ak pre všetky $v_1<a>v_2$ existuje práve jedno α také, že $v_1<a>v_2 \to \alpha \in P$, tak hovoríme o *deterministickom*, inak o *nedeterministickom* (k,l)L systéme.

Príklad 2.3.1. Majme deterministický $(2,1)L$ systém $S = (\{a\}, g, P, aaaa)$, kde

$$P = \{gg<a>a \ \rightarrow \ a, \quad ga<a>a \rightarrow a, \quad aa<a>a \ \rightarrow \ a^2, \quad aa<a>g \ \rightarrow \ a\}$$

Prvé 3 kroky odvodenia vyzerajú nasledovne:

$$aaaa \Rightarrow aaa^2a \Rightarrow aaa^2a^2a \Rightarrow aaa^2a^2a^2a^2a$$

Prvých niekoľko slov tohto systému vyzerá nasledovne:

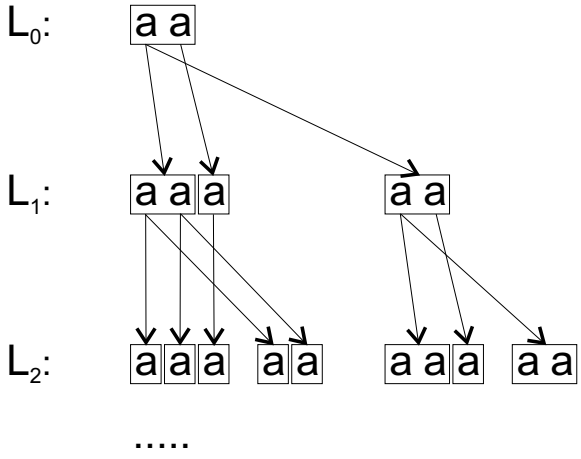
$$\begin{aligned} L_0 &= \{aaaa\} \ &= \{a^4\} \\ L_1 &= \{aaa^2a\} &= \{a^5\} \\ L_2 &= \{aaa^4a\} &= \{a^7\} \\ L_3 &= \{aaa^8a\} &= \{a^{11}\} \\ &\ldots \end{aligned}$$

Ľahko zistíme, že systém generuje jazyk

$$L(S) = \{a^{2^n+3}\}$$

Príklad 2.3.2. Majme nedeterministický $(1,2)L$ systém $S = (\{a\}, g, P, aa)$, kde $P = \{g<a>ag \rightarrow aa, g<a>aa \rightarrow a, a<a>ag \rightarrow a, a<a>gg \rightarrow a \mid \varepsilon\}$

Pravidlá $g<a>gg \rightarrow a$ a $a<a>aa \rightarrow a$ sa počas výpočtu nikdy nepoužijú, ale kvôli úplnosti pravidiel je vhodné ich uviesť.



Pomerne jednoducho sa presvedčíme, že systém generuje jazyk $L = \{aa, aaa\}$

Jednými z výsledkov tejto práce sú aj tvrdenia a dôkazy viet o triedach $(k,l)L$ systémov, preto ich rozoberieme podrobne v ďalšej kapitole.

Kapitola 3

Vety o $(k,l)L$ systémoch

3.1 Základné tvrdenia o triedach $(k,l)L$ jazykov

Ukážeme širšiu neuzavretosť tried $\mathcal{L}((k,l)L)$ na zjednotenie.

Veta 3.1.1. Existujú také jazyky L_1 a L_2 z triedy $\mathcal{L}(0L)$, pre ktoré *neexistujú* konštanty $k,l \geq 0$ a $(k,l)L$ systém S generujúci jazyk $L = L_1 \cup L_2$.

Dôkaz: Nech $L_1 = \{a^{2^n}\}$ a $L_2 = \{a^{3^n}\}$. Oba jazyky L_1 aj L_2 sú v triede jazykov $\mathcal{L}(0L)$. Pokúsme sa zostrojiť systém $S = (\Sigma, g, P, w_0)$ generúci jazyk $L = L_1 \cup L_2$.

1. S' musí byť deterministický.
   Dokážeme sporom. Nech by v P bolo pravidlo $u_1<a>u_2 \to a^x|a^y$ a nech $x < y$. Nech $a^m \in L$ a jedno a sa prepísalo pomocou $u_1<a>u_2 \to a^x$. Potom aj a^{m-x+y} patrí do L. Teda symbol a sa prepísal pravidlom $u_1<a>u_2 \to a^y$ a nie $u_1<a>u_2 \to a^x$. $x-y$ je konštanta. Lenže vzdialenosti slov (vzhľadom na dĺžku) sa stále zväčšujú a teda niekdedy určite prekročia rozdiel $x-y$. Vytvorili by sa aj slová nepatriace do L. Teda S musí byť deterministický.

2. Nech P'_g sú pravidlá, v ktorých kontexte sa nachádzajú špeciálne znaky g, P'_a sú ostatné pravidlá - teda pravidlo $a^k<a>a^l \to a^r$.
   Dĺžku slova v systéme S' po n krokoch odvodenia možno popísať rekurenciou

   $l_0 = |w_0|$

   $l_n = (l_{n-1} - p)r + q$

   kde

   p - je celková dĺžka kontextu

   q - počet písmen, ktoré vyrobia pravidlá P'_g

   r - konštanta z pravidla $a^k<a>a^l \to a^r$, teda počet symblolov a, ktoré toto pravidlo vytvorí

Po vyriešení rekurencie dostávame vzťah pre dĺžku slova po n-tom kroku odvodenia:

$$l_n = (|w_0| + \tfrac{q-pr}{r-1})r^n + \tfrac{pr-q}{r-1}$$

Ak chceme vytvoriť slová rádovo dlhé 2^n, musí platiť $r = 2$, ak chceme vytvoriť slová rádovo dlhé 3^n, musí platiť $r = 3$. Keďže máme deterministický systém, tak systém nevieme zostrojiť.

□

Veta 3.1.2. Triedy jazykov $(k,l)L$ nie sú uzavreté na prienik a prienik s regulárnym jazykom.

Veta bude dokázaná po tvrdeniach o vzťahoch $(k,l)L$ jazykov a jazykov chomského hierarchie v závere kapitoly.

Dôsledok 3.1.1. Triedy jazykov $0L$, $(k,0)L$, $(k,l)L$ nie sú uzavreté na zjednotenie.

Veta 3.1.3. Každý konečný netriviálny jazyk patrí do triedy $\mathcal{L}((k,l)L)$ kde $k > 0$ alebo $l > 0$.

Dôkaz vety je analogický ako dôkaz vety 2.2.2.

Veta 3.1.4. Každý jednopísmenkový jazyk patriaci do triedy $\mathcal{L}((k,l)L)$ patrí aj do triedy $\mathcal{L}((l,k)L)$.

Dôkaz: Ku $(k,l)L$ systému S vytvoríme $(l,k)L$ systém S' jednoducho tak, že otočíme axiomu, ľavý a pravý kontext a pravé strany pravidiel. Teda nech $S = (\Sigma, g, P, w_0)$ potom $S' = (\Sigma, g, P', w_0')$ pričom:

$$w_0' = w_0^R$$

$$P' = \{u_1{<}a{>}u_2 \to u_3 | u_2{<}a{>}u_1 \to u_3^R \in P\}$$

□

3.2 Hierarchia tried $(k,l)L$ jazykov

Lema 3.2.1. $\forall k,l \geq 0$

$\mathcal{L}((k,l)L) \subsetneq \mathcal{L}((k+1,l)L)$

$\mathcal{L}((k,l)L) \subsetneq \mathcal{L}((k,l+1)L)$

Dôkaz: Nech $L \in \mathcal{L}((k,l)L)$, potom automaticky $L \in \mathcal{L}((k+1,l)L)$ a $L \in \mathcal{L}((k,l+1)L)$.

Ukážeme, že existuje jazyk L a $(k+1,l)L$ systém S taký, že $L \in \mathcal{L}(S)$, ale neexistuje $(k,l)L$ systém S' taký, že $L \in \mathcal{L}(S')$.

$L = \{a^{k+l+1}a^{(k+l+2)^n} \mid n \geq 1\}$

$S = (\Sigma, g, w_0, P)$
$\Sigma = \{a\}$
$w_0 = \{a^{k+l+1}a^{k+l+2}\}$
Pravidlá v množine P vyzerajú nasledovne:

$g^i a^{k+1-i}<a>a^l \to a$ pre všetky $0 < i \leq k+1$

$a^{k+1}<a>a^{l-i}g^i \to a$ pre všetky $0 < i \leq l$

$a^{k+1}<a>a^l \to a^{k+l+2}$

Nech $w \in L$ teda $w = a^{k+l+1}a^{(k+l+2)^i}$ pre konkrétne $i > 0$. Nasledujúce odvodené slovo bude $a^{k+1}a^{(k+l+2)^i*(k+l+2)}a^l = a^{k+l+1}a^{(k+l+2)^{i+1}}$. Teda systém vygeneruje všetky slová z jazyka L a žiadne iné.
Analogicky vieme urobiť systém $(k,l+1)L$, ktorý generuje jazyk L.

Pokúsme sa nájsť $(k,l)L$ systém $S' = (\Sigma, g, w_0', P')$ generujúci L.

1. S' musí byť deterministický.

   Dokážeme sporom. Nech by v P' bolo pravidlo $u_1<a>u_2 \to a^x|a^y$ a nech $x < y$. Nech $a^m \in L$ a jedno a sa prepísalo pomocou $u_1<a>u_2 \to a^x$. Potom aj a^{m-x+y} patrí do L. Teda symbol a sa prepísal pravidlom $u_1<a>u_2 \to a^y$ a nie $u_1<a>u_2 \to a^x$. $x-y$ je konštanta. Lenže vzdialenosti slov (vzhľadom na dĺžku) sa stále zväčšujú a teda niekdedy určite prekročia rozdiel $x-y$. Vytvorili by sa aj slová nepatriace do L. Teda S' musí byť deterministický.

2. Nech P_g' sú pravidlá, v ktorých kontexte sa nachádzajú špeciálne znaky g, P_a' sú ostatné pravidlá - teda pravidlo $a^k<a>a^l \to a^r$.
   Dĺžku slova v systéme S' po n krokoch odvodenia možno popísať rekurenciou:

$l_0 = |w'_0|$

$l_n = (l_{n-1} - p)r + q$

kde

p - je celková dĺžka kontextu

q - počet písmen, ktoré vyrobia pravidlá P'_g

r - konštanta z pravidla $a^k<a>a^l \to a^r$, teda počet symblolov a, ktoré toto pravidlo vytvorí

Po vyriešení rekurencie dostávame vzťah pre dĺžku slova po n-tom kroku odvodenia:

$l_n = (|w'_0| + \frac{q-pr}{r-1})r^n + \frac{pr-q}{r-1}$

Ak chceme vytvoriť slová rádovo dlhé $(k+l+2)^n$, musí platiť $r = k+l+2$, a teda musí existovať pravidlo $a^k<a>a^l \to a^{k+l+2} \in P'$.

3. V každom vygenerovanom slove je viac symbolov, na ktoré sa dá aplikovať pravidlo $a^k<a>a^l \to a^{k+l+2}$, ako symbolov, na ktoré sa dajú aplikovať iné pravidlá. Keďže zároveň L je deterministický, vetná forma sa počas generovania stále predlžuje. Znamená to, že musíme začať generovať od najkratšieho slova a nemôžeme vygenerovať slovo, pokiaľ sme už nevygenerovali všetky kratšie slová.

   Nech $u_1 = x = a^{k+l+1}a^{(k+l+2)^1}$, $u_2 = a^{k+l+1}a^{(k+l+2)^2}$. Vieme, že $u_1 \Rightarrow^1 u_2$, $|u_1| = k+l+1+k+l+2 = 2k+2l+3$, $|u_2| = k+l+1+(k+l+2)^2 = k^2+l^2+2kl+5k+5l+5$. Pri ďalšom odvodení zo slova u_1 sa $k+l$ symbolov prepíše pravidlami P'_g. Zvyšných $k+l+3$ symbolov sa prepíše pravidlom P'_a. Teda vznikne vetná forma aspoň s $(k+l+3)(k+l+2) = k^2+l^2+2kl+5k+5l+6$ symbolmi, čo je viac ako $|u_2|$ - spor.

Slovo u_2 v systéme S' nevieme odvodiť.

Ukázali sme, že neexistuje $(k,l)L$ systém, ktorý by generoval jazyk L.

□

Veta 3.2.1. Triedy jazykov $\mathcal{L}((0,l)L)$ a $\mathcal{L}((k,0)L)$ sú pre ľubovoľné $k,l > 0$ neporovnateľné.

Dôkaz: Ukážeme, že pre ľubovoľné $k,l > 0$ existujú jazyky

1. $L_1 \in \mathcal{L}((0,1)L) \subseteq \mathcal{L}((0,l)L) \wedge L_1 \notin \mathcal{L}((k,0)L)$
2. $L_2 \notin \mathcal{L}((0,l)L) \wedge L_2 \in \mathcal{L}((1,0)L) \subseteq \mathcal{L}((k,0)L)$

1. Nech $S = (\{a,b\}, P, g, a^{k+2})$

$$P = \{<a>g \to aab, <a>a \to a, <b>g \to ab\}$$

Prvých niekoľko slov systému S je $a^{k+2}, a^{k+3}b, a^{k+4}b, \ldots$. Ľahko zistíme, že tento systém generuje jazyk

$$L_1 = \{a^{k+2}\} \cup \{a^{k+3+i}b \mid i \geq 0\}$$

Pokúsme sa nájsť $(k,0)L$ systém $S' = (\Sigma, P', g, w_0')$ generujúci jazyk L_1.

(a) Ak by bola axioma $a^{k+3+i}b$. Na odvodenie slova a^{k+2} potrebujeme pravidlo $P_b = a^k<b> \to a^m, m \geq 0$. Do L_1 patrí aj slovo $w = a^{k+3+p}b, p > i$. Použitím pravidla P_b sa z neho odstráni jediný symbol b. Aby vzniknuté slovo opäť patrilo do jazyka L_1, musíme slovo w prepísať na a^{k+2}. Keďže w môže byť ľubovoľne dlhé, nutnou podmienkou je, aby $P_a = a^k<a> \to \varepsilon \in P'$ a zároveň aby P_a bolo deterministické. To ale znamená, že vygenerované slovo nikdy neprekročí istú dĺžku. To je spor, lebo L_1 obsahuje ľubovoľne dlhé slová dlhšie ako $k+4$.

(b) Ak by bola axioma a^{k+2}. Na vygenerovanie symbolu b potrebujeme pravidlo, ktoré možno aplikovať na jediný symbol axiomy. To znamená, že $P_b = g^i a^{k-i}<a> \to a^j b \in P'$ pre konkrétne i. To znamená, že na začiatku slova možno vytvoriť symbol b. Aby v slove nevzniklo viac symbolov b, musíme pri prepisovaní všetky už vygenerované symboly b zmazať. To ale znamená, že symbol b musíme vždy nanovo vytvoriť, a teda pravidlo P_b musí byť deterministické. Ďalej musíme zaručiť, aby sa všetky symboly za vygenerovaným symbolom b zmazali. Symboly pred symbolom b vytvárajú iba pravidlá $P_g = g^n a^{k-n}<a> \to a^m$. To ale znamená, že vygenerované slovo nikdy neprekročí istú dĺžku a teda systém nedokáže vytvoriť ľubovoľne dlhé slová dlhšie ako $k+4$.

2. Analogicky ako v 1. časti. Použitý jazyk bude

$$L_2 = \{a^{k+2}\} \cup \{ba^{k+3+i} \mid i \geq 0\}$$

□

Teraz si ukážeme, že sila $(k,l)L$ systému závisí predovšetkým od dĺžky kontextu a nie od jeho rozdelenia na ľavý a pravý.

Veta 3.2.2. Pre ľubovoľné $k \geq 1, l \geq 0$ platí :

$$\mathcal{L}((k+1,l)L) \subseteq \mathcal{L}((k,l+1)L)$$

Veta 3.2.3. Pre ľubovoľné $k \geq 0, l \geq 1$ platí :

$$\mathcal{L}((k,l+1)L) \subseteq \mathcal{L}((k+1,l)L)$$

Veta 3.2.4. Pre ľubovoľné $k,l,m,n \geq 1$ platí :

$$\mathcal{L}((k,l)L) = \mathcal{L}((m,n)L) \iff k+l = m+n$$

Ešte pred dôkazom týchto viet uvedieme algoritmus, ktorý prerobí $(k+1,l)L$ systém na $(k,l+1)L$ systém generujúci rovnaký jazyk.

Algoritmus 3.2.1. Nech $S = (\Sigma, P, g, w_0)$ je $(k+1,l)L$ systém, $k > 1, l \geq 0$. Zostrojíme k nemu $(k,l+1)L$ systém $S' = (\Sigma', P', g', w_0')$ taký, že $\mathcal{L}(S) = \mathcal{L}(S')$.

$\Sigma' = \Sigma$

$g' = g$

$w_0' = w_0$

P' je definovaná nasledovne:
Nech $x_1, x_2, x_3 \in \Sigma$; $u_1, u_2, u_3 \in (\Sigma \cup \{g\})^*$; $w, t \in \Sigma^*$

(a) Pre každé pravidlo $g^k x_1 < x_2 > u_2 \to t \in P$ také,
že existuje pravidlo $g^{k+1} < x_1 > x_2 u_2 \to w \in P$,
vytvoríme v P' pravidlo $g'^k < x_1 > x_2 u_2 \to wt$

(b) Pre každé pravidlo $u_1 x_1 x_2 < x_3 > u_2 \to w \in P$
vytvoríme v P' pravidlo $u_1 x_1 < x_2 > x_3 u_2 \to w \in P'$

(c) Pre každé pravidlo $g^{k+1} < x_1 > g^l \to w \in P$
vytvoríme v P' pravidlo $g^k < x_1 > g^{l+1} \to w$

(d) Pre každé $u_1 \neq g^k, |u_1| = k$ vytvoríme v P' pravidlo $u_1 < x_1 > g^{l+1} \to \varepsilon$

Všetky pravidlá P' vzniknú iba použitím týchto možností.

Analogicky vieme zostrojiť algoritmus, ktorý prepíše $(k,l+1)L$ systém na $(k+1,l)L$ systém generujúci rovnaký jazyk.

Správnosť algoritmu dokážeme.

Dôkaz: Ukážeme, že systém S', ktorý dostaneme prepísaním systému S, naozaj generuje ten istý jazyk L. Dôkaz matematickou indukciou:

1. $w_0' = w_0$
2. Nech $u, w \in \Sigma^*$. Ak $u \Rightarrow^1 w$ v systéme S, potom aj $u \Rightarrow^1 w$ v systéme S'

Ak má slovo u iba jeden symbol, tak sa v S' podľa (c) prepíše na také isté slovo w ako v S.
Nech $u_1 \dots u_n$ sú symboly slova u. Ak sa zo slova $u_1 \dots u_n$ odvodí v systéme S slovo $w_1 \dots w_n$, potom pre všetky $2 < i \leq n$ v systéme S' u_{i-1} simuluje u_i

tým, že odvodí w_n (podľa (b)). (V pôvodnom systéme u_i odvodí w_i.) Ďalej u_1 v systéme S' simuluje odvodenie u_1 aj u_2 v systéme S. (podľa (a)). Nakoniec sa u_n v systéme S' prepíše na ε (podľa (d)). Teda sme v systéme S' zo slova $u_1 \dots u_n$ odvodili slovo $w_1 \dots w_n$.

□

Príklad 3.2.1. Nech $S = (\{a\}, g, P, aa)$, kde

$$P = \left\{ \begin{array}{lll} ga{<}a{>}g{\to}aa, & gg{<}a{>}a{\to}a \mid \varepsilon, & aa{<}a{>}a{\to}a \\ aa{<}a{>}g{\to}a\ , & ga{<}a{>}a{\to}a\ \ , & gg{<}a{>}g{\to}a \end{array} \right\}$$

Preveďme $(2,1)L$ systém S na $(1,2)L$ systém $S' = (\{a\}, g, P', aa)$. Vytvorené pravidlá pre systém S' sú nasledovné:

Podľa (a) vytvoríme pravidlá $\{g{<}a{>}ag \to a^3 \mid a^2,\ \ g{<}a{>}aa \to a^2 \mid a\}$.
Podľa (b) vytvoríme pravidlá $\{a{<}a{>}ag \to a,\ \ a{<}a{>}aa \to a\}$.
Podľa (c) vytvoríme pravidlo $\{g{<}a{>}gg \to a\}$.
Podľa (d) vytvoríme pravidlo $\{a{<}a{>}gg \to \varepsilon\}$

Ľahko overíme, že $L(S') = L(S) = \{aaa, aa\}$

Vety 3.2.2, 3.2.3 sú priamo výsledkom algoritmu 3.2.1.

Dôkaz vety 3.2.4
$\Leftarrow$
Ak $k = l = 1$ potom aj $m = n = 1$ a tvrdenie platí. Bez ujmy na všeobecnosti nech $k > 1$. Keďže $k = m + n - l$ potom $\mathcal{L}((k,l)L) = \mathcal{L}((m+n-l,l)L) \subseteq \mathcal{L}((m+n-l-1,l+1)L) \subseteq \cdots \subseteq \mathcal{L}((m,l-l+n)L) = \mathcal{L}((m,n)L)$. Podobne ukážeme, že $\mathcal{L}((m,n)L) \subseteq \mathcal{L}((k,l)L)$.
$\Rightarrow$
Dôkaz obmenou. Bez ujmy na všeobecnosti nech $k+l > m+n$. Nech $k+l+q = m+n, q > 0$. Podľa lemy 3.2.1 $\mathcal{L}((k,l)L) \subsetneq \mathcal{L}((k+q,l)L)$ a podľa "$\Leftarrow$ " platí $\mathcal{L}((k+q,l)L) = \mathcal{L}((m,n)L)$. Teda $\mathcal{L}((k,l)L) \subsetneq \mathcal{L}((m,n)L)$.

□

O tom, kedy je jedna trieda jazykov ostro silnejšia ako druhá, hovorí nasledujúca veta.

Veta 3.2.5. Pre ľubovoľné $k, l, m, n \geq 1$ platí :

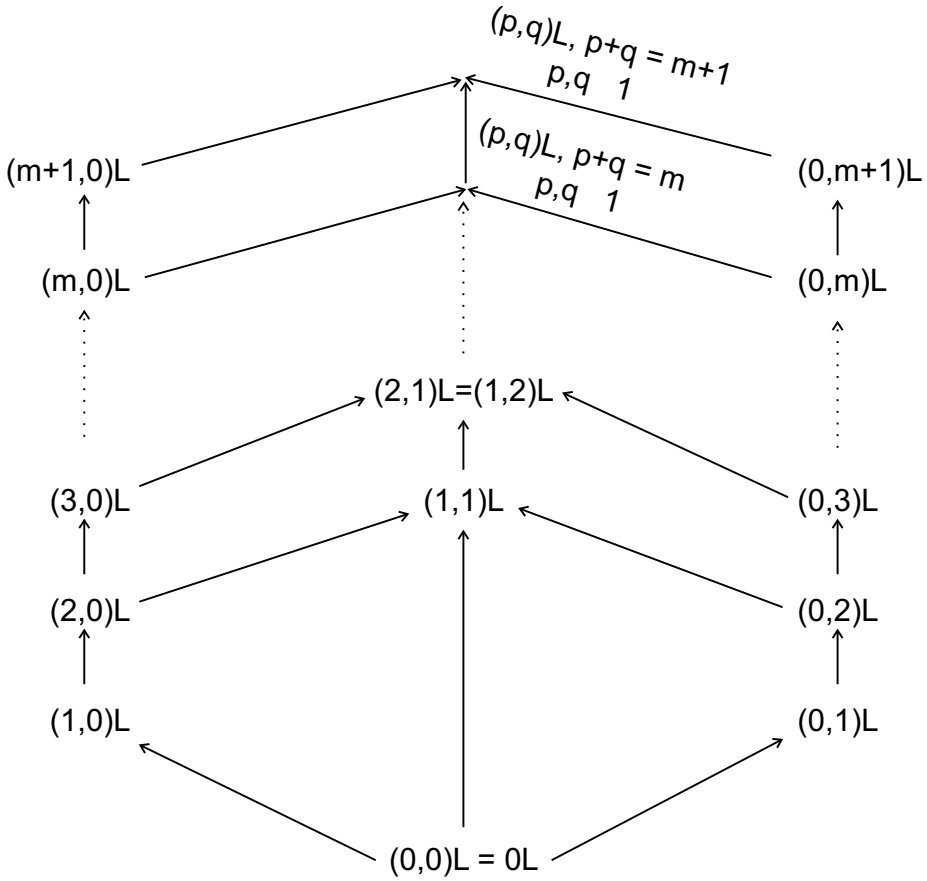
$$\mathcal{L}((k,l)L) \subsetneq \mathcal{L}((m,n)L) \iff k + l < m + n$$

Dôkaz:
" $\Leftarrow$ "
Podľa vety 3.2.4 platí $\mathcal{L}((m,n)L) = \mathcal{L}((m+n-l, n-n+l)L) = \mathcal{L}((m+n-l,l)L)$.
$k < m + n - l$ a teda za pomoci lemy 3.2.1 platí
$\mathcal{L}((k,l)L) \subsetneq \mathcal{L}((k+1,l)L) \subsetneq \cdots \subsetneq \mathcal{L}((m+n-l,l)L)$.

" $\Rightarrow$ "
Dokážeme obmenu tvrdenia.
$k+l \geq m+n \Rightarrow \mathcal{L}((k,l)L) \supset \mathcal{L}((m,n)L)$
Ak $k+l = m+n$, potom podľa vety 3.2.4 $\mathcal{L}((k,l)L) = \mathcal{L}((m,n)L)$.
Ak $k+l > m+n$, potom podľa " $\Leftarrow$ " platí $\mathcal{L}((k,l)L) \supsetneq \mathcal{L}((m,n)L)$.

□

Výsledky z tejto podkapitoly môžeme zhrnúť a prehľadne zakresliť do obrázku. Nasledovný obrázok ukazuje hierarchiu tried $(k,l)L$ systémov v závislosti od dĺžky ľavého a pravého kontextu. Šípka určuje ostrú nerovnosť medzi dvomi triedami. Pokiaľ sa z jednej triedy do druhej nedá dostať postupnosťou v smere šípok, tak sú tieto triedy neporovnateľné.



3.3 Porovnanie tried $(k,l)L$ jazykov s Chomského hierarchiou

Veta 3.3.1. Každá z tried $\mathcal{L}_{CF} - \mathcal{R}$, $\mathcal{L}_{CS} - \mathcal{L}_{CF}$ obsahuje jazyky, ktoré sú v triede $\mathcal{L}((k,l)L)$ pre vhodné k,l aj jazyky, ktoré v tejto triede nie sú.

Dôkaz: $(k,l)L$ jazyky v príslušných triedach sú:

1. $\{a^i b a^i \mid i \geq 0\} \in \mathcal{L}_{CF} - \mathcal{R}$
   Príslušný $0L$ ($(0,0)L$) systém bude napríklad
   $S = (\{a,b\}, \{b \to aba, a \to a\}, b)$.
   Odvodenie do hĺbky 3: $b \Rightarrow aba \Rightarrow aabaa \Rightarrow a^3ba^3$.

2. $\{a^n b^n c^n \mid n \geq 0\} \in \mathcal{L}_{CS} - \mathcal{L}_{CF}$
   Príslušný $(1,0)L$ systém bude napríklad
   $S = (\{a,b,c\}, g, \{g<a> \to aa, a<b> \to bb, b<c> \to cc, a<a> \to a, b<b> \to b, c<c> \to c, \}, abc)$.
   Odvodenie do hĺbky 3: $abc \Rightarrow a^2b^2c^2 \Rightarrow a^3b^3c^3 \Rightarrow a^4b^4c^4$.

Jazyky v príslušných triedach, ktoré nie sú v triede $(k,l)L$ pre ľubovoľné k,l:

1. $\{a^{n_1}b^{2n}a^{n_2}b^{3n}a^{n_3} \mid n_1, n_2, n_3, n > 0\} \in \mathcal{L}_{CF} - \mathcal{R}$
   Bezkontextová gramatika generujúca takýto jazyk je napríklad
   $G = (\{S,A,B\}, \{a,b\}, \{S \to AbbBbbbA, A \to aA|a, B \to bbBbbb|A\}, S\})$.
   Dôvod, prečo nemôže existovať $(k,l)L$ systém pre žiadne k,l generujúci takýto jazyk L, je nasledovný. Určite sa v priebehu odvodzovania vyskytne vetná forma $a^{n_1}b^{2n}a^{n_2}b^{3n}a^{n_3}$, kde $n1, n2, n3 \geq k+l$. Potom ale na základe kontextu nevieme rozlíšiť skupiny znakov b. Teda sa následne môžu prepísať obe takým istým spôsobom a systém vygeneruje slovo nepatriace do jazyka L.

2. $\{a^{2^n}\} \cup \{a^{3^n}\}, n \geq 1 \in \mathcal{L}_{CS} - \mathcal{L}_{CF}$, dokázané ako veta 3.1.1.

□

Veta 3.3.2. Každý regulárny jazyk patrí do niektorej triedy jazykov $\mathcal{L}((k,l)L)$.

Predvedieme algoritmus, ktorý dokáže každý deterministický konečný automat bez prechodov na epsilon previesť na $(k,l)L$ systém, pričom oba generujú ten istý jazyk. Následne ukážeme použitie algoritmu na príklade.

Algoritmus 3.3.1. Nech $A = (K, \Sigma_A, \delta, q_0, F)$ je deterministický konečný automat bez ε-ových prechodov. Potom $S = (\Sigma_S, g, P, w_0)$ je $(|K|, |w_0|)L$ systém, pričom $L(A) = L(S)$.

S je definovaný nasledovne:

$\Sigma_S = \Sigma_A$

w_0 je najkratšie slovo dĺžky aspoň 1 akceptované automatom A

Množinu prepisovacích pravidiel vyrobíme nasledovne:

$P = P_1 \cup P_2 \cup P_3$

Najskôr zadefinujeme množiny M_A a $M_A(q)$.

Množina M_A obsahuje slová zložené zo znakov, ktoré automat A môže prečítať pri prechode z počiatočného stavu do ľubovoľného akceptačného stavu s tým, že urobí najviac $2c$ krokov, kde c je maximum z najkratších ciest od počiatočného stavu k akceptačnému stavu v stavovom grafe automatu. Ak sa počiatočný a niektorý akceptačný stav rovnajú, v množine M_A je aj slovo ε. Teda množina M_A obsahuje všetky slová jazyka $L(A)$ kratšie ako $2|w_0|$ (obsahuje aj dlhšie).

$M_A = \{w \mid w = a_1 \dots a_n \wedge w \in \Sigma_A^* \wedge q_F \in F \wedge q_0 = p_0 \wedge q_F = p_n \wedge q_0 \neq q_f \wedge \delta(p_0, a_1) = p_1, \delta(p_1, a_2) = p_2, \dots, \delta(p_{n-1}, a_n) = p_n \wedge n \leq 2c\} \cup \{\varepsilon | q_0 = q_f \wedge q_f \in F\}$

Množina $M_A(p)$ obsahuje všetky slová zložené zo znakov, ktoré automat A môže prečítať pri prechode zo stavu p do stavu p po kružnici alebo slučke.

$M_A(p) = \{w | w = a_1 \dots a_n \wedge w \in \Sigma_A^* \wedge \delta(p, a_1) = p_1, \delta(p_1, a_2) = p_2, \dots, \delta(p_{n-1}, a_n) = p \wedge (p_i = p_j) \Rightarrow (i = j) \wedge \ i, j = 1 \dots n-1\} \cup \{\varepsilon\}$.

Nech: $w_0 = a_1 \dots a_n$, $|K| = k$, $|w_0| = l$, potom

$$P_1 = \{g^{k-i} a_1 \dots a_{i-1} < a_i > a_{i+1} \dots a_n g^{l-i+1} \to \varepsilon \mid 1 < i \leq l\} \cup \{g^k < a_1 > a_2 \dots a_n g \to w | w \in M_A\}$$

Pravidlá množiny P_1 zmažú všetky znaky axiomy až na prvý znak. Prvý znak axiomy sa nedeterministicky prepíše na ľubovoľné slovo z množiny M_A. Systém, ktorý by obsahoval iba pravidlá P_1, by vedel odvodiť všetky slová jazyka $L(A)$ menšie alebo rovnako dlhé ako $2|w_0|$.

$$P_2 = \{ \quad g^{k-|w_1|} w_1 < x > w_2 \to w_3 \quad | \quad (\exists a_1, \dots a_n \in \Sigma, \exists p_1, \dots p_n \in K, \exists q \in K) (w_1 = a_1 \dots a_n \wedge \delta(q_0, a_1) = p_1 \wedge \delta(p_1, a_2) = p_2 \wedge \ \dots \ \wedge \delta(p_{n-1}, a_n) = q \wedge (\forall i, j 0 < i, j < n)(p_i = p_j \Rightarrow i = j) \wedge w_3 \in M_A(g))\}$$
$$w_1, w_2, w_3 \in \Sigma^*, x \in \Sigma$$

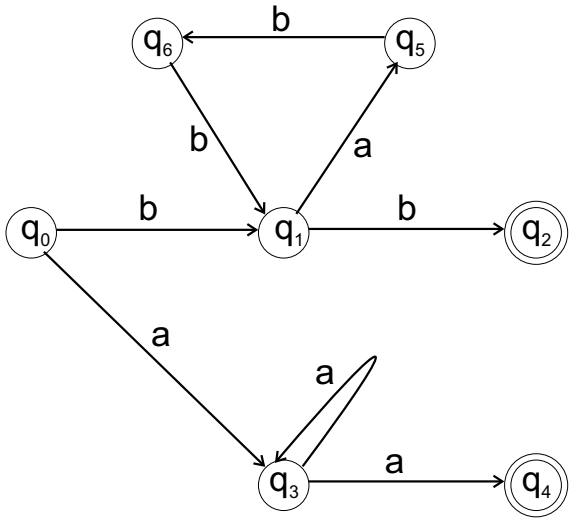
Pravidlá množiny P_2 simulujú kružnice v stavovom grafe automatu A. Systém S na týchto miestach pripája postupnosť písmen, ktorými by prešiel automat A pri prechode takejto kružnice. Pravé kontexty pravidiel z množiny P_2 neobsahujú

znak g a preto pravidlá z množiny P_1, ktoré vymazávali časti axiomy, ostanú deterministické. Keďže pracujeme s deterministickým automatom A a každý ľavý kontext pravidiel z množiny P_2 začína znakom g, potom tieto ľavé kontexty jednoznačne určujú miesto (stav automatu A), kde pripájame podslová vzniknuté z kružníc.

Množina P_3 obsahuje pravidlá s kombináciami ľavých a pravých kontextov, ktoré neboli definované v množinách P_1 a P_2. Pravidlá z množiny P_3 prepisujú znaky samé na seba. Množina výsledných pravidiel bude teda zjednotenie pravidiel P_1, P_2 a P_3.

Pre lepšie pochopenie automatu uvedieme príklad použitia.

Príklad 3.3.1. Nech $A = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{a, b\}, \delta, q_0, \{q_2, q_4\})$ je konečný automat, prechodová funkcia δ je definovaná pomocou stavového grafu:



K automatu A zostrojíme $(7,2)L$ systém $S = (\Sigma, g, P, w_0)$, kde

$\Sigma = \{a, b\}$

$w_0 = aa$

$P = P_1 \cup P_2 \cup P_3$

$M_A = \{bb, bb^3b, bb^6b, aa, aaa, aaaa\}$

$$P_1 = \left\{ \begin{array}{l} g^7<a>ag \to b^2 \mid b^5 \mid b^8 \mid a^2 \mid a^3 \mid a^4 \\ g^6a<a>gg \to \varepsilon \end{array} \right\}$$

$P_2 = P_{q_0} \cup P_{q_1} \cup P_{q_2} \cup P_{q_3} \cup P_{q_4} \cup P_{q_5} \cup P_{q_6}$

V ďalšom texte x a y sú ľubovoľné znaky abecedy Σ.

$$M_A(q_0) = \{\varepsilon\}$$

$$P_{q_0} = \left\{ \begin{array}{l} g^7{<}a{>}xy \to a \\ g^7{<}b{>}xy \to b \end{array} \right\}$$

$$M_A(q_1) = \{\varepsilon, abb\}$$

$$P_{q_1} = \left\{ \begin{array}{l} g^6b{<}a{>}xy \to a \mid abba \\ g^6b{<}b{>}xy \to b \mid abbb \end{array} \right\}$$

$$M_A(q_2) = \{\varepsilon\}$$

$$P_{q_2} = \left\{ \begin{array}{l} g^5bb{<}a{>}xy \to a \\ g^5bb{<}b{>}xy \to b \end{array} \right\}$$

$$M_A(q_3) = \{\varepsilon, a\}$$

$$P_{q_3} = \left\{ \begin{array}{l} g^6a{<}a{>}xy \to a \mid aa \\ g^6a{<}b{>}xy \to b \mid ab \end{array} \right\}$$

$$M_A(q_4) = \{\varepsilon\}$$

$$P_{q_4} = \left\{ \begin{array}{l} g^5aa{<}a{>}xy \to a \\ g^5aa{<}b{>}xy \to b \end{array} \right\}$$

$$M_A(q_5) = \{\varepsilon, bba\}$$

$$P_{q_5} = \left\{ \begin{array}{l} g^5ba{<}a{>}xy \to a \mid bbaa \\ g^5ba{<}b{>}xy \to b \mid bbab \end{array} \right\}$$

$$M_A(q_6) = \{\varepsilon, bab\}$$

$$P_{q_6} = \left\{ \begin{array}{l} g^4bab{<}a{>}xy \to a \mid baba \\ g^4bab{<}b{>}xy \to b \mid babb \end{array} \right\}$$

P_3 obsahuje všetky pravidlá, ktorých kombinácie ľavých a pravých kontextov nie sú zahrnuté v množinách P_1 a P_2. Pravidlá z množiny P_3 prepisujú znaky samé na seba.

Ľahko sa možno presvedčiť, že systém S generuje jazyk $L(A)$.

Veta 3.3.3. Existuje jazyk z triedy rekurzívne vyčísliteľných jazykov, ktorý nepatrí do žiadnej triedy $\mathcal{L}((k,l)L)$.

Takýmto jazykom je napríklad už spomínaný jazyk $\{a^{2^n}\} \cup \{a^{3^n}\}, n \geq 1$.

Veta 3.3.4. Existuje jazyk $L \in \mathcal{L}_{RE} - \mathcal{L}_{CS}$ ktorý patrí do triedy $\mathcal{L}((k,l)L)$ pre nejaké $k, l \in \mathbb{N}$.

Dôkaz: Ukážeme, že trieda jazykov $\mathcal{L}((k,l)L) \not\subseteq \mathcal{L}_{REC}$.
Nech G je ľubovoľná frázová gramatika generujúca rekurzívne vyčísliteľný jazyk L. Ukážeme, že vieme zostrojiť $(k,l)L$ systém S a regulárny jazyk R, aby $L = L(S) \cap R$. Ak by $\mathcal{L}((k,l)L) \subseteq L_{REC}$ potom
$A = \{L' \cap R' \mid L' \in \mathcal{L}((k,l)L), R' \in \mathcal{R}\} \subseteq \mathcal{L}_{REC}$. Lenže L je rekurzívne vyčísliteľný jazyk a nie rekurzívny a zároveň $L \in A \subseteq \mathcal{L}_{REC}$, čo je spor.

□

Ostáva ešte popísať algoritmus na prerobenie ľubovoľnej frázovej gramatiky na $(k,l)L$ systém S a regulárny jazyk R, aby $L = L(S) \cap R$.

Algoritmus 3.3.2. Nech $G = (N, T, P, \sigma)$ je ľubovoľná frázová gramatika. Nech $z, z', z_i, 0 < i < k \notin N \cup T$, a nech k je najdlhšia ľavá strana pravidla v P. Zostrojíme $(1,k)L$ systém S a regulárny jazyk R, aby $L(G) = L(S) \cap R$.
$R = T^*$, $S = (\Sigma', g, P', z\sigma)$, pričom :

$\Sigma' = N \cup T \cup \{z, z', z_1, \ldots, z_{k-1}\}$

Nech $x, y \in T \cup N, u \in (T \cup N \cup g)^k$, potom $P' = \{$

$z{<}x{>}u \to xz$

$z{<}x{>}u_1u_2 \to wz_i \;\Big|\; \begin{array}{l} xu_1 \Rightarrow w \in P, \\ i = |u_1| \\ u_1 \in (T \cup N)^{|w|-1} \\ u_2 \in (T \cup N)^{k-|w|} \end{array}$

$z_i{<}x{>}u \to z_{i-1} \mid k > i > 1$

$z_1{<}x{>}u \to z$

$x{<}z{>}g^k \to z'x$

$g{<}z'{>}u \to z|\varepsilon$

$x{<}z{>}u \to \varepsilon$

$x{<}z'{>}u \to \varepsilon$

$x{<}y{>}z'u_1 \to z'y \mid u_1 \in (T \cup N \cup g)^{k-1}$

všetky ostatné znaky sa prepíšu samy na seba

$\}$

Výpočet takto zostrojeného systému prebieha nasledovne: Na začiatku vetnej formy je znak z. Tento znak podobne ako aj znaky $z', z_i 0 < i < k$ reprezentujú miesto, na ktorom sa vo vetnej forme niečo môže prepísať. Takýto znak je vo vetnej forme vždy maximálne jeden. Počas výpočtu sa v každom kroku znak z vo vetnej forme buď posunie doprava, alebo sa prepíše nasledujúci znak podľa prepisovacích pravidiel frázovej gramatiky s tým, že za prepísanými znakmi sa pridá nový znak z_i. Ten ukazuje, koľko nasledujúcich znakov treba z vetnej formy zmazať. Po zmazaní i znakov sa opäť vytvorí znak z a výpočet pokračuje obvyklým spôsobom až do konca vetnej formy. Pokiaľ sa z dostane až na koniec vetnej formy, vráti sa pomocou znaku z' naspäť na začiatok vetnej formy. Na začiatku vetnej formy sa buď z' prepíše na z (presnejšie povedané z' sa zmaže a

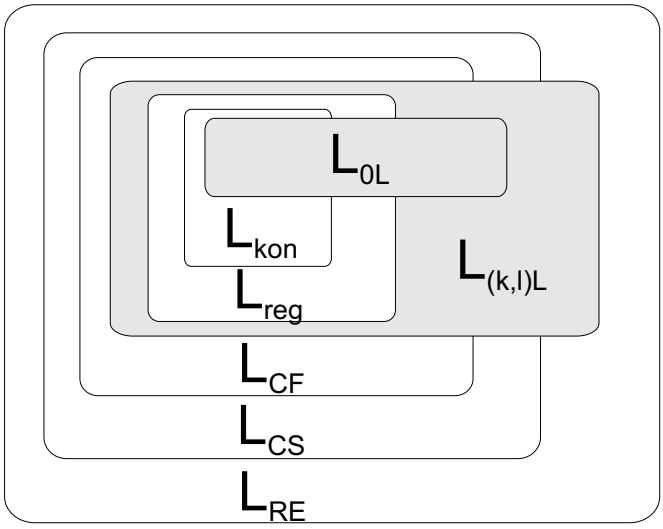
nasledujúci znak vytvorí aj z), alebo sa z' odstráni. Odstránením symbolov z sa výpočet skončí. Použitím symbolov z sa nám podarilo zachovať sekvenčnosť frázových gramatík, avšak použili sme neterminály. Na ich odstránenie treba urobiť prienik vzniknutého jazyka s regulárnym jazykom T^*.

Veta 3.3.5. Triedy jazykov $(k,l)L$ nie sú uzavreté na prienik a prienik s regulárnym jazykom.

Dôkaz: Vieme, že jazyk $L_1 = \{a^{2^n}\} \cup \{a^{3^n}\}$ nevieme urobiť žiadnym $(k,l)L$ systémom. Ak použijeme algoritmus 3.3.2, tak získame $(k,l)L$ jazyk L_2 a regulárny jazyk L_3 také, že $L_1 = L_2 \cap L_3$. Pripomeňme, že každý regulárny jazyk vieme urobiť vhodným $(k,l)L$ systémom.

□

Uvedené vety nám umožňujú zakresliť vzťah tried $(k,l)L$ jazykov a jazykov Chomského hierarchie do prehľadného obrázka.



Kapitola 4

$(k,l)L$ Generátor

$(k,l)L$ generátor je program alebo zariadenie, ktoré keď na vstup dostane definíciu $(k,l)L$ systému a hĺbku odvodenia n, tak na výstupe vyrobí slovo, ktoré tento systém po n krokoch odvodí. Pri návrhu takéhoto generátora sa intuitívne ponúka spôsob, pri ktorom vygenerujeme najskôr prvé slovo, uložíme ho v pamäti, ďalej druhé, tretie až n-té. Treba si však uvedomiť, že slová v L systémoch sú často exponenciálne dlhé od hĺbky odvodenia. Pritom tieto slová často používame na kreslenie rastlín, stromov, fraktálov. To znamená, že už nakreslenú časť slova môžeme zabudnúť. Preto je vhodný spôsob, keď výstup z generátora priamo kreslíme a neukladáme ho zbytočne celý do pamäte.

V tejto kapitole predvedieme $(k,l)L$ generátor, ktorý odvodí slovo $(k,l)L$ systému po n krokoch odvodenia, pričom použije iba $O(n)$ pamäte. Súčasťou tejto práce je aj implementovaný generátor pre deterministické systémy.

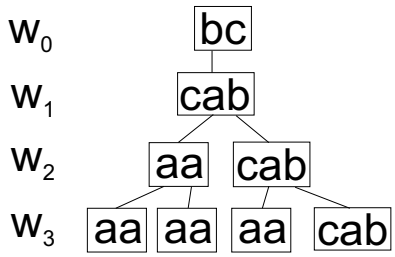
Ešte pred tým, ako predstavíme samotný $(k,l)L$ generátor, ukážeme princíp jeho fungovania na $0L$ systémoch.

4.1 $0L$ generátor

$0L$ generátor odvádza výsledné slovo w tak, že prechádza strom odvodenia slova w metódou *do hĺbky* (deep first search). Každej úrovni odpovedajúcej kroku odvodenia prislúcha časť pamäte, nazvime ju front. $0L$ generátor bude používať fronty $F_0, F_1, \ldots F_n$. Znaky z najspodnejšieho frontu F_n systém vyberá a zapisuje do výsledného slova. Ak je najspodnejší front prázdny, potom systém prepisuje znak z najspodnejšieho neprázdneho frontu, prepísaný znak z frontu vyberie a do frontu pod ním vloží pravú stranu vybratého pravidla. Vo fronte F_0 je zapísaná axioma. Žiaden front okrem F_0 nepoužije viac pamäte, ako je dĺžka k najdlhšej pravej strany nejakého pravidla. To znamená, že celková použitá pamäť bude $kn + |w_0|$, čo spadá pod $O(n)$.

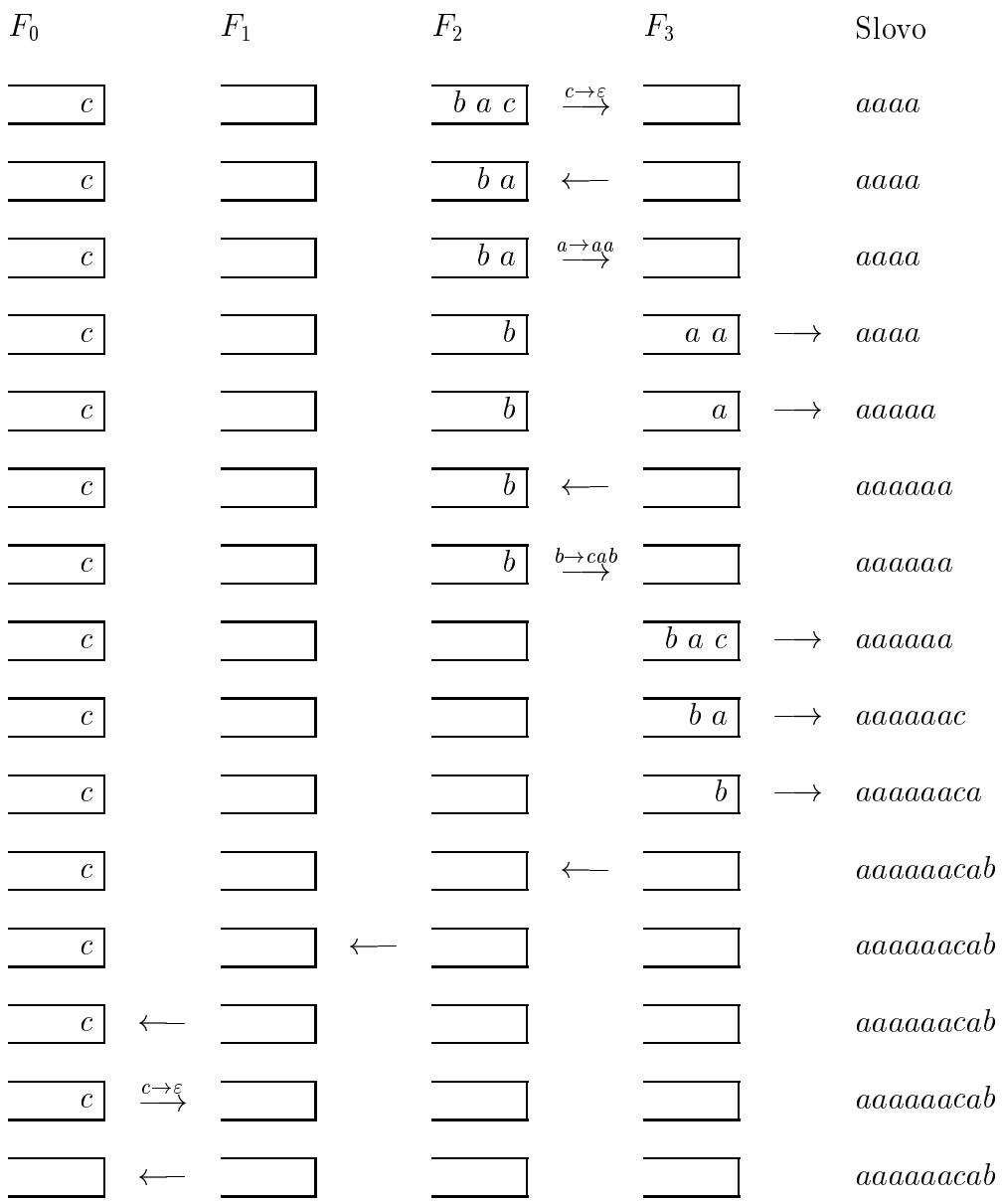
Fungovanie 0L generátora ukážeme na príklade:

Príklad 4.1.1. $S = (\{a,b,c\}, P, bc)$, kde $P = \{a \to aa, b \to cab, c \to \varepsilon\}$ je $0L$ systém. Strom odvodenia vyzerá nasledovne:



Pozrime sa na odvonenie slova po troch krokoch odvodenia, teda $n = 3$.

F_0		F_1		F_2		F_3		Slovo
$c\ b$	$\overset{b\to cab}{\longrightarrow}$							ε
c		$b\ a\ c$	$\overset{c\to\varepsilon}{\longrightarrow}$					ε
c		$b\ a$	$\longleftarrow$					ε
c		$b\ a$	$\overset{a\to aa}{\longrightarrow}$					ε
c		b		$a\ a$	$\overset{a\to aa}{\longrightarrow}$			ε
c		b		a		$a\ a$	$\longrightarrow$	ε
c		b		a		a	$\longrightarrow$	a
c		b		a	$\longleftarrow$	a		aa
c		b		a	$\overset{a\to aa}{\longrightarrow}$			ε
c		b				$a\ a$	$\longrightarrow$	aa
c		b				a	$\longrightarrow$	aaa
c		b			$\longleftarrow$			$aaaa$
c		b	$\longleftarrow$					$aaaa$
c		b	$\overset{b\to cab}{\longrightarrow}$					$aaaa$



Po skončení výpočtu bude výstupné slovo *aaaaaacab*.

V každom kroku výpočtu je ľubovoľný front buď prázdny, alebo je v ňom uložená pravá strana nejakého pravidla, prípadne jej ešte neprepísaný zvyšok. Počet znakov vo fronte v každom kroku výpočtu možno teda ohraničiť dĺžkou najdlhšej pravej strany pravidiel, a to je pre každý $0L$ systém konštanta. Množstvo pamäte potrebnej na odvodenie výstupného slova bude teda závisieť len od počtu frontov.

4.2 Príklad $(k,l)L$ generátora

$(k,l)L$ generátor odvádza výsledné slovo w podobne ako $0L$ generátor tak, že prechádza strom odvodenia slova w metódou do hĺbky (deep first search). Podobne

aj $(k,l)L$ generátor bude používať fronty $F_0, F_1, \ldots F_n$ avšak ku každému frontu patria deterministické konečné automaty rozpoznávajúce ľavý a pravý kontext.

Príklad 4.2.1. $S = (\{a\}, g, P, aa), P = \{g<a>gg \to a^3, g<a>ag \to a^3, g<a>aa \to a^2, a<a>gg \to \varepsilon, a<a>ag \to a, a<a>aa \to \varepsilon\}$ je (1,2)L systém. Prvých niekoľko odvodených slov tohto systému vyzerá nasledovne:

$w_0 = aa$

$w_1 = aaa$

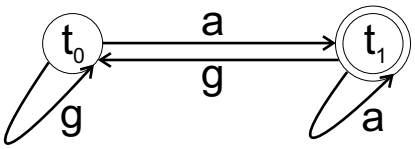
$w_2 = aaa$

$w_3 = aaa$

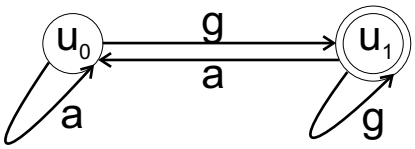
...

Tento systém je deterministický, $L(S) = \{aaa\}$. Skonštruujeme k nemu $(1,2)L$ generátor, ktorý túto množinu generuje. Generátor používa fronty F_0, F_1, F_2, F_3 a ku každému z nich patria deterministické konečné automaty, ktoré rozpoznávajú kontexty.

Automat A_g

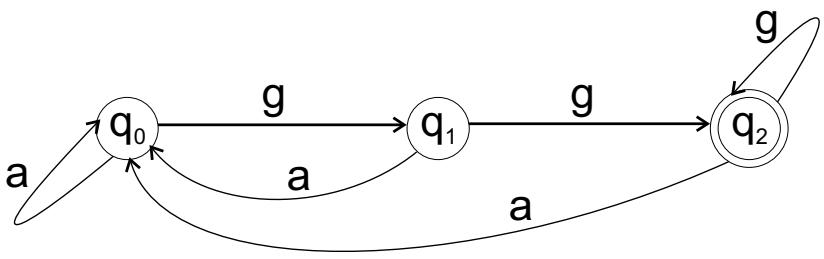


Automat A_a

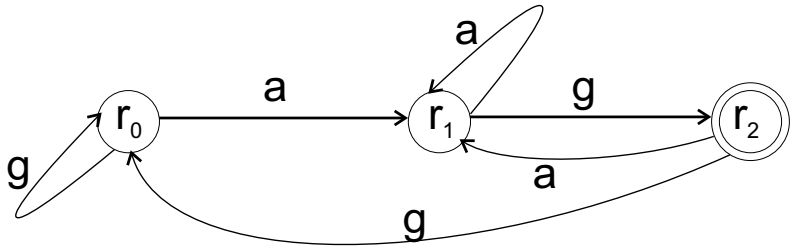


Na obrázku sú nakreslené automaty Ag, Aa na rozpoznávanie ľavých kontextov znaku na začiatku frontu. Automat Ag rozpoznáva kontext g, automat Aa rozpoznáva kontext a. Počiatočným stavom každého automatu je jeho najľavejší stav.

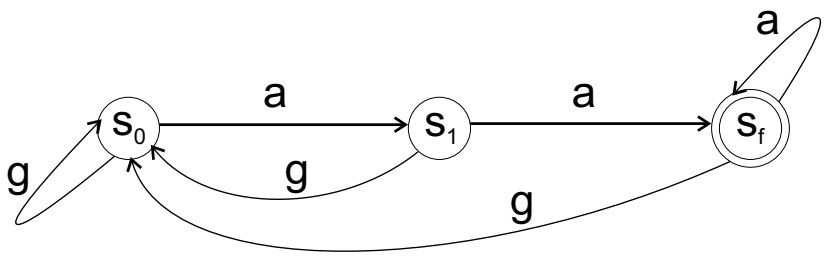
Na ďalšom obrázku sú nakreslené automaty Bgg, Bag, Baa pre rozpoznávanie pravých kontextov znaku na začiatku frontu. Automat Bgg rozpoznáva kontext gg, automat Bag rozpoznáva kontext ag, automat Baa rozpoznáva kontext aa.



Automat B_{gg}



Automat B_{ag}



Automat B_{aa}

Počiatočným stavom každého automatu je jeho najľavejší stav.

V podkapitole 4.4 si ukážeme, že pri praktickej realizácii generátora sú aj vhodnejšie prostriedky na rozpoznávanie kontextov ako množina konečných automatov. Pre jednoduchšie pochopenie však najskôr uvádzame metódu s konečnými automatmi.

Výpočet prebieha podobne ako výpočet $0L$ generátora s tým rozdielom, že pri prepisovaní sa berú do úvahy aj stavy automatov. Na prepísanie znaku sa vyberie pravidlo, ktorého ľavý a pravý kontext rozpoznávajú automaty, ktoré sú práve v akceptačnom stave.

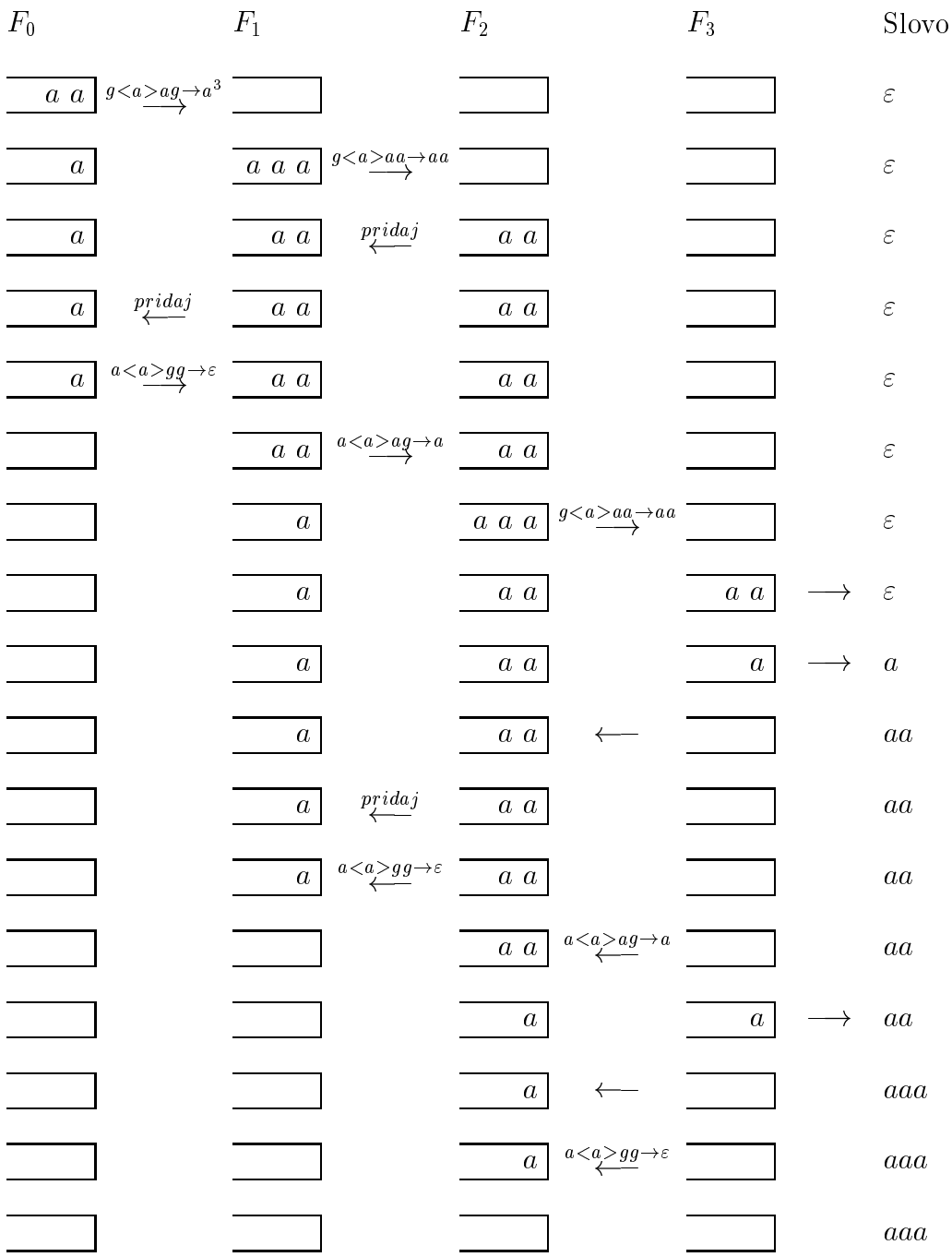
Prepísaný znak potom príde na vstup všetkých automatov pre rozpoznávanie ľavého kontextu, ktoré patria frontu. Tým sa zachová informácia o ľavom kontexte vrchola frontu.

Na vstup automatov pre pravý kontext dáva (k,l)L generátor znaky, ktoré nasledujú vo fronte za znakom na začiatku frontu.

Znak z vrcholu frontu F bude prepisovaný podľa toho, v akom ľavom a pravom kontexte sa nachádza, teda podľa toho, ktoré automaty sú v akceptačnom stave. Môže však nastať situácia, že vo fronte je príliš málo znakov na to, aby sme poznali celý pravý kontext dĺžky l. Vtedy sa výpočet vráti k predchádzajúcemu frontu (ak sa dá) a prepíše znak zo začiatku tohto frontu. Tým do frontu F môžu pribúdať ďalšie znaky. Môže sa stať, že front F nemá žiaden predchádzajúci, alebo všetky predchádzajúce fronty sú prázdne. V tomto prípade už do frontu F nemôžu pribudnúť ďalšie znaky, preto na vstup automatov pre pravý kontext pôjde potrebný počet špeciálnych symbolov g.

Nasledujúca schéma zachytáva priebeh výpočtu $(1, 2)L$ generátora, ktorý generuje množinu $\mathcal{L}_3(S)$.

Poznámka: Šípka medzi frontami naznačuje pravidlo, ktorým sa v dalšom kroku prepíše znak na začiatku frontu.



Výstupné slovo bude aaa.

Počet znakov vo fronte v každom kroku výpočtu možno ohraničiť pre konkrétny $(k, l)L$ systém konštantou. (Najviac znakov bude vo fronte vtedy, keď potrebuje-

me prepísať znak na začiatku frontu a poznáme jeho kontext iba do hĺbky $l - 1$ a do frontu pribudne najdlhšia pravá strana pravidiel). Množstvo pamäte potrebnej na odvodenie výstupného slova bude opäť závisieť len od počtu frontov.

4.3 Formálna definícia $(k,l)L$ generátora

Definícia 4.3.1. $(k,l)L$ generátor je devätica $G = (n, \Sigma, g, w_0, A, B, \delta, f, h)$ s týmito vlastnosťami:

$n \in N$ je maximálna hĺbka výpočtu

Σ je abeceda generátora

$g \notin \Sigma$ je špeciálny symbol

$w_0 \in \Sigma^+$ je počiatočný stav nultého frontu

$A = \{A_{a_1 \dots a_k} | A_{a_1 \dots a_k}$ je deterministický konečný automat rozpoznávajúci kontext $a_1 \dots a_k$, pričom $a_1 = a_2 = \cdots = a_i = g, a_{i+1}, a_{i+2}, \dots, a_k \in \Sigma, 0 \leq i \leq k\}$

$B = \{B_{b_1 \dots b_l} | B_{b_1 \dots a_l}$ je deterministický konečný automat rozpoznávajúci kontext $b_1 \dots b_l$, pričom $b_1, b_2, \dots, b_i \in \Sigma, b_{i+1} = b_{i+2} = \cdots = b_l = g, , 0 \leq i \leq l\}$

$\delta : \Sigma \longrightarrow 2^{A \times B \times \Sigma^*}_{KON}$ je prechodová funkcia

$f : A \longrightarrow \{1, 2, \dots, |A|\}$ je bijektívna funkcia, ktorá očísluje automaty z množiny A. Nazývame ju číslovacia funkcia množiny A.

$h : B \longrightarrow \{1, 2, \dots, |B|\}$ je bijektívna funkcia, ktorá očísluje automaty z množiny B. Nazývame ju číslovacia funkcia množiny B.

Príklad 4.3.1. $(1,2)L$ generátorom je napríklad $\overline{G} = (3, a, g, a, A, B, \delta, f, h)$, kde $A = \{A_g, A_a\}$, $B = \{B_{gg}, B_{ag}, B_{aa}\}$ pričom

$$A_g = (\{t_0, t_1\}, \{g, a\}, \begin{pmatrix} \Delta_g & g & a \\ t_0 & t_1 & t_0 \\ t_1 & t_1 & t_0 \end{pmatrix}, t_0, \{t_1\})$$

$$A_a = (\{u_0, u_1\}, \{g, a\}, \begin{pmatrix} \Delta_a & g & a \\ u_0 & u_0 & u_1 \\ u_1 & u_0 & u_1 \end{pmatrix}, u_0, \{u_1\})$$

$$B_{gg} = (\{q_0, q_1, q_2\}, \{g, a\}, \begin{pmatrix} \Delta_{gg} & g & a \\ q_0 & q_1 & q_0 \\ q_1 & q_2 & q_0 \\ q_2 & q_2 & q_0 \end{pmatrix}, q_0, \{q_2\})$$

$$B_{ag} = (\{r_0, r_1, r_2\}, \{g, a\}, \begin{pmatrix} \Delta_{ag} & g & a \\ r_0 & r_0 & r_1 \\ r_1 & r_2 & r_1 \\ r_2 & r_0 & r_1 \end{pmatrix}, r_0, \{r_2\})$$

$$B_{aa} = (\{s_0, s_1, s_2\}, \{g, a\}, \begin{pmatrix} \Delta_{aa} & g & a \\ s_0 & s_0 & s_1 \\ s_1 & s_0 & s_2 \\ s_2 & s_0 & s_2 \end{pmatrix}, s_0, \{s_2\})$$

a kde

$$\delta(a) = \{(A_g, B_{gg}, aaa), (A_g, B_{ag}, aaa), (A_g, B_a, aa),$$
$$(A_a, B_{gg}, \varepsilon), (A_a, B_{ag}, a), (A_g, B_{aa}, \varepsilon)\}$$

pričom číslovacie funkcie sú definované nasledovne:

$$f(A_g) = 1, f(A_a) = 2$$
$$h(B_{gg}) = 1, h(B_{ag}) = 2, h(B_{aa}) = 3$$

Tento generátor je formalizáciou výpočtu z predchádzajúceho príkladu.

Definícia 4.3.2. Konfiguráciou $(k,l)L$ generátora nazývame prvok

$$(i, \vec{j}, F_0, F_1, \ldots, F_n, \omega, \mathcal{A}_{A,f}, \mathcal{B}_{B,h})$$

kde i je okamžitá hĺbka výpočtu, F_j pre všetky $j \in \{0, \ldots, n\}$ je j-ty front, ω je výstupné slovo, $\mathcal{A}_{A,f}$ je matica stavov typu $n \times |A|$ zahŕňajúca stavy automatov množiny A, $\mathcal{B}_{B,h}$ je matica stavov typu $n \times |B|$ zahŕňajúca stavy automatov množiny B a $\vec{j} \in \{0, 1, \ldots, l\}^n$ je vektor, ktorého zložky j_s sú informáciou o prezretom pravom kontexte vrchola frontu F_s.

Konfiguráciou $(k,l)L$ generátora $\overline{G}$ je napríklad devätica

$$(0, \vec{0}, aa, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \begin{pmatrix} t_1 & u_0 \\ t_1 & u_0 \\ t_1 & u_0 \end{pmatrix}, \begin{pmatrix} q_1 & r_2 & s_0 \\ q_0 & r_0 & s_0 \\ q_0 & r_0 & s_0 \end{pmatrix})$$

Definícia 4.3.3. Krokom výpočtu nazývame reláciu $\vdash$ nad konfiguráciami, pre ktorú platí:

(a) $(i,(j_0,\ldots,j_{i-1},l,j_{i+1},\ldots,j_{n-1}),F_0,\ldots,F_{i-1},aF_i,F_{i+1},\ldots,F_n,\omega,\mathcal{A},\mathcal{B}) \vdash$
$(i{+}1,(j_0,\ldots,j_{i-1},l{-}1,j_{i+1},\ldots,j_{n-1}),F_0,\ldots,F_{i-1},F_i,F_{i+1}\alpha,\ldots,F_n,\omega,\overline{\mathcal{A}},\mathcal{B})$,
kde

$$\mathcal{A}=\begin{pmatrix} q_{0,1} & \cdots & q_{0,|A|} \\ \vdots & \ddots & \vdots \\ q_{n-1,1} & \cdots & q_{n-1,|A|} \end{pmatrix} \quad \text{a} \quad \mathcal{B}=\begin{pmatrix} Q_{0,1} & \cdots & Q_{0,|B|} \\ \vdots & \ddots & \vdots \\ Q_{n-1,1} & \cdots & Q_{n-1,|B|} \end{pmatrix}$$

ak sú splnené tieto podmienky:

(i) $i<n$

(ii) $(A_{a_1\ldots a_k},B_{b_1\ldots b_l},\alpha)\in\delta(a)$ a
$q_{i,f(A_{a_1\ldots a_k})}\in F^{A_{a_1\ldots a_k}}$ a
$Q_{i,h(B_{b_1\ldots b_l})}\in F^{B_{b_1\ldots b_l}}$

(iii) $\mathcal{A}\overset{i,a}{\sqsubseteq}\overline{\mathcal{A}}$

(b) $(n,\vec{j},F_0,\ldots,F_{n-1},aF_n,\omega,\mathcal{A},\mathcal{B})\vdash(n,\vec{j},F_0,\ldots,F_{n-1},F_n,\omega a,\mathcal{A},\mathcal{B})$

(c) $(n,\vec{j},F_0,\ldots,F_{n-1},\varepsilon,\omega,\mathcal{A},\mathcal{B})\vdash(n-1,\vec{j},F_0,\ldots,F_{n-1},\varepsilon,\omega,\mathcal{A},\mathcal{B})$

(d) $(i,\vec{j},F_0,\ldots,F_n,\omega,\mathcal{A},\mathcal{B})\vdash(i-1,\vec{j},F_0,\ldots,F_n,\omega,\mathcal{A},\mathcal{B})$ ak sú splnené tieto podmienky:

(i) $0<i<n$

(ii) $|F_i|-1\le j_i<l$

(iii) aspoň jeden z frontov $F_0,\ldots,F_{i-1}$ je neprázdny

(e) $(i,(j_0,\ldots,j_{i-1},j_i,j_{i+1},\ldots,j_{n-1}),\varepsilon,\ldots,\varepsilon,F_i,\ldots,F_n,\omega,\mathcal{A},\mathcal{B})\vdash$
$(i,(j_0,\ldots,j_{i-1},j_i+1,j_{i+1},\ldots,j_{n-1}),\varepsilon,\ldots,\varepsilon,F_i,\ldots,F_n,\omega,\mathcal{A},\overline{\mathcal{B}})$ ak sú splnené tieto podmienky:

(i) $i<n$

(ii) $0\le|F_i|-1\le j_i<l$

(iii) $\mathcal{B}\overset{i,g}{\sqsubseteq}\overline{\mathcal{B}}$

(f) $(i,(j_0,\ldots,j_{i-1},j_i,j_{i+1},\ldots,j_{n-1}),F_0,\ldots,F_n,\omega,\mathcal{A},\mathcal{B})\vdash$
$(i,(j_0,\ldots,j_{i-1},j_i+1,j_{i+1},\ldots,j_{n-1}),F_0,\ldots,F_n,\omega,\mathcal{A},\overline{\mathcal{B}})$ ak sú splnené tieto podmienky:

(i) $i<n$

(ii) $|F_i| - 1 > j_i < l$

(iii) $\mathcal{B} \overset{i,a}{\sqsubseteq} \overline{\mathcal{B}}$, pričom $F_i = \gamma_1 a \gamma_2$ a $|\gamma_1| = j_i + 1$

(g) $(i, \vec{j}, \varepsilon, \ldots, \varepsilon, \varepsilon, F_{i+1}, \ldots, F_n, \omega, \mathcal{A}, \mathcal{B}) \vdash$
$(i+1, \vec{j}, \varepsilon, \ldots, \varepsilon, \varepsilon, F_{i+1}, \ldots, F_n, \omega, \mathcal{A}, \mathcal{B})$ ak sú splnené tieto podmienky:

(i) $i < n$

(ii) aspoň jeden z frontov $F_{i+1}, \ldots, F_n$ je neprázdny

(h) Všetky dvojice relácie $\vdash$ získame pomocou (a) až (g).

Časť (a) definície 4.3.3 určuje, za akých podmienok $(k,l)L$ generátor prepisuje znak na začiatku frontu $F_i, i \neq n$. Časti (b), (c) určujú, ako sa pracuje s frontom F_n. Časti (d), (e) určujú, ako sa pracuje s frontom $F_i, i \neq n$, keď v ňom nie je dostatočne veľa (teda aspoň l) znakov. Časť (e) určuje, za akých podmienok treba dať na vstup automatov pre pravý kontext špeciálny symbol g. Časť (f) určuje, za akých podmienok treba dať na vstup automatov pre pravý kontext určitý znak z frontu $F_i, i \neq n$.

Nasledujúce konfigurácie $(k,l)L$ generátora $\overline{G}$ sú v relácii podľa

(a)

$$(0, (2,0,0), aa, \varepsilon, \varepsilon, \varepsilon, \begin{pmatrix} t_1 & u_0 \\ t_1 & u_0 \\ t_1 & u_0 \end{pmatrix}, \begin{pmatrix} q_1 & r_2 & s_0 \\ q_0 & r_0 & s_0 \\ q_0 & r_0 & s_0 \end{pmatrix}) \vdash$$

$$(1, (1,0,0), a, aaa, \varepsilon, \varepsilon, \varepsilon, \begin{pmatrix} t_0 & u_1 \\ t_1 & u_0 \\ t_1 & u_0 \end{pmatrix}, \begin{pmatrix} q_1 & r_2 & s_0 \\ q_0 & r_0 & s_0 \\ q_0 & r_0 & s_0 \end{pmatrix})$$

(b)

$$(3, (1,1,1), \varepsilon, a, aa, aa, \varepsilon, \begin{pmatrix} t_0 & u_1 \\ t_0 & u_1 \\ t_0 & u_1 \end{pmatrix}, \begin{pmatrix} q_2 & r_0 & s_0 \\ q_1 & r_2 & s_0 \\ q_0 & r_1 & s_2 \end{pmatrix}) \vdash$$

$$(3, (1,1,1), \varepsilon, a, aa, a, a, \begin{pmatrix} t_0 & u_1 \\ t_0 & u_1 \\ t_0 & u_1 \end{pmatrix}, \begin{pmatrix} q_2 & r_0 & s_0 \\ q_1 & r_2 & s_0 \\ q_0 & r_1 & s_2 \end{pmatrix})$$

(c)

$$(3,(1,1,1),\varepsilon,a,aa,\varepsilon,aa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\end{pmatrix}) \vdash$$

$$(2,(1,1,1),\varepsilon,a,aa,\varepsilon,aa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\end{pmatrix})$$

(d)

$$(2,(1,1,1),a,aa,aa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\\ q_0 & r_1 & s_1\end{pmatrix}) \vdash$$

$$(1,(1,1,1),a,aa,aa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\\ q_0 & r_1 & s_1\end{pmatrix})$$

(e)

$$(0,(1,0,0),aa,\varepsilon,\varepsilon,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_0 & r_1 & s_1\\ q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\end{pmatrix}) \vdash$$

$$(0,(2,0,0),aa,\varepsilon,\varepsilon,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\end{pmatrix})$$

(f)

$$(0,(0,0,0),aa,\varepsilon,\varepsilon,\varepsilon,\varepsilon,\begin{pmatrix}t_1 & u_0\\ t_1 & u_0\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\end{pmatrix}) \vdash$$

$$(0,(1,0,0),aa,\varepsilon,\varepsilon,\varepsilon,\varepsilon,\begin{pmatrix}t_1 & u_0\\ t_1 & u_0\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_0 & r_1 & s_1\\ q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\end{pmatrix})$$

Poznámka: Reflexívny a tranzitívny uzáver relácie $\vdash$ označujeme $\vdash^*$.

Definícia 4.3.4. Jazykom $(k,l)L$ generátora G nazývame množinu

$$L(G)=\{w\in\Sigma^* \mid (0,\vec{0},w_0,\varepsilon,\ldots,\varepsilon,\varepsilon,\mathcal{A},\mathcal{B}) \vdash^* (i,\vec{j},\varepsilon,\varepsilon,\ldots,\varepsilon,w,\overline{\mathcal{A}},\overline{\mathcal{B}})\}$$

kde $\mathcal{A}$ je iniciálna matica stavov a $\mathcal{B}$ je matica počiatočných stavov.

Konfiguráciu $(0,\vec{(}0),w_0,\varepsilon,\ldots,\varepsilon,\varepsilon,\mathcal{A},\mathcal{B})$, kde $\mathcal{A}$ je iniciálna matica stavov a $\mathcal{B}$ je matica počiatočných stavov, budeme nazývať počiatočnou a

$$(i,\vec{(}j),\varepsilon,\varepsilon,\ldots,\varepsilon,w,\overline{\mathcal{A}},\overline{\mathcal{B}})$$

koncovou konfiguráciou.

Poznámka: Zápisy $K_1 \vdash^{(a)} K_2, K_1 \vdash^{(b)} K_2, \ldots, K_1 \vdash^{(g)} K_2$ znamenajú, že konfigurácie K_1, K_2 sú v relácii $\vdash$ podľa časti $(a), (b), \ldots, (g)$ definície 4.3.3.

Definícia 4.3.5. Akceptujúcim výpočtom $(k,l)L$ generátora G nazývame postupnosť konfigurácií začínajúcu počiatočnou a končiacu koncovou konfiguráciou, pričom každé po sebe idúce konfigurácie sú v relácii $\vdash_G$.

Pre názornosť uvedieme príklad akceptujúceho výpočtu generátora $\overline{G}$ z príkladu 4.3.1. Slovo aaa patrí do jazyka $(1,2)L$ generátora $\overline{G}$, pretože:

$$(0,(0,0,0),aa,\varepsilon,\varepsilon,\varepsilon,\varepsilon,\begin{pmatrix}t_1 & u_0\\ t_1 & u_0\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\end{pmatrix}) \vdash^{(f)}$$

$$(0,(1,0,0),aa,\varepsilon,\varepsilon,\varepsilon,\varepsilon,\begin{pmatrix}t_1 & u_0\\ t_1 & u_0\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_0 & r_1 & s_1\\ q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\end{pmatrix}) \vdash^{(e)}$$

$$(0,(2,0,0),aa,\varepsilon,\varepsilon,\varepsilon,\varepsilon,\begin{pmatrix}t_1 & u_0\\ t_1 & u_0\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\end{pmatrix}) \vdash^{(a)}$$

$$(1,(1,0,0),a,aaa,\varepsilon,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_1 & u_0\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_0 & s_0\\ q_0 & r_0 & s_0\end{pmatrix}) \vdash^{(f)}$$

$$(1,(1,1,0),a,aaa,\varepsilon,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_1 & u_0\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_1 & s_1\\ q_0 & r_0 & s_0\end{pmatrix}) \vdash^{(f)}$$

$$(1,(1,2,0),a,aaa,\varepsilon,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_1 & u_0\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\\ q_0 & r_0 & s_0\end{pmatrix}) \vdash^{(a)}$$

$$(2,(1,1,0),a,aa,aa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\\ q_0 & r_0 & s_0\end{pmatrix}) \vdash^{(f)}$$

$$(2,(1,1,1),a,aa,aa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\\ q_0 & r_1 & s_1\end{pmatrix}) \vdash^{(d)}$$

$$(1,(1,1,1),a,aa,aa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\\ q_0 & r_1 & s_1\end{pmatrix}) \vdash^{(d)}$$

$$(0,(1,1,1),a,aa,aa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\\ q_0 & r_1 & s_1\end{pmatrix}) \vdash^{(e)}$$

$$(0,(2,1,1),a,aa,aa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_0 & r_1 & s_2\\ q_0 & r_1 & s_1\end{pmatrix}) \vdash^{(a)}$$

$$(1,(1,1,1),\varepsilon,aa,aa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_0 & r_1 & s_2\\ q_0 & r_1 & s_1\end{pmatrix}) \vdash^{(e)}$$

$$(1,(1,2,1),\varepsilon,aa,aa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\\ q_0 & r_1 & s_1\end{pmatrix}) \vdash^{(a)}$$

$$(2,(1,1,1),\varepsilon,a,aaa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\\ q_0 & r_1 & s_1\end{pmatrix}) \vdash^{(f)}$$

$$(2,(1,1,2),\varepsilon,a,aaa,\varepsilon,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_1 & u_0\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\end{pmatrix}) \vdash^{(a)}$$

$$(3,(1,1,1),\varepsilon,a,aa,aa,\varepsilon,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\end{pmatrix}) \vdash^{(b)}$$

$$(3,(1,1,1),\varepsilon,a,aa,a,a,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\end{pmatrix}) \vdash^{(b)}$$

$$(3,(1,1,1),\varepsilon,a,aa,\varepsilon,aa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\end{pmatrix}) \vdash^{(c)}$$

$$(2,(1,1,1),\varepsilon,a,aa,\varepsilon,aa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\end{pmatrix}) \vdash^{(d)}$$

$$(1,(1,1,1),\varepsilon,a,aa,\varepsilon,aa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\\ q_0 & r_1 & s_2\end{pmatrix}) \vdash^{(e)}$$

$$(1,(1,2,1),\varepsilon,a,aa,\varepsilon,aa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_2 & r_0 & s_0\\ q_0 & r_1 & s_2\end{pmatrix}) \vdash^{(a)}$$

$$(2,(1,1,1),\varepsilon,\varepsilon,aa,\varepsilon,aa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_2 & r_0 & s_0\\ q_0 & r_1 & s_2\end{pmatrix}) \vdash^{(e)}$$

$$(2,(1,1,2),\varepsilon,\varepsilon,aa,\varepsilon,aa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\end{pmatrix}) \vdash^{(a)}$$

$$(3,(1,1,1),\varepsilon,\varepsilon,a,a,aa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\end{pmatrix}) \vdash^{(b)}$$

$$(3,(1,1,1),\varepsilon,\varepsilon,a,\varepsilon,aaa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\end{pmatrix}) \vdash^{(c)}$$

$$(2,(1,1,1),\varepsilon,\varepsilon,a,\varepsilon,aaa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\end{pmatrix}) \vdash^{(e)}$$

$$(2,(1,1,2),\varepsilon,\varepsilon,a,\varepsilon,aaa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_2 & r_0 & s_0\\ q_2 & r_0 & s_0\end{pmatrix}) \vdash^{(a)}$$

$$(3,(1,1,1),\varepsilon,\varepsilon,\varepsilon,\varepsilon,aaa,\begin{pmatrix}t_0 & u_1\\ t_0 & u_1\\ t_0 & u_1\end{pmatrix},\begin{pmatrix}q_2 & r_0 & s_0\\ q_2 & r_0 & s_0\\ q_1 & r_2 & s_0\end{pmatrix}) \vdash^{(c)}$$

Takto pracujúci $(k,l)L$ generátor odvodí práve všetky slová, ktoré odvodí $(k,l)L$ systém, podľa ktorého bol zostrojený. K podrobnému dôkazu tohto tvrdenia sa možno dočítať v [4] kapitola 6.3.

Výpočet $(k,l)L$ generátora prebieha v krokoch. V každom kroku potrebujeme

maximálne k pamäte pre uloženie konfigurácie, kde k je konštanta závislá od príslušného $(k, l)L$ systému. Teda celková použitá pamäť bude iba lineárne závislá od hĺbky odvodenia. Pamäťová zložitosť výpočtu $(k, l)L$ generátora je $\mathcal{O}(n)$. Formálny dôkaz možno nájsť v [4] kapitola 6.4.

4.4 Optimalizácia $(k,l)L$ generátora

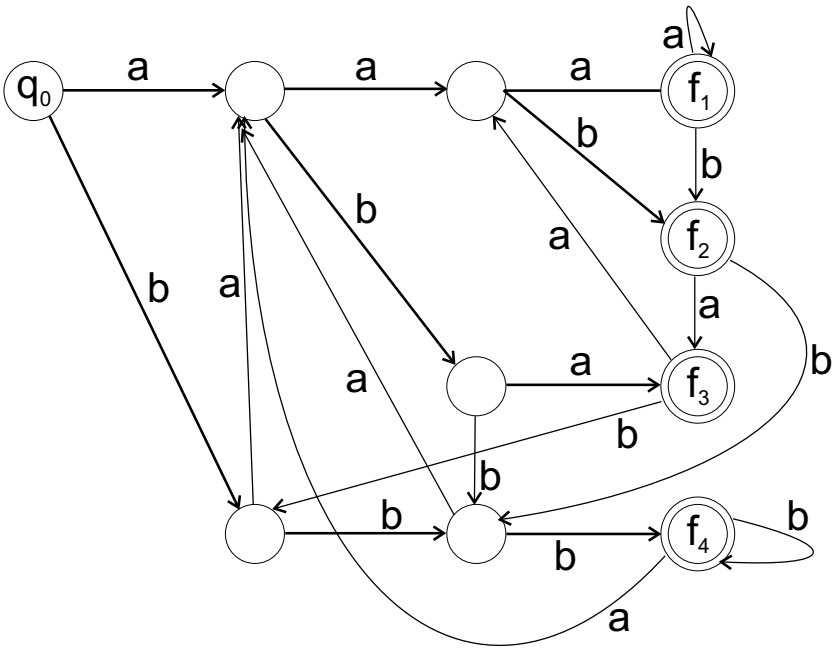
Uvedieme možnosti, ako upraviť návrh generátora pre optimálne využitie pamäte i procesorového času. Uvedené vylepšenia zlepšia výkonnosť a zmenšia nároky na pamäť. Asymptoticky sa však tieto charakteristiky zlepšiť nedajú.

4.4.1 Optimalizácia konečných automatov

V kapilote 4.3 sme uviedli návrh $(k,l)L$ generátora, ktorý pracuje s pamäťovou zložitosťou $O(n)$. Keď však zistíme, koľko automatov budeme potrebovať na rozpoznanie kontextu dĺžky k, zistíme, že ich bude $(|\Sigma|+1)^k$. To nie je veľmi výhodné, pokiaľ pracujeme s dlhými kontextami. Počet všetkých stavov takýchto automatov bude $(|\Sigma|+1)^k(k+1)$. Ukážeme si, ako túto situáciu zlepšiť. Zavedieme nový typ automatu. Použijeme jeden automat tohto typu pre pravý kontext a druhý pre ľavý kontext. Takéto nové automaty budú zastúpené v každom fronte.

Najskor neformálne popíšeme nový typ automatu, uvedieme formálnu definíciu, a napokon ukážeme, ako prispôsobiť pôvodný návrh generátora, aby sme ho mohli použiť.

Príklad 4.4.1. Nasledovný automat rozpoznáva vzorky $w_1 = aaa$,$w_2 = aab$, $w_3 = aba$ a $w_4 = baa$.



Takýto automat je v podstate známy *Aho-Corasic*-ov automat na rozpoznávanie reťazcov (viz. literatúra [6]), s tým, že má očíslované akceptačné stavy.

Index akceptačného stavu určuje číslo vzorky, ktorú automat rozpoznal.

My sa však zameriame len na vzorky s rovnakou dĺžkou, keďže všetky pravidlá $(k,l)L$ systému musia mať kontext rovnako dlhý. Takisto vieme, že ak sme už prešli prvých k krokov, musí byť automat v každom ďalšom kroku v akceptačnom stave. Podľa definície $(k,l)L$ systému vieme, že množina pravidiel by mala byť úplná a teda obsahuje všetky možné vzorky dĺžky k. Po zvážení týchto kritérií sa celý automat podstatne zjednoduší.

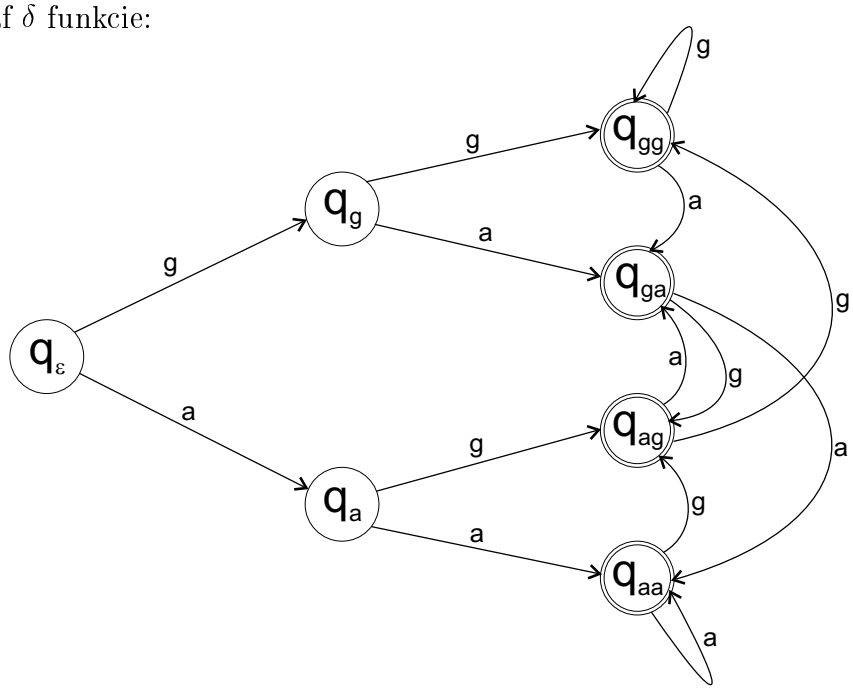
Označenie: Nový typ automatu nazveme *optimalizovaný automat rozpoznávajúci kontext.*

Uvedieme príklad automatu na rozpoznanie všetkých vzoriek dĺžky 2, ktoré sú tvorené znakmi $\{g,a\}$.

Príklad 4.4.2. Nech $A = (K, \Sigma, \delta, q_\varepsilon, F)$, pričom

$\Sigma = \{g,a\}$

graf δ funkcie:

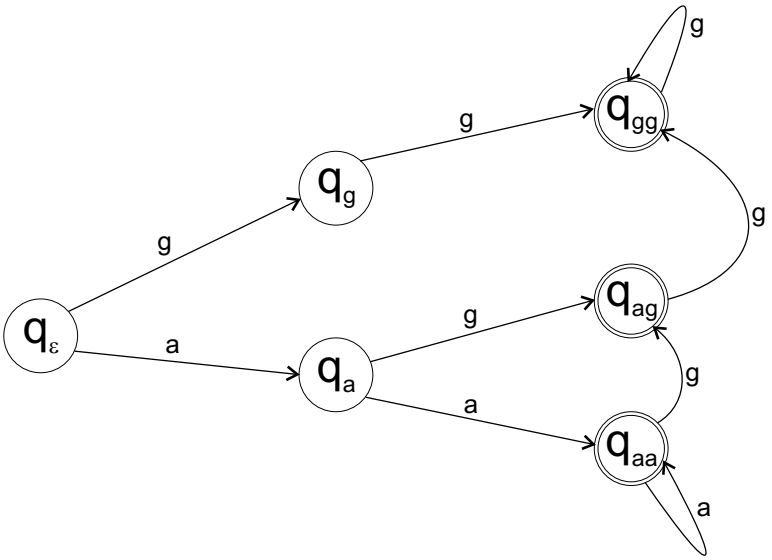


$F = \{q_{gg}, q_{ga}, q_{aa}, q_{ag}\}$

Uvedený automat rozpoznáva vzorky gg, ag, aa a ga.

V praxi však často nepotrebujeme všetky vzorky. Pokiaľ vieme, že niektoré vzorky sa vo výpočte určite nevyskytnú, možno ich v stavovom grafe automatu odstrániť. Napríklad v pravom kontexte sa nikdy nevyskytne vzorka $g^i a$ a v ľavom ag^j. Automat pre rozpoznanie pravého kontextu z príkladu 4.2.1 rozpoznáva 3 vzorky gg, ag, aa. Stavový graf nového automatu pre tieto vzorky bude vyzerať

nasledovne:



Všimnime si, že pokiaľ urobíme aspoň $k-1$ krokov, v každom ďalšom kroku sa automat pohybuje iba medzi akceptačnými stavmi. Pokiaľ sa takýto automat zasekne, tak sa vo výpočte vyskytla vzorka, ktorú sme neuvažovali. Pokiaľ nebude povedané inak, budeme vždy uvažovať automat s kompletnou sadou vzoriek.

Uvedieme formálnu definíciu optimalizovaného automatu na rozpoznávanie kontextu.

Definícia 4.4.1. Optimalizovaný automat rozpoznávajúci kontext dĺžky k pre vzorky w_1 až w_n je konečný automat $A = (K, \Sigma, \delta, q_\varepsilon, F)$, pričom:

$K = \{w \mid w \text{ je prefix ľubovoľnej vzorky } w_i\}$

$\Sigma = \{$ množina všetkých znakov vzoriek $\}$

$\delta(q_u, a) = q_w \iff (w = au \wedge |u| < k) \vee (u = a_1 \ldots a_k \wedge w = a_2 \ldots a_k a)$

$F = \{q_w \mid |w| = k\}$

Takto zavedený automat bude mať $\sum_{i=0}^{k} (|\Sigma| + 1)^i$ stavov, čo je $\frac{(|\Sigma|+1)^{k+1}-1}{|\Sigma|}$.

Dôvodov, prečo používať takýto typ automatu, je hneď niekoľko. Pri klasickej implementácii (veľa automatov) treba pri zisťovaní pravidla na prepísanie znaku zistiť, ktorý z automatov pre ľavý kontext a ktorý z automatov pre pravý kontext je v akceptačnom stave. Taký je však vždy maximálne jeden pre ľavý resp. pravý kontext. V novom riešení sú iba 2 automaty, jeden pre ľavý kontext, druhý pre pravý kontext. V nich sú akceptačné stavy označené indexom vzorky, ktorú

akceptovali. Teda namiesto hľadania akceptujúceho automatu sa stačí pozrieť na index akceptačného stavu.

V pôvodnom riešení nás každý krok výpočtu každého automatu stál procesorový čas. V novom riešení pracuje iba jeden automat pre každý s kontextov a preto sa počet krokov a teda aj čas strávený na krokoch výpočtu zmenší až $(|\Sigma| + 1)^k$ násobne, kde k je dĺžka spracovávaného kontextu. Nové riešenie teda znamená aj časové urýchlenie.

Veľa automatov znamená veľa pamäte. Pri implementácií je vhodné definovať prechodovú funkciu zvlášť a každý front si pamätá iba stav svojho automatu pre ľavý a pravý kontext. Veľkosť potrebnej pamäte na ukladanie aktuálnych stavov sa tiež zmenší $(|\Sigma| + 1)^k$ násobne.

Ako sme už spomenuli upravený generátor bude mať namiesto množín automatov $\mathcal{A}$ a $\mathcal{B}$ iba dva konečné automaty rozpoznávajúce kontext A a B (definícia 4.3.1). Patrične sa zmení aj konfigurácia takéhoto generátora (definícia 4.3.2). Namiesto $n(|\Sigma| + 1)^k + n(|\Sigma| + 1)^l$ aktuálnych stavov pôvodných konečných automatov na rozpoznávanie kontextu bude v každej konfigurácií $2n$ aktuálnych stavov nových automatov. Relácia $\sqsubseteq$ namiesto kroku výpočtov mnohých automatov predstavuje krok výpočtu nového automatu. Podobne sa upravia aj pravidlá pre krok výpočtu (definícia 4.3.1). Upravený generátor neurobil žiadnu principiálnu zmenu v priebehu výpočtu. Všetky princípy fungovania ostanú nezmenené.

4.4.2 Optimalizácia prepisovania pravidla

Ak uvážime veľkosť frontov, ktoré potreboval pôvodný generátor, zistíme, že je mininálne l, kde l je dĺžka pravého kontextu a maximálne dĺžka najdlhšej pravej strany spomedzi pravidiel. Uvedieme spôsob, pri ktorom nám vystačí front taký veľký, ako je dĺžka pravého kontextu l.

Pôvodný generátor pri prepisovaní zoberie znak z vrcholu fronty, a prepíše ho podľa toho, ktorý z automatov pre ľavý kontext a ktorý z automatov pre pravý kontext sú v akceptačnom stave. Pokiaľ vo fronte nie je dostatok symbolov pre zistenie pravého kontextu, generátor doplní znaky prepísaním znaku z vrchola predchádzajúceho frontu. Prepisovanie prebieha vcelku, teda do frontu sa dostane celá pravá strana použitého pravidla. V krajnom prípade vo fronte chýba 1 znak a prepísaním pravidla z predchádzajúceho frontu sa doplní najdlhšia pravá strana spomedzi pravých strán pravidiel.

Nové riešenie pracuje podobne, až na fakt, že pravidlo neprepisuje celé. Pokiaľ vo fronte chýba $m < l$ znakov (kde l je dĺžka pravého kontextu), tak predchádzajúci front prepíše najviac m znakov, a zapamätá si, ktoré pravidlo prepisuje a počet znakov, ktoré už prepísal. Pokiaľ sa k takémuto frontu výpočet generátora

vráti, tak prepíše zvyšné znaky, maximálne však l znakov a výpočet generátora pokračuje nasledujúcim frontom. Pokiaľ generátor prepísal celú pravú stranu pravidla, tak sa generátor správa ako po prepísaní pravidla v pôvodnom riešení.

4.5 Implementácia IL generátora

V tejto kapitole rozoberieme prácu s aplikáciou (k,l)L Generátor. Aplikácia je implementáciou návrhu z podkapitoly 4.3 s použitím vylepšení popísaných v podkapitole 4.4. Pri implementácií bol použitý programovací jazyk C++ a vývojový nástroj Borland C++ builder verzia 5.0. Aplikáciu možno nájsť na pribalenom CD-rom.

Hardvérové a softvérové požiadavky:

Operačný systém Windows NT/2000/XP

Podpora slovenského jazyka a fontov (nakoľko texty v programe sú písané po slovensky)

Ľubovolný osobný počítač, na ktorom tento operačný systém funguje.

Využitie programu:

Po zadefinovaní (k,l)L systému program umožňuje zobraziť vetnú formu systému po ľubovoľnom počte krokov (maximálne 4000).

Obmedzenia programu:

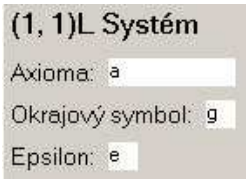
- Ľavý a pravý kontext môžu byť dlhé maximálne 10 znakov.
- Pravé strany pravidiel môžu byť dlhé maximálne 50 znakov.
- Počet pravidiel je obmedzený na 4000.
- Definícia (k,l)L systému vyžaduje kompletnú sadu pravidiel.
- Maximálna hĺbka odvodenia je obmedzená na 4000.

Použitie programu:
Pred spustením samotného generovania je nutné načítať zadanie (k,l)L systému. To možno urobiť buď zadaním nového systému, alebo otvorením uloženého systému.

Nový systém: Klikneme na položku *Nový* v menu *S*úbor.

Otvorí sa okno, kde nastavíme dĺžky ľavého a pravého kontextu a klikneme tlačidlo *vytvor*.

Následne zvolíme položky axioma, a znaky reprezentujúce špeciálne symboly ako g a ε. Povolené znaky sú malé a veľké písmená anglickej abecedy.



Následne pokračujeme pridávaním, prípadne odoberaním pravidiel.

Pridávanie pravidla:
Klikneme na tlačidlo *pridaj pravidlo*

V zobrazenom okne vyplníme jednotlivé časti pravidla. Program sám kontroluje prípustnú dĺžku a povolené znaky položiek. Potvrdíme kliknutím na *Pridaj*.

Zmazanie pravidla:
Označíme pravidlo, ktoré chceme zmazať a klikneme tlačidlo *vymaž pravidlo*.

Otvorenie systému:
Klikneme na položku *Otvoriť* v menu *S*úbor.

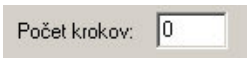
Vyberieme súbor s nahratým (k,l)L systémom.

Uloženie systému: Klikneme na položku *Uložiť* v menu *S*úbor.

Vyberieme súbor na uloženie.

Generovanie:

Spustenie generovania:
Zvolíme hĺbku odvodenia.

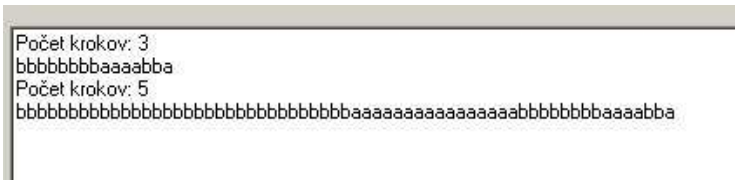


Ak sú vyplnené všetky potrebné údaje (axioma, Okrajový symbol, ε, hĺbka odvodenia), klikneme na tlačidlo *Spusti*

Prerušenie generovania:
Generovanie je možné prerušiť stlačením tlačidla *Zastav*

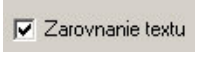
Výsledky generovania:

- Výsledky generovania sa zobrazujú v strednej časti hlavného okna programu.
- Výpis výsledku funguje konzolovým spôsobom. To znamená, že predchádzajúce výsledky sa pri novom generovaní nemažu. Pri zmene pravidiel alebo celého systému sa do výpisu pridá riadok *Zmena systému* kvôli lepšej prehľadnosti.
- Do výsledkovej konzoly je možné písať a tak dopĺňať vlastné poznámky.



Ostatné ovládacie prvky:
Na vymazanie výsledkovej časti slúži tlačidlo *Vyčisti*.

Zaškrtávacie políčko *Zarovnanie textu* zapína a vypína zalamovanie dlhých riadkov.



Práca s (k,l)L Generátorom je užívateľsky nenáročná. Program je určnený ako pomôcka k diplomovej práci.

Záver

Teória IL systémov nie je tak prepracovaná ako teória 0L a iných systémov bez interakcie. V literatúre je o IL systémoch veľa tvrdení bez dôkazov. V tejto práci som sa pokúsil situáciu napraviť. Okrem implementácie IL generátora a príkladov sú v práci moje dôkazy a algoritmy. (napríklad vety až o rozdelení kontextu v podkapitole 3.2, vety porovnávajúce triedy hierarchie IL jazykov a triedy chomského hierarchie v podkapitole 3.3 a iné) Ďalším prínosom do oblasti je vylepšenie návrhu IL generátora a jeho implementácia.

Ďalším rozšírením práce by mohol byť detailnejší a jemnejší návrh nedeterministického IL generátora, teda navrhnúť samotné odvodzovanie a premyslieť spôsob akým prezentovať vygenerované výsledky. V počítačovej grafike sa lepšie uplatnia upravené typy L-systémov. Preto by mohlo byť ďalším rozšírením navrhnúť IL-generátor, ktorý dokáže odfiltrovať pomocné značky (takzvané ignorované symboly) v priebehu výpočtu. Pamäťová zložitosť takéhoto generátora by mala ostať $\mathcal{O}(n)$. Prípadne sa zamerať na často používané zátvorkované (bracketed) L-systémy, o ktorých sa dá viac dočítať v [10].

Literatúra

[1] T. H. Cormen and C. E. Leiserson and R. L. Rivest: *Introduction to Algorithms*, MIT Press, Massachusetts, 1990.

[2] John E. Hopcroft, Jeffrey D. Ullman: *Formálne jazyky a automaty*, Alfa SNTL, 1978.

[3] Michal Chytil: *Automaty a gramatiky*, Matematický seminář SNTL, 1984.

[4] Dagmar Lúčna: *L Generátory*, FMFI UK, Bratislava, 1994.

[5] G. Rozenberg and A.Salomaa: *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.

[6] A.V. Aho and M.J. Corasick: *Efficient string matching*, Communications of the ACM, New York, 1975.

[7] G. Rozenberg and A.Salomaa: *The Book of L*, Springer-Verlag, Berlin, 1986.

[8] G. Rozenberg and A.Salomaa: *Lindenmayer Systems*, Springer-Verlag, Berlin, 1993.

[9] J. Dassow, G. Paun: *Regulated rewriting in Formal Language Theory*, Springer-Verlag, Berlin, 1995.

[10] P. Prusinkiewicz and A. Lindenmayer: *Algorithmic Beauty of Plants*, Springer-Verlag, New York, 1990.