Teória paralelných výpotov

verzia 0.8

12.02.2001

Obsah

Kapitola 1

Paralelné gramatiky

Skúsme sa trochu zamyslie nad tým, aký paralelizmus by sme v gramatikách mohli zavies. Jednou z moností by bolo, na rozdiel od gramatík Chomského hierarchie, kde sa v jednom kroku odvodenia prepisuje iba jeden neterminál, prepísa naraz vetky neterminály vo vetnej forme. Inou by mohlo by prepisovanie rovnakých neterminálov na jeden krok a ako uvidíme, existuje mnostvo alích modifikácií

1.1 Lindenmayerove systémy (L-systémy)

(: Pôvodná motivácia, ktorá dala vzniknú teoretickému modelu, o ktorom si teraz nieo bliie povieme, nebola ani zaleka taká blízka teórii jazykov, ako by si niekto mohol chybne myslie. Pán Lindenmeyer, poda ktorého je tento model pomenovaný, skúmal správanie sa istého druhu fylamentóznych organizmov, ktoré vyzerali asi takto: tvorila ich reaz buniek, priom kadá bunka (okrem krajných, ktoré majú po jednom) mala práve dvoch susedov, ktorí ju mohli, resp. nemuseli v jej správaní ovplyvova. Celý mechanizmus fungoval akoby v taktoch tak, e sa niekedy pár buniek rozhodlo, e sa rozmnoí (presnejie, e sa kadá z nich rozmnoí), niekedy zasa niektoré bunky odumreli, a teda z reazca akoby zmizli a inokedy sa s nimi ni nedialo :)

1.1.1 OL-systémy

Definícia 1.1.1. *OL-systém je trojica $G = (V, P, w)$, kde V je abeceda symbolov, $P \subseteq V \times V^*$ je mnoina prepisovacích pravidiel, $proj_1(P) = V$ (teda musí existova pravidlo pre kadý symbol abecedy L-systému), $w \in V^+$ nazývame axiom, alebo poiatoné slovo*[1]

Definícia 1.1.2. *Krok odvodenia je relácia $\Rightarrow$ na V^* definovaná nasledovne: $u \Rightarrow v$ práve vtedy, ak $u = a_1 \dots a_n$, $\forall i \;\, a_i \in V$, $v = b_1 \dots b_n$, $\forall i \;\, b_i \in V^*$ a $\forall i \;\, a_i \to b_i \in P$*

Definícia 1.1.3. *Jazyk*[2] *generovaný OL-systémom G je $L(G) = \{ \; u \in V^* \;\mid\; w \overset{*}{\Rightarrow} u \; \}$*

Definícia 1.1.4. *DOL-systém je taký OL-systém, kde $\forall a \in V \;\; \exists! \;\; a \to u \in P, u \in V^*$ (takémuto OL-systému hovoríme aj deterministický)*

[1] Je dobré si hne na zaiatku uvedomi rozdiel medzi L-systémami a gramatikami, s ktorými sme sa stretli v Chomského hierarchii, poiatoné slovo L-systému nemusí tvori iba jeden symbol, ba dokonca tu nerozliujeme medzi terminálmi a neterminálmi

[2] teda prakticky kadá vetná forma, ktorú z poiatoného slova dostaneme (vrátane jeho samého), patrí do jazyka $L(G)$, ak G je L-systém

Definícia 1.1.5. *POL-systém (propagating-pokraujúci) je taký OL-systém, ktorý je bez-ε*

Príklad 1.1.1. $G_1 = (\{a\}, \{a \to a^2\}, a)$ potom $L(G_1) = \{\ a^{2^n}\ \mid n \geq 0\ \}$
Ako vidie, OL-systém G_1 nám úplne jednoducho (pouitím jediného pravidla) umouje generova jazyk, na ktorý by nám v Chomského hierarchii nestaili ani bezkontextové prostriedky. Sila OL-systémov spoíva v paralelnom prepisovaní, kedy v jednom kroku odvodenia musíme poui prepisovacie pravidlá na vetky symboly. Skúsme ale do G_1 prida jedno pravidlo:
$G_1' = (\{a\}, \{a \to a, a \to a^2\}, a)$ potom $L_{G_1'} = a^+$ a veká sila je razom pre. Je tomu tak preto, lebo pridaním pravidla $a \to a$ sme umonili akoby rozsynchronizova odvodenie, a teda symboly sa teraz neprepisujú naraz, ale kadý v inom ase

Príklad 1.1.2. $G_2 = (\{a, b\}, \{a \to b, b \to ab\}, a)$ potom jazyk $L(G_2)$ budú tvori slová, ktorých dky sú leny Fibonacciho postupnosti, teda opä máme jeden dos zloitý, zrejme nie bezkontextový jazyk, ktorý dokáeme OL-systémom pomerne jednoducho generova

Príklad 1.1.3. $G_3 = (\{a, b, c\}, \{a \to abcc, b \to bcc, c \to c\}, a)$ potom dky slov jazyka $L(G_3)$ tvoria postupnos tvorcov (druhých mocnín) prirodzených ísel

Definícia 1.1.6. *$\mathcal{AFL}$ (Abstract Family of Languages) je kadá trieda jazykov obsahujúca nejaký neprázdny jazyk a uzavretá na $\cup, \cdot, +, h_\varepsilon, h^{-1}, \cap\mathcal{R}$*

Veta 1.1.1. *$\mathcal{L}_{OL}$ je anti-$\mathcal{AFL}$ (t.j. nie je uzavretá na iadnu $\mathcal{AFL}$ operáciu)*

Dôkaz: Pre kadú $\mathcal{AFL}$ operáciu ukáeme, e trieda $\mathcal{L}_{OL}$ na u nie je uzavretá

- $\cup$: $\{a\}, \{a^2\} \in \mathcal{L}_{OL}$ a sporom predpokladajme, e $L = \{a, a^2\} \in \mathcal{L}_{OL}$. Potom existuje nejaký OL-systém G, ktorý jazyk $\{a, a^2\}$ generuje. Ten ale musí ma nejaký axiom, môu nasta dve monosti:

  1. axiom je a - potom ale v G existuje pravidlo $a \to a^2$, lebo keby neexistovalo, tak by sme slovo a^2 nikdy nevyrobili, resp. by sme vyrobili iné slová, ktoré do L nepatria. Kee máme toto pravidlo, tak môeme vyrobi aj slová, ktoré do L nepatria (napr. a^4), o je spor s tým, e G generuje L
  2. axiom je a^2 - potom ak z neho chceme vyrobi a, tak musíme v G ma pravidlo $a \to \varepsilon$, ale aj $a \to a$, lebo keby sme ho nemali, tak vaka paralelnému prepisovaniu symbolov by sme dostali $\varepsilon \in L$, ale my vieme, e $\varepsilon \notin L$. Ale ak tu u máme tieto dve pravidlá, tak ε chtiac-nechtiac niekedy vyrobíme, o je opä v spore s tým, e $\varepsilon \notin L$

  Teda dostávame $L \notin \mathcal{L}_{OL}$ [3]

- $\cdot$: Zvolíme $L_1 = \{a\} \in \mathcal{L}_{OL}, L_2 = \{\varepsilon, a\} \in \mathcal{L}_{OL}$ ale $L_1 \cdot L_2 = \{a, a^2\} \notin \mathcal{L}_{OL}$ ako sme v predchádzajúcej asti ukázali

- $+$: Definujme $L = \{aa\} \cup \{b^{2^n} \mid n \geq 2\} \in \mathcal{L}_{OL}$, dôkaz urobíme sporom, nech G je OL-systém pre L^+:

  1. uvedomme si, e axiomom G môe by jedine aa, ak by tomu tak nebolo, axiomom by muselo by b^i pre nejaké $i \geq 4$, potom by sme ale nutne museli ma pravidlo $b \to a$ alebo $b \to a^2$, ale potom by sme vyrobili aj slovo tvaru $b^i a^j,\ \ i, j \neq 0$, ktoré do jazyka L^+ nepatrí

[3] Dostávame sa teda k mono troku prekvapujúcemu výsledku. Ukazuje sa, e i ke L-systémy v predchádzajúcom texte zvládli taký krkolomný jazyk ako bol $L(G_1)$, neporadia si s evidentne regulárnym jazykom obsahujúcim iba dve slová

2. $\varepsilon \notin L^+ \rightsquigarrow G$ je POL-systém
3. $a^4 \in L^+ \rightsquigarrow a \to a^2 \in P$ (nesmú tu by pravidlá tvaru $a \to a, a \to a^3$, pretoe by sme mohli vyrobi aj slovo $ab^i \notin L^+$)
4. $a^2 \overset{*}{\Rightarrow} b^4 \rightsquigarrow a \to b^i \in P$ pre $1 \leq i \leq 3$. Ak teraz pouijeme $a \to a^2$ a $a \to b^i$ dostaneme $a^2 \Rightarrow a^2 b^i \notin L^+$

- h_ε : Uvaujme jazyk $L = \{\varepsilon, a, a^2\} \in \mathcal{L}_{OL}$ a definujme homomorfizmus h nasledovne: $h(a) = a^2$, potom $h(L) = \{\varepsilon, a^2, a^4\} \notin \mathcal{L}_{OL}$, o sa dokáe rozbitím na prípady podobne ako pre $\cup$
- h^{-1} : Uvaujme jazyk $L = \{a\} \in \mathcal{L}_{OL}$ a homomorfizmus h daný predpisom $h(a) = a^2$. Potom $h^{-1}(L) = \emptyset \notin \mathcal{L}_{OL}$
- $\cap\mathcal{R}$: Uvaujme jazyk $L_1 = \{\varepsilon, a, a^2\} \in \mathcal{L}_{OL}$ a regulárny jazyk $L_2 = \{a^3\}^*$, dostávame rovnos $L_1 \cap L_2 = \{\varepsilon\} \notin \mathcal{L}_{OL}$ [4]

$\square$

Dôsledok 1.1.1. $\mathcal{L}_{OL}$ nie je uzavretá na substitúciu ani na zobrazenie a-prekladaom a nie je uzavretá ani na $\cap$ a ^{C} (komplement)

Dôkaz: o sa týka uzavretosti tejto triedy na zobrazenie a-prekladaom, princíp dôkazu je podobný ako pri predchádzajúcich uzáverových vlastnostiach. Kee $\mathcal{L}_{OL}$ nie je uzavretá na $\cap\mathcal{R}$, tak nemôe by uzavretá ani na $\cap$ veobecne. Uzavretos na ^{C} nechávame na itatea $\square$

Veta 1.1.2. $\mathcal{L}_{OL}$ *je uzavretá na zrkadlový obraz*

Dôkaz: Idea je rovnaká ako pre bezkontextové gramatiky. Je daný jazyk L. Nech $G = (V, P, w)$ je OL-systém taký, e $L(G) = L$. Zostrojíme OL-systém $G' = (V', P', w')$ tak, e $V = V'$, ak $a \to b_1 \dots b_n \in P$, potom $a \to b_n \dots b_1 \in P'$ a ak $w = w_1 \dots w_m, \forall i \;\; w_i \in V$, tak $w' = w_m \dots w_1$. Je zrejmé, e $L(G') = L^R$ $\square$

Veta 1.1.3. *Nech* $L \in \mathcal{L}_{OL}$ *je jazyk* $L \subseteq a^*$*, potom* $L^* \in \mathcal{L}_{OL}$

Dôkaz: Nech $L = L(G)$ kde $G = (\{a\}, P, a^m), m \geq 1$

1. L je konený:
   (a) $L = \{a\}$ alebo $L = \{\varepsilon, a\} \rightsquigarrow L^* = a^* \in \mathcal{L}_{OL}$
   (b) nech $L = \{w_1, \dots, w_n\}$, kde $w_1 \neq \varepsilon, w_1 \neq a$ (jazyk musí obsahova slovo dlhie ako $|a|$), potom $L^* = L(G')$, kde $G' = (\{a\}, \{a \to w_1, \dots, a \to w_n, a \to \varepsilon\}, w_1)$
2. L je nekonený: $\forall i \;\; 1 \leq i \leq m$ nech v_i je najkratie slovo v Ltaké, e $|v_i| \cong i \; mod \; m$ ak existuje, inak ho nedefinujeme. Nech G' je OL-systém tvaru $G' = (\{a\}, P', a^m)$, kde $P' = \{a \to \varepsilon\} \cup \{a \to u \mid a^m \underset{G}{\Rightarrow} u \text{ alebo } \exists i \;\; v_i \underset{G}{\Rightarrow} u\}$. Tvrdíme, e $L(G') = L^*$

   $\subseteq$: Kadé u v pravých stranách P' patrí do L a teda vetky slová generované OL-systémom G' patria do L^*, pretoe zaíname z $a^m \in L$, v alom kroku prepíeme kadý symbol bu na ε (a teda sa ho zbavíme), alebo ho prepíeme na slovo $\in L$, vetná forma má teda v kadom kroku tvar $w = w_1 \dots w_n, \;\; \forall i \;\; i = 1 \dots n \;\; w_i \in L \rightsquigarrow w \in L^*$

[4] $\{\varepsilon\} \notin \mathcal{L}_{OL}$, pretoe kadý OL-systém, ktorý by tento jazyk generoval, by musel ma ako axiom ε, o je z definície axiomu nemoné

$\supseteq$: Opä budeme kvôli lepej zrozumitenosti rozliova dva prípady:

A) $L \subseteq L(G')$ ukáeme MI vzhadom na poet krokov odvodenia nasledovne:

1° $a^m \in L$, súasne $a^m \in L(G')$

2° ak $x \in L$ a $x \underset{G}{\Rightarrow} y$, potom $x \underset{G'}{\Rightarrow} y$, lebo $x = (a^m)^j v_i$ pre nejaké i, j

$$\overbrace{\underbrace{aa\ldots a}_{m}\underbrace{aa\ldots a}_{m}\ldots\underbrace{aa\ldots a}_{m}\underbrace{\underbrace{aa\ldots a}_{m}\ldots\underbrace{aa\ldots a}_{m}\underbrace{aa\ldots a}_{i}}_{v_i}}^{x}$$

Pri prepisovaní x na y budeme postupova nasledovne: prvé písmenko zo skupiny a^m prepíeme na to, na o by sa to prepísalo celé v G, zvyok prepíeme na ε, to isté urobíme v asti v_i, inak povedané $y = u_1 u_2 \ldots u_j u_{j+1}$, kde $a^m \underset{G}{\rightarrow} u_i, 1 \leq i \leq j, v_i \underset{G}{\rightarrow} u_{j+1}$, teda $x \underset{G'}{\rightarrow} y$

B) Nech $w \in L^*$: ak $w = \varepsilon, w \in L$, tak je to zrejmé z A). Nech $w = w_1 \ldots w_k$, $w_i \in L$, $\forall i \;\; w_i \neq \varepsilon, k \geq 2, \forall i \;\; a^m \overset{*}{\underset{G'}{\Rightarrow}} w_i$

$a^m \overset{*}{\Rightarrow} (a^m)^k x$ pre nejaké $x \in a^*$ teda $a^m \overset{*}{\underset{G'}{\Rightarrow}} a^{mk} \overset{*}{\underset{G'}{\Rightarrow}} w^1 \ldots w_k$ teda sme nali nejaké odvodenie slova w, no ete musíme zabezpei, aby slová $w_1, \ldots, w_k$ vznikali synchrónne na rovnaký poet krokov (musíme akosi ntiahnu odvodenie tých w_i, ktoré vznikajú skôr ako ostatné. Urobíme to nasledovne: ak $a \underset{G'}{\Rightarrow} y$, tak odvodenie natiahneme[5] $a^m \underset{G'}{\Rightarrow} a^m a^t \underset{G'}{\Rightarrow} y\varepsilon$. Nali sme teda (existuje) odvodenie, kde odvodenia w_i majú rovnakú dku $\forall i$

□

Poznámka 1.1.1. *Ke $L \subseteq a^*$ je konený, tak $L^* \in \mathcal{L}_{OL}$*

Definícia 1.1.7. *M-mnoina je mnoina prirodzených ísel $M(n, m_1, \ldots, m_s)$, ktorej tvar je $M(n, m_1, \ldots, m_s) = \{n\} \cup$*

Veta 1.1.4. *(Charakterizácia nekonených OL-jazykov obsahujúcich ε)*
Nech $L \subseteq a^*$ je nekonený a obsahuje ε. Potom $L \in \mathcal{L}_{OL} \iff$ ke existuje M-mnoina M_L taká, e $L = \{a^i \mid i \in M_L\}$

Dôkaz: Dokáeme obe implikácie:

"$\Rightarrow$" Kee $L \in \mathcal{L}_{OL}$, tak existuje nejaký OL-systém G, ktorý ho generuje, urite musí obsahova (kee $\varepsilon \in L$) pravidlo $a \rightarrow \varepsilon$. Bez ujmy na veobecnosti môeme predpoklada, e pravidlá v G majú nasledovný tvar: $P = \{\; a \rightarrow \varepsilon, a \rightarrow a^{m_1}, \ldots, a \rightarrow a^{m_s} \;\}$, priom $m_1 < \cdots < m_s$ a $G = (\{a\}, P, a^n)$. M-mnoinu navrhneme nasledovne $M_L = \{n\} \cup \{k_1 m_1 + \cdots + k_s m_s\}$. Tvrdíme, e $L(G) = L(M_L)$

$\subseteq$: pre axiom a^n máme v M_L charakterizaný prvok n, vezmeme si teraz nejaké odvodenie v G a k nemu nájdeme prísluný prvok $m \in M_L$, ktorý ho charakterizuje. Odvodenie má tvar[6] $a^n \underset{G}{\Rightarrow} a^{m_{i_1}} \ldots a^{m_{i_k}} (k \leq n) \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} a^{k_1 m_1} a^{k_2 m_2} \ldots a^{k_s m_s}$, priom niektoré k_i (mono aj vetky, vaka $a \rightarrow \varepsilon$) sa môe rovna 0, staí zvoli $m = k_1 m_1 + \cdots + k_s m_s$, priom $\forall i$ je k_i rovnaké ako v odvodení

[5] toto máme existenciou neepsilonových pravidiel zabezpeené

[6] takto musí vyzera kadé odvodenie, lebo v kadom kroku sa a prepíe na nejaké a^{m_i} alebo na ε

$\supseteq$: zoberme prvok $m \in M_L$ a k nemu hadajme odvodenie slova $w \in L(G)$ takého, e $|w| = m$. Ak $m = n$, tak $w = a^n$ je axiom jazyka $L(G)$ a sme hotoví, majme teda $m = k_1m_1 + \cdots + k_sm_s \in M_L$. Odvodenie nájdeme nasledovným spôsobom: najskôr si vyrobíme dostatone vea symbolov a, konkrétne toko, aby ich bolo viac ako $\Sigma_{i=1}^{s} k_i$ (to ide, pretoe L je nekonený jazyk a tak v G musí existova pravidlo s potom symbolov na pravej strane väim ako 1), ke tieto symboly máme vyrobené, v jedinom aom kroku vyrobíme slovo w tak, e prvých $\Sigma_{i=1}^{s} k_i$ symbolov prepíeme na $a^{k_1m_1} \ldots a^{k_sm_s}$ (to ide jednoducho tak, e na prvých k_1 symbolov aplikujeme pravidlo[7] $a \rightarrow a^{m_1}$, na aích k_2 symbolov pravidlo $a \rightarrow a^{m_2}$ a tak alej, a na posledných k_s symbolov aplikujeme pravidlo $a \rightarrow a^{m_s}$), na zvyné symboly aplikujeme pravidlo $a \rightarrow \varepsilon$ a sme hotoví, pretoe sme nali odvodenie w v G

"$\Leftarrow$" Je daná M-mnoina $M_L(n, m_1, \ldots, m_s)$ taká, e $L(M_L) = \{a^i \mid i \in M_L\}$, potom $L(M_L) \in \mathcal{L}_{OL}$. Naou úlohou je teda nájs OL-systém G' taký, e $L(G') = L(M_L)$. Zvome $G' = G$. Dôkaz je potom úplne identický dôkazu prvej implikácie

$\square$

Poznámka 1.1.2. *$L_1 = \{a^{2^n} \mid n \geq 0\} \in \mathcal{L}_{OL}$, ale $L_1 \cup \{\varepsilon\} \notin \mathcal{L}_{OL}$. Tu sa ukazuje, aká dôleitá je existencia, resp. neexistencia prázdneho slova v jazyku*

1.1.2 Porovnanie $\mathcal{L}_{OL}$ s Chomského hierarchiou

Veta 1.1.5. *Kadá z tried $\mathcal{R}, \mathcal{L}_{CF} - \mathcal{R}, \mathcal{L}_{CS} - \mathcal{L}_{CF}$ obsahuje aj jazyky, ktoré sú v $\mathcal{L}_{OL}$, aj jazyky, ktoré nie sú v $\mathcal{L}_{OL}$*

Dôkaz: OL-jazyky v jednotlivých triedach sú:

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

Jazyky v prísluných triedach, ktoré nie sú v $\mathcal{L}_{OL}$:

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

$\square$

Poznámka 1.1.3. *Konené jazyky v $\mathcal{L}_{OL}$ sú (v abecede $\{a\}$) tvaru:*

1. *$\{a^m\}$ kde $m \geq 1$*
2. *$\{\varepsilon, a^m\}$ kde $m \geq 1$*
3. *$\{a^m, a^{m-1}, \ldots, \varepsilon\}$ kde $m \geq 1$*

[7] nesmieme zabúda na to, e pracujeme v OL-systéme, a teda v jednom kroku odvodenia musíme pravidlá aplikova na vetky symboly vo vetnej forme

Veta 1.1.6. *Kadý OL-systém (jazyk) obsahujúci ε, ktorý je v abecede $\{a\}$ je regulárny*

Dôkaz: Nech $L \in \mathcal{L}_{OL}$. Rozlíime dva prípady:

1. L je konený: vieme, e kadý konený jazyk je regulárny
2. L je nekonený: poda vety 1.1.4 vieme, e ak $L \in \mathcal{L}_{OL}$, tak existuje M-mnoina M_L taká, e $L = \{a^i \mid i \in M_L\}$. Nech $M_L = \{n, m_1, \dots, m_k\}$. Potom regulárna gramatika pre jazyk L bude: $G = (\{\sigma, A\}, \{a\}, \{\sigma \to a^n, \sigma \to A, A \to a^{m_1}A, \dots, A \to a^{m_k}A, A \to \varepsilon\}, \sigma)$

$\square$

Poznámka 1.1.4. *Kadý jazyk generovaný OL-systémom, v ktorom $a \to a \in P$ pre kadé $a \in V$, je bezkontextový*[8]

Poznámka 1.1.5. *Kadý jazyk $L \in \mathcal{L}_{CF}$ je tvaru $L' \cap R$ pre $L' \in \mathcal{L}_{OL}, R \in \mathcal{R}$*

Lema 1.1.1. *Nech $G = (V, P, w_0)$ je OL-systém, potom $\forall w \in L(G)$ existuje odvodenie, ktoré pouíva maximálne $k|w|$ priestoru, kde k je kontanta závislá od G*

Dôkaz: Definujme najskôr OL-systém G' nasledovne: $G' = (V \cup \{x_0\}, P \cup \{x_0 \to w_0\}, x_0)$, $x_0 \notin V$, platí $L(G') = L(G) \cup \{x_0\}$. Teraz ku kadému odvodeniu v G' uvaujme jeho strom[9]. Hovoríme, e odvodenie je redukované, ke jemu zodpovedajúci strom spa nasledujúcu podmienku: neexistuje podstrom T taký, e súasne platí:

1. vetky listy T sú ε
2. T obsahuje vetvu, v ktorej majú dva vrcholy rovnaké návestie

Pre kadé odvodenie slova w existuje redukované odvodenie (obr. 1.1), toto získame jednoducho tak, e ztotoníme rovnaké uzly v jednej vetve v podstrome T (ktorý obsahuje ako listy iba ε) originálneho stromu odvodenia.

Ozname m dku najdlhej pravej strany spomedzi vetkých pravidiel G' a $n = |V \cup \{x_0\}|$. Tvrdíme, e staí zvoli:

$$k = 3m^n$$

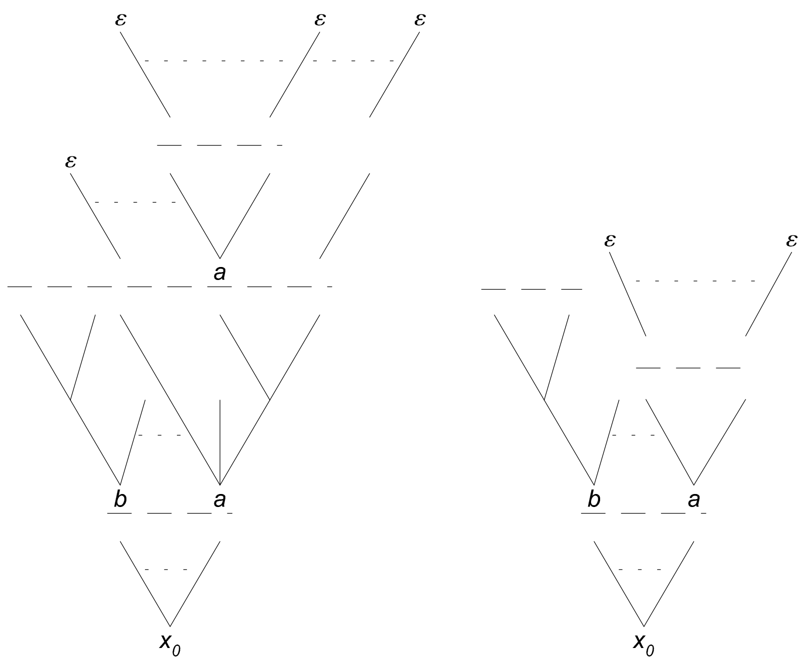
Pre $\forall w \neq \varepsilon, w \in L(G')$ s odvodením $x_0 \Rightarrow w_0 \Rightarrow w_1 \Rightarrow \cdots \Rightarrow w_n \equiv w$ bez újmy na veobecnosti predpokladajme, e toto je redukované. Výskyt symbolu a v jednom zo slov w_i nazveme neproduktívny $\Longleftrightarrow$ ak tento výskyt je návestím korea podstromu, ktorého vetky listy majú návestie ε. Inak výskyt nazveme produktívny. Pokia $w \neq \varepsilon$, tak w_i obsahuje najmenej jeden produktívny symbol, navye poet produktívnych symbolov vo $w_i \leq |w|$. Staí nám ukáza, e $\forall i$, ke Q je podslovo w_i a platí $|Q| = 3m^n$, potom Q obsahuje najmenej jeden produktívny symbol. Pre $i < n$ iadne podslovo w_i nie je dlhie ako $3m^n$. Pre $i = n + j, j \geq 0$ predpokladajme, e Q je podslovo $w_i, |Q| = 3m^n$. Q môeme zapísa nasledovne:

$$Q = Q_1Q_2Q_3, |Q_1| = |Q_2| = |Q_3| = m^n$$

Predpokladajme, e Q obsahuje iba neproduktívne symboly. Uvaujme nejaký výskyt symbolu a v Q_2. Existuje jediný výskyt symbolu b vo w_{i-n} taký, e a je návestie v strome T, ktorého kore má návestie b. alej, vobou Q, m, n vetky listy v T majú návestia ε. To je spor, lebo potom existuje v T vetva s dvoma uzlami s rovnakým návestím $\square$

[8] pravidlo $a \to a, \forall a \in V$ nám umoní si pri kadom kroku vybra jeden symbol a nahradi ho príslunou pravou stranou pravidla a na ostatné symboly poui pravidlo $a \to a$

[9] ako pre bezkontextové gramatiky - kore tvorí x_0, návestia vrcholov sú symboly $\in V \cup \{x_0\}$, hrany sú orientované a mnoina orientovaných hrán vycházdajúca z vrchola v má tvar $E(v) = \{v_1, \dots, v_k\} \Longleftrightarrow v \to v_1 \dots v_k \in P \cup \{x_0 \to w_0\}$



Obr. 1.1: Vytváranie redukovaného odvodenia

Veta 1.1.7. *(O lineárnom priestore)*
Nech A je Turingov stroj[10] *(TS) taký, e existuje k také, e pre kadé slovo w tento TS pouije pri práci na w najviac* $k|w|$ *polí́ok. Potom* $L(A) \in \mathcal{L}_{ECS}$

Veta 1.1.8. $\mathcal{L}_{OL} \subseteq \mathcal{L}_{ECS}$

Dôkaz: Pouijeme lemu 1.1.1 a na základe vety o lineárnom priestore, ktorá potom pre OL-jazyky platí, dostávame $L(G) \in \mathcal{L}_{ECS}$ $\square$

1.1.3 Rozírené OL-systémy (EOL-systémy)

Ako sa pri OL-systémoch ukázalo, paralelné prepisovanie symbolov nám istú silu pridalo, no fakt, e do jazyka sme museli zahrnú vetko, o sme vyrobili (kadú vetnú formu), nám vekú as náho optimizmu odobral. EOL-systémy nám dovolia opä (ako pri gramatikách Chomského hierarchie) filtrova vetné formy rozdelením mnoiny symbolov na terminály a neterminály. Do akej miery sa nám tým ná model zosilní (malo by by jasné, e jeho sila bude minimálne taká, ako sila OL-systémov) si ukáeme na príklade neskôr

Definícia 1.1.8. *EOL-systém je tvorica* $G = (N, T, P, w), P \subseteq (N \cup T) \times (N \cup T)^*$ *a* $\forall a \in (N \cup T)$ *existuje v P pravidlo*

Definícia 1.1.9. *Krok odvodenia je relácia* $\Rightarrow$ *na* $(N \cup T)^*$ *definovaná nasledovne:* $u \Rightarrow v$ *práve vtedy, ke* $u = a_1 \dots a_n, \forall i \ a_i \in (N \cup T), v = b_1 \dots b_n, \forall i \ b_i \in (N \cup T)^*$ *a* $\forall i \ a_i \rightarrow b_i \in P$

Definícia 1.1.10. *Jazyk generovaný EOL-systémom G je* $L(G) = \{x \in T^* \mid w \overset{*}{\Rightarrow} x\}$

[10] definície, popis modelu a iné v [1]

Príklad 1.1.4. $G_1 = (\{\sigma\}, \{a, b\}, \{\sigma \to a, \sigma \to b, a \to a^2, b \to b^2\}, \sigma)$, potom zjavne jazyk $L(G_1) = \{a^{2^n} \mid n \geq 0\} \cup \{b^{2^n} \mid n \geq 0\} \notin \mathcal{L}_{OL}$
Tento príklad ukazuje, e EOL-systémy sú silnejie ako obyajné OL-systémy a tým sme vlastne dali odpove na otázku, ktorú sme si poloili na zaiatku kapitoly, v tomto prípade sa sila neterminálu prejavila v tom, e nám umonila spravi zjednotenie dvoch jazykov

Príklad 1.1.5. $G_2 = (\{A\}, \{a, b\}, \{A \to A, A \to a, a \to a^2, b \to b\}, AbA)$ potom jazyk generovaný G je $L(G_2) = \{a^{2^n} b a^{2^m} \mid m, n \geq 0\} \notin \mathcal{L}_{OL}$

Príklad 1.1.6. $G_3 = (\{S, A, B, C, \bar{A}, \bar{B}, \bar{C}, F\}, \{a, b, c\}, P, S)$

$$P = \{ \quad \begin{array}{llll} S \to ABC & a \to F & b \to F & c \to F \\ A \to A\bar{A} & A \to a & \bar{A} \to \bar{A} & \bar{A} \to a \\ B \to B\bar{B} & B \to b & \bar{B} \to \bar{B} & \bar{B} \to b \\ C \to C\bar{C} & C \to c & \bar{C} \to \bar{C} & \bar{C} \to c \\ F \to F \ \} & & & \end{array}$$

Mono niekoho prekvapí fakt, e $L(G_3) = \{a^n b^n c^n \mid n \geq 1\}$, no ak sa nad systémom G_3 troku zamyslíme, tak sa nám táto skutonos ihne vyjasní. Ide tu toti o rafinované vyuitie neterminálu F na to, aby terminálne slová vznikali naraz (teda slová z jazyka $L(G_3)$ vznikajú v jednom kroku odvodenia prepísaním vetkých neterminálov na terminálne symboly). Ako si v nasledovnej vete ukáeme, tento jav nie je ani zaleka taký zriedkavý, ako by sa mohlo zda

Veta 1.1.9. *(Synchronizovaný tvar EOL-systému)*
Nech $G = (N, T, P, w)$ je EOL-systém pre jazyk L. Potom existuje taký EOL-systém G', $L(G') = L(G)$, e vetky jeho terminálne slová vznikajú z neterminálnych vetných foriem v jednom kroku odvodenia[11]

Dôkaz: Ukáeme kontrukciu synchronizovaného EOL-systému G'. Najskôr si vytvoríme nové neterminály, ktoré budeme potrebova:

- $\forall a \in T$ vytvoríme ξ_a (ak $a, b \in T$, tak $\xi_a \neq \xi_b$)
- F, σ', budú predstavova FALSE, resp. nový poiatoný symbol

Ozname $N' = \{\xi_a \mid a \in T\}$. Pre jednoduchos zaveme niekoko pojmov, ktoré neskôr vyuijeme:

1. Nech S je ubovoná mnoina reazcov z $(N \cup T)^*$, pre u definujeme mnoinu $A(S)$ nasledovne:
   (a) reazec $b_1 \ldots b_n \in A(S) \overset{def}{\Longleftrightarrow}$ ak existuje reazec $a_1 \ldots a_n \in S$ taký, e bu $b_i = a_i$ alebo $a_i \in T$ a $b_i = \xi_i$ (teda $b_i \in N'$), potom $A(S) \subset (N \cup N' \cup T)^*$
   (b) ak $S \neq \emptyset$, potom $A(S) \neq \emptyset$, lebo $S \subseteq A(S)$
2. $\delta_P(a)$ nazveme mnoinu vetkých pravých strán pravidiel pre symbol a z P

Definujme synchronizovaný EOL-systém G' nasledovne: $G' = (N \cup N' \cup \{F, \sigma'\}, T, P', \sigma')$, priom

$$\begin{array}{lll} P': & \delta_{P'}(\sigma') = A(\{w\}) & \\ & \forall a \in (T \cup \{F\}) & (a \to F) \in P' \\ & \forall a \in N & \delta_{P'}(a) = A(\delta_P(a)) \\ & \forall \xi_a \in N' & \delta_{P'}(\xi_a) = A(\delta_P(a)) \end{array}$$

[11] teda ke sa vo vetnej forme objaví prvý terminál, tak nutnou podmienkou, aby sme niekedy vygenerovali terminálne slovo je, aby v kroku odvodenia, kedy sa do vetnej formy dostal, "zterminálnela" vetná forma celá

Malo by by zrejmé, e teminálne slová musia v G' vznika naraz, pretoe ak sa náhodou nejaký terminál do venej formy dostane skôr ako ostatné, tak sa v aom kroku prepíe nutne na F a z F sa u nikdy terminál nestane. Rovnako zrejmé by malo by $L(G') = L(G)$, pretoe nové neterminály ξ_i vo vetnej forme akoby nahrádzali terminálne symboly, a tak simulovali odvodenie v pôvodnom systéme G $\square$

Veta 1.1.10. *Kadý konený jazyk je v* $\mathcal{L}_{EOL}$

Dôkaz: Nech L je konený, potom existuje regulárna gramatika G taká, e $L = L(G)$. Táto regulárna gramatika je ale zárove (po doplnení pravidiel $a \to a\ \forall a \in (N \cup T)$) aj hadaným EOL-systémom $\square$

Lema 1.1.2. *Nech G je EOL-systém, potom existuje EOL-systém G', ktorého vetky pravidlá obsahujúce $a \in T$ na avej strane sú tvaru $a \to a$*

Dôkaz: Zkontruujeme EOL-systém G' nasledovne: pre kadé $a \in T$ zavedieme nový neterminál ξ_a a upravíme pravidlá:

1. tie pravidlá, kde sa vyskytujú terminály, nahradíme novými tak, e vetky terminály (na avej i pravej strane) zmeníme na prísluné nové neterminály ($a \in T$ nahradíme $\xi_a \in N$)
2. pre kadé $a \in T$ pridáme pravidlo $a \to a$
3. pre kadé nové $\xi_a \in N$ pridáme pravidlo $\xi_a \to a$

$\square$

Veta 1.1.11. *Trieda $\mathcal{L}_{EOL}$ je uzavretá na $\cup, \cdot, +, \cap\mathcal{R}, h_\varepsilon$ a nie je uzavretá na h^{-1}*

Dôkaz: Dokáeme iba dve vlastnosti, väina ostatných dôkazov sa a na malé technické detaily vemi nelíi od známych kontrukcií pre bezkontextové gramatiky:

- $\cup$: Máme EOL-systémy G_1 pre L_1 a G_2 pre L_2. Ku G_1, G_2 spravíme normálne tvary G_1', G_2' poda kontrukcie z lemy 1.1.2. Bez ujmy na veobecnosti môeme predpoklada, e $N_1' \cap N_2' = \emptyset$. Vytvoríme G_3 pre $L_1 \cup L_2$ nasledovne: zavedieme nový neterminál σ_3 tak, e $\sigma_3 \notin N_1'$ a $\sigma_3 \notin N_2'$. alej $N_3 = N_1' \cup N_2' \cup \{\sigma_3\}$, $T_3 = T_1' \cup T_2'$, $P_3 = P_1' \cup P_2' \cup \{\sigma_3 \to \sigma_1' | \sigma_2'\}$ a nakoniec $G_3 = (N_3, T_3, P_3, \sigma_3)$
- h^{-1} : Zoberme jazyk $L \in \mathcal{L}_{EOL}, L = \{a^{2^n} \mid n \geq 0\}$ a definujme homomorfizmus h nasledovne: $h(a) = a, h(b) = \varepsilon$. Potom $h^{-1}(L) = \{w \in \{a,b\}^* \mid \#_a w = 2^n, n \geq 0\}$. Ukáeme, e $h^{-1}(L) \notin \mathcal{L}_{EOL}$. Nech G je EOL-systém pre $h^{-1}(L)$. Zoberme $w_0 \in h^{-1}(L)$, nech $\#_a w_0 = n$. Zoberme ubovoné dve a, medzi ktorými sú iba b. Ak je medzi týmito dvoma a "príli vea"[12] b, tak takéto slovo nedokáeme vyrobi (nemáme na to dostatok asu), o je spor s tým, e G je EOL-systém pre $h^{-1}(L)$

$\square$

Veta 1.1.12. $\mathcal{L}_{CF} \subsetneq \mathcal{L}_{EOL}$

Dôkaz: Kadá bezkontextová gramatika vyhovuje definícii EOL-systému (po doplnení pravidiel $a \to a\ \forall a \in (N \cup T)$), teda platí nevlastná inklúzia $\mathcal{L}_{CF} \subseteq \mathcal{L}_{EOL}$. e platí aj vlastná inklúzia, sme ukázali v príklade 1.1.6, kde sme nali EOL-systém pre jazyk $L = \{a^n b^n c^n \mid n \geq 1\} \in \mathcal{L}_{CS} - \mathcal{L}_{CF}$ $\square$

[12] tento poet vieme uri, závisí napr. od dky najdlhej pravej strany pravidla v G, bliie v [2]

Poznámka 1.1.6. *$1L$ a $2L$-systémy sú nejakou analógiou kontextových gramatík, i ke o ich sile by sa dali vies podobné diskusie ako pri OL-systémoch. Sná len toko, e obojstranný kontext je silnejí ako jednostranný a oba systémy sú silnejie ako OL-systémy*

1.1.4 TOL-systémy (table)

Tieto systémy nebudeme striktne definova, povieme si len základné veci, ktorými sa od klasických OL-systémov odliujú. TOL-systém $G = (V, P_1, \ldots, P_k, w_0)$ má niekoko (konkrétne k) sád pravidiel, v danom kroku odvodenia pouijeme na vetnú formu iba jednu sadu pravidiel

Príklad 1.1.7. $G = (\{a\}, \{a \to a^2\}, \{a \to a^3\}, a)$, potom $L(G) = \{a^{2^n} a^{3^m} \mid n, m \geq 0\} \notin \mathcal{L}_{OL}$

Poznámka 1.1.7. *Trieda $\mathcal{L}_{TOL}$ zdiea "zlé" uzáverové vlastnosti triedy $\mathcal{L}_{OL}$. Podobne ako pre $\mathcal{L}_{OL}$ platí aj pre $\mathcal{L}_{TOL}$ vzah $\mathcal{L}_{TOL} \subseteq \mathcal{L}_{ECS}$*

1.2 aie paralelné gramatiky

1.2.1 Indické paralelné gramatiky

Definícia 1.2.1. *Indická paralelná gramatika (IP) je tvorica $G = (N, T, P, \sigma)$, priom $\sigma \in N$ $N \cap T = \emptyset, P \subseteq N \times (N \cup T)^*$*

Definícia 1.2.2. *Krok odvodenia IP G je relácia $\Rightarrow$ ke sa vetky výskyty zvoleného neterminálu prepíu naraz tým istým pravidlom*

Príklad 1.2.1. $G_1 = (\{\sigma\}, \{a\}, \{\sigma \to \sigma\sigma, \sigma \to a\}, \sigma)$ potom $L(G_1) = \{a^{2^n} \mid n \geq 0\}$

Príklad 1.2.2. Dyckov jazyk nad dvoma písmenkami D_1 (jazyk správne uzátvorkovaných výrazov) nie je $\mathcal{L}_{IP}$

Dôkaz: Sporom predpokladajme, e $D_1 \in \mathcal{L}_{IP}$, potom existuje IP gramatika G taká, e $L(G) = D_1$. Ukáeme, e by musela ma nekonene vea neterminálov, o nie je moné. Pre jednoduchos oznamе $L = D_1$. Vytvoríme v L hierarchiu tvarov slov nasledovne: definujme rekurentne podmnoiny L a uvedomme si, koko najmenej neterminálov treba na generovanie slov z kadej z nich:

$L_1 = \{\ a^n b^n \mid n \geq 1\ \}^+$ treba aspo 2 neterminály

$L_{i+1} = \{\ a^n w b^n \mid w \in L_i, n \geq 1\ \}^+$ treba aspo $i + 2$ neterminálov

Ako vidno, táto hierarchia je nekonená, teda na generovanie $(\bigcup_{i=1}^{\infty} L_i) \subseteq L$ je treba nekonene vea neterminálov $\square$

Veta 1.2.1. $\mathcal{L}_{IP} \cap \mathcal{L}_{CF} = \mathcal{L}_{DBL}$

Poznámka 1.2.1. *$\mathcal{L}_{DBL}$ je trieda Derivation Bounded Languages, ku kadému jazyku z tejto triedy existuje k také, e poet neterminálov v kadej vetnej forme je mení ako k*

Veta 1.2.2. $\mathcal{L}_{IP} \subseteq \mathcal{L}_{ECS}$

Veta 1.2.3. *$\mathcal{L}_{IP}$ je uzavretá na $\cup, \cdot, *, h$*

Veta 1.2.4. *$\mathcal{L}_{IP}$ je neporovnatená s $\mathcal{L}_{EOL}$*

Veta 1.2.5. *$\mathcal{L}_{IP} \subseteq \mathcal{L}_{ETOL}$ (extended TOL)*

1.2.2 Ruské paralelné gramatiky

Definícia 1.2.3. *Ruská paralelná gramatika je pätica* $G = (N, T, P_1, P_2, \sigma)$*, priom* $\sigma \in N$*,* $N \cap T = \emptyset$*,* P_1, P_2 *sú konené mnoiny pravidiel,* $P_1, P_2 \subseteq N \times (N \cup T)^*$

Definícia 1.2.4. *Krok odvodenia: vetky výskyty zvoleného neterminálu sa prepíu naraz istým pravidlom z* P_1*, alebo sa prepíe práve jeden neterminál pravidlom z mnoiny* P_2

1.2.3 Absolútne paralelné gramatiky

Definícia 1.2.5. *Absolútne paralelná gramatika (AP) je tvorica* $G = (N, T, P, \sigma)$*, kde* $\sigma \in N$*,* $N \cap T = \emptyset$*,* P *je konená mnoina pravidiel tvaru* $(A_1, \ldots, A_n) \to (w_1, \ldots, w_n)$*, kde* $\forall A_i \in N$*,* $w_i \in (N \cup T)^*$

Definícia 1.2.6. *Krok odvodenia je relácia* $w \Rightarrow v$ *práve vtedy, ke* $w = u_1 A_1 u_2 A_2 \ldots u_n A_n u_{n+1}$ *a* $v = u_1 w_1 u_2 w_2 \ldots u_n w_n u_{n+1}$*, priom* $u_1, \ldots u_{n+1} \in T^*$ *(t.j. naraz prepisujeme vetky neterminály vo vetnej forme, na kadú monos výskytu neterminálov musíme ma pravidlo)*

Príklad 1.2.3. Pozrime sa na AP gramatiku:
$G = (\{S\}, \{a, b, c\}, \{(S) \to (SSS), (S, S, S) \to (aS, bS, cS), (S, S, S) \to (a, b, c)\}, S)$ potom $L(G) = \{a^n b^n c^n \mid n \geq 1\}$

Veta 1.2.6. $\mathcal{L}_{AP}$ *je* $\mathcal{AFL}$

Veta 1.2.7. $\mathcal{L}_{AP} \cap 2^{a^*} \subseteq \mathcal{R}$ *(t.j. jednopísmenkové AP jazyky sú regulárne)*

Dôkaz: Nech $L \in \mathcal{L}_{AP}$ a $G = (N, T, P, A_p)$ je AP gramatika taká, e $L(G) = L$. Bez újmy na veobecnosti môeme predpoklada, e $T = \{a\}$. Nech $N = \{A_1, \ldots, A_n\}$. Definujme N' nasledovne: $N' = \{A_p\} \cup \{A_{i_1,\ldots,i_k} \mid \forall j \; A_{i_j} \in N, \; A_{i_1}, \ldots, A_{i_k}$ sa nachádzajú na pravej strane niektorého (rovnakého) z pravidiel P v tomto poradí$\}$. Bez újmy na veobecnosti môeme predpoklada, e vetky pravidlá z P sú tvaru $(A_{i_1}, \ldots, A_{i_m}) \to (a^{k_1}\pi_1, \ldots, a^{k_m}\pi_m)$, kde $\forall i \; \pi_i \in N^*$. Definujme $\varphi_i = i_1, \ldots, i_l \iff \pi_i = A_{i_1} \ldots A_{i_l}$ Vytvoríme mnoinu pravidiel P' nasledovne:

$$A_{i_1,\ldots,i_k} \to a^{w_1 + \cdots + w_k} A_{\varphi_{i_1},\ldots,\varphi_{i_k}} \in P' \iff (A_{i_1}, \ldots, A_{i_k}) \to (a^{w_1}\pi_1, \ldots, a^{w_k}\pi_k) \in P$$

Teraz definujme $G' = (N', \{a\}, P', A_p)$. G' je zrejme regulárna gramatika a z kontrukcie vyplýva, e $L(G') = L(G)$ $\square$

Veta 1.2.8. $\mathcal{L}_{AP} = \mathcal{L}_{2DAP}$

Poznámka 1.2.2. $\mathcal{L}_{2DAP}$ *je trieda jazykov, ktoré sú generované dvojsmernými deterministickými a-prekladami*

Kapitola 2

Generatívne systémy

Doteraz sme sa zaoberali iba paralelnými modelmi, ktorých spolonou rtou bola akási “gramatiková filozofia”, teda mali sme mnoinu symbolov, poiatoný symbol, resp. poiatoné slovo a akúsi mnoinu prepisovacích pravidiel, ktoré striktne riadili odvodenie a celý mechanizmus generoval nejaký jazyk. Ukáeme si troku iný pohad na vec, nový z hadiska spôsobu prepisovania, kedy zostane zachovaná “viditená gramatiková as”, teda symboly, poiatoný symbol, zmení sa vak prepisovací mechanizmus. Ako uvidíme neskôr, pôjde o akúsi kombináciu gramatikového a automatového pohadu na jazyky. Pomocou tohto mechanizmu bude moné simulova u známe gramatiky, i u z Chomského hierarchie, alebo skôr spomenuté paralelné gramatiky. Zaujímavý (a vemi dôleitý) na tomto modeli je fakt, e pomerne jednoducho umoní porovna paralelné a sekvenné formalizmy z hadiska zloitosti, a tým ukáza, v ktorej oblasti nám paralelizmus pomôe a kde naopak nemá zmysel sa ním vemi zaobera.

2.1 Definície a oznaenia

Definícia 2.1.1. *Abstraktný generatívny systém je tvorica $G = (N, T, f, \sigma)$ kde N je mnoina neterminálov, T je mnoina terminálov, f je konene zadaná funkcia $(N \cup T)^* \to 2^{(N \cup T)^*}$, σ je poiatoný neterminál priom N, T nie sú nutne disjunktné abecedy.*

Definícia 2.1.2. *Krok odvodenia abstraktného generatívneho sytému je relácia $\Rightarrow$ na $(N \cup T)^*$ definovaná takto: $u \Rightarrow v$ práve vtedy ke $v \in f(u)$.*

Definícia 2.1.3. *Jazyk generovaný abstraktným generatívnym systémom G je mnoina $L(G)$, priom $L(G) = \{w \in T^* \mid \sigma \stackrel{*}{\Rightarrow} w\}$.*

V alom sa budeme zaobera peciálnym typom abstraktných generatívnych systémov, a to generatívnymi systémami.

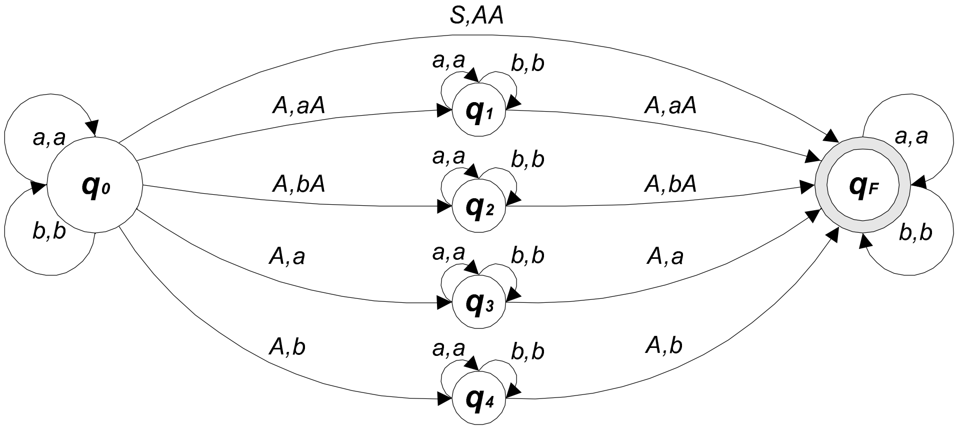
Definícia 2.1.4. *1-a-preklada je a-preklada $M = (K, \Sigma_1, \Sigma_2, H, q_0, F)$, v ktorom mnoina tvoríc je tvaru*[1] *$H \subseteq K \times \Sigma_1 \times {\Sigma_2}^* \times K$.*

Definícia 2.1.5. *Generatívny systém (g-systém) je abstraktný generatívny systém, v ktorom f je zobrazenie 1-a-prekladaom. Zapisujeme $G = (N, T, M, \sigma)$ kde M je 1-a-preklada.*

[1] 1-a-preklada nemá prunú ítaciu hlavu t.j. môe íta práve jeden symbol

Definícia 2.1.6. *Krok odvodenia g-systému je relácia $\Rightarrow$ na $(N \cup T)^*$ definovaná takto: $u \Rightarrow v$ práve vtedy*[2] *ke $v \in M(u)$.*

Príklad 2.1.1. Zostrojíme g-systém $G = (N, T, M, S)$ pre jazyk $L = \{ww \mid w \in \{a, b\}^*\}$. Neterminály budú $N = \{S, A\}$, terminály $T = \{a, b\}$, a-preklada M bude (obr.2.1).



Obr. 2.1: 1-a-preklada g-systému pre jazyk L z príkladu 2.1.1

Oznaenie: $\mathcal{G}$ trieda vetkých g-systémov.
$\mathcal{G}_\varepsilon$ trieda vetkých bez-ε g-systémov, t.j. 1-a-preklada nedáva na výstup ε.

2.2 Porovnanie generatívnej sily g-systémov s gramatikami Chomského hierarchie

Veta 2.2.1. $\mathcal{L}_\mathcal{G} = \mathcal{L}_{RE}$

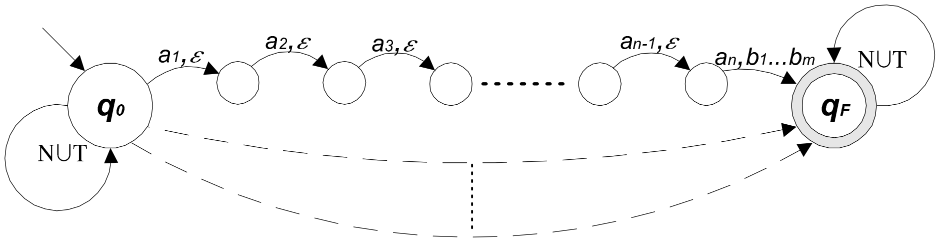
Dôkaz: Dokáeme obe inklúzie:

$\subseteq$: Táto inklúzia vyplýva z Turingovej tézy, ale iste by sme si vedeli predsavi aj TS, ktorý dostane na pásku slovo a na druhej páske simuluje odvodenie tohto slova g-systémom od poiatoného neterminálu a porovnáva, i je slovo z druhej pásky zhodné so vstupom na prvej páske.

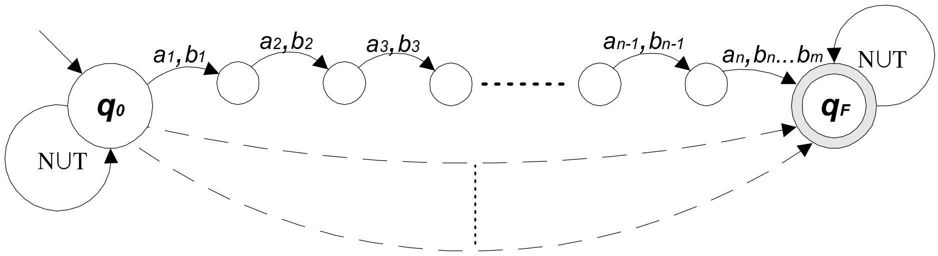
$\supseteq$: Pomocou g-systémov chceme simulova frázové gramatiky, t.j. ku kadej frázovej gramatike G treba zostroji g-systém G' taký, e $L(G') = L(G)$. G' zostrojíme nasledovne:
ku kadému pravidlu $a_1a_2 \ldots a_n \rightarrow b_1b_2 \ldots b_m$ urobíme v g-systéme cestu ako na obrázku 2.2.

□

[2] pre vstupné slovo u 1-a-preklada M vygeneruje výstup v



Obr. 2.2: Kontrukcia g-systému ku frázovej gramatike



Obr. 2.3: Kontrukcia g-systému ku kontextovej gramatike

Veta 2.2.2. $\mathcal{L}_{\mathcal{G}_\varepsilon} = \mathcal{L}_{CS}$

Dôkaz: Dokáeme obe inklúzie:

$\subseteq$: K bez-ε g-systému zostrojíme LBA akceptujúci ten istý jazyk. LBA si, rovnako ako TS v predchádzajúcej vete, vyrobí druhú pásku, na ktorej bude simulova odvodenie vstupného slova simulovaným g-systémom. Kee g-systém je bez ε, nemôe sa sta, e pri odvodení nejakého slova w vznikne vetná forma dky väej ako je $|w|$. Preto na simuláciu staí priestor ohraniený dkou vstupného slova, a teda vystaíme s LBA.

$\supseteq$: Kontrukcia je podobná ako v predchádzajúcej vete (obr.2.3) s vyuitím toho, e v kontextových gramatikách nie je pravá strana pravidla kratia ako avá, take v kontrukcii 1-a-prekladaa nepotrebujeme pravidlá s ε výstupom.

$\square$

Definícia 2.2.1. *Kopírovací cyklus v 1-a-prekladai je tvorica z H tvaru* (q, a, a, q).

Definícia 2.2.2. *g-systém je sekvenný, ak jediné cykly v 1-a-prekladai sú kopírovacie cykly v poiatonom alebo koncovom stave.*

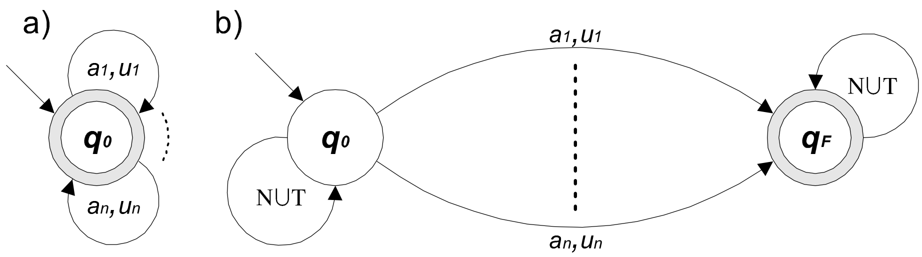
Oznaenie: $\mathcal{S}$ trieda vetkých sekvenných g-systémov.
$\mathcal{S}_\varepsilon$ trieda vetkých bez-ε sekvenných g-systémov.

Veta 2.2.3. $\mathcal{L}_{\mathcal{S}} = \mathcal{L}_{RE}, \mathcal{L}_{\mathcal{S}_\varepsilon} = \mathcal{L}_{CS}$

Dôkaz: Kontrukcia je tá istá ako vo vete 2.2.1, resp. 2.2.2, lebo ahko vidno, e zostrojený g-systém je sekvenný. □

2.3 Modelovanie známych gramatík pomocou g-systémov

Príklad 2.3.1. EOL: Nech mnoina pravidiel EOL je $P = \{a_1 \to u_1, a_2 \to u_2, \ldots, a_n \to u_n\}$ potom prísluný g-systém bude vyzera ako na obrázku 2.4a.



Obr. 2.4: Modelovanie EOL-systémov a CF-gramatík pomocou g-systémov

CF: Nech mnoina pravidiel CF je $P = \{a_1 \to u_1, a_2 \to u_2, \ldots, a_n \to u_n\}$ potom prísluný g-systém bude vyzera ako na obrázku 2.4b.

Poznámka 2.3.1. *Gramatika G je bezkontextová práve vtedy, ke existuje sekvenný g-systém G' s dvoma stavmi taký, e $L(G) = L(G')$. Implikácia "$\Rightarrow$" vyplýva z kontrukcie na obrázku 2.4b, dôkaz opanej implikácie nie je triviálny a presahuje rámec tohto textu.*

Veta 2.3.1. *Absolutne paralelné gramatiky sú ekvivalentné s g-systémami spajúce nasledujúce podmienky (ozn. $\tilde{\mathcal{G}}$ trieda takýchto g-systémov):*

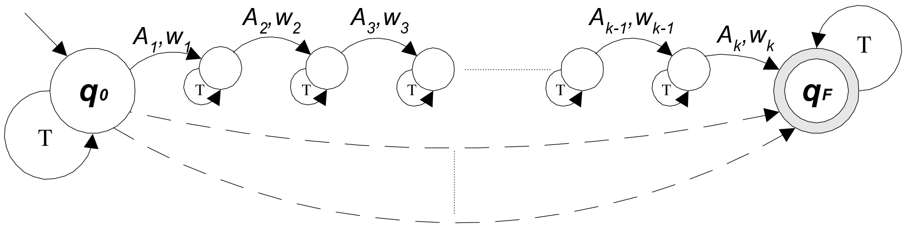
1. *kadý cyklus je kopírovací cyklus pre terminál*
2. *v kadom stave je kopírovací cyklus pre kadé $a \in T$*
3. *terminálne symboly sú prepisované len v kopírovacích cykloch*

Dôkaz: Máme dokáza, e $\mathcal{L}_{AP} = \mathcal{L}_{\tilde{\mathcal{G}}}$

$\subseteq$: K danej AP gramatike G zkontruujeme ekvivalentný g-systém $G' \in \tilde{\mathcal{G}}$ nasledovne. Kadému pravidlu v G tvaru $(A_1, \ldots A_k) \to (w_1, \ldots w_k))$ zodpovedá v 1-a-prekladai g-systému G' práve jedna cesta znázornená na obrázku 2.5.

$\supseteq$: Pre kadú cestu v 1-a-prekladai z q_0 do q_F (mimo kopírovacích cyklov) v AP gramatike urobíme pravidlo: $(A_1, \ldots A_k) \to (w_1, \ldots w_k))$.

□



Obr. 2.5: Kontrukcia g-systému k AP gramatike

2.4 Miery zloitosti

- $STATE(G)$ = poet stavov 1-a-prekladaa g-systému G
- $ARC(G)$ = poet pravidiel 1-a-prekladaa g-systému G
- $STATE_U(L) = \min\{STATE(G) \mid L = L(G), G \in U\}$
- $ARC_U(L) = \min\{ARC(G) \mid L = L(G), G \in U\}$
- $TIME(G, w) = \min\{k \mid \sigma \overset{k}{\underset{G}{\Rightarrow}} w\}; w \in L(G)$
- $TIME(G, n) = \max\{TIME(G, w) \mid w \in L(G), |w| \leq n\}$
- $TIME_U(f(n)) = \{L \mid \exists G \in U, L = L(G), TIME(G, n) = O(f(n))\}$
- $\overline{TIME}_U(f(n)) = \{L \mid \forall G \in U, L = L(G), TIME(G, n) = \Omega(f(n))\}$
- $SPACE(G, w) = \min\{SPACE(\alpha)\}$; kde $\alpha : \sigma \Rightarrow u_1 \Rightarrow \ldots \Rightarrow u_n$ je odvodenie w v G a $SPACE(\alpha) = \max\{|u_i| \mid 0 \leq i \leq n\}$

Veta 2.4.1. *Pre kadý nekonený jazyk L platí:*

1. $L \in \overline{TIME}_{\mathcal{S}}(n)$
2. $L \in \overline{TIME}_{\mathcal{G}}(\log n)$

Dôkaz:

1. Nech $G \in \mathcal{S}$. Zamyslime sa nad tým, aké najdlhie slovo môe vygenerova G na n krokov. G dokáe v jednom kroku prepisova len jeden súvislý úsek vetnej formy. Túto vetnú formu dokáe G predi o najviac

   $m = \max\{$ súet dok výstupov na jednej ceste v 1-a-prekladai - dka tejto cesty $\}$

   Majme odvodenie $\sigma = w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_i \Rightarrow w_{i+1} \Rightarrow \ldots \Rightarrow w_n$. Potom platí $|w_{i+1}| \leq |w_i| + m$. Kee $|w_0| = |\sigma| = 1$, tak $|w_n| \leq 1 + nm$. Z toho teda nakoniec plynie, e G potrebuje na vygenerovanie slova dky n aspo lineárny as.

2. Nech $G \in \mathcal{G}$. G môe v kadej vetnej forme prepísa vetky symboly. Ozname

   $$m = \max\{\text{ výstup v 1-}a\text{-prekladai }\}$$

   Potom platí $|w_{i+1}| \leq |w_i|.m$. Kee $|w_0| = |\sigma| = 1$, tak $|w_n| \leq m^n$. Z toho plynie, e G potrebuje na vygenerovanie slova dky n aspo logaritmický as

□

2.5 Porovnanie sekvenných a paralelných g-systémov

Veta 2.5.1. $\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{\mathcal{S}}, \mathcal{L}_{\mathcal{G}_\varepsilon} = \mathcal{L}_{\mathcal{S}_\varepsilon}$

Dôkaz: Tvrdenie priamo vyplýva z viet 2.2.1, 2.2.2 a 2.2.3 a hovorí nám, e paralelné a sekvenné g-systémy sú z hadiska generatívnej sily ekvivalentné, no nedáva nám iadny návod, ako ku paralelnému g-systému nájs sekvenný, rovnako ni nehovorí o tom, aký vplyv bude ma táto kontrukcia na popisnú zloitos.

Ukáeme si, ako k ubovonému g-systému $G \in \mathcal{G}$ zkontruujeme g-systém $G'' \in \mathcal{S}$ s ním ekvivalentný (t.j. $L(G) = L(G'')$) a potom si povieme, ako sa "zosekvennenie" odrazí na zloitosti.

Nech $G = (N, T, M, \sigma)$ je ubovoný g-systém, $L(G) \in \mathcal{L}_{\mathcal{G}}$, poiatoný stav M je q_0 a koncový q_F. Uvedomme si, o potrebujeme na G prepracova, aby sme ho "zosekvennili". V prvom rade potrebujeme ma v poiatonom aj koncovom stave kopírovacie cykly pre vetky[3] symboly, aby sme sa vo vetnej forme mohli presunú ubovone aleko, potom nieo a-prekladaom prepísa a vetnú formu dokopírova do konca. V druhom rade potrebujeme odstráni z G vnútorné cykly (teda vetky také, ktoré nie sú kopírovacími cyklami v poiatonom, resp. koncovom stave). Ke sa nám toto podarí, budeme s kontrukciou hotoví. G'' budeme kontruova v dvoch krokoch:

1. Zostrojíme $G' = (N', T, M', \underline{\overline{\sigma}})$ ekvivalentný s G s uitonými technickými vlastnosami, ktoré neskôr vyuijeme:

   (a) zavedieme nový poiatoný stav q'_0 a nový koncový stav q'_F a v týchto stavoch zkontruujeme kopírovacie cykly[4] $\forall a \in (N' \cup T)$, N' definujeme neskôr, oznamе mnoinu týchto kopírovacích cyklov $C = \{(q, a, a, q) \mid q \in \{q'_0, q'_F\}, a \in (N \cup T)\}$.

   (b) G' bude udrova informáciu o prvom a poslednom symbole vetnej formy tak, e ich oznaí: prvý symbol horným prúkom, posledný dolným prúkom. Oznaené symboly budú nové neterminály, v G' musíme vytvori nové podiarknuté aj nadiarknuté symboly, teda $N' = N \cup \{\overline{a} \mid a \in (N \cup T)\} \cup \{\underline{a} \mid a \in (N \cup T)\} \cup \{\underline{\overline{\sigma}}\}$. Poiatoný neterminál G' bude $\underline{\overline{\sigma}}$. Vetná forma bude vyzera nasledovne: $\overline{a}_1 a_2 a_3 \ldots a_{n-1} \underline{a_n}$.

   Teraz pome kontruova G' tak, aby sme zabezpeili obe "dobré" vlastnosti, ktoré sme uviedli a zárove aby boli G a G' ekvivalentné. Zrejme bude potrebné upravi prechodovú mnoinu 1-a-prekladaa. Ak H bola prechodová mnoina M, tak $H_{M'}$ bude prechodová mnoina M'. Zatia definujme H' nasledovne (nech $a, d, A, B \in (N \cup T), b, c \in (N \cup T)^*$):

   $$H' = H \cup C \cup \{(q'_0, \overline{A}, \overline{a}b, q_1) \mid (q_0, A, ab, q_1) \in H\} \cup \{(q, \underline{B}, c\underline{d}, q'_F) \mid (q, B, cd, q_F) \in H\}$$

   Ak sa nad H' trochu zamyslíme, tak je jasné, e v q'_0, resp. v q'_F síce máme kopírovacie cykly, ale nebudeme ich pouíva. Keby sme toti pouili ubovoný kopírovací cyklus v q'_0, tak sa u z tohto stavu nikdy nedostaneme, pretoe z neho vedú iba ípky na nadiarknutý, teda prvý, symbol vetnej formy. o sa týka stavu q'_F, tak do neho sa mono dosta iba na podiarknutý, teda posledný, symbol vetnej formy.

   Aby sme nau kontrukciu príli nepretechnizovali, nebudeme sa zaobera detailami ohadom poiatoného neterminálu $\underline{\overline{\sigma}}$ a vynecháme aj prípady, kedy M "príli vea" vymazáva, teda ak sa v odvodení môe vyskytnú vetná forma obsahujúca jediný symbol, prípadne ε. Pozrime sa teraz na g-systém, ktorý sme zkontruovali. Keby sme uvaovali $G \in \mathcal{G}_\varepsilon$, tak u teraz máme (a na prvý a posledný symbol, s ktorým budeme pracova neskôr) G' ekvivalentný s G a splnené (a)

[3]ako sa neskôr ukáe, tak nie nutne pre úplne vetky, staí pre niektoré

[4]Zrejme by nestailo doda kopírovacie cykly do q_0, resp. q_F. Mohlo by sa toti ahko sta, e by sme takto zkontruovaným 1-a-prekladaom akceptovali aj to, o 1-a-preklada v G neakceptoval

aj (b). Ale my uvaujeme $G \in \mathcal{G}$. Uvedomme si, aké nepríjemnosti nám môe spôsobi skutonos, e 1-a-preklada môe zapisova na výstup ε. Môe sa sta, e v H existuje $(q_0, A, \varepsilon, q_1)$. Poda doterajej kontrukcie by sme do H' pridali $(q_0', \overline{A}, \varepsilon, q_1)$. Tým by sme ale vymazali prvý symbol vetnej formy a 1-a-preklada by nepracoval v aích krokoch odvodenia správne. Podobne sa nám môe sta, e v H existuje (q, B, ε, q_F), potom by sme do H' pridali $(q, \underline{B}, \varepsilon, q_F')$, teda by sme si zmazali posledný (podiarknutý) symbol vetnej formy a v aom kroku odvodenia by nastali problémy s akceptovaním. Ukazuje sa teda, e predchádzajúcu kontrukciu mono poui iba na tie (q, a, u, p), kde $u \neq \varepsilon$ (zámerne sme do tretej komponenty vdy písali reazec $w \in (N' \cup T)^+$).

Pre ε-výstupy 1-a-prekladaa pouijeme nasledovnú kontrukciu:

(a) 1-a-prekladau M' pridáme nové stavy tak, e pre kadý stav q 1-a-prekladaa M vyrobíme nový stav $\overline{q}$. V týchto stavoch si budeme pamäta, e sme si zmazali prvý symbol vetnej formy a ke budeme robi v danom kroku odvodenia prvý ne-ε výstup, tak prvý symbol z neho bude zárove aj prvým symbolom vetnej formy, a tak mu pridáme iaru hore. Teraz to vetko formálne zapíeme. Oznanme $\overline{H}$prechodovú mnoinu, ktorú definujeme nasledovne ($a, b \in (N \cup T)$, $u \in (N \cup T)^*$):

$$\begin{aligned} \overline{H} = \{(q_0', \overline{A}, \varepsilon, \overline{q}_1) \mid (q_0, A, \varepsilon, q_1) \in H\} \cup & \quad \text{pamätáme si, e sme zmazali } \overline{A} \\ \cup \{(\overline{q}, a, \varepsilon, \overline{p}) \mid (q, a, \varepsilon, p) \in H\} \cup & \quad \text{stále sme nezapísali prvý symbol} \\ \cup \{(\overline{q}, a, \overline{b}u, p) \mid (q, a, bu, p) \in H\} & \quad \text{zapíeme } \overline{b}u \end{aligned}$$

(b) 1-a-prekladau M' pridáme nové stavy tak, e pre kadý stav q 1-a-prekladaa M vyrobíme nový stav $\underline{q}$. Tieto stavy budeme pouíva na to, aby sme si nezmazali posledný (podiarknutý) symbol vetnej formy. Tu to vak nebude také jednoznané ako pri prvom symbole. Pretoe 1-a-preklada sa nemôe vraca, tak po tom, ako zmae posledný symbol, u nemôe oznai iarou dole nejaký iný. Na to, e raz zmae posledný symbol vetnej formy, musí prís "dostatone skoro", vyuijeme monos nedeterminizmu. Uhádneme, e istý symbol vo vetnej forme je posledný, ktorý reálne zapisujeme, a e to, o vznikne prepísaním symbolov za ním budú iba ε, a potom len kontrolujeme, i sme hádali správne. Formálne zapísané v prechodovej mnoine $\underline{H}$ ($w \in (N \cup T)^*$):

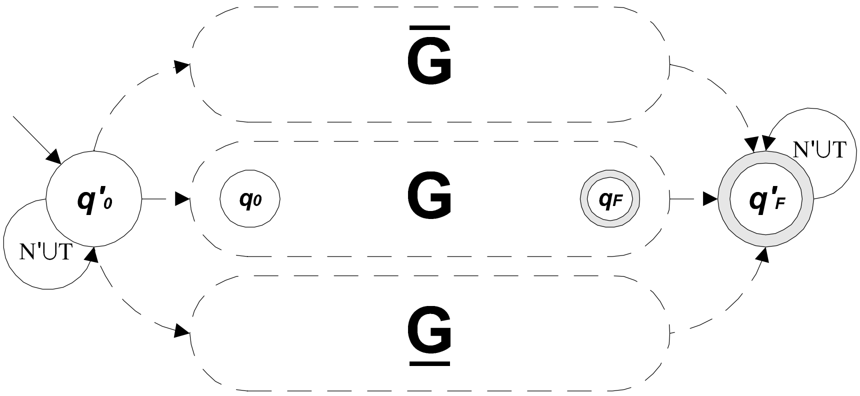
$$\begin{aligned} \underline{H} = \{(q, a, w\underline{b}, \underline{p}) \mid (q, a, wb, p) \in H\} \cup & \quad \text{nedet. zapíe nový posledný symbol} \\ \cup \{(\underline{q}, a, \varepsilon, \underline{p}) \mid (q, a, \varepsilon, p) \in H\} \cup & \quad \text{stále mae vetky symboly} \\ \cup \{(\underline{q}, \underline{a}, \varepsilon, q_F') \mid (q, a, \varepsilon, q_F) \in H\} & \quad \text{ak zmae posledný symbol, akceptuje} \end{aligned}$$

Vytvorili sme akoby virtuálne g-systémy $G, \underline{G}, \overline{G}$ (obr.2.6), priom v jednom kroku odvodenia sa môeme medzi nimi pohybova. Neexistujú vak iadne ípky $G \to \overline{G}$, $\underline{G} \to G$, $\underline{G} \to \overline{G}$.

Na to, aby sme mali $L(G) = L(G')$ ete potrebujeme v istom kroku odznai prvý aj posledný oznaený symbol. Avak kee máme v 1-a-prekladai g-systému G' ípky z q_0' do iného stavu, resp. ípky do q_F' z iného stavu (teda nie kopírovacie cykly v týchto stavoch) iba na oznaený symbol, tak s odznaenou vetnou formou u alej nepohneme. Z toho plynie, e symboly mono odznai len v poslednom kroku odvodenia, teda vtedy, ak vetky symboly vetnej formy, okrem prvého a posledného, sú terminály a aj prvý a posledný symbol (odhliadnuc od oznaenia) sú terminály. Teda opä nedeterministicky hádame posledný krok. Formálne definujme:

$$H_T = \{(q_0', \overline{a}, a, q_T), (q_T, a, a, q_T), (q_T, \underline{a}, a, q_F') \mid a \in T\}$$

q_T je nový stav 1-a-prekladaa g-systému G'. Zachovali sme konzistentnos v tom zmysle, e z q_0' vychádzajú, okrem kopírovacích cyklov, iba ípky na prvý (nadiarknutý) symbol a do q_F' vchádzajú, okrem kopírovacích cyklov, iba ípky na posledný



Obr. 2.6: 1-a-preklada g-systému G'

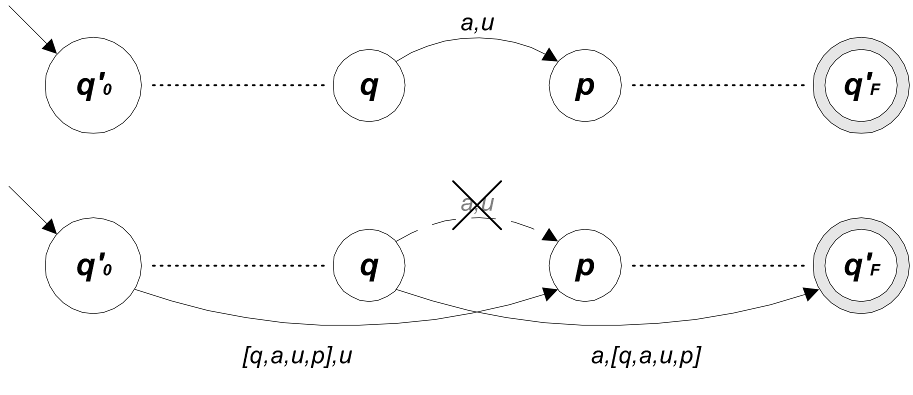
(podiarknutý) symbol vetnej formy, teda máme $L(G) = L(G')$, priom $G' = (N', T, M', \underline{\overline{\sigma}})$. o sa týka M' pre nás je zaujímavé, ako sa zmenila prechodová mnoina: ke v M to bola H, tak v M' to bude $H_{M'} = H' \cup \overline{H} \cup \underline{H} \cup H_T$. Pokia ide o stavy a abecedy, ich kontrukcia by mala by z u uvedeného zrejmá.

2. Ku $G' = (N', T, M', \underline{\overline{\sigma}})$ zostrojíme ekvivalentný g-systém G'' tak, e preruíme vetky vnútorné cykly (teda tie, ktoré nie sú kopírovacie v poiatonom alebo koncovom stave). Vonkajou ípkou 1-a-prekladaa nazveme takú, ktorá bu vychádza z poiatoného stavu, alebo vchádza do akceptaného stavu (v naom prípade ide o stavy q'_0, q'_F). Ostatné ípky nazveme vnútorné. Pri odstraovaní cyklov budeme postupova nasledovne:

    (a) stavy budú rovnaké ako v G'

    (b) vetky vonkajie ípky necháme tak ako sú

    (c) kadú vnútornú ípku nahradíme dvoma novými nasledovne: ak $(q, a, u, p) \in H_{M'}$, tak $(q, a, u, p) \notin H_{M''}, (q, a, [q, a, u, p], q'_F) \in H_{M''}, (q'_0, [q, a, u, p], u, p) \in H_{M''}$, priom $[q, a, u, p]$ budú nové neterminály[5] (obr.2.7)



Obr. 2.7: Nahrádzanie vnútorných ípok v G''

[5]Ak sa itate trochu zamyslí, malo by mu by jasné, e v kadom kroku môe by vo vetnej forme najviac jeden takýto neterminál

Môe sa naskytnú otázka: preo sme nenahradzovali aj vonkajie ípky? Odpove je vemi jednoduchá: pretoe ony iadne cykly vytvára nemohli, tak sme G' kontruovali. Malo by by zrejmé, e takto sme odstránili vetky cykly, okrem kopírovacích v poiatonom a koncovom stave, lebo teraz vetky ípky vychádzajú z q'_0 alebo vchádzajú do q'_F a toto sú jediné ípky v G''. Rovnako zrejmé by malo by $L(G') = L(G'')$, v G'' sme iba nahradili jeden paralelný krok n sekvennými, kde n je poet ípok v jednom paralelnom kroku.

Máme zkontruovaný g-systém G'' taký, e $L(G) = L(G'')$ a $G'' \in \mathcal{S}$, teda nae "zosekvenovanie" g-systému G je na konci. □

Nasledujúce tvrdenia hovoria nieo o zloitosti g-systému ktorý sme práve zostrojili v predchádzajúcej kontrukcii a priamo z nej vyplývajú.

Veta 2.5.2. *(Vzah popisnej zloitosti sekvenných a paralelných g-systémov)*

- $\forall L \ STATE_{\mathcal{S}}(L) \leq 3.STATE_{\mathcal{G}}(L) + 3$
- $ARC_{\mathcal{S}}(L) \leq 32.STATE_{\mathcal{G}}(L) + 22.\#\Sigma_L$

Je dobré si uvedomi, e zosekvennením g-systému sme nepouili iadny priestor navye, o om hovorí nasledujúca veta.

Veta 2.5.3. $SPACE_{\mathcal{S}}(f(n)) = SPACE_{\mathcal{G}}(f(n))$ *pre* $\forall f(n)$

Veta 2.5.4. *Pre ubovoný jazyk L a ubovoný $G \in \mathcal{G}$ taký, e $L = L(G)$ platí:*

$$L \in TIME_{\mathcal{S}}(SPACE_{\mathcal{G}}(G,n).TIME_{\mathcal{G}}(G,n))$$

Veta 2.5.4 hovorí o akomsi hornom odhade potu krokov sekvenných g-systémov v porovnaní s paralelnými. Na bliie pochopenie tohto tvrdenia si staí uvedomi, e sekvenný g-systém môe v jednom kroku vaka tomu, e nemá vnútorné cykly, prepísa len istý, vo veobecnosti malý, úsek vetnej formy na rozdiel od paralelného g-systému, ktorý môe v jednom kroku prepísa celú vetnú formu. Teda ak chce sekvenný g-systém odsimulova jeden krok paralelného g-systému, musí urobi $\{$ rádovo dka vetnej formy $\}$ krokov. Ak chceme sekvenne odsimulova celý výpoet paralelného g-systému, dostávame spomínané tvrdenie.

Ke sa na tento problém pozrieme z opanej strany (ke chceme jazyk generovaný sekvenným g-systémom, generova paralelným g-systémom), podobná úvaha nám umouje pretransformova dolný odhad potu krokov pre sekvenné g-systémy na dolný odhad potu krokov pre paralelné g-systémy. Dostávame teda nasledujúce tvrdenie.

Veta 2.5.5. $\overline{TIME}_{\mathcal{S}_\varepsilon}(f(n)) \subseteq \overline{TIME}_{\mathcal{G}_\varepsilon}(\frac{f(n)}{n})$

Toto tvrdenie nám ukazuje, e vzah paralelných a sekvenných g-systémov je lineárny a teda, e paralelizmus nám vo veobecnosti vemi nepomôe (aspo nie nejak dramaticky), ako si vak neskôr ukáeme, existujú tzv. "rýchlo generovatené" jazyky, kde je rozdiel výraznejí.

Príklad 2.5.1. Je známe, e $L = \{wcw \mid w \in \{a,b\}^*\} \in \overline{TIME}_{\mathcal{S}_\varepsilon}(n^2)$, teda poda vety 2.5.5 $L \in TIME_{\mathcal{G}_\varepsilon}(n)$. Paralelizmus teda tomuto jazyku vemi nepomôe, nie je toti exponenciálny

2.6 Normálové tvary g-systémov

Definícia 2.6.1. *Hovoríme, e g-systém je v normálovom tvare, ak spa nasledujúce podmienky:*

1. *existuje kopírovací cyklus pre* $\forall a \in N \cup T$ *v* q_0 *aj v* q_F

2. *iba kopírovacie cykly vstupujú do* q_0 *a vystupujú z* q_F

3. *terminálne symboly sa iba kopírujú*

4. *g-systém je bez* ε

Oznaenie: $\mathcal{N}$ je trieda vetkých g-systémov v normálnom tvare

Veta 2.6.1. $\mathcal{L}_{\mathcal{N}} = \mathcal{L}_{\mathcal{G}_\varepsilon}$

Dôkaz: Inklúzia $\subseteq$ je zrejmá vaka tvrtej podmienke. Dokáeme opanú inklúziu:
Nech $G = (N, T, M, \sigma) \in \mathcal{G}_\varepsilon$. Chceme zostroji $G'' \in \mathcal{N}$ taký, e $L(G'') = L(G)$, teda chceme. aby G'' spal vetky tyri podmienky z definície $\mathcal{N}$. Kee $G \in \mathcal{G}_\varepsilon$ tak podmienka 4 je automaticky splnená. G'' zostrojíme v dvoch krokoch:

1. Najskôr zostrojíme $G' = (N', T, M', \sigma)$ taký, e $L(G') = L(G)$ a G' bude spa tretiu podmienku z definície $\mathcal{N}$:
   - $N' = N \cup \{\xi_a \mid a \in T\}$
   - $\forall a \in T$: vetky výskyty terminálu a v H nahradíme novým neterminálom ξ_a
   - v q_0 nedeterministicky uhádneme, e vetná forma obsahuje u len neterminály typu ξ_a a celú ju zterminálnime. Do M' pridáme nový stav q' a do H' pridáme tvorice: pre $\forall\ a \in T$ (q_0, ξ_a, a, q') a (q', ξ_a, a, q'), priom q' bude akceptaným stavom.

2. Na G' pouijeme kontrukciu podobnú s prvou asou kontrukcie z kapitoly 2.5, ktorá nám zabezpeí splnenie podmienok 1 a 2. Vimnime si, e kee pracujeme s bez ε g-sytémami, tak nemusíme strojnásobova stavy, a teda vychádzajú aj lepie odhady popisnej zloitosti.

□

Dôsledok 2.6.1. Pre kadý jazyk $L \in \mathcal{L}_{\mathcal{G}_\varepsilon}$ platí:

1. $STATE_{\mathcal{N}}(L) \leq STATE_{\mathcal{G}_\varepsilon}(L) + 3$

2. $ARC_{\mathcal{N}}(L) \leq 12.ARC_{\mathcal{G}_\varepsilon}(L) + 14.\#\Sigma_L$

3. $SPACE_{\mathcal{N}}(n) = SPACE_{\mathcal{G}_\varepsilon}(n) = \mathcal{L}_{CS}$

4. $TIME_{\mathcal{N}}(f(n)) = TIME_{\mathcal{G}_\varepsilon}(f(n))$

Oznaenie: $STATE_{\mathcal{N}}(n) = \{L \mid \exists G \in \mathcal{N}, STATE(G) \leq n, L = L(G)\}$

Veta 2.6.2. $STATE_{\mathcal{N}}(n)$ *je* $\mathcal{AFL}$ *pre*[6] $\forall n > 1$

Oznaenie: $\mathcal{N}_{i,j}$ - trieda vetkých sekvenných g-systémov v normálnom tvare, ktoré majú najviac i stavov a najviac j neterminálov, priom sú dovolené ε prechody v 1-a-prekladai.

[6]pre $n = 1$ by boli problémy s h^{-1}

Veta 2.6.3. $\mathcal{L}_{\mathcal{N}_{6,2}} = \mathcal{L}_{\mathcal{N}_{5,3}} = \mathcal{L}_{\mathcal{N}_{4,4}} = \mathcal{L}_{RE}$

Pre $\mathcal{N}_{6,2}$ tvrdenie platí, ak g-systém zaína svoju prácu z poiatoného slova, otvoreným problémom je, i platí aj pri pouití poiatoného neterminálu.

Dôsledkom tejto vety sú niektoré normálové tvary pre frázové gramatiky:

Dôsledok 2.6.2. K ubovonému jazyku $L \in \mathcal{L}_{RE}$ existuje taká frázová gramatika, e vetky jej pravidlá sú tvaru:

1. $\sigma \to u$ alebo $AB \to \varepsilon$ a $CD \to \varepsilon$ kde $N = \{\sigma, A, B, C, D\}$ a $u \in (N \cup T)^*$
2. $\sigma \to u$ alebo $ABBBA \to \varepsilon$ kde $N = \{\sigma, A, B\}$ a $u \in (N \cup T)^*$
3. $\sigma \to u$ alebo $ABC \to \varepsilon$ kde $N = \{\sigma, A, B, C\}$ a $u \in (N \cup T)^*$

Otvorenými problémami zostávajú:

- $\mathcal{N}_{5,2} \stackrel{?}{=} \mathcal{L}_{RE}$
- $\mathcal{N}_{4,2} \stackrel{?}{=} \mathcal{L}_{RE}$
- $\mathcal{N}_{4,3} \stackrel{?}{=} \mathcal{L}_{RE}$

2.7 Charakterizácia triedy $TIME_{\mathcal{G}}(f(n))$ pomocou sekvenného priestoru

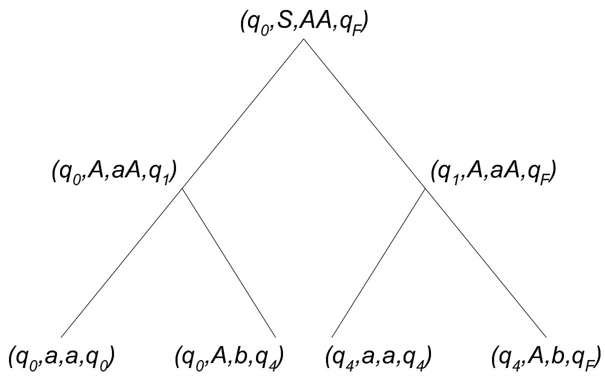
Definícia 2.7.1. *Nech D je odvodenie slova $w \in L(G)$ g-systémom $G = (N, T, M, \sigma)$, kde $M = (K, \Sigma, \Sigma, H, q_0, q_F)$. Strom odvodenia (ozn. T_D) nazývame strom, ktorého vrcholy sú oznaené ako tvorice z H, priom oznaenie korea T_D je tvaru (q_0, σ, u, q_F), kde $u \in (N \cup T)^*$. Vrchol s oznaením $(q, a, b_1 b_2 \dots b_k, p)$ má k synov $(p_1, b_1, u_1, p_2)(p_2, b_2, u_2, p_3) \dots (p_k, b_k, u_k, p_{k+1})$, kde $\forall i u_i \in (N \cup T)^*$. Postupnos tvoríc na kadej úrovni stromu je výpoet 1-a-prekladaa. Zreazením tretích komponent tvoríc na k-tej úrovni dostaneme k-tu vetnú formu výpotu G. Teda zreazením tretích komponent tvoríc na poslednej úrovni[7] dostaneme slovo w.*

Príklad 2.7.1. Zoberme si g-systém a jeho 1-a-preklada pre jazyk $L = \{ww \mid w \in \{a, b\}^*\}$ z príkladu 2.1.1. Zoberme si jedno konkrétne odvodenie $S \Rightarrow AA \Rightarrow aAaA \Rightarrow abab$. Jednotlivým krokom tohto odvodenia zodpovedá v 1-a-prekladai výpoet:

- $S \Rightarrow AA \rightsquigarrow (q_0, S, AA, q_F)$
- $AA \Rightarrow aAaA \rightsquigarrow (q_0, A, aA, q_1)(q_1, A, aA, q_F)$
- $aAaA \Rightarrow abab \rightsquigarrow (q_0, a, a, q_0)(q_0, A, b, q_4)(q_4, a, a, q_4)(q_4, A, b, q_F)$

Strom tohto odvodenia je na obrázku 2.8.

[7] hbka stromu odvodenia $\geq TIME(G, w)$



Obr. 2.8: Strom zodpovedajúci odvodeniu $S \Rightarrow AA \Rightarrow aAaA \Rightarrow abab$

Lema 2.7.1. *$TIME_{\mathcal{G}}(f(n)) \subseteq 1NSPACE(f(n))$ pre $\forall f(n)$.*

Dôkaz: Nech $G = (N, T, M, \sigma)$ je g-systém pracujúci v ase $O(f(n))$. Chceme skontruova Turingov stroj A s jednosmernou vstupnou páskou, ktorý bude simulova G v priestore $O(f(n))$. Uvaujme slovo $w \in L(G)$ a jeho prísluný strom odvodenia T. Chceme ukáza, e A akceptuje w práve vtedy, ke $w \in L(G)$. A bude na svojej pracovnej páske zapisova cesty z T vedúce od korea k listom (obr.2.9). A bude háda tieto cesty a overova, i jednotlivé tvorice na rovnakej úrovni T tvoria výpoet 1-a-prekladaa M a i zreazenie výstupov tvoríc v listoch T tvoria w.

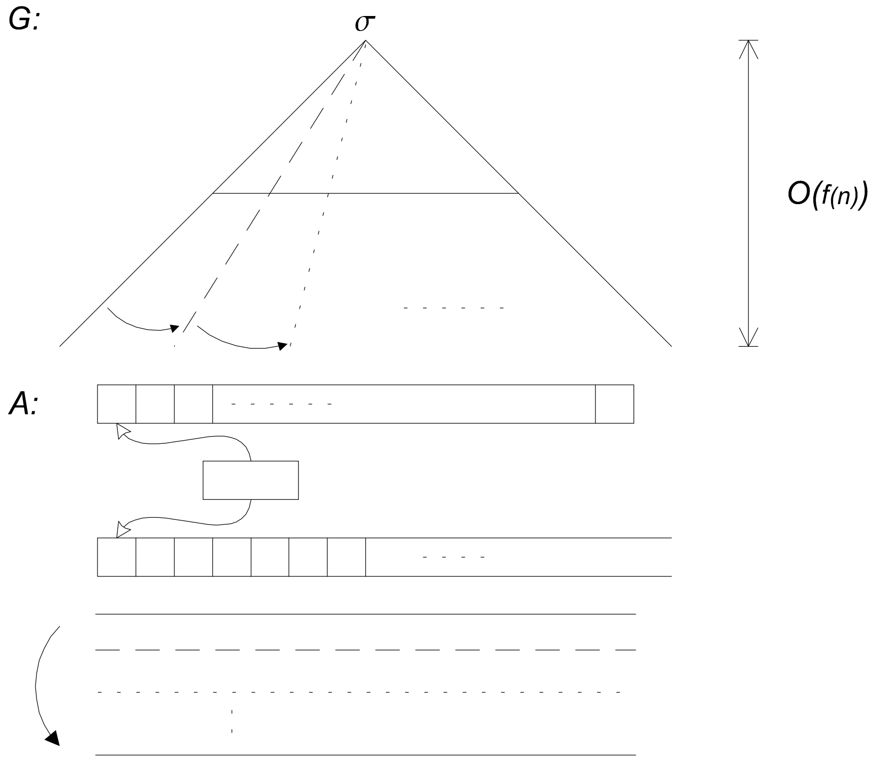
A najprv uhádne a zapíe na pracovnú pásku cestu z korea do najavejieho listu α, priom musí skontrolova i prvá tvorica je tvaru (q_0, σ, u, q_F) a ostatné tvorice musia zaina v stave q_0 a prepisova prvý symbol výstupu predchádzajúcej tvorice. Naviac výstup listu α musí by prefixom slova w. Ak nejaká z týchto podmienok neplatí, tak A neuhádol najavejiu cestu správne a zasekne sa. Ak A uhádol, tak nahradí tvoricu α jeho najavejím bratom β (to samozrejme tie uhádne (obr.2.10a) a zaznaí si, e β je druhým synom ich spoloného otca γ. A overí, i β je dobre uhádnutý t.j. poiatoný stav β je rovnaký ako koncový stav α, β prepisuje druhý symbol výstupu γ a výstup β je rovnaký ako alia as w (bez prefixu, ktorý bol vo výstupe α). Takto A pokrauje a kým neuhádne a neoverí posledného syna γ. Potom A nahradí γ jeho najavejím bratom δ (podobne ako bol α nahradený β). Teda A robí prehadávanie do hbky, priom si treba uvedomi, e pri návrate na vyiu úrove v strome, si A musí pamäta koncové stavy jednotlivých vrcholov, aby mohol pri hádaní alích vrcholov overi správnu následnos.

Týmto spôsobom A pokrauje a kým vo výstupoch listov nenájde celé slovo w. Potom u A vie, e vetky ostatné neoverené listy musia ma na výstupe ε (obr.2.10b). Teda zvyné cesty bude A hada s tým, e v listoch musí by na výstupe ε. Ke A dosiahol najpravejí list (to je, ako inak, opä uhádnuté), tak A overí i vetky koncové stavy vo vetkých tvoriciach na celej pracovnej páske sú q_F. Ak je to tak, potom A akceptuje w.

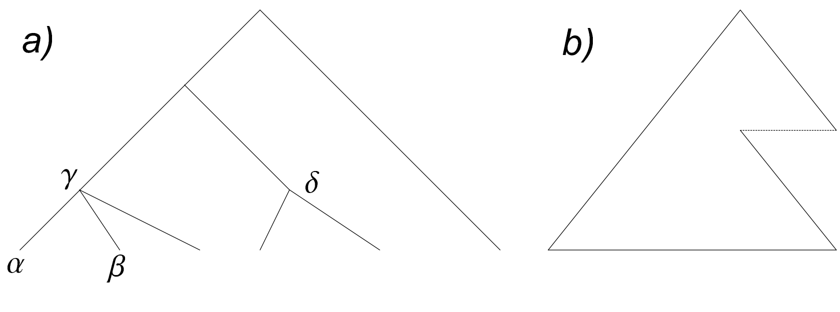
Z kontrukcie A je zrejmé, e A akceptuje w práve vtedy, ke $w \in L(G)$. Kee $w \in L(G)$, tak hbka stromu odvodenia nie je väia ako $c.f(|w|)$ kde c je kontanta nezávislá od w, teda hbka stromu odvodenia je $O(f(n))$, a teda A na akceptovanie slova w nepotrebuje viac políok na páske ako $O(f(n))$ z oho konene plynie, e $L(G) \in 1NSPACE(f(n))$. $\square$

Lema 2.7.2. *$1NSPACE(f(n)) \subseteq TIME_{\mathcal{G}}(f(n))$ pre $f(n) = \Omega(\log n)$.*

Dôkaz: Nech A je nedeterministický Turingov stroj s jednosmernou vstupnou páskou, ktorý akceptuje v priestore $O(f(n))$. Bez ujmy na veobecnosti predpokladajme, e A má len jednu jednosmerne nekonenú pracovnú pásku.



Obr. 2.9: Simulácia g-systému G Turingovým strojom A



Obr. 2.10: Nahrádzanie hrán v 1-a-prekladai M

Oíslujme políka pracovnej pásky 0,1,2. . . Chceme skontruova g-systém G, ktorý simuluje A v ase $O(f(n))$. Jedna vetná forma G obsahuje informáciu o políku pracovnej pásky A poas celého výpotu. Nasledujúca vetá forma obsahuje informáciu o nasledujúcom políku at. Kee A pracuje na najviac $O(f(n))$ políkach, tak G potrebuje na vygenerovanie slova dky n najviac $O(f(n))$ vetných foriem (t.j. G pracuje v ase $O(f(n))$). Samozrejme G musí zarui konzistentnos medzi jednotlivými políkami pásky, stavmi A a symbolmi na vstupnej páske poda δ-funkcie A.

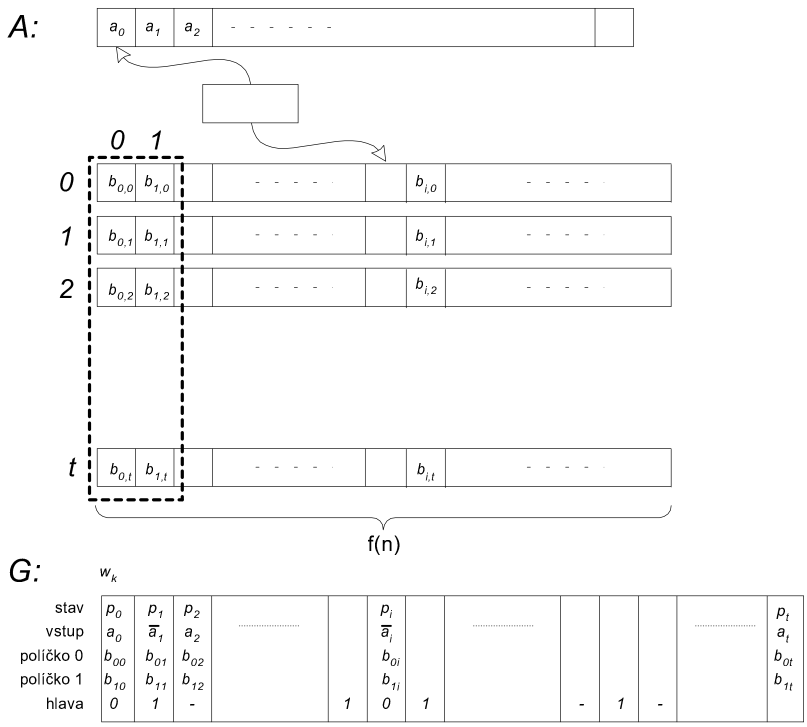
Uvaujme jeden výpoet A na vstupnom slove $w \in L(A)$. Nech s je priestor a t je as potrebný na výpoet w. Odvodenie slova w g-systémom G je tvaru:

$$\sigma = w_0 \stackrel{k}{\Rightarrow} w_k \Rightarrow w'_k \Rightarrow w_{k+1} \Rightarrow w'_{k+1} \Rightarrow \ldots \Rightarrow w_{k+s} \Rightarrow w'_{k+s} \Rightarrow w$$

kde vetná forma w_{k+j} drí obsah j-teho a $j+1$. políka A v celom výpote $A(w)$ a vetná forma w'_{k+j} je pouitá na overenie i uhádnuté obsahy sú legálne vzhadom na δ-funkciu A.
i-ty symbol vetnej formy w_{k+j} je päposchodový symbol obsahujúci:

- stav p_i, v ktorom je A v ase i (p_{t+1} je akceptujúci)
- symbol a_i, ktorý v ase i íta vstupná hlava A, priom tento symbol si oznaíme, ak v ase i A posúva hlavu na vstupe
- symbol $b_{j,i}$, ktorý je na j-tom políku pracovnej pásky A v ase i
- symbol $b_{j+1,i}$, ktorý je na $j+1$. políku pracovnej pásky A v ase i
- peciálny symbol 0, ak hlava bola na j-tom políku, peciálny symbol 1, ak hlava bola na $j+1$. políku a v ostatných prípadoch peciálny symbol -

V prvých k-krokoch G odvodí (uhádne[8]) vetnú formu w_k dky $t+1$ t.j. postupnos stavov, ktorými A prechádza poas výpotu na w, vstupné slovo a asy, v ktorých A pohne hlavou na vstupe, obsahy nultého a prvého políka pracovnej pásky A poas celého výpotu a asy výskytu hlavy na nultom políku pracovnej pásky. Schématicky vyzerá vetná forma ako na obrázku 2.11.



Obr. 2.11: Simulácia TS A g-systémom G

V kroku $k+1$ G overí i, to o uhádol v predchádzajúcom kroku je v súlade s δ-funkciou A a vygeneruje[9] w'_k. Ak to bolo uhádnuté dobre, tak G vygeneruje w_{k+1} t.j. prvé a druhé poschodie

[8]to sa dá na $O(f(|w|))$ krokov kee A pracuje v priestore $O(f(|w|))$ a teda v ase $O(|w|.c^{f(|w|)})$ pre nejakú kontantu c t.j. poet vetkých moných konfigurácií A pri výpote w. g-systém vie tokoto symbolov vygenerova v logaritmickom ase, o je v tomto prípade $O(f|w|)$.

[9]tento krok je naozaj iba overovací, to znamená, e ak G uhádol dobre, tak $w'_k = w_k$ inak sa G zasekne

bude rovnaké ako v w'_k, tvrté poschodie w'_k bude vo w_{k+1} tretím poschodím, a kde bol v piatom poschodí w'_k peciálny symbol 1, tam bude v piatom poschodí w_{k+1} peciálny symbol 0, ostatné G uhádne[10] (obr.2.12). V alom kroku G overí i to o uhádol teraz je legálne vzhadom na δ-funkciu A a vygeneruje $w'_{k+1} \cdots$

Overovanie vo veobecnosti vyzerá tak, e G akoby sa naraz pozeral na dva susedné päposchodové symboly, take vidí isté malé okolie (2 symboly) pracovnej pásky v istom malom asovom úseku (2 takty TS) a k zmenám na pracovnej páske hadá príslunú as δ-funkcie, ktorá je schopná takéto zmeny spôsobi. Ak takúto as δ-funkcie A nájde, tak tieto dva päposchodové symboly môu stá veda seba a G sa posunie o jeden päposchodový symbol. Takto môe G overi celú vetnú formu.

G pokrauje v striedaní hádacích a overovacích krokov, a kým neodvodí vetnú formu bez peciálnych symbolov výskytu hlavy na pracovnej páske A t.j. v piatom poschodí vetnej formy sa nevyskytuje 0 ani 1. To znamená, e G odsimuloval celý výpoet A.

G: w_{k+1}

stav	p_0	p_1	p_2			p_i							p_t
vstup	a_0	$\overline{a}_1$	a_2	....		$\overline{a}_i$		....				....	a_t
políčko 1	b_{10}	b_{11}	b_{12}			b_{1i}							b_{1t}
políčko 2	b_{20}	b_{21}	b_{22}			b_{2i}							b_{2t}
hlava	-	0	1		0	1	-		-	-	0		

Obr. 2.12: Posun symbolov vo vetnej forme

V poslednom kroku G prepíe vetnú formu tak, e kadý päposchodoví symbol sa prepíe na symbol z druhého poschodia (t.j. symbol zo vstupnej pásky A) ak bol tento symbol oznaený, inak sa celý päposchodový symbol prepíe na ε.

Zrejme $w \in L(G) \iff$ ak w je akceptované A. Naviac w je vygenerované v ase
$O(f(n)) + O(2f(n)) = O(f(n))$. Samozrejme dva kroky odvodenia G $w_l \Rightarrow w'_l \Rightarrow w_{l+1}$ sa dajú nahradi jedným, potom w je vygenerované v ase $O(f(n)) + O(f(n)) = O(f(n))$. Tento spôsob odvodenia sme zvolili iba pre lepiu itatenos dôkazu. $\square$

Predchádzajúce dve lemy nám umoujú vyslovi nasledujúce tvrdenie:

Veta 2.7.1. *$TIME_{\mathcal{G}}(f(n)) = 1NSPACE(f(n))$ pre $f(n) \geq \log n$.*

Ak si uvedomíme, e pre $f(n) \geq n$ je pracovná páska Turingovho stroja dos dlhá, aby sme si na nej zapamätali vstup, potom $1NSPACE(f(n)) = NSPACE(f(n))$.

alím jednoduchým pozorovaním zistíme, e ak $f(n) \geq n$, tak priestor na TS a na g-systémoch je ekvivalentný, teda $NSPACE(f(n)) = SPACE_{\mathcal{G}}(f(n))$.

Môeme teda vyslovi nasledovné dôsledky predchádzajúcej vety:

Dôsledok 2.7.1. $TIME_{\mathcal{G}}(f(n)) = NSPACE(f(n))$ pre $f(n) \geq n$.

Dôsledok 2.7.2. $TIME_{\mathcal{G}}(f(n)) = SPACE_{\mathcal{G}}(f(n))$ pre $f(n) \geq n$.

[10] t.j. G uhádne symboly na druhom políku pracovnej pásky A poas celého výpotu a asy výskytu hlavy na prvom políku

2.8 Niektoré vlastnosti "rýchlo generovatených" jazykov

"Rýchlo generovatené" jazyky nazývame také jazyky z triedy $TIME_{\mathcal{G}}(f(n))$, pre ktoré $f(n) < n$.

Veta 2.8.1. *$TIME_{\mathcal{G}}(\log^p n)$ je $\mathcal{AFL}$ pre*[11] *$p \geq 1$.*

Veta 2.8.2. *$TIME_{\mathcal{G}}(\log^p n) \subsetneq TIME_{\mathcal{G}}(\log^q n)$ pre $q > p > 1$.*

Dôsledok 2.8.1. Hierarchia $TIME_{\mathcal{G}}(\log^p n)$ pre $p > 1$ je nekonená.

Veta 2.8.3. *Pre kadý jazyk $L \in \mathcal{L}_{RE}$ existuje $L' \in TIME_{\mathcal{G}}(\log n)$ a existuje homomorfizmus h taký, e $L = h(L')$.*

Dôkaz: Táto veta nám hovorí, e ku kadému jazyku $L \in \mathcal{L}_{RE}$ vieme nájs g-systém G pracujúci v logaritmickom ase a homomorfizmus h taký, e $h(L(G)) = L$. Veta 2.2.1 nám hovorí, e vieme k L nájs prísluný g-systém G, ale nezaruuje, e bude pracova v logaritmickom ase. G upravíme na G' nasledovne:

- Do 1-a-prekladaa G zavedieme nový terminál γ
- Kadú tvoricu tvaru (q_0, σ, u, p) nahradíme tvoricou $(q_0, \sigma, \gamma u, p)$
- Pridáme tvoricu $(q_0, \gamma, \gamma\gamma, q_0)$

ahko vidno, e takto upravený g-systém G' generuje jazyk

$$L' = \{\gamma^{2^m} w \mid w \in L(G) \text{ a } m \text{ je poet krokov odvodenia } w \text{ v } G\}$$

Zamyslime sa teraz nad asovou zloitosou G'. Zoberme si nejaké slovo $u = w\gamma^{2^k} \in L'$ pre nejaké k. Zrejme $|u| \geq 2^k$, ale G' toto slovo vygeneruje v k krokoch, teda v logaritmickom ase, a teda $L' \in TIME_{\mathcal{G}}(\log n)$. Homomorfizmus h zvolíme takto:

- $\forall a \in T : h(a) = a$
- $h(\gamma) = \varepsilon$

To znamená, e h vymae vetky γ zo slov $w \in L'$, ktoré sme zaviedli g-systémom G'. Teda $h(L') = L$. Ak sa nad touto kontrukciou ete raz zamyslíme, zistíme, e my sme g-systém nejakým spôsobom "nezrýchovali", ale to, e sme jazyk dostali do logaritmickej asovej zloitosti sme dosiahli tým, e dku slov z tohto jazyka sme natoko zväili, e pri zachovaní potu krokov odvodenia bude tento jazyk vygenerovaný v logaritmickom ase vzhadom na túto zväenú dku slova. $\square$

Dôsledok 2.8.2. Trieda $TIME_G(f(n))$ nie je uzavretá na ubovoný homomorfizmus.

2.9 Záverom o g-systémoch

Je vhodné si uvedomi niekoko významných faktov, ktoré nám model generatívnych systémov priniesol.

Pre známe paralelné gramatiky, ktoré dokáeme simulova na g-systémoch, dostaneme priestorové ohranienie najviac $1NSPACE(f(n))$. Podobne, ak navrhneme nový paralelný model, ktorý vieme "tesne" simulova na g-systémoch, dostaneme priestorové ohranienie $1NSPACE(f(n))$. Naviac pre kadý jazyk $L \in 1NSPACE(f(n))$ existuje nejaký typ paralelnej gramatiky, ktorá L dokáe generova v ase $f(n)$.

[11] rýchlejie ako v logaritmickom ase g-systém nedokáe pracova

Kapitola 3

Kooperujúce distribuované systémy gramatík ($CDGS$)

V tejto kapitole ukáeme aliu monos paralelizmu, kde viac gramatík pracuje na jednej spolonej vetnej forme: gramatika dostane vetnú formu a pracuje na nej tak dlho, ako je jej urené...

Definícia 3.0.1. *CDGS je $(n+2)$-tica $\Gamma = (T, G_1, G_2, ..., G_n, S)$, kde $\forall i G_i = (N_i, T_i, P_i)$ je "bezkontextová gramatika bez poiatoného symbolu". $T \subseteq \bigcup_i T_i$, $S \in \bigcup_i N_i$ je poiatoný symbol*

Definícia 3.0.2. *Nech $\Gamma = (T, G_1, G_2, ..., G_n, S)$. Krok odvodenia je relácia $\overset{\le k}{\underset{\Gamma}{\Longrightarrow}}$, kde $\overset{\le k}{\underset{\Gamma}{\Longrightarrow}} = \overset{\le k}{\underset{G_i}{\Longrightarrow}}$ $i \in \{1, 2, ..., n\}$ definovaná nasledovne: $\overset{\le k}{\underset{G_i}{\Longrightarrow}} = (\bigcup_{j=1}^{k} \overset{j}{\underset{G_i}{\Longrightarrow}})$, podobne definujeme aj: $\overset{\ge k}{\underset{\Gamma}{\Longrightarrow}}$, $\overset{=k}{\underset{\Gamma}{\Longrightarrow}}$. Definujeme $x \overset{\tilde{t}}{\underset{\Gamma}{\Longrightarrow}} y$[1]: $x \overset{t}{\underset{\Gamma}{\Longrightarrow}} y = x \overset{t}{\underset{G_i}{\Longrightarrow}} y$ $i \in \{1, 2, ...n\}$ platí práve vtedy, ak: $x \overset{*}{\underset{G_i}{\Longrightarrow}} y$ a $\nexists z \neq y : y \underset{G_i}{\Rightarrow} z$*

Definícia 3.0.3. *Nech $f \in \{t, *, =1, =2, ..., \le 1, \le 2, ..., \ge 1, \ge 2, ...\}$ a nech Γ je $CDGS$. Potom jazyk definovaný systémom pri spôsobe prepisovania f je*

$$L_f(\Gamma) = \{w \in T^* \mid \exists r, i_1, i_2, ..., i_r \; S \overset{f}{\underset{G_{i_1}}{\Longrightarrow}} w_1 \overset{f}{\underset{G_{i_2}}{\Longrightarrow}} w_2 \overset{f}{\underset{G_{i_3}}{\Longrightarrow}} ... \overset{f}{\underset{G_{i_r}}{\Longrightarrow}} w_r \equiv w\}$$

Príklad 3.0.1. $\Gamma = (\{a, b, c\}, G_1, G_2, S)$
$G_1 = (\{A, B\}, \{A', B', a, b, c\}, \{A \to aA'b, B \to cB', A \to ab, B \to c\})$ a
$G_2 = (\{S, S', A', B'\}, \{A, B\}, \{S \to S', S' \to AB, A' \to A, B' \to B\})$, potom

$$L_{=1}(\Gamma) = L_*(\Gamma) = L_{\le k}(\Gamma) = L_{\ge 1}(\Gamma) = L_t(\Gamma) = \{a^n b^n c^m \mid n, m \ge 1\}, k \ge 1$$

$$L_{=2}(\Gamma) = L_{\ge 2}(\Gamma) = \{a^n b^n c^n \mid n \ge 1\}$$

$$L_{=k}(\Gamma) = L_{\ge k}(\Gamma) = \emptyset \text{ pre } k \ge 3$$

Príklad 3.0.2. $\Gamma = (\{a\}, G_1, G_2, G_3, S)$
$G_1 = (\{S\}, \{A\}, \{S \to AA\})$

[1]alej budeme písa len $x \overset{t}{\underset{\Gamma}{\Longrightarrow}} y$

$G_2 = (\{A\}, \{S\}, \{A \to S\})$ a
$G_3 = (\{A\}, \{a\}, \{A \to a\})$. Potom $L_t(\Gamma) = \{a^{2^n} \mid n \geq 1\}$

Príklad 3.0.3. $\Gamma = (\{a,b,c\}, G_1, G_2, G_3, S)$
$G_1 = (\{S, A, A'\}, \{a,b,c\}, \{S \to S, S \to AcA, A' \to A\})$
$G_2 = (\{S, A, A'\}, \{a,b,c\}, \{A \to aA', a \to a\})$ a
$G_3 = (\{S, A, A'\}, \{a,b,c\}, \{A \to bA', A \to b\})$. Potom $L_{=2}(\Gamma) = L_{\geq 2}(\Gamma) = \{wcw \mid w \in \{a,b\}^+\}$

Skúmali sa viaceré monosti voby T:

Definícia 3.0.4. *Akceptaný týl definujeme nasledovne:*

arb $T \subseteq \bigcup_i T_i$ *(arbitrary)*

ex $T = \bigcup_i T_i$ *(exactly)*

all $T = \bigcap_i T_i$

one $T = T_i$ *pre nejaké* i

Oznaenie: $f \in \{*, t, =1, =2, ..., \leq 1, \leq 2, ..., \geq 1, \geq 2, ...\} = D$ $D' = \{*, =1, \geq 1, \leq 1, \leq 2, ...\}$[2] $A \in \{arb, ex, all, one\}$. (CD_nCF, f, A) oznauje triedu s bezkontextovými komponentami (najviac n) s akceptaným týlom A. (CD_*CF, f, A) oznauje triedu s ubovoným potom bezkontextových komponent s akceptaným týlom A.

Veta 3.0.1. $\mathcal{L}(CD_*CF, f, arb)$=$\mathcal{L}(CD_*CF, f, ex)$=$\mathcal{L}(CD_*CF, f, all)$= $\mathcal{L}(CD_*CF, f, one)$

Dôkaz:

$\mathcal{L}(CD_*CF, f, arb) = \mathcal{L}(CD_*CF, f, all)$

- $\subseteq$: Nech $\Gamma \in \mathcal{L}(CD_*CF, f, arb)$. Zostrojíme ekvivalentnú $\Gamma' \in \mathcal{L}(CD_*CF, f, all)$: $\Gamma = (T, G_1, ..., G_n, S)$ $\Gamma' = (T, G'_1, ..., G'_n, G_{n+1}, S')$
  - (a) $f = t$ $G'_i = (N'_i, T'_i, P'_i)$, kde $N'_i = \{A' \mid A \in N_i\}$, $T'_i = \{a' \mid a \in T_i\} \cup T$ -potrebujeme dosiahnu, aby T bolo prienikom T'_i-iek - môu tam by nejaké navye. $P'_i = \{A' \to w' \mid A \to w \in P_i\}$, kde $w' = a'_1, ...a'_n$ ak $w = a_1, ...a_n$. $G_{n+1} = (N_{n+1}, T_{n+1}, P_{n+1})$, kde $N_{n+1} = \{a' \mid a \in (\bigcup_i N_i \cup \bigcup_i T_i)\} \cup \{F\}$ $T_{n+1} = T$[3] $P_{n+1} = \{a' \to a \mid a \in T\} \cup \{a' \to F \mid a \notin T\} \cup \{F \to FF\}$
  - (b) $f \neq t$ Rovnaká kontrukcia ako v prípade (a), ale $P_{n+1} = \{a' \to a' \mid a \in T\} \cup \{a' \to a \mid a \in T\}$
- $\supseteq$: Táto inklúzia triviálne platí, lebo ak $T = \bigcap_i T_i$, tak potom aj $T \subseteq \bigcup_i T_i$

$\mathcal{L}(CD_*CF, f, arb) = \mathcal{L}(CD_*CF, f, ex)$

- $\subseteq$: Nech $\Gamma \in \mathcal{L}(CD_*CF, f, arb)$. Zostrojíme ekvivalentnú $\Gamma' \in \mathcal{L}(CD_*CF, f, ex)$: $\Gamma = (T, G_1, ..., G_n, S)$ $\Gamma' = (T, G'_1, ..., G'_n, G_{n+1}, S')$

[2] v D' nie sú tie, ktoré nás nútia robi viac ako 1 krok
[3] Týmto sme zabezpeili, e $(\bigcap_i T'_i \cap T_{n+1}) = T$

(a) $f = t$ $G'_i = (N'_i, T'_i, P'_i)$, kde $N'_i = \{A' \mid A \in N_i\} \cup \{a' \mid a \in T_i\}$, $T'_i = T$ -potrebujeme dosiahnu, aby T bolo rovné zjednoteniu T'_i-iek. $P'_i = \{A' \to w' \mid A \to w \in P_i\}$, kde $w' = a'_1, ...a'_n$ ak $w = a_1, ...a_n$.
$G_{n+1} = (N_{n+1}, T_{n+1}, P_{n+1})$, kde $N_{n+1} = \{a' \mid a \in (\bigcup_i N_i \cup \bigcup_i T_i)\} \cup \{F\}$ $T_{n+1} = T$ $P_{n+1} = \{a' \to a \mid a \in T\} \cup \{a' \to F \mid a \notin T\} \cup \{F \to FF\}$

(b) $f \neq t$ Rovnaká kontrukcia ako v prípade (a), ale $P_{n+1} = \{a' \to a' \mid a \in T\} \cup \{a' \to a \mid a \in T\}$

$\supseteq$: Táto inklúzia triviálne platí, lebo ak $T = \bigcup_i T_i$, tak potom aj $T \subseteq \bigcup_i T_i$

$\mathcal{L}(CD_*CF, f, arb) = \mathcal{L}(CD_*CF, f, one)$

$\subseteq$: Nech $\Gamma \in \mathcal{L}(CD_*CF, f, arb)$. Zostrojíme ekvivalentnú $\Gamma' \in \mathcal{L}(CD_*CF, f, one)$: $\Gamma = (T, G_1, ..., G_n, S)$ $\Gamma' = (T, G'_1, ..., G'_n, G_{n+1}, S')$
$G'_i = (N'_i, T'_i, P'_i)$, kde $N'_i = N_i$, $T'_i = T_i$, $P'_i = P_i$. $G_{n+1} = (N_{n+1}, T_{n+1}, P_{n+1})$, kde $N_{n+1} = \emptyset$, $T_{n+1} = T$ - gramatikou G_{n+1} sme dosiahli to, e urite existuje i také, e platí: $T = T_i$ pre nejaké i

$\supseteq$: Táto inklúzia triviálne platí, lebo ak $T = T_i$ pre nejaké i, tak potom aj $T \subseteq \bigcup_i T_i$

$\square$

V alej asti tejto kapitoly platí: A=*all* a nebudeme ho explicitne písa.

Veta 3.0.2. [4]

- $\mathcal{L}(CD_*CF, f) = \mathcal{L}_{CF}$ $\forall f \in D'$
- $\mathcal{L}_{CF} = \mathcal{L}(CD_1CF, f) \subsetneq \mathcal{L}(CD_2CF, f) \subseteq \mathcal{L}(CD_nCF, f) \subseteq \mathcal{L}(CD_*CF, f) \subseteq \mathcal{L}_{CFMatrix}$[5] $\forall f \in D - D'$, $n \geq 3$
- $\mathcal{L}(CD_nCF, = k) \subseteq \mathcal{L}(CD_nCF, = s.k)$ $\forall k, n, s \geq 1$ [6]
- $\mathcal{L}(CD_nCF, \geq k) \subseteq \mathcal{L}(CD_nCF, \geq k+1)$ $\forall n, k \geq 1$
- $\mathcal{L}(CD_*CF, \geq) \subseteq \mathcal{L}(CD_*CF, =)$, *kde* $\mathcal{L}(CD_*CF, \geq) = \mathcal{L}(CD_*CF, \geq 1) \cup \mathcal{L}(CD_*CF, \geq 2) \cup ...$ *a* $\mathcal{L}(CD_*CF, =) = \mathcal{L}(CD_*CF, = 1) \cup \mathcal{L}(CD_*CF, = 2) \cup ...$
- $\mathcal{L}_{CF} = \mathcal{L}(CD_1CF, t) = \mathcal{L}(CD_2CF, t) \subsetneq \mathcal{L}(CD_3CF, t) = \mathcal{L}(CD_*CF, t) = \mathcal{L}(ETOL)$[7]

Dôkaz: Za vetky len jeden príklad: $\mathcal{L}(CD_*CF, t) \subseteq \mathcal{L}(CD_3CF, t)$:
Nech $\Gamma \in \mathcal{L}(CD_*CF, t)$. Zostrojíme $\Gamma' \in \mathcal{L}(CD_3CF, t)$. V Γ' to bude vyzera nasledovne: V prvej gramatike budú schované vetky gramatiky z Γ. alie dve gramatiky budú slúi na prepínanie v tej jednej[8].
$\Gamma = (T, G_1, G_2, ..., G_n, S)$[9], kde $G_i = (N_i, T_i, P_i)$
$\Gamma' = (T, G'_1, G'_2, G'_3, [S, 1])$

[4] Toto je: "Kilometrová veta s plno tvrdeniami na zamyslenie sa"

[5] Maticové bezkontextové gramatiky - istým spôsobom sa reguluje, akým spôsobom sa pouívajú pravidlá. P: mnoina -tíc; vyberieme jednu z nich a u musíme poui vetky pravidlá, ktoré sú v nej

[6] toto tvrdenie sa nepodarilo doposia dokáza veobecnejie, len pre násobky

[7] tabukové rozírené $0L$ - systémy

[8] Musia by dve, lebo keby sme mali iba jednu a kee sa nachádzame v mode t, táto jedna gramatika by sa nám zacyklila

[9] Tu je nutné predpoklada, e n je párne. Ak by tomu tak nebolo, pridáme gramatiku, v ktorej bude $P = \emptyset$

$G'_1 = (N'_1, T'_1, P'_1)$, kde $N'_1 = \{[A,i] \mid A \in N_i\}$, $T'_1 = \bigcup\limits_{i=1}^{n} T_i$, $P'_1 = \{[A,i] \to [w',i] \mid A \to w \in P_i,\ 1 \le i \le n\}$, kde w' je vlastne w, ibae vetky staré neterminály sú nahradené novými.
$G'_2 = (N'_2, T'_2, P'_2)$, kde $N'_2 = \{[A,i] \mid A \in \bigcup\limits_{j=1}^{n} N_j,\ i = 1,...,n\}$, $T'_2 = \emptyset$ a $P'_2 = \{[A,i] \to [A,i+1] \mid i \equiv 1 (mod\ 2)\}$
$G'_3 = (N'_3, T'_3, P'_3)$, kde $N'_3 = N'_2$, $T'_3 = \emptyset$ a $P'_3 = \{[A,i] \to [A,i+1] \mid i \equiv 0(mod\ 2)\}$
$\cup \{[A,n] \to [A,1] \mid [A,n] \in N'_3\}$ $\Box$

Príklad 3.0.4.

G_1: $S \to aAB \mid ...$	$[S,1] \to a[A,1][B,1] \mid ...$
G_2:	
G_3: $A \to bAS \mid ...$	$[A,3] \to b[A,3][S,3] \mid ...$
$S \underset{G_1}{\Rightarrow} aAB \underset{G_3}{\Rightarrow} abASB \Rightarrow ...$	$[S,1] \underset{G'_1}{\Rightarrow} a[A,1][B,1] \underset{G'_2}{\Rightarrow} a[A,2][B,1] \underset{G'_2}{\Rightarrow} a[A,2][B,2] \underset{G'_3}{\Rightarrow}$ $\underset{G'_3}{\Rightarrow} a[A,3][B,2] \underset{G'_3}{\Rightarrow} a[A,3][B,3] \underset{G'_1}{\Rightarrow} ab[A,3][S,3][B,3] \Rightarrow ...$

3.1 Niektoré otázky popisnej zloitosti

Definícia 3.1.1. *Definujeme miery:*

$Var(\Gamma) = \#(\bigcup\limits_i N_i)$ - *poet neterminálov*

$Prod(\Gamma) = \sum\limits_i \# P_i$ - *suma potu pravidiel*

$Symb(\Gamma) = \sum\limits_i (\sum\limits_{A \to w \in P_i} (\mid w \mid + 2))$

Definícia 3.1.2. *Pre miery $M \in \{Var, Prod, Symb\}$ a triedu gramatík X a jazyk L definujeme:* $M_X(L) = min\{M(\Gamma) \mid \Gamma \in X,\ \ L = L(\Gamma)\}$

Definícia 3.1.3. *Pre mieru M a triedy gramatík X a Y a triedu jazykov $\mathcal{L} = \mathcal{L}(X) \cap \mathcal{L}(Y)$ takú, e $M_Y(L) \le M_X(L)\ \ \forall L \in \mathcal{L}$ oznaíme:*

$Y \overset{M}{=} X \ \Leftrightarrow\ M_Y(L) = M_X(L)\ \ \forall L \in \mathcal{L}$

$Y \overset{M}{<}_1 X \ \Leftrightarrow\ \exists L \in \mathcal{L}\ \ M_Y(L) < M_X(L)$

$Y \overset{M}{<}_2 X \ \Leftrightarrow\ \forall n\ \exists L_n \in \mathcal{L}\ \ M_X(L_n) - M_Y(L_n) > n$

$Y \overset{M}{<}_3 X \ \Leftrightarrow\ \exists L_n \in \mathcal{L},\ \ n \ge 1$ *také, e* $\lim\limits_{n \to \infty} \frac{M_Y(L_n)}{M_X(L_n)} = 0$

$Y \overset{M}{<}_4 X \ \Leftrightarrow\ \exists p$ *a* $\exists L_n \in \mathcal{L},\ \ n \ge 1$ *také, e* $M_X(L_n) > n$ *a* $M_Y(L_n) \le p$

Veta 3.1.1. *Porovnanie (CD_*CF, f, A) a CFG :*

	$*$	t	$\le k$	$= k$	$\ge k$
VAR	$=$	$<_4$	$=$	$<_4$	$<_4$
$PROD$	$=$	$<_3$	$=$	$<_4$	$<_4$
$SYMB$	$=$	$<_3$	$=$	$<_3$	$<_3$

Dôkaz: Príklad: $(CD * CF, t) \overset{Var}{<}_4 CFG$ (VAR)
Uvaujme $L_n = \bigcup_{i=1}^{n} b(a^i b)^+$
Potom $Var_{CFG}(L_n) = n + 1$
$P_n = \{S_0 \to bS_i \mid 1 \le i \le n\} \cup \bigcup_{i=1}^{n} \{S_i \to a^i bS_i,\ \ S_i \to a^i b\}$
S menej neterminálmi to neide, lebo pomieaním pravidiel by sme dostali zlé slová.
$Var_{CD_* CF,t}(L_n) \le 3$
$\Gamma = (\{a, b\}, G_1, G_2, ..., G_{n+1}, S)$, kde
$G_i = (\{A\}, \{a, b\}, \{A \to a^i b\});\ \ 1 \le i \le n$
$G_{n+1} = (\{S, S', A\}, \{a, b\}, \{S \to bS', S' \to AS', S' \to A\})$ - táto gramatika pracuje ako prvá. $\square$

Kapitola 4

Paralelné komunikujúce systémy gramatík ($PCGS$)

Dostávame sa k aliemu typu paralelizmu. Paralelný komunikujúci systém gramatík v sebe integruje viacero gramatík nejakého typu, ktoré sú zosynchronizované poda akýchsi globálnych hodín, take pracujú v taktoch. Ozname si tieto gramatiky $G_1, G_2, \ldots, G_n$. Kadá z týchto gramatík pracuje na svojej vetnej forme poda svojich pravidiel. Naviac týmto gramatikám dodáme peciálny neterminál Q (*query* symbol). Ak sa vo vetnej forme nejakej gramatiky vyskytne symbol Q_i, znamená to, e v alom takte sa Q_i zmení na vetnú formu vygenerovanú gramatikou G_i, priom G_i zane generova odznova. Toto presunutie sa vykoná len vtedy, ke vetná forma G_i neobsahuje, iadny symbol *query*. Teraz pristúpime k formálnemu zadefinovaniu tohto modelu.

4.1 Definície a oznaenia

Definícia 4.1.1. *PCGS (Parallel Communicating Grammar Systems) stupa n je $(n+3)$-ica $\Gamma = (N, K, T, G_1, \ldots, G_n)$ kde N je mnoina neterminálov, K je mnoina komunikaných (query) symbolov tandardne oznaovaných $K = \{Q_1, \ldots, Q_n\}$, T je mnoina terminálov, pre kadé i $G_i = (N \cup K, T, P_i, S_i)$ sú bez-ε gramatiky ubovoného typu[1], priom $S_i \in N$ je poiatoný neterminál a v P_i nie sú pravidlá obsahujúce na avej strane query.*

Oznaenie: $V_\Gamma = N \cup K \cup T$

Definícia 4.1.2. *n-vetná forma (konfigurácia) je n-tica slov $(x_1, \ldots, x_n)$ kde $x_i \in V_\Gamma^*$.*

Definícia 4.1.3. *Krok odvodenia je relácia $\underset{\Gamma}{\Rightarrow}$ na n-vetných formách definovaná nasledovne: $(x_1, \ldots, x_n) \underset{\Gamma}{\Rightarrow} (y_1, \ldots, y_n)$ práve vtedy ke nastane jeden z prípadov:*

1. *(prepisovací krok) x_i neobsahuje query symbol a*
    - *x_i obsahuje neterminál, potom $x_i \underset{G_i}{\Rightarrow} y_i$*
    - *x_i je terminálne slovo, potom $y_i = x_i$*

[1] väinou sa pouívajú regulárne alebo bezkontextové typy gramatík, lebo pri zloitejích typoch u máme veké problémy sledova, o taký systém vôbec robí

2. *(komunikaný krok)* x_i *obsahuje query symboly* $Q_{j_1}, \ldots, Q_{j_s}$ *potom*

   - *ak* x_{j_k} *neobsahuje query symbol, tak v* x_i *nahradíme* Q_{j_k} *vetnou formou* x_{j_k} *a* $y_{j_k} = S_{j_k}$
   - *ak* x_{j_k} *obsahuje nejaký query symbol tak v* x_i *necháme* Q_{j_k}

   pre vetky ostatné x_i *neobsahujúce query symboly platí* $y_i = x_i$.

Práve vyslovená definícia je trochu komplikovaná, pretoe neberie do úvahy nejaký konkrétny typ gramatiky, ale je pouitená pre akýkovek typ gramatiky Chomskeho hierarchie. Kee v alom sa budeme zaobera hlavne $PCGS$ s regulárnymi komponentami, vyslovíme teraz definíciu kroku odvodenia pre tieto $PCGS$, ktorá je o nieo jednoduchia.

Definícia 4.1.4. *Krok odvodenia je relácia* $\underset{\Gamma}{\Rightarrow}$ *na n-vetných formách definovaná nasledovne:* $(x_1, \ldots, x_n) \underset{\Gamma}{\Rightarrow} (y_1, \ldots, y_n)$ *práve vtedy ke nastane jeden z prípadov:*

1. x_i *neobsahuje query symbol, teda*
   - $x_i = w_i A$ *kde* $w_i \in T^*$ *a* $A \in N$*, potom* $x_i \underset{G_i}{\Rightarrow} y_i$
   - $x_i = w_i$ *je terminálne slovo, potom* $y_i = x_i$
2. x_i *obsahuje query symbol, teda* $x_i = w_i Q_j$*, a potom*
   - *ak* x_j *neobsahuje query symbol, tak* $y_i = w_i x_j$ *a* $y_j = S_j$
   - *ak* x_j *obsahuje nejaký query symbol, tak* $y_i = x_i$

   pre vetky ostatné x_i *neobsahujúce query symboly platí* $y_i = x_i$

Uvedomme si, e odvodenie v $PCGS$ pozostáva z prepisovacích a komunikaných krokov. Prepisovací krok nastane vtedy, ak sa v n-vetnej forme nevyskytuje ani jeden komunikaný symbol, a potom vetky gramatiky spravia jeden krok odvodenia na svojich vetných formách. Ak nejaký komponent n-vetnej formy je terminálne slovo, tak sa v alom nemení a kým nejaká gramatika nepoiada o jej obsah. Ak sa v nejakej vetnej forme vyskytne neterminál, ktorý sa nedá prepisa, tak sa odvodenie zasekne. V komunikanom kroku sa nahradia vetky *query* príslunými vetnými formami, ak tie neobsahujú *query*. Je zrejmé, e komunikaných krokov môe po sebe nasledova viac, a kým sa celá n-vetná forma "nevyistí" od *query*. Môe sa sta, e komunikácia sa zacyklí a nebude moné vykona iaden prepisovací krok. Vtedy sa odvodenie zasekne.

Definícia 4.1.5. *Jazyk generovaný PCGS systémom* Γ *je mnoina terminálnych slov vygenerovaných gramatikou* G_1. *Teda* $L(\Gamma) = \{x \in T^* \mid (S_1, \ldots, S_n) \overset{*}{\underset{\Gamma}{\Rightarrow}} (x, v_2, \ldots, v_n), v_i \in V_\Gamma^*\}$.

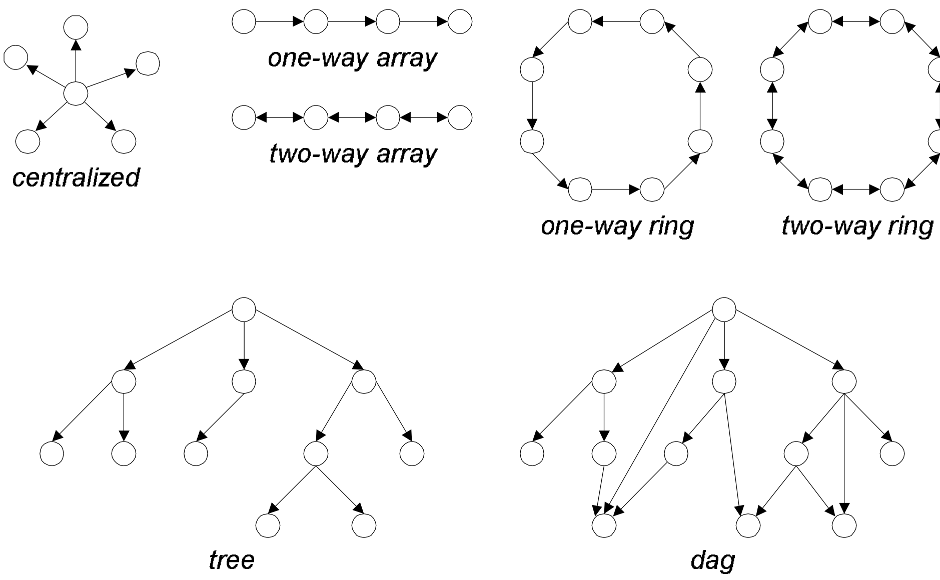
4.2 Parametre uvaované na $PCGS$

1. komunikaná truktúra - Môeme si predstavi orientovaný graf, ktorého vrcholy sú gramatiky a hrana z G_i vedie do G_j ak G_i môe generova Q_j. Komunikané truktúry delíme na dva základné typy:

   centralizované - *query* môe generova len gramatika G_1

   necentralizované - vetky ostatné napr. dag(directed acyclic graph), tree, two-way array, one-way array, two-way ring, one-way ring, ...

2. typ gramatík v komponentoch



Obr. 4.1: Príklady komunikaných truktúr *PCGS*

3. poet komponentov (gramatík)
4. poet komunikaných krokov
5. systémy s resetom resp. bez resetu - t.j. i sa vetná forma po presunutí svojho obsahu do inej vetnej formy zmení na poiatoný neterminál alebo nie.

Oznaenie: $xPCGS_nX$ - trieda $PCGS$-systémov kde $x \in \{centr, tree, dag, \dots\}$ je typ komunikanej truktúry[2], n je poet komponentov ($(n = *)$ oznauje ubolný poet komponentov) a $X \in \{REG, LIN, CF, \dots\}$ je typ komponentov[3].

4.3 Generatívna sila *PCGS*

Na bliie pochopenie sily $PCGS$ si uvedieme najskôr zopár príkladov.

Príklad 4.3.1. $\Gamma_1 = (\{S_1, S_2, S_3\}, K, \{a, b, c\}, G_1, G_2, G_3)$, kde mnoiny pravidiel prísluných gramatík sú:

- $P_1 = \{S_1 \to aS_1, S_1 \to aQ_2, S_2 \to bQ_3, S_3 \to c\}$
- $P_2 = \{S_2 \to bS_2\}$
- $P_3 = \{S_3 \to cS_3\}$

[2]ak nie je uvedený typ komunikanej truktúry, máme na mysli triedu vetkých $PCGS$-systémov bez ohadu na truktúru

[3]lineárne gramatiky (LIN) sú bezkontextové gramatiky, ktoré majú na pravej strane pravidiel najviac jeden neterminál

Skúsme si napísa pár krokov odvodenia:
$(S_1, S_2, S_3) \Rightarrow (aS_1, bS_2, cS_3) \Rightarrow \ldots k - krokov \ldots \Rightarrow (a^k S_1, b^k S_2, c^k S_3) \Rightarrow$
$(a^{k+1} Q_2, b^{k+1} S_2, c^{k+1} S_3) \Rightarrow (a^{k+1} b^{k+1} S_2, S_2, c^{k+1} S_3) \Rightarrow (a^{k+1} b^{k+2} Q_3, bS_2, c^{k+2} S_3) \Rightarrow$
$(a^{k+1} b^{k+2} c^{k+2} S_3, bS_2, S_3) \Rightarrow (a^{k+1} b^{k+2} c^{k+3}, b^2 S_2, cS_3)$
Teraz u vidíme, e $L(\Gamma_1) = \{a^n b^{n+1} c^{n+2} \mid n \geq 1\}$, a teda centralizovaný $PCGS$ s troma regulárnymi komponentami vygeneroval jazyk, ktorý nie je regulárny, ba dokonca ani bezkontextový.

Príklad 4.3.2. $\Gamma_2 = (\{S_1, S_2, A\}, K, \{a, b, c\}, G_1, G_2)$, kde

- $P_1 = \{S_1 \to aS_1, S_1 \to Q_2, A \to aS_1, A \to c\}$
- $P_2 = \{S_2 \to A, A \to bA\}$

Napíme si pár krokov odvodenia:
$(S_1, S_2) \Rightarrow (aS_1, A) \stackrel{*}{\Rightarrow} (a^k S_1, b^{k-1} A) \Rightarrow (a^k Q_2, b^k A) \Rightarrow (a^k b^k A, S_2) \Rightarrow (a^k b^k a S_1, A) \Rightarrow \ldots$
ahko vidno, e $L(\Gamma_2) = (\{a^n b^n \mid n \geq 1\})^+ c$, o je opä jazyk, ktorý nie je ani bezkontextový.

Príklad 4.3.3. $\Gamma_3 = (\{S_1, S_2\}, K, \{a, b, c, d\}, G_1, G_2)$

- $P_1 = \{S_1 \to cS_1 d, S_1 \to cQ_2 d\}$
- $P_2 = \{S_2 \to aS_2 b, S_2 \to ab\}$

Vimnime si niektoré moné odvodenia:

1. $(S_1, S_2) \stackrel{*}{\Rightarrow} (c^{k-1} S_1 d^{k-1}, a^{k-1} S_2 b^{k-1}) \Rightarrow (c^k Q_2 d^k, a^k S_2 b^k) \Rightarrow (c^k a^k S_2 b^k d^k, S_2)$ a tu sa Γ_3 zasekne
2. $(S_1, S_2) \stackrel{*}{\Rightarrow} (c^{k-1} S_1 d^{k-1}, a^{k-1} S_2 b^{k-1}) \Rightarrow (c^k Q_2 d^k, a^k b^k) \Rightarrow (c^k a^k b^k d^k, S_2)$
3. $(S_1, S_2) \stackrel{*}{\Rightarrow} (c^{k-1} S_1 d^{k-1}, a^{k-1} S_2 b^{k-1}) \Rightarrow (c^k S_1 d^k, a^k b^k) \stackrel{*}{\Rightarrow} (c^{k+i} Q_2 d^{k+i}, a^k b^k) \Rightarrow$
$(c^{k+i} a^k b^k d^{k+i}, a^k b^k)$

Teda $L(\Gamma_3) = \{c^l a^k b^k d^l \mid l \geq k \geq 1\}$. $PCGS$ s dvoma lineárnymi komponentami vygeneroval jazyk mimo bezkontextovú triedu jazykov.

Príklad 4.3.4. $\Gamma_4 = (\{S_1, S_2\}, K, \{a, b, c\}, G_1, G_2)$

- $P_1 = \{S_1 \to S_1, S_1 \to Q_2 c Q_2\}$
- $P_2 = \{S_2 \to aS_2, S_2 \to bS_2, S_2 \to a, S_2 \to b\}$

ahko vidno, e $L(\Gamma_4) = \{wcw \mid w \in \{a, b\}^+\}$, a teda $PCGS$ s dvoma bezkontextovými komponentami vygeneroval jazyk, ktorý nie je bezkontextový.

Príklad 4.3.5. $\Gamma_5 = (\{S_1, S_2, S_3\}, K, \{a, b, c, d\}, G_1, G_2, G_3)$

- $P_1 = \{S_1 \to aS_1, S_2 \to aQ_2, S_3 \to d\}$
- $P_2 = \{S_2 \to bS_2, S_2 \to bQ_3\}$
- $P_3 = \{S_3 \to cS_3\}$

Vimnime si odvodenie, v ktorom nasleduje po sebe viac komunikaných krokov:
$(S_1, S_2, S_3) \stackrel{*}{\Rightarrow} (a^k S_1, b^k S_2, c^k S_3) \Rightarrow (a^{k+1} Q_2, b^{k+1} Q_3, c^{k+1} S_3) \Rightarrow (a^{k+1} Q_2, b^{k+1} c^{k+1} S_3, S_3) \Rightarrow$
$(a^{k+1} b^{k+1} c^{k+1} S_3, S_2, S_3) \Rightarrow (a^{k+1} b^{k+1} c^{k+1} d, bS_2, cS_3)$
ahko vidno, e $L(\Gamma_5) = \{a^n b^n c^n d \mid n \geq 1\} \in \mathcal{L}_{CS} - \mathcal{L}_{CF}$.

Príklad 4.3.6. V tomto príklade ukáeme ako *PCGS* naplno vyuije silu svojej komunikácie a vyrobíme jazyk $L(\Gamma) = \{w^{2^n}c \mid w \in \{a,b\}^*, n \geq 1\}$:
$\Gamma_6 = (\{S_1, S_2, S_3\}, K, \{a,b\}, G_1, G_2, G_3)$

- $P_1 = \{S_1 \to aS_1, S_1 \to bS_1, S_1 \to Q_2, \omega_1 \to Q_3, \omega_2 \to S_1', S_1' \to c\}$
- $P_2 = \{S_2 \to S_2, S_2 \to Q_1, S_1 \to \omega_1, S_1' \to \omega_1\}$
- $P_3 = \{S_3 \to S_3, S_3 \to Q_1, S_1 \to \omega_1, \omega_1 \to \omega_2, S_1' \to \omega_1\}$

Pozrime sa na to ako funguje odvodenie v Γ:
$(S_1, S_2, S_3) \overset{*}{\Rightarrow} (wS_1, S_2, S_3) \Rightarrow (wS_1, Q_1, Q_1) \Rightarrow (S_1, wS_1, wS_1) \Rightarrow (Q_2, w\omega_1, w\omega_1) \Rightarrow$ $(w\omega_1, S_2, w\omega_1) \Rightarrow (wQ_3, S_2, w\omega_2) \Rightarrow (ww\omega_2, S_2, S_3) \Rightarrow (wwS_1', Q_1, Q_1) \overset{*}{\Rightarrow} (w^4S_1', Q_1, Q_1) \overset{*}{\Rightarrow}$ $(w^{2^n}S_1', S_2, S_3) \Rightarrow (w^{2^n}c, \dots)$
Pravidlá v Γ sú zostavené tak dômyselne, e gramatiky G_2, G_3 naraz vygenerujú komunikaný symbol Q_1, teda prenesú si vetnú formu vygenerovanú prvou gramatikou a následne G_1 poiada o vetné formy G_2 a G_3, teda obsah vetnej formy G_1 sa zdvojnásobí. Ak by gramatiky takto "nespolupracovali", tak sa Γ zasekne.
Vimnime si ete, e Γ vygeneruje slovo $w^{2^n}c$ v ase $O(n)$. G_1 na zaiatku vygeneruje w a potom pri kadom zdvojnásobení Γ pouíva u len kontantný poet krokov odvodenia. Treba si uvedomi, e je to moné len vaka tomu, e pri modeli *PCGS* máme komunikáciu v podstate zadarmo, lebo v jednom kroku dokáeme prenies ubovone veké slovo.

Veta 4.3.1. *Niekoko porovnaní PCGS s triedami Chomského hierarchie:*

1. $\mathcal{L}(PCGS_nREG) - \mathcal{L}_{\mathcal{LIN}} \neq \emptyset$ *pre* $n \geq 2$
2. $\mathcal{L}(PCGS_nREG) - \mathcal{L}_{\mathcal{CF}} \neq \emptyset$ *pre* $n \geq 3$
3. $\mathcal{L}(PCGS_nLIN) - \mathcal{L}_{\mathcal{CF}} \neq \emptyset$ *pre* $n \geq 2$

Dôkaz: Pozri príklady 4.3.2, 4.3.1 a 4.3.3. $\square$

Veta 4.3.2. $\mathcal{L}_{LIN} - \mathcal{L}(centrPCGS_*REG) \neq \emptyset$

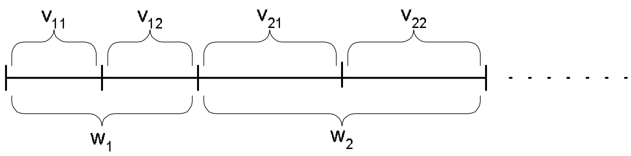
Dôkaz: Uvaujme jazyk $L = \{a^nb^mcb^ma^n \mid n,m \geq 1\}$.
Zrejme $L \in \mathcal{L}_{LIN}$. Ukáeme, e $L \notin \mathcal{L}(centrPCGS_*REG)$.
Bez ujmy na veobecnosti môme predpoklada, e G_1 negeneruje a-ka, lebo ak by generovala tak to isté dokáe aj iný komponent a G_1 môe poiada o jej výstup. Teda a-ka generujú dve iné gramatiky (G_2, G_3), lebo potrebujeme rovnaký poet a-iek na zaiatku aj na konci vetnej formy. Podobne musia existova aj dve gramatiky (G_4, G_5) na generovanie b-iek. Ak po nejakom pote krokov G_1 vygeneruje Q_2, tak v konenom (dos malom) pote krokov musí vygenerova aj Q_3, lebo G_3 sa nemá ako dozvedie, e u nemá generova a-ka, kee uvaujeme centralizovanú komunikanú truktúru. Z toho ale plynie, e v tomto konenom pote krokov musí G_1 vygenerova Q_4 a Q_5, lebo uvaujeme regulárne gramatiky. Teda G_4 a G_5 majú obmedzený as na generovanie b-iek, a teda poet b-iek nemôe by ovea väí[4] ako poet a-iek, o je spor s tým, e potrebujeme vygenerova aj slová s ubovone vekým rozdielom medzi m a n. $\square$

[4] m môe by väie od n najviac lineárne v závislosti od pravidiel v G_2, G_3 a v G_4, G_5

Veta 4.3.3. $\mathcal{L}(centrPCGS_2REG) \subsetneq \mathcal{L}_{CF}$

Dôkaz: Poda vety 4.3.2 staí dokáza u len inklúziu $\subseteq$:
Majme $\Gamma = (N, K, T, G_1, G_2) \in centrPCGS_2REG$. Chceme zostroji bezkontextovú gramatiku $G = (N', T, P, S)$ takú, e $L(\Gamma) = L(G)$. Rozírime mnoinu neterminálov takto:
$N' = N \cup \{[A, B] \mid A \in N, B \in N\} \cup \{\overline{A} \mid A \in N\}$, neskôr si ukáeme ako tieto nové neterminály budeme vyuíva. Najskôr sa zamyslime nad tým, ako bude vyzera výsledné slovo $w \in L(\Gamma)$. w bude ma nasledujúce vlastnosti (obr.4.2):

1. w sa dá dekomponova na $w = w_1 w_2 \dots w_s$ pre nejaké $s \geq 1$
2. $\forall i$ $w_i = v_{i_1} v_{i_2}$ priom v_{i_1} generuje G_1 a v_{i_2} generuje G_2
3. v_{i_1} a v_{i_2} sú generované na rovnaký poet krokov



Obr. 4.2: Tvar slova generovaného $centrPCGS_2REG$

G musí by schopná generova slová s takýmito vlastnosami. Na zabezpeenie podmienky 3 bude G generova jednotlivé podslová w_i odstredu t.j. nejaký peciálny neterminál sa bude postupne obklopova terminálmi poda pravidiel G_1, G_2.
V P budú pravidlá:

$S \to [S_1, B]B \ \forall B \in N$
: Teda máme peciálnu sadu neterminálov $[S_1, B]$, ktoré urujú, e podslovo v_{1_1} sa zaína generova z poiatoného neterminálu S_1 a podslovo v_{1_2} bude ma posledný neterminál B. To v podstate znamená, e G sa nedeterministicky rozhodne pre neterminál B.

$[A, B] \to a[A', B']b$ ak $A \to aA' \in P_1$ a $B' \to bB \in P_2$
: Tieto pravidlá nám zabezpeujú postupné simulovanie odvodenia slov v_{i_1} odpredu a slov v_{i_2} odzadu, priom toto odvodenie bude legálne vzhadom na pravidlá G_1, G_2.

$[Q_2, S_2] \to \varepsilon$
: Toto pravidlo zabezpeuje, e ak G na zaiatku dobre uhádla posledný neterminál v_{i_2}, tak po konenom pote krokov simulácia G_1 dospela ku komunikanému neterminálu Q_2 a simulácia G_2 v spätnom odvodení dospela k poiatonému neterminálu S_2, teda neterminál $[Q_2, S_2]$ sa u nebude alej rozvíja, a teda ho vymaeme.

$A \to [A, B]B \ \forall A, B \in N$
: Tieto pravidlá slúia na správne nadväzovanie podslov w_i, w_{i+1}. To znamená, e týmto pravidlom G uhádne akým neterminálom bude koni alie podslovo.

Ukáme si teraz ako funguje simulácia. Zoberme si nejaké odvodenie v Γ.
$(S_1, S_2) \Rightarrow (u_1A_1, v_1B_1) \overset{*}{\Rightarrow} (u_1u_2 \dots u_kQ_2, v_1v_2 \dots v_kB_k) \Rightarrow (u_1 \dots u_kv_1 \dots v_kB_k, S_2) \Rightarrow \dots$
Prísluná as odvodenia v G bude vyzera takto:
$S \Rightarrow [S_1, B_k]B_k \Rightarrow u_1[A_1, B_{k-1}]v_kB_k \Rightarrow u_1u_2[A_2, B_{k-2}]v_{k-1}v_kB_k \overset{*}{\Rightarrow}$
$u_1 \dots u_k[Q_2, S_2]v_1 \dots v_kB_k \Rightarrow u_1 \dots u_kv_1 \dots v_kB_k \Rightarrow \dots$
G najskôr nedeterministicky vybrala pravidlo $S \to [S_1, B_k]B_k$ tak, aby správne uhádla neterminál, ktorým bude alej pokraova G_1 v odvodení po komunikanom kroku (t.j. neterminál, ktorý bol prenáaný v komunikanom kroku). Potom postupne simulovala odvodenie G_1, G_2 a nakoniec vymazala neterminál $[Q_2, S_2]$ z vetnej formy. Takto ostal vo vetnej forme len neterminál B_k, ktorým zane aliu simuláciu G_1. Teraz G pouije nejaké z pravidiel $B_k \to [B_k, B]B$ na uhádnutie neterminálu, ktorý bude prenáaný v nasledujúcom komunikanom kroku at.

Ete sa musíme zamyslie nad tým, ako simuláciu ukoni. Môu nasta dva prípady:

1. G_1 u nepoiada G_2 o jej vetnú formu (teda nevyprodukuje Q_2) a sama vyprodukuje terminálne slovo. Pre túto monos dodáme do P tieto pravidlá:

   - $B \to \overline{B}\ \forall B \in N$
   - $\overline{A} \to u\overline{B}$ ak $A \to uB \in P_1$

   Tým, e sme pouili "pruhované" neterminály a nemáme pravidlo typu $\overline{A} \to A$ sme zmarili aliu[5] komunikáciu, a teda simulujeme G_1 a kým nevygeneruje terminálne slovo.

2. G_2 skoní skôr (t.j. vygeneruje terminálne slovo) ako G_1 poiada o jej vetnú formu. V Γ sa vetná forma G_2 nemení a aká, kým G_1 vygeneruje Q_2. Pre túto monos dodáme do P tieto pravidlá:

   - $A \to [A, \varepsilon]\ \forall A \in N$
   - $[A, \varepsilon] \to u[A', \varepsilon]$ ak $A \to uA' \in P_1$
   - $[A, \varepsilon] \to u[A', B]\ \forall B \in N$ ak $A \to uA' \in P_1$

   Teda pravidlá typu $[A, \varepsilon] \to u[A', \varepsilon]$ simulujú akaknie G_2 na komunikáciu a pravidlá typu $[A, \varepsilon] \to u[A', B]$ opä slúia na uhádnutie posledného neterminálu, ktorý sa objavil vo vetnej forme G_2.

$\square$

Lema 4.3.1. *(Pumpovacia lema pre centrPCGS$_n$REG)*
Nech $L \in \mathcal{L}(centrPCGS_nREG)$. Potom existuje prirodzené íslo M, e $\forall w \in L$ také, e $|w| > M$ existuje m $1 \leq m \leq n$ a existujú x_i, y_i tak, e sú splnené nasledovné podmienky:

1. $w = x_1y_1x_2y_2 \dots x_my_mx_{m+1}$
2. $\forall i\ y_i \neq \varepsilon$
3. $\forall k \geq 0\ :\ x_1{y_1}^kx_2{y_2}^k \dots x_m{y_m}^kx_{m+1} \in L$

Dôkaz: Nech $\Gamma = (N, K, T, G_1, \dots, G_n) \in centrPCGS_nREG$. Kadá gramatika má vo svojej vetnej forme najviac jeden neterminál a ten je posledným symbolom tejto vetnej formy. Teda konfigurácia Γ je tvaru $c = (x_1A_1, x_2A_2, \dots, x_nA_n)$. Budeme hovori, e dve konfigurácie $c_1 = (x_1A_1, x_2A_2, \dots, x_nA_n)$ a $c_2 = (y_1B_1, y_2B_2, \dots, y_nB_n)$, kde $\forall i\ x_i, y_i \in T^+$ a $\forall i\ A_i, B_i \in N \cup \{\varepsilon\}$

[5] k dispozícii máme samozrejme aj pravidlo $S \to \overline{S}$ teda iadna komunikácia nemusí nasta

sú ekvivalentné (ozn. $c_1 \equiv c_2$), ak $\forall i: \; A_i = B_i$.
Uvaujme slovo $w \in L(\Gamma)$ a jeho odvodenie minimálnej dky. Ak M je poet vetkých moných rôznych výskytov neterminálov vo vetných formách[6], tak za predpokladu, e $|w| > M$, existujú v tomto odvodení dve konfigurácie c_1 a c_2 spajúce nasledovné podmienky:

1. $c_1 \equiv c_2$
2. ak je v odvodení medzi konfiguráciami c_1 a c_2 pouitý komunikaný symbol Q_i, $2 \leq i \leq n$, tak $x_i = y_i$

Teda odvodenie je tvaru:
$(S_1, S_2, \dots, S_n) \overset{*}{\Rightarrow} (x_1A_1, x_2A_2, \dots, x_nA_n) \overset{*}{\Rightarrow} (x_1z_1A_1, x_2z_2A_2, \dots, x_nz_nA_n) \overset{*}{\Rightarrow} (w, \dots)$
Ak je v odvodení medzi konfiguráciami c_1 a c_2 pouitý komunikaný symbol Q_i, $2 \leq i \leq n$, tak poda podmienky (2) platí, e $z_i = \varepsilon$. Pre ostatné komponenty konfigurácie nastáva jedna z nasledujúcich moností:

1. $z_1 \in T^+$ t.j. z_1 je neprázdne terminálne slovo.
2. existuje j, $2 \leq j \leq n$ také, e Q_j nie je pouité v odvodení medzi konfiguráciami c_1 a c_2 ale je pouité v odvodení, ktoré zaína konfiguráciou c_2, naviac $z_j \in T^+$ teda z_j je neprázdne terminálne slovo.

Predpokladajme, e ani jedna z týchto moností nenastane. Potom jednotlivé komponenty konfigurácií c_1 a c_2 sú totoné. Z toho vyplýva, e as odvodenia ($c_1 \overset{*}{\Rightarrow} c_2$) medzi konfiguráciami c_1 a c_2 môeme z odvodenia vynecha, ím dostaneme kratie odvodenie slova w, o je spor s predpokladom, e sme uvaovali najkratie odvodenie w.
Teda jedna z uvedených moností urite nastane. V tomto prípade môeme postupnos krokov odvodenia medzi c_1 a c_2 opakova k krát pre ubovoné[7] k. Po týchto k iteráciach sa komponent j, pre ktorý z_j bolo neprázdne terminálne slovo, zmení na $x_j {z_j}^k A_j$. Ak po týchto k iteráciach dokoníme odvodenie asou pôvodného odvodenia $c_2 \overset{*}{\Rightarrow} (w, \dots)$, tak dostaneme nové slovo $w' \in L(\Gamma)$, ktoré sa od pôvodného w líi napumpovaním asti z_j.
Ak si uvedomíme, e poet podslov, ktoré môu by takto pumpované, môe by najviac n, tak sme s dôkazom tejto lemy hotoví. □

Podobné pumpovacie lemy platia aj pre triedy *tree*$PCGS_n$*REG* a *dag*$PCGS_n$*REG*, avak pre veobecnú necentralizovanú komunikanú truktúru *PCGS* sa pumpova nedá.

Veta 4.3.4. $\mathcal{L}(centrPCGS_{n-1}REG) \subsetneq \mathcal{L}(centrPCGS_nREG)$ *pre* $n \geq 2$

Dôkaz: Inklúzia $\subseteq$ je zrejmá.
Treba ukáza, e existuje jazyk L taký, e $L \in \mathcal{L}(centrPCGS_nREG)$ a $L \notin \mathcal{L}(centrPCGS_{n-1}REG)$. Uvaujme jazyky $L_n = \{{a_1}^{k+1}{a_2}^{k+2}{a_3}^{k+3} \dots {a_n}^{k+n} \mid k \geq 0\}$. Zostrojme *centr*$PCGS_n$*REG* $\Gamma_n = (\{S_1, \dots, S_n\}, K, \{a_1, \dots, a_n\}, G_1, \dots, G_n)$, kde

- $P_1 = \{S_1 \to a_1S_1, S_n \to a_n\} \cup \{S_i \to a_iQ_{i+1} \mid 1 \leq i \leq n-1\}$
- $P_j = \{S_j \to a_jS_j\}$ pre $2 \leq j \leq n$

Zrejme[8] $L(\Gamma_n) = L_n$. Chceme ukáza, e $L_n \notin \mathcal{L}(centrPCGS_{n-1}REG)$. Sporom, predpokladjme, e $L_n \in \mathcal{L}(centrPCGS_{n-1}REG)$. Poda lemy 4.3.1 pre tento jazyk existuje prirodzené íslo M.

[6]tých je $M = |N \cup K|^n$
[7]ak túto postupnos krokov úplne vynecháme, tie dostaneme legálne odvodnie v Γ, teda pripúame aj $k = 0$
[8]itateovi to môe by zrejmejie, ak sa pozrie na príklad 4.3.1

Zoberme si slovo $w = {a_1}^{M+1}{a_2}^{M+2}{a_3}^{M+3} \dots {a_n}^{M+n} \in L_n$. Zjavne $|w| \geq M$. Slovo w môme pumpova najviac na $n-1$ miestach. Aby toto slovo po pumpovaní ostalo v L_n potrebujeme ho pumpova na n miestach, a to je spor s predpokladom. □

Dôsledok 4.3.1. Hierarchia $\mathcal{L}(centrPCGS_nREG)$, $n \geq 1$ je nekonená.

Podobnými dokazovacími technikami (t.j. vyuitím pumpovacích liem pre nejaký jazyk) by sme sa dopracovali k nasledujúcim dvom tvrdeniam:

Veta 4.3.5. *Hierarchie nasledujúcich tried sú nekonené:*

- $\mathcal{L}(treePCGS_nREG)$, $n \geq 1$
- $\mathcal{L}(dagPCGS_nREG)$, $n \geq 1$

Hoci pre triedy $PCGS$ s komunikanými truktúrami *array* a *ring* nie sú známe pumpovacie lemy, inými spôsobmi sa dájú dokáza nasledujúce tri tvrdenia:

Veta 4.3.6. *Hierarchie nasledujúcich tried sú nekonené:*

- $\mathcal{L}(two-way-arrayPCGS_nREG)$, $n \geq 1$
- $\mathcal{L}(two-way-ringPCGS_nREG)$, $n \geq 1$
- $\mathcal{L}(one-way-ringPCGS_nREG)$, $n \geq 1$

Veta 4.3.7. $\mathcal{L}(PCGS_{n-1}REG) \subsetneq \mathcal{L}(PCGS_nREG)$ *pre* $n \geq 2$

Dôkaz: Uvaujme jazyky $L_n = \{{a_1}^k{a_2}^k \dots a_{2n-2}^k \mid k \geq 1\}$. Dá sa skontruova $\Gamma \in PCGS_nREG$ tak, e $L(\Gamma) = L_n$. Potom zrejme $L_n \in \mathcal{L}(PCGS_nREG)$. Treba ukáza, e $L_n \notin \mathcal{L}(PCGS_{n-1}REG)$. Tento dôkaz je vak dos technický, a preto ho tu nebudeme uvádza[9]. □

Dôsledok 4.3.2. Hierarchia $\mathcal{L}(PCGS_nREG)$, $n \geq 1$ je nekonená.

Poznámka 4.3.1. *Otvorenými problémami zostávajú nekonenos hierarchií* $\mathcal{L}(centrPCGS_nX), \mathcal{L}(PCGS_nX)$ *pre* $n \geq 1$ *a* $X \in \{CF, CS\}$.

4.4 Porovnanie $PCGS$ so sekvennými triedami

Veta 4.4.1. *Nech* Γ *je* $treePCGS_mREG(f(n))$, *kde* $f(n)$ *je poet komunikaných krokov potrebných na vygenerovanie slova dky* n. *Potom existuje*[10] *nedeterministický* $(m-1)$ *poítadlový automat* M *pracujúci v lineárnom ase a akceptujúci jazyk* $L(\Gamma)$ *priom vykoná* $2f(n)$ *obratov*[11] *a* $f(n)$ *testov na nulu*[12].

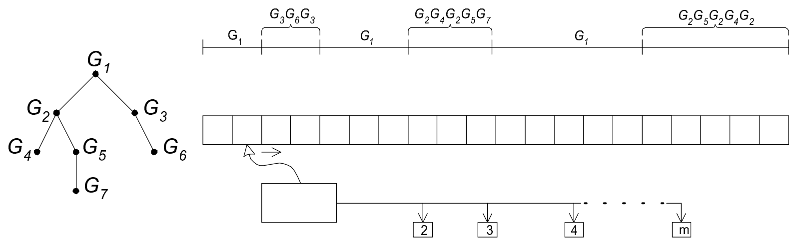
[9] podrobnosti mono nájs v [5]

[10] n poítadlový automat je zásobníkový automat s n zásobníkmi, ktoré pracujú nad jednopísmenkovou abecedou

[11] zmena smeru hlavy v zásobníku resp. zmena inkrementácie counteru na dekrementáciu a opane

[12] dosiahnutie dna zásobníka resp. counter dosiahol nulu

Dôkaz: Nech Γ má komponenty $G_1, \ldots, G_m$. Chceme zostroji nedeterministický $(m-1)$ poítadlový automat M simulujúci Γ. Celá simulácia Γ nám bude jasnejia, ak sa zamyslíme nad tým, ako môe vyzera slovo $w \in L(\Gamma)$ pri stromovej komunikanej truktúre. Slovo $w = w_{i_1} w_{i_2} \ldots w_{i_k}$, $\forall j \ i_j \in \{1, \ldots, m\}$ pozostáva z podslov w_{i_j} generovaných gramatikami G_i, priom dky týchto podslov závisia od konkrétnej komunikanej truktúry (obr.4.3).



Obr. 4.3: Viacpoítadlový automat a tvar slova generovaného *treePCGS*

Vieme, e kadá regulárna gramatika sa dá simulova pomocou nedeterministického koneného automatu[13]. Podobne budú v pravidlách náho automatu M simulované pravidlá regulárnych komponentov $G_1, \ldots, G_m$. M bude pouíva svoje poítadlá $C_2, \ldots, C_m$ na zabepeenie toho, aby jednotlivé gramatiky $G_1, \ldots, G_m$ neboli simulované dlhie ako to vyaduje daná situácia (konfigurácia). V kadej konfigurácii M bude íslo $c(C_i)$ $\forall i \in \{2, \ldots, m\}$ uloené v C_i uchováva rozdiel medzi potom prepisovacích krokov G_i simulovaných M a potom prepisovacích krokov otca G_i v stromovej komunikanej truktúre (to znamená, e ak G_i je poiadaná svojim otcom o vyprodukovanú vetnú formu, tak túto vetnú formu môe G_i generova najviac $c(C_i)$ krokov).
M nedeterministicky striedavo simuluje gramatiky $G_1, \ldots, G_m$ a zárove overuje, i slovo generované gramatikami je zhodné so slovom na vstupnej páske. Na zaiatku M simuluje G_1 a priebene overuje vstupnú pásku. Zárove po kadom odsimulovanom prepisovacom kroku G_1 inkrementuje poítadlá vetkým synom G_1 a ostatné poítadlá zostanú nezmenené. Simulácia G_1 skoní, ke G_1 vygeneruje komunikaný symbol Q_i pre nejaké i. M zane simulova gramatiku G_i z poiatoného neterminálu S_i. Poas tejto simulácie po kadom prepisovacom kroku G_i sa dekrementuje poítadlo C_i a inkrementujú sa poítadlá vetkých synov G_i. Ak G_i vyprodukuje terminálnu vetnú formu, tak M zastaví a akceptuje vstupné slovo práve vtedy, ke bol vstup preítaný a do konca. Ak C_i je prázdne ($c(C_i) = 0$) a G_i v poslednom kroku vygeneroval na konci neterminál A, tak M zane simulova otca G_i (v tomto prípade G_1) priom pokrauje v prepisovaní z neterminálu A. Ak G_1 nemôe prepísa A (t.j. nemá pravidlá na A), tak M neakceptuje vstupné slovo. Ak G_i vegeneruje komunikaný symbol Q_j pre nejaké j, tak M zane simulova G_j (kde G_j je synom G_i) z poiatoného neterminálu S_j. M od tohto momentu po kadom prepisovacom kroku G_j dekrementuje C_j a inkrementuje poítadlá vetkých svojich synov. M takto rekurzívne simuluje gramatiky a kým poslená gramatika nevygeneruje terminálnu vetnú formu.
Z popisu práce M by malo by zrejmé, e poet obratov je $2f(n)$ a poet testov na nulu je $f(n)$, lebo poítadlo C_i zane klesa práve vtedy, ke je vygenerovaný komunikaný symbol Q_i.
Ak sa v pravidlách gramatík nenachádzajú iadne *chainrules*[14], tak M pracuje v ase n. Ak sú v gramatikách takéto pravidlá, tak M pracuje v ase $O(n)$, pretoe existuje kontanta d taká, e pre kadé $w \in L(\Gamma)$ existuje odvodenie, ktoré v kadých d krokoch vygeneruje aspo jeden terminál. $\square$

Uvedomme si, e $m-1$ poítadlový automat vieme simulova viacpáskovým TS pracujúcim v

[13]pozri [1]
[14]pravidlá typu $A \to B$ kde A, B sú neterminály

rovnakom ase, a e poítadlá uchovávajú ísla vekosti najviac n,take v binárnom kódovaní sa zmestia do priestoru $O(\log n)$ Z týchto dvoch poznatkov plynie nasledujúce tvrdenie.

Veta 4.4.2. $\mathcal{L}(treePCGS_mREG(f(n)) \subseteq NTIME(n) \cap NSPACE(\log n)$

Poznámka 4.4.1. *Predchádzajúca veta nehovorí, e vieme zostroji ekvivalentný TS pracujúci v lineárnom ase a logaritmickom priestore. Hovorí len, e vieme zostroji ekvivalentný TS pracujúci v lineárnom ase a (iný) ekvivalentný TS pracujúci v logaritmickom priestore.*

Skúsme sa zamyslie nad simuláciou $PCGS_mREG(f(n))$ priamoiaro pomocou m-páskového TS. Ten by na kadej páske simuloval prepisovanie vetnej formy jednej gramatiky. Komunikácia medzi gramatikami v tomto prípade znamená presunutie obsahu jednej pásky na druhú. Kee vekos pásky je $O(n)$ a komunikácií je $f(n) \in O(n)$, tak asová zloitos je $O(n^2)$. Pri takejto simulácii sa nám ahko môe sta, e nejaký kúsok vetnej formy bol presúvaný medzi páskami $O(n)$ krát a nakoniec sa aj tak nedostal do výsledného terminálneho slova. V nasledujúcej vete budeme toto "zbytoné" presúvanie eliminova a dostaneme dobrý výsledok pre $dagPCGS$.

Veta 4.4.3. $\mathcal{L}(dagPCGS_*REG) \subseteq NTIME(n)$

Dôkaz: Nech $\Gamma = (N, K, T, G_1, \ldots, G_m) \in dagPCGS_mREG$. Chceme zostroji m-páskový[15] nedeterministický TS A pracujúci v lineárnom ase a akceptujúci jazyk $L(\Gamma)$.
V δ-funkcii si A uchováva pravidlá vetkých m gramatík a v stavoch si drí aktuálne neterminály[16] vetkých gramatík. Na páskach $T_1, \ldots, T_m$ sú vetné formy generované gramatikami $G_1, \ldots, G_m$ alebo T_i obsahuje blank symbol B, ak sa A nedeterministicky rozhodne, e vetnú formu, ktorú generuje G_i sa neobjaví vo výslednom terminálnom slove generovanom Γ t.j eliminovanie "zbytoných" presúvaní z jednej pásky na druhú.
Na zaiatku bude A v stave so zaiatoným neterminálom pre kadú gramatiku. Pre kadé $i = 1..m$ A uhádne, i sa vetná forma generovaná gramatikou G_i bude nachádza vo výslednom terminálnom slove. Ak A rozhodne, e vetná forma generovaná G_i bude vo výslednom slove, tak A bude simulova odvodenie gramatiky G_i na páske T_i. Ak A rozhodne, e nebude, tak A zapíe na pásku T_i symbol B a alej nesimuluje odvodenie G_i na páske T_i. Takto A simuluje iba to, o sa vo výslednom slove naozaj objaví.
Jeden krok simulácie A pozostáva z jedného prepisovacieho kroku kadej z gramatík, pre ktoré sa A rozhodol, e ich bude simulova. To môeme, lebo ako sme si povedali A si v stave uchováva aktuálne neterminály kadej z gramatík. Takto A pokrauje, a kým sa na páske T_1 neobjaví terminálne slovo, alebo a kým sa aspo na jednej páske neobjaví *query* symbol.
Ak sa na T_1 objaví terminálne slovo, tak A porovná obsah pásky T_1 so vstupom a ak sa rovnajú, tak A akceptuje vstupné slovo.

Ak aktuálny neterminál G_i je Q_j a

- ani T_i ani T_j neobsahuje B, tak A prekopíruje obsah T_j na miesto Q_j na páske T_i a celý obsah pásky T_j prepíe na S_j (poiatoný neterminál gramatiky G_j). Po tomto pokrauje A v simulácii G_i z neterminálu, ktorý bol posledným symbolom pri kopírovaní. A uhádne, i alia vetná forma generovaná G_j bude asou výsledného slova alebo nie. Poda toho zane simulova G_j z poiatoného neterminálu S_j alebo na T_j zapíe B.
- bu T_i alebo T_j obsahuje B, tak to znamená, e A uhádol nesprávne[17] a výpoet sa zasekne.

[15] to znamená, e TS A má m hláv

[16] kee gramatiky sú regulárne, vo vetnej forme danej gramatiky sa vyskytuje najviac jeden neterminál

[17] Ak T_j obsahuje B, tak T_i sa objaví vo výsledku, a teda aj T_j bude vo výsledku, kee si ju G_i vyiadala, teda A zle uhádol. Ak T_i obsahuje B, tak T_j sa má objavi vo výsledku, ale nemá sa ako dosta cez T_i, take A zle uhádol.

- T_i aj T_j obsahuje B, tak A nezmení obsah T_i a nedeterministicky uhádne, i sa alia vetná forma generovaná G_j objaví vo výslednom terminálnom slove. Poda toho A zane na T_j simulova G_j od poiatoného neterminálu S_j alebo na T_j nechá B.

Ak sa naraz na viacerých páskach objavia komunikané symboly, tak A kopíruje vetné formy v uritom poradí tak, aby nekopíroval nejaký komunikaný symbol. To sa urite dá, pretoe uvaujeme komunikanú truktúru dag, ktorá nepripúa cykly.
Pre kadé slovo $w \in L(\Gamma)$ existuje postupnos správnych nedeterministických rozhodnutí TS A, ktorá vedie k odvodeniu tohto slova na páske T_1. Kee komunikaná truktúra nepripúa cykly, tak kadý symbol akceptovaného slova w bol kopírovaný z jednej pásky na druhú najviac $m-1$ krát, o je kontantne vea. Teda na prekopírovanie vetkých symbolov slova w potrebujeme as $O(n)$. Na páskach generujeme iba symboly, ktoré sa objavia vo výslednom slove w, teda na vygenerovanie vetkých symbolov slova w potrebujeme as $O(n)$. Konene na porovnanie terminálneho slova na páske T_1 so vstupom potrebujeme as $O(n)$. Teda celkovo pracuje A v ase $O(n)$. $\square$

Uvedieme ete niekoko tvrdení, ktoré hovoria o tom, e ani zvýenie potu gramatík, ani zvýenie potu komunikaných liniek v stromovej truktúre nedokáe vykompenzova obmedzenie potu komunikaných krokov.

Veta 4.4.4. $\mathcal{L}(centrPCGS_2REG(n)) - \mathcal{L}(treePCGS_*REG(f(n))) \neq \emptyset$ *pre* $f(n) \notin \Omega(n)$

Dôkaz: Uvedieme[18] len, e dôkaz uvauje jazyk

$$L = \{a^{i_1}b^{i_1}a^{i_2}b^{i_2}\dots a^{i_k}b^{i_k}c \mid k \geq 1, i_j \geq 1 \; pre \; j \in \{1,\dots,n\}\}$$

o ktorom sa ukáe, e $L \in \mathcal{L}(centrPCGS_2REG(n))$ a $L \notin \mathcal{L}(treePCGS_*REG(f(n)))$. Vyuíva sa pri tom simulácia $treePCGS_mREG(f(n))$ pomocou $m-1$ poítadlového automatu ako to bolo uvedené vo vete 4.4.1. $\square$

Veta 4.4.5. *Pre* $f(n) \notin \Omega(n)$ *platí:*

- $\mathcal{L}(one-way-arrayPCGS_m(f(n))) \subsetneq \mathcal{L}(one-way-arrayPCGS_m(n))$ *pre* $m \geq 2$
- $\mathcal{L}(centrPCGS_m(f(n))) \subsetneq \mathcal{L}(centrPCGS_m(n))$ *pre* $m \geq 2$
- $\mathcal{L}(treePCGS_m(f(n))) \subsetneq \mathcal{L}(treePCGS_m(n))$ *pre* $m \geq 2$

Veta 4.4.6. $\mathcal{L}(centrPCGS_{k+1}(k)) - \mathcal{L}(treePCGS_*(k-1)) \neq \emptyset$ *pre ubovonú kontantu* k.

Veta 4.4.7. *Pre ubovoné* $k \geq 1$ *a ubovoné* $x \in \{centr, tree, one-way-array\}$ *platí:*

- $\mathcal{L}(xPCGS_{k+1}(k-1)) \subsetneq \mathcal{L}(xPCGS_{k+1}(k))$
- $\mathcal{L}(xPCGS_*(k-1)) \subsetneq \mathcal{L}(xPCGS_*(k))$

4.5 Niektoré alie vlastnosti *PCGS*

Veta 4.5.1. $\mathcal{L}(PCGS_*CF)$ *je úplná AFL.*

Veta 4.5.2. $centrPCGS_*CF <_4^{Var} CF$

Veta 4.5.3. $centrPCGS_*CF <_4^{Prod} CF$

[18] podrobnosti mono nájs v [6]

Kapitola 5

Alternujúce stroje

V predchádzajúcich kapitolách sme si ukázali viaceré paralelné modely. I ke lo o rôzne pohady na paralelizmus, vetky mali jedno spoloné: pozerali sa na vec z "gramatikového" pohadu, teda vo vetkých prípadoch sme mali jednu, prípadne viacero gramatík, ktoré osi generovali. Teraz si ukáeme jednu automatovú charakterizáciu a porovnáme ju so sekvennými triedami. Pre itatea, ktorý sa s pojmom alternovania ete nestretol, povedzme nasledovných pár viet. Modely alternujúcich strojov sa uvaujú pre vetky známe zariadenia, my si ich popíeme na najveobecnejom prípade, teda na Turingových strojoch, pre itatea iste nebude problémom analogicky si zadefinova alternujúce konené automaty, alternujúce zásobníkové automaty, prípadne alternujúce lineárne ohraniené automaty. O niektorých týchto zariadeniach si v závere kapitoly ukáeme pár zaujímavých vecí, napr. ako alternovanie ovplyvní, resp. neovplyvní ich generatívnu silu a podobne. Zariadenie (alternujúci stroj) má tzv. existenné a tzv. univerzálne stavy, ktoré môe ubovone kombinova, a ktoré si alej bliie popíeme. Alternujúci Turingov stroj sa v existenných stavoch správa rovnako, ako u známy nedeterministický model Turingovho stroja. Odliné je správanie sa v univerzálnych stavoch. Zariadenie sa akoby rozdelí na n nových strojov (kde n je poet konfigurácií dosiahnutených na 1 krok z jeho momentálnej konfigurácie), ktoré dostanú novú pamä (rovnakú ako pôvodný stroj) a alej nezávisle od seba pracujú, táto operácia sa nazýva FORK a itateovi môe by troku v reálnejej podobe známa z niektorých operaných systémov (UNIX), kde jeden proces vytvára nové a prideuje im pamä.

5.1 Definície a oznaenia

Definícia 5.1.1. *Alternujúci Turingov stroj (ATS) je 6-tica $A = (K, \Sigma, \Gamma, \delta, q_0, F)$, kde K je konená mnoina stavov rozdelená na dve disjunktné asti $K = K_1 \cup K_2$, K_1 je mnoina existenných stavov, K_2 je mnoina univerzálnych stavov, Σ je abeceda vstupných symbolov, Γ je pracovná (pásková) abeceda, q_0 je poiatoný stav a F je mnoina akceptaných stavov, δ je prechodová funkcia*

$$\delta : K \times \Gamma \to 2^{K \times \Gamma \times \{-1,0,1\}}$$

Definícia 5.1.2. *Konfiguráciu ATS A definujeme rovnako ako pre nedeterministické Turingove stroje (NTS), bliie napr. [1]*

Definícia 5.1.3. *Krok výpotu ATS A je relácia $\vdash$ na konfiguráciách definovaná rovnako ako pre NTS* itate si mono spomenie na pojem stromu konfigurácií, definovaný pre nedeterministický Turingov stroj, ktorý prehadne reprezentoval jeho výpoet na konkrétnom slove[1]. Podobný strom definujeme aj pre alternujúce stroje. Okrem toho, e nám tento pojem pomôe pri definovaní výpotu ATS, mal mi výraznou mierou prispie k pochopeniu alternovania samotného.

Definícia 5.1.4. *Úplný strom konfigurácií pre ATS A a vstupné slovo w je strom, ktorého vrcholy sú oznaené konfiguráciami taký, e:*

1. *kore je poiatoná konfigurácia na slove w*
2. *kadý vrchol má za priamych nasledovníkov práve toko vrcholov, koko konfigurácii je v relácii* $\vdash$ *s konfiguráciou oznaujúcou daný vrchol*
3. *tieto vrcholy sú oznaené týmito konfiguráciami*

Poznámka 5.1.1. *Úplný strom konfigurácií nemusí by konený*

Definícia 5.1.5. *Výpoet ATS A (na slove w) je podstrom úplného stromu konfigurácií na slove w taký, e:*

1. *obsahuje kore*
2. *s kadým univerzálnym vrcholom* ($q \in K_2$) *obsahuje vetkých jeho priamych nasledovníkov*
3. *s kadým existenným vrcholom obsahuje práve jedného jeho priameho nasledovníka, ak existuje*

Definícia 5.1.6. *Akceptujúci výpoet ATS A na slove w je taký výpoet na w, ktorý je konený a kadá listová konfigurácia je akceptaná*

Definícia 5.1.7. *Jazyk akceptovaný ATS A je*

$$L(A) = \{w \in \Sigma^* \mid \textit{existuje akceptaný výpoet } A \textit{ na } w\}$$

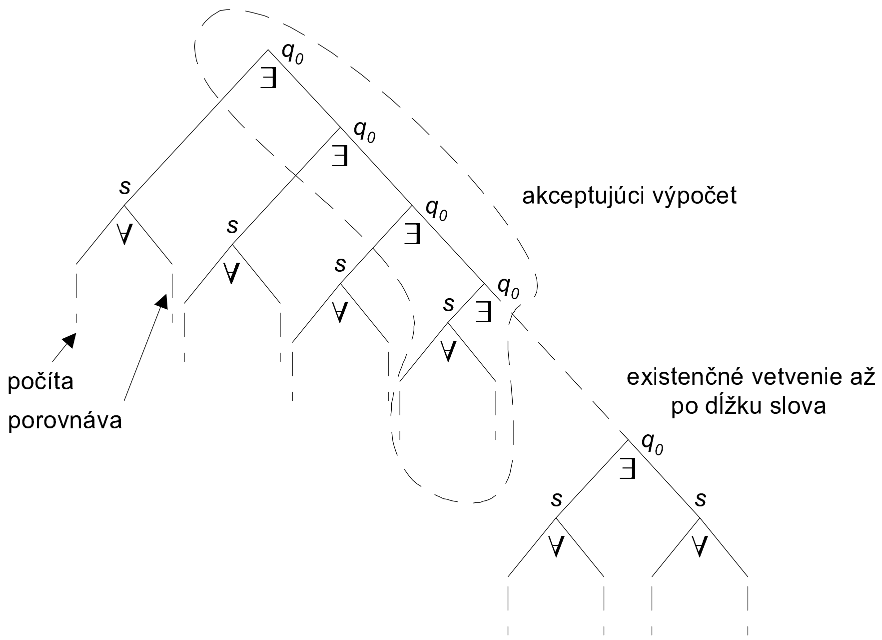
Príklad 5.1.1. Ukáeme si, ako alternovanie pomôe dvojhlavým koneným automatom[2] v generatívnej sile. Zoberme si jazyk $L = \{ww \mid w \in \{a,b\}^*\}$. Dalo by sa ukáza, e ho nie je moné akceptova dvojhlavým koneným automatom, no zostrojíme pre alternujúci dvojhlavý konený automat, ktorý bude pracova nasledovne:

1. dostane na vstup slovo, nedeterministicky uhádne jeho polovicu
2. nastane FORK, automat alternuje v univerzálnom stave
   - 1. proces overuje, i vstup je tvaru *ww* tak, e hlavy sa synchrónne pohybujú po vstupe, v jednom kroku kadá íta symbol, ak obe preítajú to isté a ke sa jedna z nich dostane na koniec vstupu, dôjde k akceptovaniu
   - 2. proces overuje, i automat uhádol polovicu správne tak, e prvá hlava sa hýbe o jedno políko vpravo, zatia o za ten istý as sa druhá hlava hýbe o dve políka vpravo, k akceptovaniu dôjde, ak obe hlavy doítajú naraz vstupné slovo

Výpoet automatu mono vidie na (obr.5.1)

[1] bol napr. dobrou pomôckou pri dôkaze ekvivalencie deteministických a nedeterministických Turingových strojov

[2] budeme uvaova zariadenie s monosou pohybu hlavy iba jedným smerom



Obr. 5.1: Úplný strom konfigurácií automatu pre jazyk $L = \{ww \mid w \in \{a,b\}^*\}$ z príkladu 5.1.1

Veta 5.1.1. $\mathcal{L}_{ATS} = \mathcal{L}_{RE}$

Dôkaz: Povieme si pár slov o oboch inklúziách:

$\subseteq$: vyplýva z Turingovej tézy, keby sme mali zkontruova TS, ktorý by simuloval ATS, tak by sme zrejme prechádzali strom konfigurácii ATS do írky, pamätali si nieo o výpote a charaktere stavov a podobne

$\supseteq$: na NTS sa mono pozera aj ako na peciálny prípad ATS, ke akoby vetky stavy NTS boli existenné

□

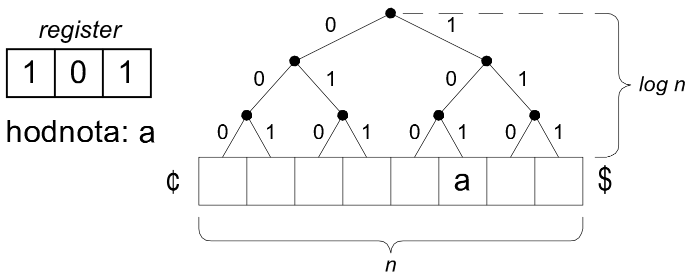
5.2 Miery zloitosti

Pre ATS si zadefinujeme dve miery, ktoré sú analógiou mier $NSPACE, NTIME$ definovaných pre NTS:

- $ASPACE$ = najväi poet pouitých políok na páske v niektorej konfigurácii akceptaného výpotu
- $ATIME$ = hbka akceptaného výpotu v strome konfigurácií

V niektorých simuláciách budeme pracova s triedou jazykov $ATIME(f(n))$, $\log n < f(n) < n$. Klasický model ATS by sme tu nemohli poui, lebo na preítanie vstupu dky n potrebujeme aspo

n krokov. V takýchto prípadoch sa pouíva model ATS s rýchlym prístupom na vstup. Môeme si ho reprezentova nasledovne: nad páskou má binárny vyhadávací strom, kadý list je práve jedno políko, strom má výku $\log n$, máme register dky $\log n$, ke do neho binárne zapíeme íslo k, tak za as $\log n$ sa dostaneme ku k-temu vstupnému políku (obr.5.2).



Obr. 5.2: Model ATS s rýchlym prístupom na vstup

5.3 Alternujúce vs. sekvenné triedy zloitosti

Kee vo väine simulácií budeme poadova predpoklad páskovej kontruovatenosti nejakej funkcie, bude dobré, ke si tento pojem zadefinujeme.

Definícia 5.3.1. *Funkcia $f(n)$ sa nazýva páskovo kontruovatená, ak existuje deterministický Turingov stroj (DTS) T, ktorý je $f(n)$ páskovo ohraniený a vyznaí na páske $f(n)$ políok*

Ukazuje sa, e takmer vetky funkcie, ktoré si mono reálne predstavi, sú páskovo kontruovatené, problémy pri kontruovatenosti vak nastávajú, ke sa pozrieme na triedu funkcií meních ako $\log n$, teda $O(\log n)$. Keby sme chceli nahliadnu medzi nekontruovatené funkcie, dobrým prostriedkom na ich zostrojovanie by bola metóda diagonalizácie.

V tejto chvíli sme u pripravení na ukázanie základného výsledku o alternujúcich Turingových strojoch.

Veta 5.3.1. $NSPACE(S(n)) \subseteq ATIME(S^2(n))$ *pre* $S(n) \geq \log n$

Dôkaz: Ak $S(n) \geq n$, nie je problém v lineárnom ase preíta vstup, ak by vak bolo $S(n) < n$, na preítanie vstupu naim ATS A' by sme potrebovali aspo lineárny as a celá simulácia by ztroskotala u na zaiatku. Preto pouijeme model ATS s rýchlym prístupom na vstup, v logaritmickom ase $O(\log n)$ sa vieme dosta na ubovoné políko vstupnej pásky, vyuívame tu ideu paralelnej práce procesov, nie kadý proces musí vidie celý vstup, aby akceptoval, resp. kadému procesu bude k práci stai malý úsek vstupu. K danému NTS A pracujúcemu v priestore $S(n)$ zostrojíme ATS A' pracujúci v ase $S^2(n)$ taký, e $L(A) = L(A')$.

Akceptujúci výpoet A na slove w dky n má (zmysluplne) dku najviac $c^{S(n)}$ pre vhodné[3] c. Polome $m = c^{S(n)}$, $\forall i\ \ k_i$ je konfigurácia A, akceptujúci výpoet má tvar:

$$k_1 \vdash k_2 \vdash \cdots \vdash k_m$$

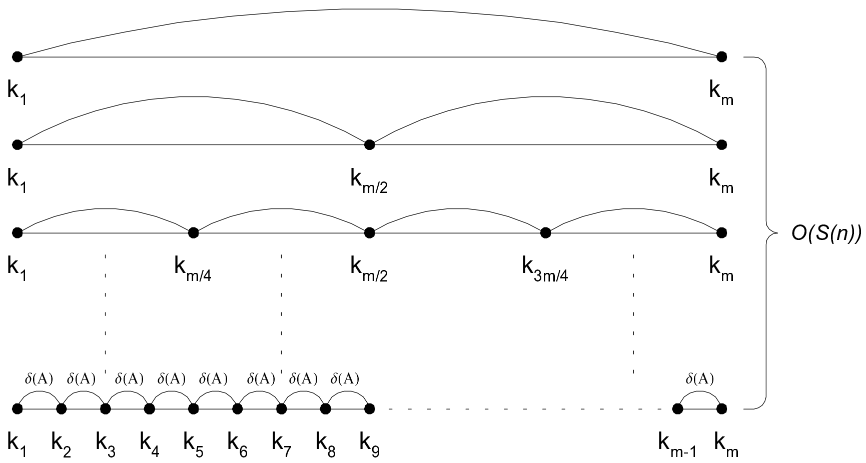
priom k_1 je poiatoná konfigurácia, k_m je akceptaná konfigurácia (ak je náhodou akceptujúci výpoet kratí, dodefinujeme δ-funkciu v akceptanom stave tak, aby sa ni nedialo, ale aby sme mohli "naahova" výpoet).

[3]ako u neraz, $c^{S(n)}$ je poet rôznych konfigurácií A na slove dky n, ke máme k dispozícii priestor $S(n)$

ATS A' bude pracova nasledovne:

1. výpoet zane v k_1
2. uhádne akceptanú konfiguráciu k_m
3. overí, i sa z k_1 do k_m dá dosta na m krokov nasledovne:
   (a) uhádne $k_{\frac{m}{2}}$
   (b) overí, i sa z k_1 dá dosta do $k_{\frac{m}{2}}$ na $\frac{m}{2}$ krokov
   (c) overí, i sa z $k_{\frac{m}{2}}$ dá dosta do k_m na $\frac{m}{2}$ krokov

Body b) a c) tretieho kroku sú vlastne rekurzívne volania kroku 3 s parametrami $k_1, k_{\frac{m}{2}}, \frac{m}{2}$, resp. $k_{\frac{m}{2}}, k_m, \frac{m}{2}$. ATS A' sa do rekurzie bude vnára[4] dovtedy, kým sa nedostane na úrove, kedy bude overova, i sa dá z k_i dosta do k_{i+1} na jeden krok $\forall i = 1, \ldots, m-1$ (obr.5.3). Táto úrove je z hadiska rekurzie elementárna, na nej staí overi, i v δ-funkcii NTS A existoval taký prechod, ktorý umonil prepísanie konfigurácie k_i na k_{i+1} (teda simulujeme jeden krok pôvodného NTS A). Je nutné si uvedomi, e rekurzia sa vykonáva paralelne.



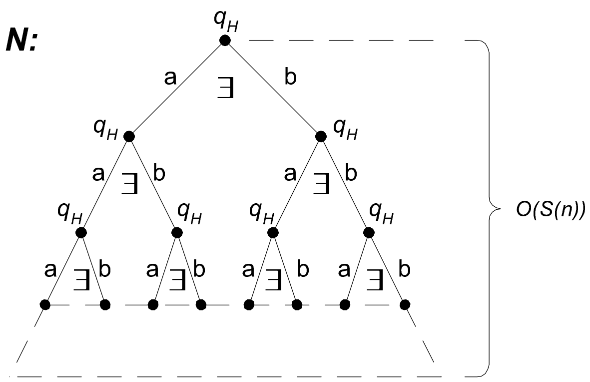
Obr. 5.3: Overovanie náveznosti konfigurácií v ATS A'

Pozrime sa na proces hádania konfigurácií a overovania dosiahnutenosti konfigurácie k_j z konfigurácie k_i na m krokov podrobnejie, z hadiska alternovania a rozdelenia mnoiny stavov ATS A' na existenné a univerzálne. Na zaiatku výpotu je A' v stave q_0 a na vstupe má slovo w, teda $k_1 = q_0 w$, tento stav je existenný. Teraz A' prejde do stavu q_H a háda k_m, pod $q_0 w$ v strome konfigurácií visí "vemi koatý" strom (ozname ho N), je na (obr.5.4), výky $S(n)$, ktorý ako vetky svoje vrcholy, vrátane listov, obsahuje vetky moné konfigurácie NTS A v priestore $S(n)$. Kvôli jednoduchosti si ho môeme popísa nasledovne[5] (na itatea nechávame domyslenie ukonenia generovania):

$$\delta(q_H, 1) = \{(q_H, a, 1), (q_H, b, 1)\}$$

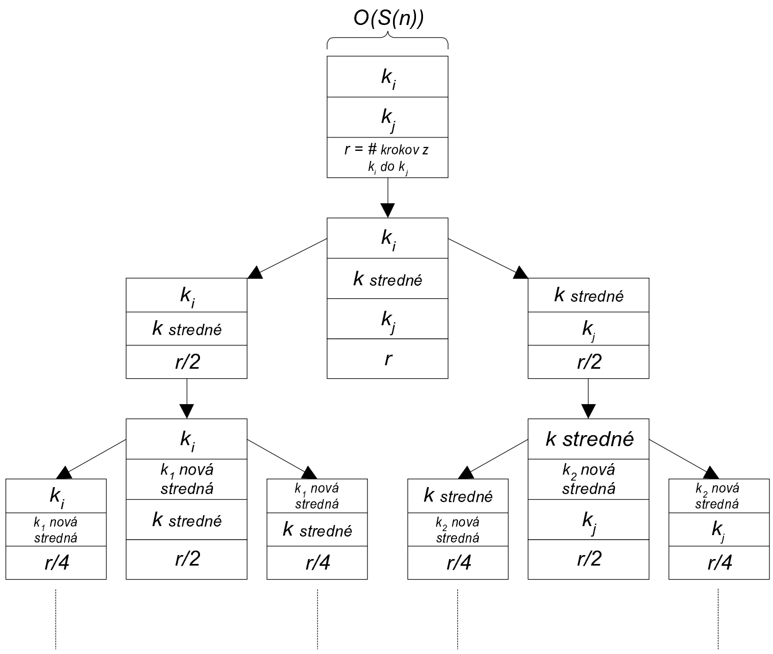
[4]kadé rekurzívne volanie si môme reprezentova z hadiska alternovania tak, e A' vytvorí nový proces, ktorý ako parametre dostane k_i, k_j, r a za úlohu má overi, i sa z k_i dá dosta do k_j na r krokov (obr.5.5)

[5]konfigurácia ako ju popisujeme, neobsahuje stav, ani pozíciu hlavy, itate si tieto veci iste rád domyslí sám



Obr. 5.4: Hádanie konfigurácie dky $S(n)$ ATS A'

Ke má A' uhádnutú akceptanú konfiguráciu k_m, overí, i sa z k_1 do k_m dá dosta na m krokov. To spraví tak, e uhádne $k_{\frac{m}{2}}$ rovnako, ako hádal k_m (teda pod kadým vrcholom stromu N visí nový strom (opä rovnaký ako N, má odliné stavy, vetky sú existenné), v ktorom A' háda $k_{\frac{m}{2}}$). Ke ju uhádne, overuje, i sa z k_1 dá dosta do $k_{\frac{m}{2}}$ na $\frac{m}{2}$ krokov a i sa z $k_{\frac{m}{2}}$ dá dosta do k_m na $\frac{m}{2}$ krokov tak, e v univerzálnom stave spraví FORK dvoch nových procesov[6], ktoré pracujú paralelne a kadý overuje polovicu toho, o mal overi materský proces, at.



Obr. 5.5: Schématické vytváranie nových procesov ATS A'

Z kontrukcie by malo by zrejmé (nebudeme to dokazova), e $L(A) = L(A')$. Pome sa teraz pozrie

[6]to je práve jedno rekurzívne volanie

na asovú zloitos ATS A', ktorý sme práve zkontruovali: máme $O(S(n))$ rekurzívnych volaní a v kadom hádame strednú konfiguráciu, resp. na zaiatku musíme uhádnu akceptanú konfiguráciu k_m. Hádanie ubovonej konfigurácie k_i trvá as $O(S(n))$, o je presne výka stromu N, lebo ju treba zapamäta na páske. Celkový poet krokov (alebo inak výka akceptaných vetiev úplného stromu konfigurácií) A' je $O(S^2(n))$ a teda $L(A') \in ATIME(S^2(n))$ □

Poznámka 5.3.1. *V predchádzajúcej vete sme pouili "paralelnú verziu" kontrukcie, ktorá bola (v sevkvennej podobe) pouitá pri dôkaze vety $NSPACE(S(n)) \subseteq DSPACE(S^2(n))$ (Savitch)*

Pri väine známych kontrukcií, ke máme dané zariadenie (Turingov stroj) a poadujeme k nemu zostroji zariadenie ekvivalentné, ale rýchlejie, urýchlenie ide na úkor pouitého priestoru, resp. analogicky ke poadujeme zmenenie pouitého priestoru, ide to na úkor asu. Ako naznaila kontrukcia z vety 5.3.1, pri alternujúcich strojoch je situácia trochu iná. Pozornému itateovi iste neunikol fakt, e priestorové ohranienie zostrojeného ATS zostalo nezmenené.

V niektorých kontrukciách bude kôli jednoduchosti vhodné uvaova binárne vetvenie úplného stromu konfigurácií ATS, teda fakt, e ubovoný prechod δ-funkcie má najviac dva prvky. Ukáeme si preto nasledovné dve tvrdenia o normálnych tvaroch pre ATS.

Lema 5.3.1. *K ubovonému ATS A pracujúcemu v ase $T(n)$ s ubovonou vekosou vetvenia[7] v úplnom strome konfigurácii existuje ATS A' pracujúci v ase $T(n)$, ktorého úplný strom konfigurácii je binárny a $L(A) = L(A')$*

Dôkaz: Iba neformálne naznaíme spôsob kontrukcie A': nech je vetvenie vo vrchole v úplného stromu konfigurácií A stupa n. Binárne vetvenie vytvoríme tak, e v avej vetve vychádzajúcej z vrchola v bude jeho najavejí nasledovník a v pravej vetci ostatní nasledovníci tak, e priamy nasledovník v bude v' a jeho nasledovníci budú zvynými nasledovníkmi v. Konfiguráciu v tomto vrchole dostaneme tak, e v konfigurácii vo v zmeníme stav na nový, ktorý ete nepatrí do mnoiny stavov A, vetvenie vrchola v' bude stupa $n-1$, teda o 1 menieho ako vetvenie v. Na vrchol v' aplikujeme algoritmus rekurzívne a v rekurzii skoníme a vtedy, ke bude vrchol stupa ≤ 2. Na itatea nechávame premysie, ako algoritmus aplikova na celý strom konfigurácií[8]. Dodajme ete, e ak vrchol v bol existenný, tak aj vetky vrcholy, ktoré pri vytváraní binárneho vetvenia vzniknú, budú existenné, podobne ak v bol univerzálny, tak vetky vzniknuté vrcholy budú univerzálne. Uvedomme si, e hbka akceptaného výpotu v úplnom strome konfigurácií A' bude iba kontantným násobkom hbky úplného stromu konfigurácií A, teda A' pracuje v ase $T(n)$ a akceptuje rovnaký jazyk ako A □

Lema 5.3.2. *K ubovonému ATS A pracujúcemu v priestore $S(n)$ s ubovonou vekosou vetvenia v úplnom strome konfigurácii existuje ATS A' pracujúci v priestore $S(n)$, ktorého úplný strom konfigurácii je binárny a $L(A) = L(A')$*

Dôkaz: V kontrukcii z lemy 5.3.1 sme nikde nemenili priestor, ktorý vyuíval A pri práci, a teda tvrdenie je jej priamym dôsledkom □

Nasledujúce tvrdenie hovorí, ako efektívne z hadiska priestorového ohranienia vieme simulova alternujúce Turingove stroje pracujúce v ase $T(n)$ na deterministických Turingových strojoch.

Veta 5.3.2. *$ATIME(T(n)) \subseteq DSPACE(T(n))$ pre $T(n) \geq \log n, T(n)$ je páskovo kontruovatená*

[7] vekosou vetvenia rozumieme maximálny poet konfigurácii dosiahnutený z ubovonej konfigurácie na jeden krok

[8] pomôcka: δ-funkcia A má iba konene vea prvkov a vetvenie v úplnom strome konfigurácií je iba grafické znázornenie δ-funkcie, resp. výpotu

Dôkaz: V celom dôkaze budeme pod pojmom strom konfigurácii rozumie podstrom úplného stromu konfigurácii, ktorý dostaneme tak, e v úplnom strome konfigurácii “odreeme” vetky vetvy v hbke $T(n)$.

K danému ATS A pracujúcemu v ase $T(n)$ zostrojíme DTS A' pracujúci v priestore $T(n)$. Poda lemy 5.3.1 môeme kvôli jednoduchosti predpoklada, e úplný strom konfigurácii A je binárny. A' musí overi, i sa v strome konfigurácií A nachádza podstrom akceptujúceho výpotu.

A' bude prehadáva strom konfigurácií A algoritmom PREORDER do hbky a priradzova vrcholom hodnoty $0, 1$ nasledovne:

- vrcholu sa môe priradi hodnota iba vtedy, ak sú u priradené hodnoty vetkým jeho nasledovníkom v strome konfigurácii alebo ak je to list[9]
- listom priradzujeme hodnoty poda toho, i sú akceptujúcimi, alebo neakceptujúcimi konfiguráciami, akceptujúcim priradíme hodnotu 1, neakceptujúcim hodnotu 0
- ak je vrchol existenný (a nie je to list) a aspo jeden z jeho nasledovníkov v strome konfigurácii má hodnotu 1, tak sa mu priradí hodnota 1, inak sa mu priradí 0
- ak je vrchol univerzálny (a nie je to list), tak sa mu priradí hodnota 1 práve vtedy, ke obaja jeho nasledovníci v strome konfigurácií majú hodnotu 1, inak sa mu priradí 0

A' akceptuje svoj vstup práve vtedy, ke bude koreu priradená hodnota 1. Keby sme chceli dosiahnu dobrý pomer as/priestor, asi by sme postupovali tak, e by sme si pamätali informáciu o celom “lúi” konfigurácii, teda vetky konfigurácie, ktorými sme od korea prechádzali a k listom, resp. ku vrcholom v hbke $T(n)$, aby sme nestrácali as pri vracaní sa v strome smerom nahor. Dosiahli by sme vak priestorové ohranienie $T^2(n)$ (pretoe by sme si museli pamäta a $T(n)$ konfigurácií, kadú dky $T(n)$), o v naom prípade nie je iadúce, preto budeme musie poui trochu rafinovanejiu kontrukciu, vzhadom na to, e nám nezálei na ase. Nebudeme si pamäta celý “lú” konfigurácii, ale iba momentálnu konfiguráciu a navigáciu k nej v strome konfigurácii. Navigácia bude dky najviac $T(n)$ a bude to postupnos z $\{0, 1\}^*$, kde 0 znamená pohyb v strome vavo smerom dole a 1 znamená pohyb vpravo smerom dole. Ke sa budeme v strome musie vraca smerom hore, tak si predchodcu konfigurácie, v ktorej práve sme, poda tejto navigácie vypoítame, na páske A' (obr.5.6) si budeme potrebova pamäta navigáciu, momentálnu konfiguráciu a pár pomocných konfigurácii, teda máme priestorové ohranienie $T(n)$ pre A' $\square$

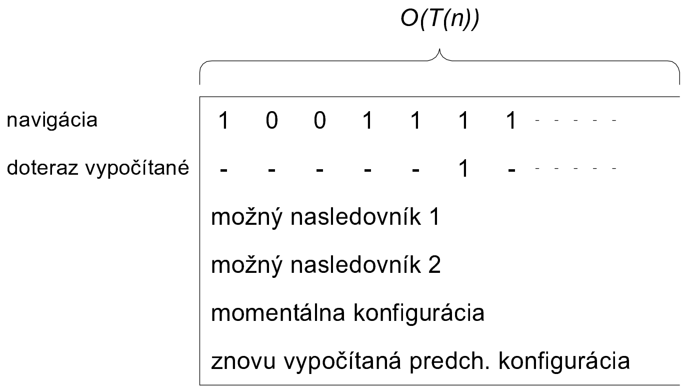
Definícia 5.3.2. *Polynomiálne triedy zloitosti definujeme nasledovne:*

- $NSPACE(Poly) \stackrel{def}{=} \bigcup_{k \geq 1} NSPACE(n^k)$
- $DSPACE(Poly) \stackrel{def}{=} \bigcup_{k \geq 1} DSPACE(n^k)$
- $ATIME(Poly) \stackrel{def}{=} \bigcup_{k \geq 1} ATIME(n^k)$

Dôsledok 5.3.1. $NSPACE(Poly) = DSPACE(Poly) = ATIME(Poly)$

Dôkaz: Tvrdenie vyplíva zo Savitchovej vety, vety 5.3.1 a vety 5.3.2 $\square$

[9]listom sa v tomto prípade myslí aj taký vrchol, ktorý síce nie je listovým v úplnom strome konfigurácii A, ale stáva sa listovým vrcholom v strome konfigurácii A



Obr. 5.6: Páska DTS A' (druhá stopa by sa dala pamäta aj v stave)

Veta 5.3.3. $ASPACE(S(n)) \subseteq \bigcup_{c>0} DTIME(c^{S(n)})$ *pre* $S(n) \geq \log n$

Dôkaz: Ukáeme si, e ke máme k ATS A pracujúcemu v priestore $S(n)$ zostroji DTS A' pracujúci v exponenciálnom ase $c^{S(n)}$ pre nejaké nezáporné c, tak nám na simuláciu postaí aj vemi hrubá sila, akou je bezosporu vygenerovanie vetkých moných konfigurácii A na páske a práca na týchto konfiguráciách. Opä budeme bez újmy na veobecnosti predpoklada, e úplný strom konfigurácií A je binárny.

A' bude pracova nasledovne:

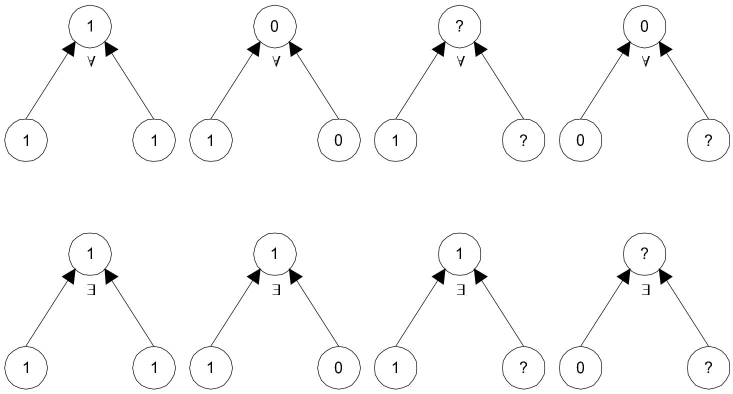
- na pásku zapíe vetkých $|\Gamma_A|^{S(n)}.S(n).|K_A| < r^{S(n)}$ konfigurácii A v lexikografickom usporiadaní, za kadou konfiguráciou si nechá priestor (jedno políko), ktorý alej vyuije[10], na to potrebuje as $S(n).r^{S(n)}$
- do vyznaeného priestoru bude kadej konfigurácii priraova hodnoty $0, 1, ?$ prechodom stromu konfigurácii A tak, e v kadom prechode priradí hodnoty $0, 1$ rodiom tých detí, ktoré u sú vyhodnotené a hodnotu ? rodiom nevyhodnotených detí (poda (obr.5.7)), akceptuje, ak bude poiatonej konfigurácii priradená hodnota 1, asová zloitos bude nasledovná:
  1. $r^{S(n)}$-krát prejde pásku a "spracuje" kadú konfiguráciu
  2. "spracova" konfiguráciu znamená zisti hodnoty jej nasledovníkov a ak sa dá, tak jej priradi hodnotu 0,1, toto je asovo ohraniené $\approx r^{S(n)}.S(n)$

Ke si celú prácu A' zosumarizujeme, dostávame asovú zloitos pribline $c^{S(n)}$ pre nejaké c (staí zvoli napr. $c = r^{10}$) a sme hotoví $\square$

Veta 5.3.4. $DTIME(T(n)) \subseteq ASPACE(\log T(n))$ *pre* $T(n) \geq n$

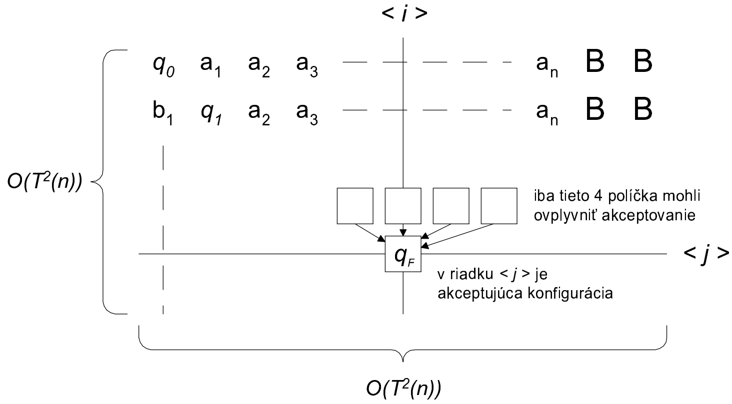
Dôkaz: Nech A_1 je k-páskový DTS pracujúci v ase $T(n)$. K nemu existuje jednopáskový DTS $A = (K_A, \Sigma, \Gamma, \delta, q_0, F_A)$ pracujúci v ase $T^2(n)$ taký, e $L(A_1) = L(A)$. K nemu zkontruujeme ATS A' pracujúci v priestore $\log T(n)$ taký, e $L(A) = L(A')$. Najskôr si zapíme konfigurácie A pod seba poda (obr.5.8) tak, e v prvom riadku bude poiatoná konfigurácia $q_0 w$ a ke oznaíme

[10]priestor si na páske vyznaíme napr. #?#, priom # slúi ako oddeova konfigurácii, nie je prvkom páskovej abecedy Γ, rovnako ani ?, ktorý hovorí, e o tomto políku zatia ni nevieme



Obr. 5.7: Priraovanie hodnôt rodiom v strome konfigurácii ATS A

$< qw_k >_l$ konfiguráciu v k-tom riadku pre nejaký stav $q \in K_A$ a pozíciu stavu (hlavy) v tejto konfigurácii l, tak platí $< qw_k >_l \vdash < pw_{k+1} >_m$, priom $m \in \{l-1, l, l+1\}$. Ak $w \in L(A)$, tak máme v tabuke zapísanú postupnos konfigurácií akceptujúceho výpotu (vzhadom na to, e A je deterministický), teda existuje i také, e pre $< q_F w_i >_j$ je $q_F \in F_A$.



Obr. 5.8: Tabuka konfigurácií jednopáskového DTS A

A' bude pracova nasledovne:

- uhádne pozíciu <riadok, stpec> akceptaného stavu q_F v akceptujúcej konfigurácii tvaru $< q_F w_i >_j$, teda háda $< i, j >$, toto zaberie $\log T^2(n)$ políok (môeme zvoli napr. binárnu reprezentáciu ísel i, j), to je $2. \log T(n) \approx \log T(n)$ políok
- uhádne, ktorý akceptujúci stav sa na pozícii $< i, j >$ nachádza
- overí, i dobre hádal pozíciu $< i, j >$ a akceptaný stav nasledovne:
    1. uhádne (zmysluplne[11]) obsahy políok $< i-1, j-1 >, < i-1, j >, < i-1, j+1 >$, $< i-1, j+2 >$, pretoe týmito obsahmi je obsah $< i, j >$ jednoznane urený

[11] poda δ-funkcie A

2. v univerzálnom vetvení overí, i hádal správne (kadá vetva overí jedno políko)

Takto sa bude A' vraca v tabuke a do prvého riadku (zrejme si bude dekrementova i a pracova s j poda pohybu hlavy A), ke bude na pozícii $< 1, 1 >$ a bude v poiatonom stave, tak akceptuje

Pri prechode tabukou konfigurácií smerom zdola nahor sa nám nemôe sta, e by sme nali viac ako jednu cestu vedúcu k akceptovaniu, pretoe A je deterministický a teda akceptaný výpoet pre dané slovo je vdy jednoznaný, ak sme v niektorej vetve nieo zle uhádli, výpoet sa urite zablokuje.

Ke sa máme bavi o priestorovej zloitosti A', jediná vec, ktorá ju ovplyvuje, je dka i, resp. j, pretoe okrem týchto dvoch ísel si pamätáme iba kontantne vea informácii (dokonca vemi málo). Ale o dke sme si u povedali, e je $\log T(n)$, take sme hotoví □

Dôsledok 5.3.2. $ASPACE(S(n)) = \bigcup_{c>0} DTIME(c^{S(n)})$ pre $S(n) \geq n$

5.4 Alternujúce konené automaty ($AFSA$)

itate sa iste stretol s viacerými modifikáciami pôvodného modelu deterministických konených automatov. Nedeterminizmus im nepomohol, rovnako ako nepomohlo napr. pridanie monosti obojsmerného pohybu po vstupnej páske. Mohlo by sa zda, e taká sila, akou je alternovanie, by mohlo pomôc týmto zariadeniam vyjs z triedy $\mathcal{R}$ a akceptova aj nejaké nie regulárne jazyky. Ako vak hovorí nasledujúca veta, opak je pravdou. Faktom toti zostáva, e ke procesom v jednotlivých vetvách neumoníme komunikáciu, resp. synchronizáciu, tak sa sila zariadenia nezväí.

Veta 5.4.1. $\mathcal{L}_{AFSA} = \mathcal{R}$

Dôkaz: Inklúzia zprava doava je triviálna a preto ju nebudeme ukazova. Na dôkaz opanej inklúzie zostrojíme k AFSA A ekvivalentný nedeterministický konený automat (NKA) A' taký, e $L(A) = L(A')$. Najskôr podobnou kontrukciou ako pre NKA zostrojíme k A ekvivalentný ATS A_1 (teda platí $L(A) = L(A_1)$) taký, e v A_1 neexistujú ε-prechody (kontrukcii sa nebudeme bliie venova, itate ju nájde napr. v [1]). Zavedieme pár oznaení, ktoré sa nám neskôr budú hodi:

- $all(q, a) = \{$mnoina vetkých stavov dosiahnutených v A_1 na jeden krok zo stavu q pri ítaní vstupného symbolu $a\}$
- $ex_i(q, a) = p$, priom $p \in \delta'(q, a)$ a ke máme prvky $\delta'(q, a)$ oíslované, resp. usporiadané, tak $\delta'(q, a) = p$ je i-ty v poradí

Teraz k bez-ε ATS $A_1 = (K, \Sigma, \delta, q_0, F)$ zostrojíme NKA $A' = (K', \Sigma', \delta', q_0', F')$ nasledovne:

- Vezmime si úplný strom konfigurácii A_1 a pozrime sa na po úrovniach. Kee v A_1 nie sú ε-prechody, tak na n-tej úrovni je zo vstupu preítaných práve n-symbolov
- K' bude nová mnoina stavov, priom jej prvky budú podmnoiny mnoiny stavov K, teda poet stavov $|K'| = 2^{|K|}$, formálne

$$[q_{i_1}, \ldots, q_{i_k}] \in K' \overset{def}{\iff} q_{i_1}, \ldots, q_{i_k} \in K$$

- $\Sigma' = \Sigma$, teda vstupná abeceda sa nemení
- δ' definujeme nasledovne:

– ak q je univerzálny a $\delta(q,a) = \{q_{i_1}, \dots, q_{i_k}\}$, tak $\delta'([q],a) = \{[q_{i_1}, \dots, q_{i_k}]\}$
– ak q je existený a $\delta(q,a) = \{q_{i_1}, \dots, q_{i_k}\}$, tak $\delta'([q],a) = \{[q_{i_1}], \dots, [q_{i_k}]\}$
– rekurzívne definujeme $\delta'([q_{i_1}, \dots, q_{i_j}, q_{i_{j+1}}, \dots, q_{i_k}], a)$ ako $\{[all(q_{i_1},a), \dots, all(q_{i_j},a), ex_1(q_{i_{j+1}},a)], \dots, [all(q_{i_1},a), \dots, all(q_{i_j},a), ex_n(q_{i_k})]\}$ priom $q_{i_1}, \dots, q_{i_j}$ sú univerzálne stavy a $q_{i_{j+1}}, \dots, q_{i_k}$ sú existenné stavy a platí $|\delta'(q_{i_k})| = n$

V stavoch A' udrujeme informácie o vetkých vetvách úplného stromu konfigurácií A_1, je dobré si uvedomi, e ak u máme mnoinu stavov $[p,q]$, a napr. $\delta'([p],a) = [r]$ a súasne $\delta'([q],a) = [r]$, tak si túto informáciu nemusíme pamäta druhý krát, ako stav si budeme pamäta iba $[r]$

- prirodzene definujeme $q'_0 = [q_0]$
- $F' = \{[q_{i_1}, \dots, q_{i_k}] \mid q_{i_1}, \dots q_{i_k} \in F\}$

Nemalo by by a také aké pochopi, e $L(A_1) = L(A')$, a teda platí aj $L(A) = L(A')$ $\square$

V niektorých kontrukciách je potrebné a zmysluplné poadova, aby zostrojený konený automat k alternujúcemu konenému automatu bol deterministický. Potom jeho stavy budú opä mnoiny stavov AFSA, ke tieto budú univerzálne, no ke prídu do hry existenné stavy, tak sa stavy rozpadnú na mnoiny, akceptané stavy potom budú také, ktorých aspo jedna zloka je akceptaná v u definovanm nedeterministickom zmysle. Stavov môe by a $2^{2^{|K|}}$, o nie je zanedbatené íslo.

Príklad 5.4.1. Ku AFSA $A = (K, \Sigma, \delta, q_0, F)$, ktorého úplný strom konfigurácií na $w = aba$ je na (obr.5.9), priom $\{q_0, \dots, q_4, p_1, p_2, r_1, \dots, r_4\} \subseteq K$, $\Sigma = \{a,b\}$, $\{r_1, r_2, r_3\} \subseteq F$ a δ je zrejmá z obrázku[12], zostrojíme NKA $A' = (K', \Sigma', \delta', q'_0, F')$ (jeho as) poda kontrukcie z vety 5.4 nasledovne:

1. $\{[q_0], [p_1,p_2], [q_1,q_3], [q_1,q_4], [q_2,q_3], [q_2,q_4], [r_1,r_2,r_3], [r_2,r_3,r_4]\} \subset K'$
2. $\Sigma' = \Sigma$
3. $q'_0 = [q_0]$
4. $[r_1, r_2, r_3] \subseteq F'$
5. δ' definujeme schématicky nasledovne:

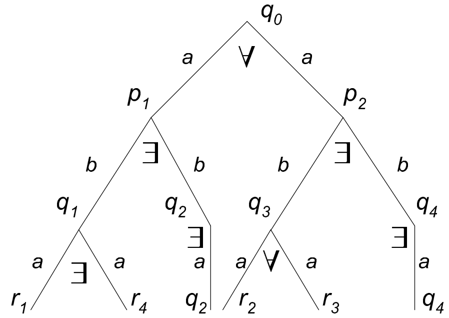
$$[q_0] \overset{a}{\rightsquigarrow} [p_1,p_2] \overset{b}{\rightsquigarrow} \begin{cases} [q_1,q_3] \overset{a}{\rightsquigarrow} \begin{cases} [r_1,r_2,r_3] \\ [r_2,r_3,r_4] \end{cases} \\ [q_1,q_4] \overset{a}{\rightsquigarrow} \begin{cases} [r_1,q_4] \\ [r_4,q_4] \end{cases} \\ [q_2,q_3] \overset{a}{\rightsquigarrow} [q_2,r_2,r_3] \\ [q_2,q_4] \overset{a}{\rightsquigarrow} [q_2,q_4] \end{cases}$$

Potom zrejme výpoet

$$[q_0]aba \vdash_{A'} [p_1,p_2]ba \vdash_{A'} [q_1,q_3]a \vdash_{A'} [r_1,r_2,r_3]$$

je akceptujúcim výpotom NTS A' na w.

[12] definovali sme iba as AFSA A potrebnú pre výpoet na w



Obr. 5.9: Úplný strom konfigurácií AFSA A na slove $w = aba$

5.5 Alternujúce zásobníkové automaty ($APDA$)

Ako sme ukázali, koneným automatom alternovanie v generatívnej sile vôbec nepomohlo. Lepia je situácia v oblasti bezkontextových jazykov, ktoré rozpoznávajú zásobníkové automaty (PDA). Tu alternovanie zvýi monos rozpoznávania jazykov a na úrove, o ktorej hovorí nasledujúce tvrdenie

Veta 5.5.1. $\bigcup_c DTIME(c^n) \subseteq \mathcal{L}_{APDA}$

Dôkaz: Tvrdenie nebudeme dokazova priamo, vyuijeme tvrdenie vety 5.3.4, poda ktorého platí $\bigcup_c DTIME(c^n) \subseteq ASPACE(n)$. My ukáeme, e platí $ASPACE(n) \subseteq \mathcal{L}_{APDA}$, potom bude z tranzitívnosti relácie $\subseteq$ plati aj $\bigcup_c DTIME(c^n) \subseteq \mathcal{L}_{APDA}$.

Na dôkaz $ASPACE(n) \subseteq \mathcal{L}_{APDA}$ potrebujeme k ATS A, ktorý pre vstup dky n pouíva n políok na pracovnej páske, zostroji APDA A' taký, e $L(A) = L(A')$. Bez újmy na veobecnosti budeme predpoklada, e úplný strom konfigurácií A je binárny[13].

A' bude pracova nasledovne[14]:

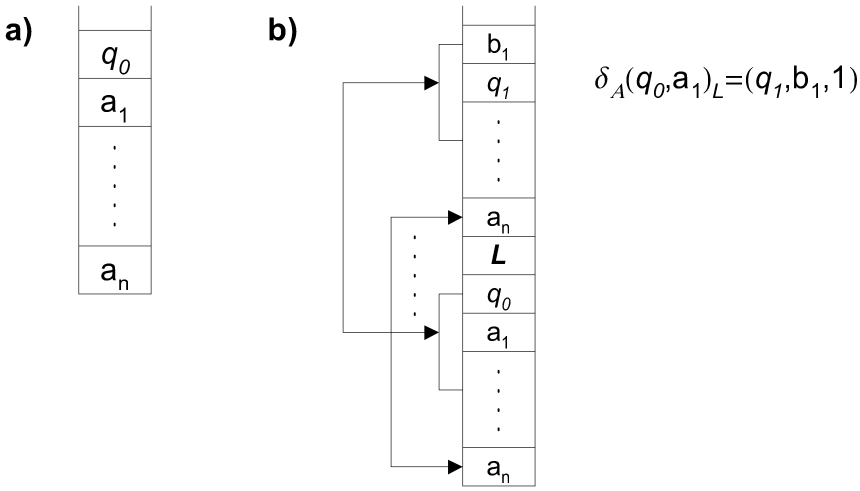
- najskôr v existennom vetvení uhádne poiatonú konfiguráciu (bude háda $n+1$ symbolov, háda aj stav) q_0w, priom $w = a_1 \dots a_n$, v obrátenom poradí, teda zásobník bude ma po $n+1$ krokoch tvar ako na (obr.5.10a)
- v univerzálnom vetvení v jednej vetve overí, i konfiguráciu hádal správne, v druhej vetve háda nasledovníkov konfigurácie q_0w vo vetvení, ktorého typ zodpovedá vetveniu v úplnom strome konfigurácií A, kadej konfigurácii priradí aj vetvu, v ktorej sa nachádzala v úplnom strome konfigurácií A (teda L,R), konfigurácie alej overí a háda aie, a kým neakceptuje, resp. neodmietne (REJECT) vstup
- do overenia, i konfigurácie na seba nadväzujú, spadá:
  - overi, i sme hádali správnu vetvu (prvky δ_A si usporiadame L,R)

[13]itate si iste vie predstavi podobnú kontrukciu ako v leme 5.3.2 pre APDA

[14]idea je nasledovná: v zásobníku bude simulova výpoet ATS A tak, e do neho bude postupne pridáva konfigurácie A a overova ich náväznos vzhadom na $\vdash$, ke pridá do zásobníka akceptanú konfiguráciu, ktorá bude nadväzova na predchádzajúce, akceptuje

– A' musí po jednotlivých symboloch prejs konfigurácie a testova ich na rovnos (obr.5.10b), resp. monos prechodu v δ_A, univerzálne sa rozvetví: v jednej vetve overí, i sa 1. symbol zhoduje s $(n+k)$-tym symbolom[15], resp.i existuje v δ_A prechod taký, aby sa symboly na seba mohli prepísa, tak, e zo zásobníka zmae $n+k$ symbolov (je dôleité uvedomi si, e do tejto chvíle sme v tejto vetve vstup neítali, tu ho pouívame na poítanie n, zrejme bude dobre vdy si v stave pamäta 4 symboly z vrchu pôvodného obsahu zásobníka pred vymazávaním), v aej overí, i 2. a $(n+1)$-vý symbol z prvej konfigurácie sedí poda δ_A s 2. a $(n+1)$-vým symbolom druhej konfigurácie podobne ako pre 1. symbol

itateovi odporúame podrobnejie rozpracova overovanie konfigurácii, najmä overenie, i sa konfigurácia nachádzala v avej, alebo pravej vetve vzhadom na svojho predka, v úplnom strome konfigurácií ATS A. Na pochopenie, e $L(A) = L(A')$ by vak úrove detailu v naej kontrukcii mala by dostaujúca □



Obr. 5.10: Zásobník APDA A' a) po uhádnutí q_0w b) pri overovaní náväznosti konfigurácií

Poznámka 5.5.1. *Platí aj obrátená inklúzia, jej dôkaz vak presahuje rozsah tohto textu, navye na ukáku, e alternovanie zásobníkovým automatom pomáha v generatívnej sile, staí veta 5.5.1*

5.6 Synchronizované alternujúce stroje

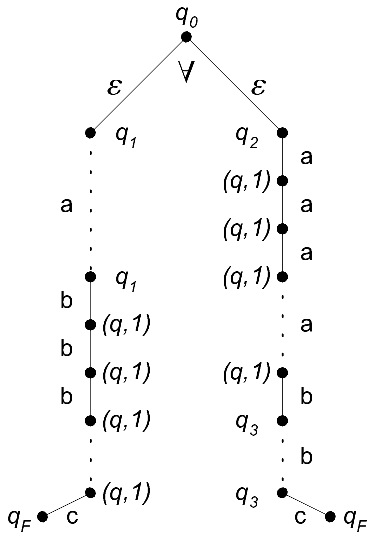
Ukáeme si jednu modifikáciu pôvodného modelu alternujúcich strojov, ke umoníme jednoduchú komunikáciu medzi paralelnými procesmi. Zavedieme nové delenie stavov alternujúceho stroja, nezávisle od delenia na existenné a univerzálne stavy, na:

- obyajné
- synchronizané - dvojica $(q, S) \to$ (stav,symbol)

Ke proces prejde do synchronizaného stavu, aká, kým vetky ostatné prejdu do synchronizaného stavu. Ke majú vetky rovnaký symbol v druhej komponente synchronizaného stavu, pokraujú,inak sa zariadenie zablokuje

[15] k je kontanta, jej urenie prenechávame na itatea

Príklad 5.6.1. Ukáeme si, e ke umoníme synchronizáciu koneným automatom, tak sa nám ich podarí posunú z triedy regulárnych jazykov. Na (obr.5.11) je znázornený synchronizovaný konený automat, ktorý akceptuje bezkontextový jazyk $L = \{a^n b^n c \mid n \geq 1\}$



Obr. 5.11: Synchronizovaný konený automat pre jazyk $L = \{a^n b^n c \mid n \geq 1\}$

Poznámka 5.6.1. *Dá sa ukáza tvrdenie, ktoré hovorí o tom, e trieda jazykov rozpoznávaných synchronizovanými konenými automatmi je presne* $\mathcal{L}_{CS}$

Kapitola 6

Booleovské obvody (BO)

6.1 Definície a oznaenia

Definícia 6.1.1. *Booleovský obvod (BO) je konený acyklický orientovaný graf, v ktorom kadému vrcholu v priradíme typ $\tau(v) \in \{B_{IN}\} \cup \{B_0\} \cup \{B_1\} \cup \{B_2\}$ a hodnotu $\mathcal{V}(v) \in \{0,1\}$. Vrchol v, pre ktorý typ $\tau(v) \in \{B_{IN}\}$, má vstupný stupe 0 a nazývame ho vstupný vrchol. Vstupom pre booleovský obvod je n-tica rôznych vstupných vrcholov oznaených $\langle x_1, \ldots, x_n \rangle$. Vrchol v, pre ktorý typ $\tau(v) \in \{B_i\}$, má vstupný stupe i a nazývame ho hradlo. Medzi hradlá typu B_0 patria kontanty "0" a "1", do B_1 patria hradlá "I" a "$\neg$" reprezentujúce booleovské funkcie identita a negácia a do B_2 patria hradlá "$\wedge$" a "$\vee$" reprezentujúce booleovské funkcie AND a OR. Vrcholy s výstupným stupom 0 nazývame výstupné. Výstupom pre booleovský obvod je m-tica rôznych výstupných vrcholov oznaených $\langle y_1, \ldots, y_m \rangle$.*

Definícia 6.1.2. *Booleovský obvod so vstupom $\langle x_1, \ldots, x_n \rangle$ a výstupom $\langle y_1, \ldots, y_m \rangle$ poíta funkciu $f : \{0,1\}^n \to \{0,1\}^m$ nasledovne. Kadý vstupný vrchol x_i má danú hodnotu $\mathcal{V}(x_i) \in \{0,1\}$. Kadé hradlo h jednoznane vyhodnotí hodnotu $\mathcal{V}(h)$ aplikovaním elementárnej booleovskej funkcie $\tau(h)$ na hodnoty vstupných vrcholov. Výsledná hodnota funkcie f je daná m-ticou hodnôt výstupných vrcholov $\langle \mathcal{V}(y_1), \ldots, \mathcal{V}(y_m) \rangle$.*

Vimnime si, e v definícii sme ohraniili poet vstupov vrchola, ale nie poet výstupov. Takisto sme nezakázali, aby vstupný vrchol bol zárove výstupným.

Ak chceme pomocou modelu BO definova jazyky, je rozumné sa obmedzi na BO s jedným výstupným vrcholom, priom slovo na vstupe z $\{0,1\}^*$ akceptujeme práve vtedy, ke hodnota výstupného vrchola bude 1. Toto so sebou prináa jednu nepríjemnos, pretoe BO má konený poet vstupných vrcholov, a teda je schopný akceptova len konené jazyky. Preto jazyky budeme definova pomocou triedy booleovských obvodov.

Definícia 6.1.3. *Nech $\{C_n\}_{n=0}^{\infty}$ je postupnos booleovských obvodov, kde BO C_n s n vstupnými a $m(n)$ výstupnými vrcholmi poíta funkciu $f_n : \{0,1\}^n \to \{0,1\}^{m(n)}$. Túto postupnos nazývame trieda booleovských obvodov $\{C_n\}$ poítajúca funkciu $f : \{0,1\}^* \to \{0,1\}^*$ definovanú takto: $f(w) \equiv f_n(w)$ práve vtedy, ke $|w| = n$.*

Definícia 6.1.4. *Nech $\{C_n\}$ je trieda (postupnos) booleovských obvodov poítajúca funkciu $f : \{0,1\}^* \to \{0,1\}$, teda kadý BO má jeden výstupný vrchol. Jazyk akceptovaný triedou $\{C_n\}$ definujeme takto: $L(\{C_n\}) = \{w \in \{0,1\} \mid f(w) = 1\}$.*

6.2 Miery zloitosti

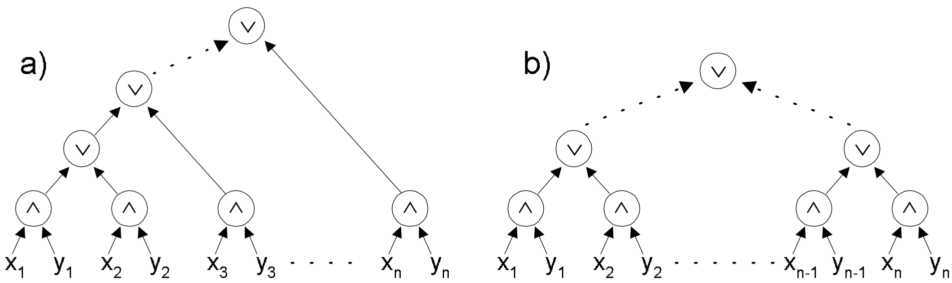
Pre booleovské obvody definujeme nasledujúce miery zloitosti:

- $DEPTH(C_n)$ = dka najdlhej cesty v obvode C_n
- $SIZE(C_n)$ = poet hradiel obvodu C_n

Ak predpokladáme, e as, ktorý potrebuje hradlo na vyhodnotenie výstupu je jedna asová jednotka, a as potrebný na prenos informácie medzi hradlami neuvaujeme, potom miera $DEPTH$ je ekvivalentná asovej náronosti výpotu na BO.

Príklad 6.2.1. Chceme vypoíta skalárny súin dvoch n-bitových vektorov $(x_1, \dots, x_n)$ a $(y_1, \dots, y_n)$. Vieme, e ich skalárny súin vypoítame takto: $(x_1 \wedge y_1) \vee \dots \vee (x_n \wedge y_n)$. Prísluný BO realizujúci tento výpoet je znázornený na obrázku 6.1a, z ktorého ahko vidie, e $SIZE(C_n) = 2n - 1 = O(n)$ a $DEPTH(C_n) = n = O(n)$.

Príklad 6.2.2. Opä chceme vypoíta skalárny súin dvoch n-bitových vektorov, tentoraz vak pouijeme iný BO C_n (obr. 6.1b). Pri zachovaní rovnakého potu hradiel sme zníili hbku obvodu na $DEPTH(C_n) = O(\log n)$.



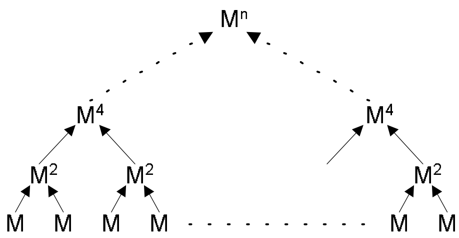
Obr. 6.1: Výpoet skalárneho súinu na BO

Príklad 6.2.3. Chceme vynásobi dve booleovské matice A a B rozmeru $n \times n$. Výsledok je matica rovnakého rozmeru C, priom jej prvky vypoítame nasledovne: $c_{i,j} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$. To je n^2 nezávislých skalárnych súinov. Ak budeme kadý takýto súin poíta poda príkadu 6.2.2 dostaneme prísluný obvod C_n, pre ktorý platí, e $SIZE(C_n) = O(n^3)$ a $DEPTH(C_n) = O(\log)$.

Príklad 6.2.4. Teraz sa pokúsime vyrobi reflexívny a tranzitívny uzáver booleovskej matice M rozmeru $n \times n$. Výsledkom je matica M^*, ktorú dostaneme nasledovným výpotom: $M^* = I \vee M \vee M^2 \vee \dots M^n$, kde $M^i = \bigwedge_{k=1}^{i} M$, o znamená, e $M^* = (I \vee M)^n$. Ak na výpoet súinu matíc pouijeme obvod z príkladu 6.2.3 a truktúru úplného binárneho stromu (obr. 6.2) dostaneme obvod C_n poítajúci M^*, pre ktorý platí $SIZE(C_n) = O(n^4)$ a $DEPTH(C_n) = O(\log^2 n)$.

6.3 BC-uniformné booleovské obvody

Postupnos booleovských obvodov, ako neskôr uvidíme, dokáe akceptova triedu jazykov $\mathcal{L}_{RE}$, ale tak ako sme ju zatial zadefinovali dokáe akceptova ete viac, pretoe postupnos BO môeme uri



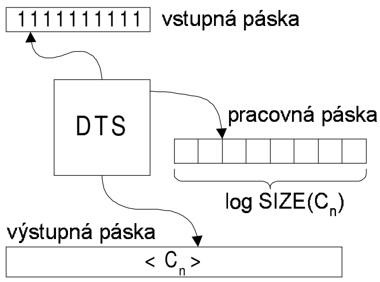
Obr. 6.2: Výpoet reflexívneho a tranzitívneho uzáveru matice

tak, e pre rôzne dky vstupu bude pouíva úplne iný mechanizmus spracovania vstupného slova, o v iných modeloch v zásade nie je moné. Preto doterají model BO trochu zoslabíme zavedením akejsi "pravidelnosti" (uniformity) do postupnosti BO.

Definícia 6.3.1. *Postupnos booleovských obvodov je uniformná, ak existuje nejaký deterministický stroj (*DTS*,* ATS*), ktorý na vstupe* 1^n *vygeneruje BO* C_n *resp. jeho kód.*

Definícia 6.3.2. *(tandardný kód BO)*
Kód booleovského obvodu C_n *(ozn.* $\langle C_n \rangle$*) je postupnos tvoríc* $\langle g, t, a, b \rangle$*, kde* $g \in \{0,1\}^+$ *je jednoznané íslo vrchola,* $t \in \{0, 1, I, \neg, \wedge, \vee, x\}$ *je typ vrchola g (x oznauje vstup),*
$a, b \in \{0,1\}^+$ *sú ísla avého a pravého vstupu vrchola g. Vstupným vrcholom* $x_1, \ldots, x_n$ *tandardne priradíme ísla* $1, \ldots, n$ *a výstupnému vrcholu (ak je len jeden) priradíme íslo 0.*

Definícia 6.3.3. *Postupnos booleovských obvodov* $\{C_n\}$ *je* BC*-uniformná (*BC*-uniformne kontruovatená[1]), ak existuje* DTS*, ktorý pre kadé n na vstupe* 1^n *vygeneruje kód booleovského obvodu* $\langle C_n \rangle$ *v priestore[2]* $\log(SIZE(C_n))$*.*



Obr. 6.3: DTS generujúci kód booleovského obvodu

ísla vrcholov v kóde obvodu C_n kódujeme binárne, take pre vekos kódu dostávame $|\langle C_n \rangle| = O(SIZE(C_n). \log SIZE(C_n))$, o nám uruje asovú zloitos DTS z definície BC-uniformity.
alím predpokladom na tento DTS je topologické usporiadanie výstupu t.j. kód vrchola C_n sa na výstupe neobjaví skôr ako kódy jeho vstupov.

[1] BC - Borodin, Cook
[2] vimnime si, e pracovný priestor neurujeme poda vekosti vstupu, ale na základe vekosti vygenerovaného výstupu

Poznámka 6.3.1. *BC-uniformita zabezpeuje len to, aby sme s jazykmi nevyli z triedy $\mathcal{L}_{RE}$, teda v návrhoch postupností BO nás príli neobmedzuje. Preto aj kadá "rozumne" navrhnutá postupnos BO je BC-uniformná.*

Oznaenie: $\mathcal{U}_{BC}DEPTHSIZE(D(n), S(n))$ - trieda jazykov. Pre kadý jazyk z tejto triedy existuje BC-uniformná postupnos booleovských obvodov $\{C_n\}$ akceptujúca daný jazyk priom $DEPTH(C_n) = O(D(n))$ a $SIZE(C_n) = O(S(n))$.

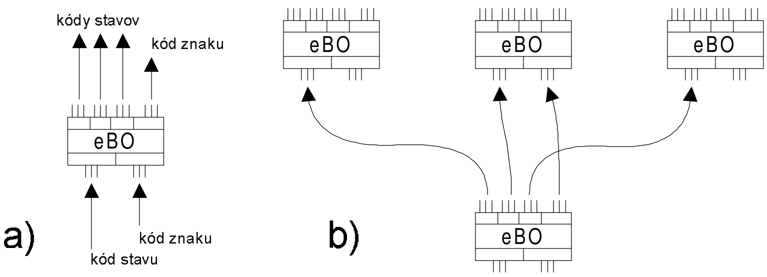
6.4 Porovnanie BO a TS

Veta 6.4.1. *Ak L je jazyk akceptovaný jednopáskovým DTS v ase $T(n)$, potom existuje BC-uniformná postupnos BO $\{C_n\}$ taká, e $L(\{C_n\}) = L$ a $SIZE(C_n) = O(T^2(n))$.*

Dôkaz: Uvaujme DTS A a jazyk L zo znenia vety. Zoberme si nejaké slovo $w = a_1 \dots a_n \in L$. Ukáeme si ako bude vyzera BO C_n akceptujúci slová dky n z jazyka L.

Najskôr binárne zakódujeme vetky symboly vstupnej a pracovnej abecedy a vetky stavy DTS A t.j. vetkým jednoznane priradíme nenulový binárny vektor. Obvod C_n bude ma $T(n)$ úrovní, priom na i-tej úrovni bude "udriava" informáciu o i-tej konfigurácii A pri výpote na slove w nasledovným spôsobom. Kadá úrove bude pozostáva z elementárnych obvodov (eBO) reprezentujúcich jedno políko pracovnej pásky A, priom jeden takýto obvod dostane na vstup kód znaku, ktorý je na danom políku, a kód stavu, v ktorom je A, ak sa hlava A nachádza na tomto políku (obr. 6.4a).

Z mechanizmu výpotu Turingovho stroja vieme, e zmeny konfigurácií sú len lokálneho charakteru t.j. v jednom kroku výpotu sa môe zmeni len jednopísmenkové okolie pozície hlavy. Podobne aj v naom obvode C_n jeden elementárny obvod môe ovplyvni len svojho nasledovníka a jeho susedov, preto sú jednotlivé elementárne obvody pospájané ako na obrázku 6.4b.

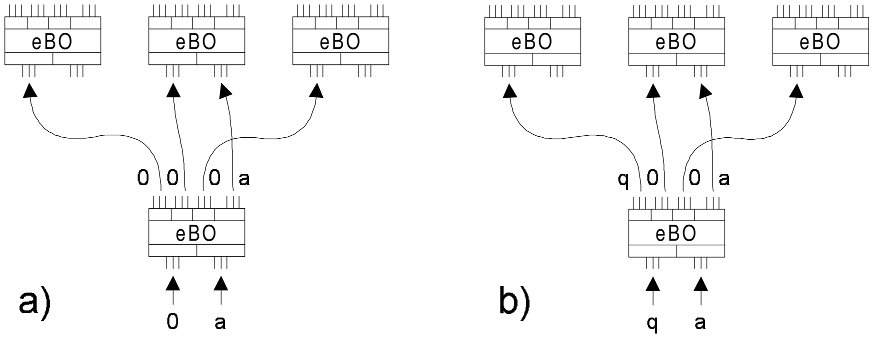


Obr. 6.4: Elementárny booleovský obvod (eBO)

Kadý elementárny obvod bude pracova nasledovne:

1. ak na vstup dostane kód nejakého znaku a nulový vektor ako kód stavu, znamená to, e hlava A nie je na políku, ktoré je reprezentované týmto obvodom a na výstup pole kód prijatého znaku a nulový vektor ako kód stavu (obr. 6.5a)

2. ak na vstup dostane kód znaku a a kód stavu q, tak poda definície δ-funkcie A pole na výstup kód nového znaku b, a poda pohybu hlavy v δ-funkcii pole príslunému obvodu v nasledujúcej úrovni kód nového stavu, ím ho informuje o tom, e v nasledujúcom kroku bude hlava nad jeho políkom a ostatným dvom pole ako kód stavu nulový vektor (obr. 6.5b pre δ-funkciu DTS, ktorá pole hlavu vavo)



Obr. 6.5: Komunikácia medzi úrovami *eBO*

Take celý obvod C_n bude ma $T^2(n)$ elementárnych obvodov ($T(n)$ úrovní pre kadú konfiguráciu a v kadej úrovni $T(n)$ obvodov pre kadé políko[3]). Vstupom pre C_n bude poiatoná konfigurácia A, teda prvý obvod v prvej úrovni dostane na vstup kód poiatoného stavu a kód prvého písmenka slova $w = a_1 \dots a_n$, alích $n-1$ obvodov dostane na vstup kódy zvyných $n-1$ písmenok slova w a nulové vektory ako kódy stavov, a ostatné obvody dostanú na vstup nulové vektory ako kódy znakov aj stavov (obr. 6.6). alej bude kadý elementárny obvod pracova ako sme u uviedli, take jednotlivé úrovne obvodu C_n budú krok po kroku zodpoveda konfiguráciám vo výpote A. Na úrovni $T(n)$ u iba skontrolujeme, i nejaký *eBO* je v akceptanom stave, ak áno, na výstup dáme jednotku inak nulu.

Z uvedeného vyplýva, e $DEPTH(C_n) = T(n)$, a kee uvaujeme konený poet stavov, symbolov abecedy a konený zápis δ-funkcie $DTS\ A$, tak dokáeme realizova elementárny obvod z koneného potu hradiel, z oho nakoniec plynie, e $SIZE(C_n) = O(T^2(n))$.

Týmto sme ukázali, e $L \subseteq L(\{C_n\})$. Opaná inklúzia (t.j. e takto zostrojená postupnos *BO* neakceptuje iadne slovo mimo jazyka L) je z kontrukcie zrejmá.

To, e takto vytvorená postupnos *BO* je naozaj *BC*-uniformná, nebudeme formálne dokazova. Obmedzíme sa len na fakt, e kadý obvod C_n tejto postupnosti bude vytváraný rovnakým postupom, o v súlade s poznámnkou 6.3.1 zabezpeuje *BC*-unifomitu. □

Veta 6.4.2. *Ak L je jazyk akceptovaný BC-uniformnou postupnosou BO $\{C_n\}$, priom $SIZE(C_n) = S(n)$, potom existuje DTS akceptujúci jazyk L v ase $S^3(n)$.*

Dôkaz: Nech $\{C_n\}$ je postupnos *BO* zo znenia vety. Chceme zozstroji $DTS\ A$ taký, e $L(A) = L$. Postupnos $\{C_n\}$ je *BC*-uniformná, teda poznáme $DTS\ A'$, ktorý vygeneruje kód $\langle C_n \rangle$. $DTS\ A$ bude na vstupnom slove w dky n pracova nasledovne:

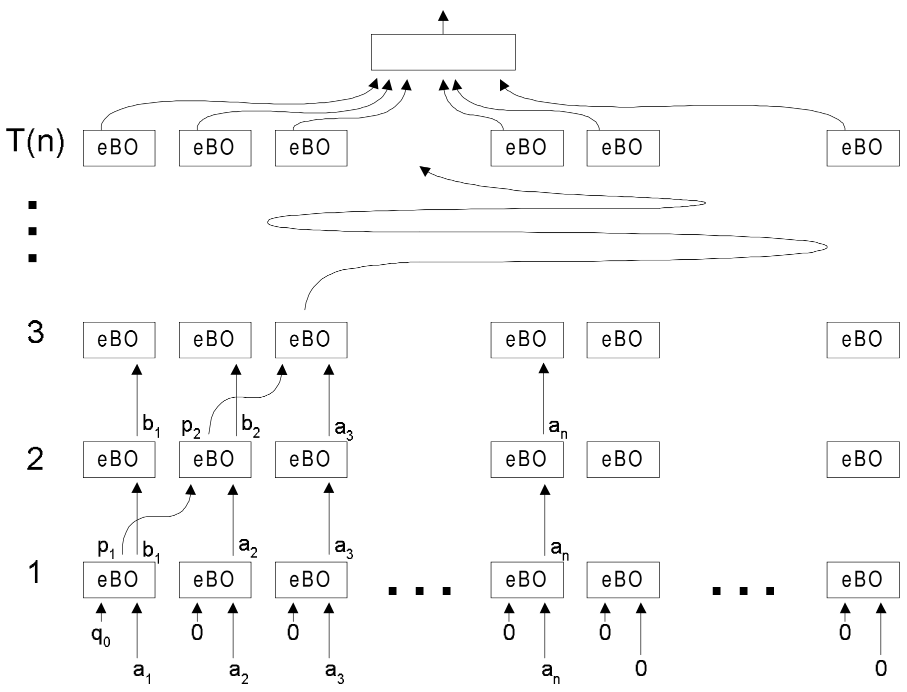
1. A si na pásku napíe kód *BO* C_n simulovaním A' so vstupom[4] 1^n. To dokáe v ase $O(S(n). \log S(n))$, lebo kód $\langle C_n \rangle$ je takejto dky.

2. A bude postupne ohodnocova[5] vrcholy *BO* C_n tak, e vstupné vrcholy ohodnotí poda príslunených hodnôt vstupu a hradlá ohodnotí poda hodnôt vstupov hradla a jeho typu. Ohodnocovanie musí samozrejme prebieha v topologickom usporiadaní. Na ohodnotenie jedného vrchola potrebuje A v najhorom prípade prejs celú pásku trikrát (dvakrát na nájdenie hodnôt vstupov a tretí krát na nájdenie a ohodnotenie samotného vrchola). Vrcholov je $S(n)$,

[3]v kadej úrovni by stailo $S(n)$ (vekos pracovného priestoru) obvodov, ale nemáme iaden predpoklad na priestor $DTS\ A$, vieme vak, e urite platí $S(n) \leq T(n)$

[4]Takýto vstup máme k dispozícii zo vstupného slova w tým, e vetky symboly budeme povaova za 1.

[5]ohodnoti hradlo znamená na páske oznai symboly prísluného hradla v kóde C_n hodnotami 0 alebo 1



Obr. 6.6: Simulácia DTS booleovským obvodom

vekos pásky je $O(S(n). \log S(n))$, teda na ohodnotenie vetkých vrcholov C_n potrebuje A as $O(S^2(n). \log S(n))$.

A akceptuje vstupné slovo práve vtedy, ke výstupný vrchol ohodnotí jednotkou. Teda ukázali sme, e $L \subseteq L(A)$. Opaná inklúzia je z kontrukcie zrejmá. Obidva kroky výpotu vykoná A v ase $O(S^2(n). \log S(n))$, o je samozrejme v $O(S^3(n))$. □

Dôsledok 6.4.1. $DTIME(Poly) = \mathcal{U}_{BC}SIZE(Poly)$

Dôkaz: Z vety 6.4.1 sme dostali $DTIME(T(n)) \subseteq \mathcal{U}_{BC}SIZE(T^2(n))$.

Z vety 6.4.2 sme dostali $\mathcal{U}_{BC}SIZE(S(n)) \subseteq DTIME(S^3(n))$.

Spojením týchto výsledkov teda dostávame $DTIME(Poly) = \mathcal{U}_{BC}SIZE(Poly)$. To znamená, e vo výpotovom modeli *BO* vieme polynomiálny sekvenný as premiea na polynomiálny paralelný priestor a opane. □

Veta 6.4.3. *Ak L je jazyk akceptovaný NTS, pracujúcim s jednou vstupnou a jednou pracovnou páskou, v priestore $S(n) \geq \log n$, potom existuje BC-uniformná postupnos BO $\{C_n\}$ taká, e $L(\{C_n\}) = L$ a $DEPTH(C_n) = O(S^2(n))$.*

Dôkaz: Uvaujme jazyk L a NTS A zo znenia vety. Zostrojíme *BO* C_n akceptujúci slová dky n z jazyka L. Zoberme si nejaké slovo $w \in L$, kde $|w| = n$. Zamyslime sa nad tým v kokých moných konfiguráciach môe by A poas výpotu na vstupnom slove w. Ak berieme do úvahy poty stavov, symbolov abecedy, políok pracovnej pásky, moné pozície hlavy na vstupnej a pracovnej páske, dostaneme, e poet vetkých moných konfigurácií je $k^{S(n)}$ pre vhodnú kontantu k.

Uvaujme alej reláciu krok výpotu $\vdash$ na týchto konfiguráciach. Za predpokladu, e binárne zakódujeme stavy a symboly abecedy, dokáeme reláciu $\vdash$ reprezentova booleovskou maticou rozmeru

$k^{S(n)} \times k^{S(n)}$, o je pre pevne stanovené n konená matica. Predpokladajme, e A má jednoznane danú akceptanú konfiguráciu. Potom otázka, i vstupné slovo w patrí do jazyka L, je vlastne otázka, i poiatoná konfigurácia A je v relácii $\stackrel{*}{\vdash}$ s akceptanou konfiguráciou. Relácia $\stackrel{*}{\vdash}$ je reflexívnym a tranzitívnym uzáverom relácie $\vdash$. Kee túto reláciu vieme reprezentova konenou booleovskou maticou, z príkladu 6.2.4 plynie, e aj reláciu $\stackrel{*}{\vdash}$ vieme reprezentova konenou booleovskou maticou, ktorú dostaneme koneným násobením mocnín matice reprezentujúcej reláciu $\vdash$. Z uvedeného príkladu tie vieme, e reflexívny a tranzitívny uzáver booleovskej matice M rozmeru $k^{S(n)} \times k^{S(n)}$ vypoítame na BO hbky rádovo $\log^2 k^{S(n)} = O(S^2(n))$.

Teda ná BO C_n so vstupom $\langle w \rangle$ najskôr zostrojí maticu relácie $\vdash$ na vstupnom slove w (to sa dá koneným BO, v ktorom je zakódovaná δ-funkcia A), a potom u spomínaným spôsobom urobí nad touto maticou jej reflexívny a tranzitívny uzáver. A akceptuje práve vtedy, ke vo výslednej matici je na i-tom riadku a j-tom stpci 1, kde i-ty riadok reprezentuje poiatonú konfiguráciu a j-ty stpec reprezentuje akceptanú konfiguráciu. Takto sme ukázali, e $L \subseteq L(\{C_n\})$, priom $DEPTH(C_n) = O(S^2(n))$. Opaná inklúzia je z kontrukcie zrejmá.

Tvrdenie o tom, e vytvorená postupnos BO je BC-uniformná, nechávame opä na poznámku 6.3.1. □

Veta 6.4.4. *Ak L je jazyk akceptovaný BC-uniformnou postupnosou BO $\{C_n\}$, priom $DEPTH(C_n) = D(n)$, potom existuje DTS akceptujúci jazyk L v priestore $O(D(n))$.*

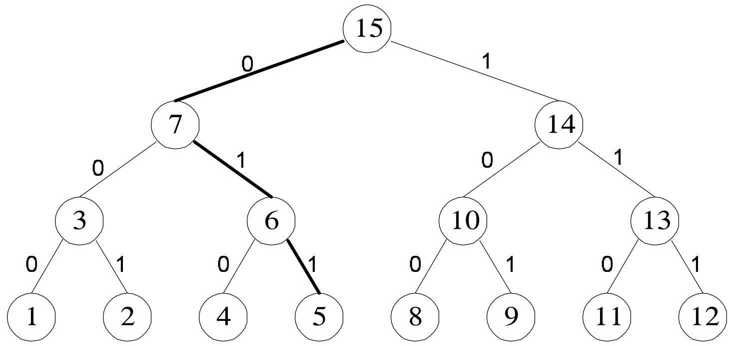
Dôkaz: Uvaujme jazyk L a postupnos BO $\{C_n\}$ zo znenia vety. Chceme zostroji prísluný DTS A. Nech $w \in L$ je dky n, ie kód $\langle w \rangle$ je akceptovaný BO C_n z postupnosti $\{C_n\}$. Vieme, e $\{C_n\}$ je BC-uniformná, take existuje generátor A' kódu $\langle C_n \rangle$ pracujúci v priestore $\log(SIZE(C_n))$. Zrejme $\log(SIZE(C_n)) \in O(D(n))$, take A má dostatok priestoru, aby mohol simulova A', ale nemá dos priestoru na uloenie celého kódu $\langle C_n \rangle$. Nemôeme teda poui techniku ohodnocovania C_n ako v dôkaze vety 6.4.2.

A bude postupne ohodnocova vrcholy C_n postorder prehadávaním C_n od výstupného vrcholu (vstupný vrchol ohodnotí poda príslunej asti vstupu $\langle w \rangle$ a vnútorný poda hodnôt jeho vstupov). A vak nemá na páske ani toko priestoru, aby si zapamätal kódy vetkých vrcholov na ceste od výstupného k práve prehadávanému vrcholu[6]. Preto si bude na páske pamäta len naviganú cestu od výstupného k práve prehadávanému vrcholu (obr. 6.7). Ak sa chce A posunú pri prehadávaní z jedného vrchola do druhého (t.j. z otca do syna alebo opane), zakadým musí spusti simuláciu A' a v jeho výstupe nájs kód prísluného vrchola. Vstupné slovo w bude A akceptova práve vtedy, ke výstupný vrchol dostane hodnotu 1.

Take A si bude na páske udriava kódy práve prehadávaného vrcholu a niekoko málo vrcholov v jeho okolí (aby mohol vrchol ohodnoti), naviganú cestu od korea k prehadávanému vrcholu a u známe ohodnotenia vrcholov, ktoré má aktuálne na páske. Na toto potrebuje A priestor rádovo $D(n)$. Na simulovanie A' potrebuje tie priestor rádovo $D(n)$, take vekos pracovnej pásky A bude $O(D(n))$.

Týmto sme ukázali, e $L \subseteq L(A)$, opaná inklúzia by opä mala by z kontrukcie zrejmá. □

[6] Cesta môe by dlhá maximálne $D(n)$ a kód kadého vrcholu je vekosti rádovo $\log SIZE(C_n)$, take by sme potrebovali priestor rádovo $D^2(n)$.



Obr. 6.7: Postorder prehadávanie booleovského obvodu

Dôsledok 6.4.2. $NSPACE(Poly) = \mathcal{U}_{BC}DEPTH(Poly)$

Dôkaz: Z vety 6.4.3 sme dostali $NSPACE(S(n)) \subseteq \mathcal{U}_{BC}DEPTH(S^2(n))$.

Z vety 6.4.4 sme dostali $\mathcal{U}_{BC}DEPTH(D(n)) \subseteq DSPACE(D(n))$.

Zo Savitchovej vety vieme, e $DSPACE(Poly) = NSPACE(Poly)$.

Spojením týchto výsledkov teda dostávame $NSPACE(Poly) = \mathcal{U}_{BC}DEPTH(Poly)$. To znamená, e vo výpotovom modeli BO vieme polynomiálny sekvenný priestor premiea na polynomiálny paralelný as a opane. □

6.5 Druhá poítaová trieda a Nick Class

V tejto asti si povieme nakoko sú paralelné výpotové modely, ktoré sme doteraz spomenuli a ktoré ete len spomenieme, vhodné na efektívne rieenie problémov, i je daný model vhodne zadefinovaný a ukáeme si akými mierami budeme posudzova vhodnos daného modelu.

Definícia 6.5.1. *Model poítaa patrí do druhej poítaovej triedy, ak sekvenný nedeterministický priestor je v polynomiálnom vzahu s asom na danom modeli.*

Predchádzajúca definícia hovorí o tom aké kritérium sme zvolili na posudzovanie toho, i je nejaký výpotový model pre nás zaujímavý (vhodný). Z doteraz spomenutých modelov patria do druhej poítaovej triedy modely Alternujúcich Turingových strojov (dôsledok 5.3.1) a BC-uniformných booleovských obvodov (dôsledok 6.4.2).

Takmer vetky paralelné modely patria do druhej poítaovej triedy. Ak uvaujeme nejaký nový model a chceme ho zaradi do druhej poítaovej triedy, tak môme urobi simuláciu daného modelu s Turingovým strojom podobne, ako sme to urobili vo vetách 6.4.3 a 6.4.4, alebo urobíme simuláciu s modelom, ktorý tam u patrí.

alou otázkou, ktorá je pre nás zaujímavá, je aké problémy môeme povaova za efektívne paralelne rieitené. Vieme, e pri sekvenných modeloch tieto problémy tvoria triedu $\mathcal{P}$, teda sú to problémy, ktoré vieme riei v polynomiálnom ase. Za efektívne rieitené povaujeme pri paralelných modeloch problémy patriace do triedy $\mathcal{NC}$ (Nick Class).

Definícia 6.5.2. *Nick Class*

- $\mathcal{NC}^i = \mathcal{U}_{BC} DEPTHSIZE(\log^i n, n^{O(1)})$
- $\mathcal{NC} = \bigcup_{i\geq 1} \mathcal{NC}^i$

Z doterajích poznatkov (pozri vety 6.4.2 a 6.4.4) vieme o práve zadefinovanej triede $\mathcal{NC}$ vyslovi pár tvrdení:

Veta 6.5.1. *Postavenie triedy $\mathcal{NC}$ medzi inými zloitostnými triedami.*

- $\mathcal{NC} \subseteq \mathcal{P}$
- $\mathcal{NC}^i \subseteq DSPACE(\log^i n)$

Doterajie teoretické výsledky ukazujú, e medzi triedami platí nasledovná hierarchia:

$$\mathcal{NC} \subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq PSPACE$$

Vieme, e $\bigcup_{i\geq 1} DSPACE(\log^i n) \subset PSPACE$. Z predchádzajúcej vety dostávame $\mathcal{NC} \neq PSPACE$. Otvoreným problémom zostáva, ktorá z inklúzií v spomenutej hierarchii tried je ostrá.

6.6 Iné uniformity pre booleovské obvody

BC-uniformita nie je jedinou monosou ako uniformova booleovské obvody. Niekedy sa nám táto uniformita na nae úely nehodí. Preto teraz ukáeme alie dva spôsoby ako mono uniformova booleovké obvody, a tie neskôr vyuijeme pri porovnaní s alternujúcimi Turingovými strojmi.

Definícia 6.6.1. *Nech $\{C_n\}$ je postupnos BO. Jej prísluným rozíreným jazykom prepojení je jazyk $L_e = \{$
$\langle n, g, p, y\rangle \mid n \in \{1\}^+, g \in \{0,1\}^+, p \in \{L, R\}^*, |p| \leq \log SIZE(C_n), y \in \{x, \wedge, \vee, \neg\} \cup \{0,1\}^+$, kde*

- *n udáva[7], e slovo $\langle n, g, p, y\rangle$ popisuje obvod C_n*
- *g je binárne kódované íslo vrchola obvodu C_n*
- *p je bu ε alebo naviganá cesta k vrcholu*
- *y je typ[8] hradla alebo íslo vrchola*

priom platí:

1. *ak $p = \varepsilon$, tak hradlo íslo g je typu $y \in \{x, \wedge, \vee, \neg\}$*
2. *ak $p \in \{L, R\}^+$, tak p-predchodca hradla íslo g má íslo y t.j. hradlo, do ktorého sa dostaneme z hradla g po naviganej ceste p, má íslo $y \in \{0,1\}^+$*

$\}$

[7] vimnime si, e n je kódované unárne
[8] x je oznaenie pre vstupný vrchol

Príklad 6.6.1. Na bliie pochopenie predchádzajúcej komplikovanej definície si ukáeme as jazyka L_e prislúchajúceho k obvodu C_4 (obr. 6.8) z nejakej postupnosti $\{C_n\}$.

vrchol 1: $\langle 1111, 1, \varepsilon, x\rangle$

vrchol 2: $\langle 1111, 10, \varepsilon, x\rangle$

vrchol 3: $\langle 1111, 11, \varepsilon, x\rangle$

vrchol 4: $\langle 1111, 100, \varepsilon, x\rangle$

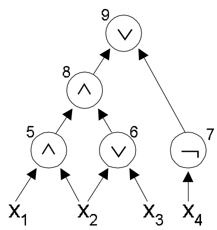
vrchol 5: $\langle 1111, 101, \varepsilon, \wedge\rangle, \langle 1111, 101, L, 1\rangle, \langle 1111, 101, R, 10\rangle$

vrchol 6: $\langle 1111, 110, \varepsilon, \vee\rangle, \langle 1111, 110, L, 10\rangle, \langle 1111, 110, R, 11\rangle$

vrchol 7: $\langle 1111, 111, \varepsilon, \neg\rangle, \langle 1111, 111, L, 100\rangle$

vrchol 8: $\langle 1111, 1000, \varepsilon, \wedge\rangle, \langle 1111, 1000, L, 101\rangle, \langle 1111, 1000, R, 110\rangle, \langle 1111, 1000, LL, 1\rangle,$ $\langle 1111, 1000, LR, 10\rangle, \langle 1111, 1000, RL, 10\rangle, \langle 1111, 1000, RR, 11\rangle$

vrchol 9: $\langle 1111, 1001, \varepsilon, \vee\rangle, \langle 1111, 1001, L, 1001\rangle, \langle 1111, 1001, LL, 101\rangle, \langle 1111, 1001, LR, 110\rangle,$ $\langle 1111, 1001, LLL, 1\rangle, \langle 1111, 1001, LLR, 10\rangle, \langle 1111, 1001, LRL, 10\rangle, \langle 1111, 1001, LRR, 11\rangle,$ $\langle 1111, 1001, R, 111\rangle, \langle 1111, 1001, RR, 100\rangle$



Obr. 6.8: Booleovský obvod C_4 z príkladu 6.6.1

Definícia 6.6.2. *Postupnos booleovských obvodov $\{C_n\}$ vekosti $S(n)$ a hbky $D(n)$ je*

1. $\mathcal{U}_E$ *- uniformná, ak existuje DTS A taký, e $L(A) = L_e$ a slová $\langle n, g, p, y\rangle$ akceptuje v ase* $\log S(n)$
2. $\mathcal{U}_{E^*}$ *- uniformná, ak existuje ATS A taký, e $L(A) = L_e$ a slová $\langle n, g, p, y\rangle$ akceptuje v ase $D(n)$ a priestore* $\log S(n)$.

Vimnime si, e priestor potrebný na akceptovanie jazyka L_e nezávisí od dky vstupného slova, ale od nejakého podslova vstupného slova. Preto nie je korektné na základe definície $\mathcal{U}_E$ uniformity písa $L_e \in DTIME(\log SIZE(C_n))$. Na vyjadrenie tejto situácie zavedieme peciálne oznaenie: $L_e \Subset DTIME(\log SIZE(C_n))$, a podobne pri $\mathcal{U}_{E^*}$ uniformite:
$L_e \Subset ATIMESPACE(DEPTH(n), logSIZE(C_n))$.

V alom budeme predpoklada, e v jazyku L_e, ktorý prislúcha postupnosti BO $\{C_n\}$, bude pouité "tesné" ("bez dier") oíslovanie vrcholov t.j. ak má obvod m vrcholov, tak ísla vrcholov budú z mnoiny $\{1, \ldots, m\}$ resp. $\{0, \ldots, m-1\}$.

Lema 6.6.1. *Nech $\{C_n\}$ je postupnos BO, L_e je jej prísluný rozírený jazyk prepojení, a nech $f(n) \in \Omega(\log SIZE(C_n))$. Potom tandardný kód $\langle C_n\rangle$ sa dá vypoíta v $DSPACE(f(n))$ práve vtedy, ke $L_e \Subset DSPACE(f(n))$.*

Dôkaz: Dokáeme obe inklúzie:

"⇒" Máme DTS A generujúci kód $\langle C_n\rangle$v priestore $f(n)$. Chceme zostroji DTS A' akceptujúci jazyk L_e v rovnakom priestore. A' bude na vstupnom slove $\langle n, g, p, y\rangle$ pracova nasledovne:

A' najskôr overí, i v obvode C_n existuje vrchol s islom g. To urobí tak, e bude simulova A na vstupe 1^n, a kým nevygeneruje slovo $\langle g, t, a, b$ v tandardom kóde. Ak $p = \varepsilon$, overí i $y = t$ (ak nie, tak slovo neakceptuje). Ak $p \in \{L, R\}^+$, tak A' potrebuje overi, i p-predchodca vrchola g má íslo y. To robí rekurzívne:

- ak $p = Lp'$, tak A' simuluje A, a kým nevygeneruje $\langle a, t', a', b'\rangle$. Ak $p' = \varepsilon$, overí typ t.j. i $t' = y$ (ak nie, tak slovo neakceptuje). Inak opä simuluje A', a kým nevygeneruje p'-predchodcu vrchola a at.
- ak $p = Rp'$, tak A' simuluje A, a kým nevygeneruje $\langle b, t', a', b'\rangle$, a pokrauje podobne ako v prvom prípade.

Na prácu A' nám staí priestor $f(n)$ (v zmysle $\Subset$ notácie), lebo v tomto priestore dokáeme simulova A a pamäta si jedno slovo tandardného kódu.

"⇐" Máme DTS A' akceptujúci jazyk L_e v $\Subset$ priestore $f(n)$. Chceme zostroji generátor A. Na vstupnom slove 1^n potrebuje A pre kadý vrchol íslo g v obvode C_n zisti jeho typ t a jeho vstupy a, b, a potom zapísa na výstup tvoricu $\langle g, t, a, b\rangle$. To bude robi nasledovne:

A bude postupne pre $g = 0, 1, 2, \ldots$ a $t = x, 0, 1, \wedge, \vee, I, \neg$ simulova A' na vstupe $\langle n, g, \varepsilon, t\rangle$. Ak nejakú takúto tvoricu A' akceptuje, znamená to, e v obvode C_n sa nachádza vrchol s íslom g a typom t. Na zistenie vstupov vrchola g bude A postupne pre $a = 0, 1, 2, \ldots$ simulova A' na vstupe $\langle n, g, L, a\rangle$. Ak A' pre nejaké a akceptuje, zistili sme íslo vrchola, ktorý je avým vstupom vrchola g. Podobne, simulovaním A' na vstupe $\langle n, g, R, b\rangle$ pre $b = 0, 1, 2, \ldots$, zistíme pravý vstup vrchola g. Teda A môe na výstup zapísa $\langle g, t, a, b\rangle$. Toto bude A vykonáva, a kým pre nejaké g A' neakceptuje ani jednu zo tvoríc $\langle n, g, \varepsilon, t\rangle$ pre vetky $t \in \{x, 0, 1, \wedge, \vee, I, \neg\}$. To znamená, e vrchol s takýmto íslom v obvode C_n nie je, a vaka predpokladu o íslovaní vrcholov "bez dier" vieme, e A u vygeneroval kódy vetkých vrcholov v obvode.

Na túto prácu staí A priestor $f(n)$, lebo v tomto prietore dokáe simulova A' a pamäta si nejakú informáciu kontantnej dky o práve generovanom vrchole.

□

Teraz si ukáeme aké vzahy sú medzi troma uniformitami booleovských obvodov, ktoré sme doteraz spomenuli.

Veta 6.6.1. *Medzi uniformitami platia nasledujúce vzahy:*

1. $\mathcal{U}_E DEPTHSIZE(D(n), S(n)) \subseteq \mathcal{U}_{BC} DEPTHSIZE(D(n), S(n))$
2. $\mathcal{U}_E DEPTHSIZE(D(n), S(n)) \subseteq \mathcal{U}_{E^*} DEPTHSIZE(D(n), S(n))$
3. *Nech* $D(n) \geq \log^2(S(n))$, *potom*
   $\mathcal{U}_{BC} DEPTHSIZE(D(n), S(n)) \subseteq \mathcal{U}_{E^*} DEPTHSIZE(D(n), S(n))$

Dôkaz: Dokáeme vetky tri inklúzie:

1. Nech $\{C_n\}$ je $\mathcal{U}_E$-uniformná postupnos BO. Z definície vieme, e prísluný jazyk prepojení $L_e \Subset DTIME(\log S(n))$. Zjavne iadny DTS nepouije viac priestoru ako asu, take $L_e \Subset DSPACE(\log S(n))$. Potom z lemy 6.6.1 plynie, e tandardný kód $\{C_n\}$ sa dá vypoíta v $DSPACE(\log S(n))$, teda $\{C_n\}$ je $\mathcal{U}_{BC}$-uniformná.
2. Nech $\{C_n\}$ je opä $\mathcal{U}_E$-uniformná, teda vieme, e $L_e \Subset DTIME(\log S(n))$. Zrejme $L_e \Subset DTIMESPACE(\log S(n), \log S(n))$. Kee DTS je peciálnym prípadom ATS, tak $L_e \Subset ATIMESPACE(\log S(n), \log S(n))$. Zrejme $D(n) \geq \log S(n)$, teda
$L_e \Subset DTIMESPACE(D(n), \log S(n))$, o znamená, e $\{C_n\}$ je $\mathcal{U}_{E^*}$-uniformná.
3. Nech $\{C_n\}$ je $\mathcal{U}_{BC}$-uniformná postupnos booleovských obvodov, teda jej tandardný kód vieme vygenerova v $DSPACE(\log S(n))$. Z lemy 6.6.1 vieme, e prísluný jazyk prepojení $L_e \Subset DSPACE(\log S(n))$. Zo simulácie DTS na ATS (veta 5.3.1) plynie $L_e \Subset ATIMESPACE(\log^2 S(n), \log S(n))$. Z predpokladu $D(n) \geq \log^2 S(n)$ dostávame $L_e \Subset ATIMESPACE(D(n), \log S(n))$, o znamená, e $\{C_n\}$ je $\mathcal{U}_{E^*}$-uniformná.

□

6.7 Porovnanie modelov BO a ATS

Prv ne prejdeme k samotnému porovnaniu modelov, uvedieme si jeden normálový tvar booleovských obvodov, ktoré neobsahujú hradlo negácie. Tento normálový tvar následne vyuijeme v alej vete.

Lema 6.7.1. *Pre kadý BO C_n existuje ekvivalentný BO C'_n obsahujúci len vrcholy typu $\wedge$, $\vee$, "0", "1", "x", "$\overline{x}$", kde "$\overline{x}$" oznauje negáciu vstupného vrchola "x".*

Dôkaz: Majme daný BO C_n. Chceme k nemu skontruova BO C'_n, ktorý nebude obsahova hradlo $\neg$. Jediným miestom v obvode C'_n, kde sa môe negácia prejavi, je na vstupe nahradením vstupného vrchola "x" jeho negáciou "$\overline{x}$".

Odstránenie hradiel $\neg$ z obvodu bude prebieha nasledovne. Obvod C_n budeme prehadáva (do írky) od výstupného vrchola k vstupným. Ak narazíme na hradlo $\neg$, budeme rozliova es moných prípadov v závislosti na tom, aký vrchol je vstupom pre nájdené hradlo:

$\wedge$ **:** Vstupom pre $\wedge$ sú nejaké hodnoty A, B. Vieme, e platí: $\neg(A \wedge B) = \neg A \vee \neg B$, teda vrcholy $\neg, \wedge$ nahradíme vrcholmi $\vee, \neg, \neg$ poda obrázka 6.9a.

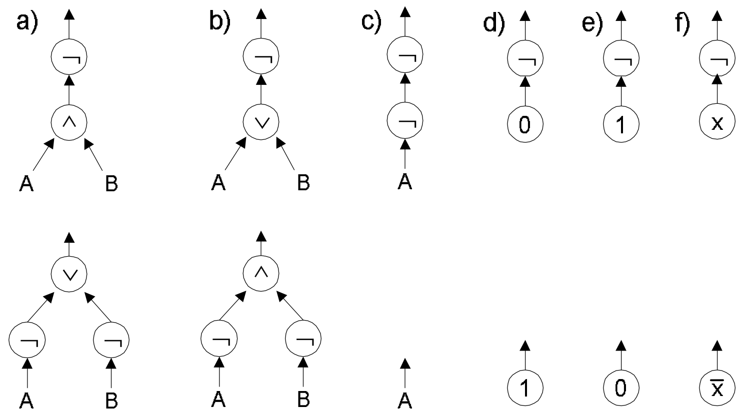
$\vee$ **:** Vstupom pre $\vee$ sú nejaké hodnoty A, B. Vieme, e platí: $\neg(A \vee B) = \neg A \wedge \neg B$, teda vrcholy $\neg, \vee$ nahradíme vrcholmi $\wedge, \neg, \neg$ poda obrázka 6.9b.

$\neg$ **:** Vstupom pre $\neg$ je nejaká hodnota A. Platí $\neg(\neg A) = A$, teda vrcholy $\neg, \neg$ z obvodu vynecháme (obr. 6.9c).

"0" : Vrcholy $\neg$,"0" nahradíme vrcholom "1" (obr. 6.9d).

"1" : Vrcholy $\neg$,"1" nahradíme vrcholom "0" (obr. 6.9e).

"x" : Vrchol $\neg$ a vstupný vrchol "x" nahradíme vstupným vrcholom "$\overline{x}$" s opanou hodnotou (obr. 6.9f).



Obr. 6.9: Modifikácia BO do normálového tvaru

V prvých dvoch prípadoch pri modifikácii obvodu opä vyuívame hradlá ¬. Treba si vak uvedomi, e tieto hradlá sú v obvode o úrove niie ako pôvodné nahradzované hradlo. Po príslunej úprave obvodu pokraujeme rekurzívne v prehadávaní obvodu v alej úrovni a sa dostaneme k vstupným vrcholom.

Takto modifikovaný obvod C'_n zrejme akceptuje rovnaký jazyk ako C_n, lebo kadá elementárna modifikácia bola logicky ekvivalentná. □

Veta 6.7.1. *Nech* $S(n) \geq n$, *potom platí:*
$\mathcal{U}_{E^*}DEPTHSIZE(D(n), S(n)) \subseteq ATIMESPACE(D(n), \log S(n))$.

Dôkaz: Nech $\{C_n\}$ je E^*-uniformná postupnos BO hbky $D(n)$ a vekosti $S(n)$ v normálnom tvare z predchádzajúcej lemy, a nech A' je ATS akceptujúci prísluný jazyk prepojení L_e v ase $D(n)$ a priestore $\log S(n)$. Chceme zostroji ATS A simulujúci $\{C_n\}$. Uvaujme nejaké slovo $w = a_1 \dots a_n \in L(\{C_n\})$ dky n (t.j. w je akceptované BO C_n). Ukáeme, ako pracuje A na vstupnom slove w.

A uhádne íslo a typ výstupného vrchola g_{out}, t_{out}. To urobí tak, e sa existenne rozvetví na vea vetiev a v kadej vetve na pracovnú pásku zapíe binárne íslo dky najviac $\log S(n)$ a nejaký typ vrchola[9]. Kadá takáto vetva overí svoje hádanie t.j. univerzálne spustí proces, v ktorom sa bude simulova A' na vstupe[10] $\langle n, g_{out}, \varepsilon, t_{out}\rangle$, teda i vrchol s íslom g_{out} a typom t_{out} v obvode C_n existuje. Ak taký vrchol existuje, potrebujeme overi, i je naozaj výstupný, teda opä vo vekom univerzálnom vetvení simuláciou A' overujeme, i pre vetky $h \in \{0,1\}^*$ také, e $|h| \leq \log S(n)$ platí $\langle n, h, L, g_{out}\rangle \notin L_e$, a zárove $\langle n, h, R, g_{out}\rangle \notin L_e$ to znamená, e vrchol g_{out} nemá nasledovníkov.

Uvaujme alej výpoet na vetve, ktorá úspene uhádla výstupný vrchol. Na pracovnej páske máme zapísané $\langle g_{out}, t_{out}, \varepsilon\rangle$. V nasledujúcom bude A háda typy vrcholov, ktoré sú vstupmi g_{out}. A sa poda typu t_{out} ($\wedge$ univerzálne, $\vee$ existenne) rozvetví priom v jednej vetve bude ma na páske zapísané $\langle g_{out}, L\rangle$ a v druhej $\langle g_{out}, R\rangle$.

Vo veobecnosti bude ma A na páske zapísané $\langle g, p\rangle$, kde g je íslo nejakého vrchola a $p \in \{L, R\}^*$

[9]Naozaj to bude tak, e pod poiatonou konfiguráciou sa rozvetví binárny strom hbky $\log S(n)$, priom pri prechode na niiu úrove A pripíe k u vygenerovanému binárnemu íslu jeden bit poda cesty v binárnom strome od korea.

[10]A má v kadej vetve na pracovnej páske zapísané iba $\langle g_{out}, t_{out}\rangle$, preto treba ete vhodne dopísa ε. n na pracovnú pásku zapísa nemôeme, lebo jej priestor $\log S(n)$ je na to malý. Nie je vak problém upravi A' tak, aby n, kee je kódované unárne, preítal zo vstupnej pásky A, kde ja zapísané vstupné slovo w dky n.

je nejaká naviganá cesta dky najviac $\log S(n)$. A v tomto prípade uhádne typ vrchola $g(p)$ (p-predchodca vrchola g) a univerzálne sa rozvetví na dve vetvy:

- V jednej overí, e hádal správne typ t.j. uhádne íslo p-predchodcu vrchola g (teda v existennom vetvení si A zapíe na pásku binárne íslo $g(p)$) a overí, e hádal správne, teda simuluje A' na vstupe $\langle n, g, p, g(p)\rangle$. Následne overí, e pre p-predchodcu vrchola g hádal správny typ, teda simuluje A' na vstupe $\langle n, g(p), \varepsilon, t_{g(p)}\rangle$.
- V druhej vetve pokrauje vo výpote. To znamená, e poda hádaného typu vrchola $g(p)$ sa ($\wedge$ univerzálne, $\vee$ existenne) rozvetví, priom v jednej vetve si k naviganej ceste p na pásku pripíe L a v druhej si pripíe R, teda A bude v situácii kedy má na páske zapísané $\langle g, p'\rangle$, kde p' je v jednom prípade pL, v druhom pR. V oboch vetvách A postupuje rovnako ako sme naznaili vyie.

  V prípade, e typ vrchola $g(p)$ je vstup, A uhádne, ktorý i-ty vstup to je a i je typu "x_i" alebo "$\overline{x}_i$". Hádanie v univerzálnom vetvení overí, teda v jednej vetve simuluje A' na vstupe $\langle n, g(p), \varepsilon, x_i\rangle$ resp. $\langle n, g(p), \varepsilon, \overline{x}_i\rangle$ a v druhej vetve výpoet koní s tým, e

  - ak je vstup typu "x_i", tak A akceptuje, ak $a_i = 1$ (neakceptuje, ak $a_i = 0$)
  - ak je vstup typu "$\overline{x}_i$", tak A akceptuje, ak $a_i = 0$ (neakceptuje, ak $a_i = 1$)

Uvaujme teraz prípad, e dka naviganej cesty p na páske dosiahne hranicu $\log S(n)$. To je problém, pretoe priestor pracovnej pásky A je ohraniený na $O(\log S(n))$ a naviac pri väej dke p by sme u ani nemohli jednoducho overova hádanie simuláciou A', lebo slová jazyka L_e sú definované s naviganou cestou dky najviac $\log S(n)$. V tomto prípade postupojeme nasledovne:

A má na páske zapísané $\langle g, p\rangle$, priom $|p| = \log S(n)$. V existennom vetvení uhádne íslo h a typ t_h vrchola $g(p)$ a univerzálne

- overí i $\langle n, g, p, h\rangle \in L_e$ a $\langle n, h, \varepsilon, t_h\rangle \in L_e$
- pokrauje vo výpote, priom na páske má zapísané $\langle h, t_h, \varepsilon\rangle$, teda sme v podobnej situácii, ako sme boli na zaiatku po uhádnutí výstupného vrchola.

Z kontrukcie by malo by vidie, e skontruovaný ATS A naozaj akceptuje práve slová z jazyka $L(\{C_n\})$.

Zamyslime sa teraz nad asovou a priestorovou zloitosou A:

- Uhádnutie výstupného vrchola trvá as $\log S(n) \leq D(n)$.
- Ak má A na páske zapísanú cestu p dky menej ako $\log S(n)$, tak A sa v hlavnom výpote (hádanie alej úrovne C_n) posunie v kontantnom ase vaka tomu, e pri rozvetvovaní staí na páske predi cestu p o L resp. R. Vetky dlhé hádania a overovania sa robia v boných vetvách výpotu, a teda ich netreba do celkového asu zaráta. Musíme si vak uvedomi, e tieto boné vetvy sú dlhé najviac $D(n)$, lebo v nich A bu háda, teda zapisuje na pásku (ale sú to vdy informácie dky najviac $D(n)$), alebo simuluje A', ten vak z definície pracuje v ase $O(D(n))$.
- Kadú $(\log S(n))$-tú úrove výpotu dosiahne dka p hranicu $\log S(n)$. A vtedy musí znova háda v hlavnom výpote íslo vrchola - na to spotrebuje as $O(\log S(n))$. Kee obvod má hbku $D(n)$, takýchto situácií sa poas výpotu vyskytne $\frac{D(n)}{\log S(n)}$ krát. Na rieenie vetkých týchto situácií spotrebuje A as $O(\frac{D(n)}{\log S(n)} \log S(n)) = O(D(n))$.

- Na koncoch jednotlivých vetiev výpotu pri zisovaní hodnoty i-teho vstupu musí A presunú hlavu nad tento symbol. Pri dke vstupu n by sme na túto operáciu potrebovali as $O(n)$, o je privea. Preto budeme uvaova, e ATS A má rýchly prístup na vstupnú pásku, teda nám bude stai as $\log n$, o je menej ako $D(n)$ vaka predpokladu $S(n) \leq n$ zo znenia vety.
- Poas celého výpotu bolo na páske zapísané íslo nejakého vrchola dky najviac $\log S(n)$, naviganá cesta dky najviac $\log S(n)$ a typ vrchola kontantnej dky, o celkovo dáva nároky na priestor $O(\log S(n))$.

Z uvedeného vyplýva, e sme kontrukciou ATS A splnili poiadavky na korektnos akceptácie ako aj na asovú a priestorovú zloitos. □

Veta 6.7.2. *Nech $S(n) \geq \log n$, potom pre vhodnú kontantu k platí:*
$ATIMESPACE(T(n), S(n)) \subseteq \mathcal{U}_E DEPTHSIZE(T(n), k^{S(n)})$

Dôkaz: K danému ATS A chceme zostroji postupnos BO $\{C_n\}$ akceptujúcu jazyk $L(A)$. Skôr ako zaneme kontruova $\{C_n\}$ zamyslime sa nad tým ako vyzerá výpoet na ATS.

Predstavme si úplný strom konfigurácií ATS A na nejakom vstupnom slove w. Na výpoet A sa môme pozera aj ako na vyhodnocovanie tohto stromu zdola. Kadý vrchol reprezentujúci akceptanú konfiguráciu na konci nejakej vetvy v strome konfigurácií ohodnotíme 1. Ostatné vrcholy na konci vetiev ako aj nekonené vetvy ohodnotíme 0. Kadý vrchol reprezentujúci univerzálnu konfiguráciu ohodnotíme 1 práve vtedy, ke vetci jeho synovia majú hodnotu 1. Vrchol reprezentujúci existennú konfiguráciu ohodnotíme 1 práve vtedy, ke aspo jeden z jeho synov má hodnotu 1. Vstupné slovo w bude A akceptova práve vtedy, ke kore stromu (vrchol reprezentujúci poiatonú konfiguráciu A na slove w) ohodnotíme 1.

Uvaujme teraz ATS A v nasledovnom normálovom tvare:
A pracuje s binárnou vstupnou aj pracovnou abecedou, A pouíva rýchly prístup na vstupnú pásku (t.j. A má peciálnu pásku dky $\log n$, na ktorú ke v binárnom kódovaní zapíe íslo j, dostane sa k j-temu symbolu na vstupe), úplný strom konfigurácií je binárny a v kadej vetve tohto stromu môe A íta vstup iba raz, a to na konci (t.j. A má peciálne ítacie stavy q^0 a q^1, do ktorých sa dostane na konci výpotu vetvy, priom v stave q^0 oakáva na vstupe[11] 0 (ak je na vstupe naozaj 0, tak akceptuje, ak je tam 1, neakceptuje) a v q^1 oakáva na vstupe 1).

Pozrime sa teraz na úplný strom konfigurácií A. Má $T(n)$ úrovní, priom v nultej úrovni bude jediný vrchol reprezentujúci poiatonú konfiguráciu. Hadaný BO C_n akceptujúci vstupné slovo w dky n bude vyzera vemi podobne. Kadému vrcholu v úplnom strome konfigurácií zodpovedá jeden vrchol v obvode C_n.

Vrcholu na úrovni t, ktorý reprezentuje konfiguráciu[12] c v strome konfigurácií zodpovedá v obvode C_n vrchol s íslom $f(t, c)$, kde $f : N \times N \to N$ je funkcia zachovávajúca tesné oíslovanie vrcholov, priom

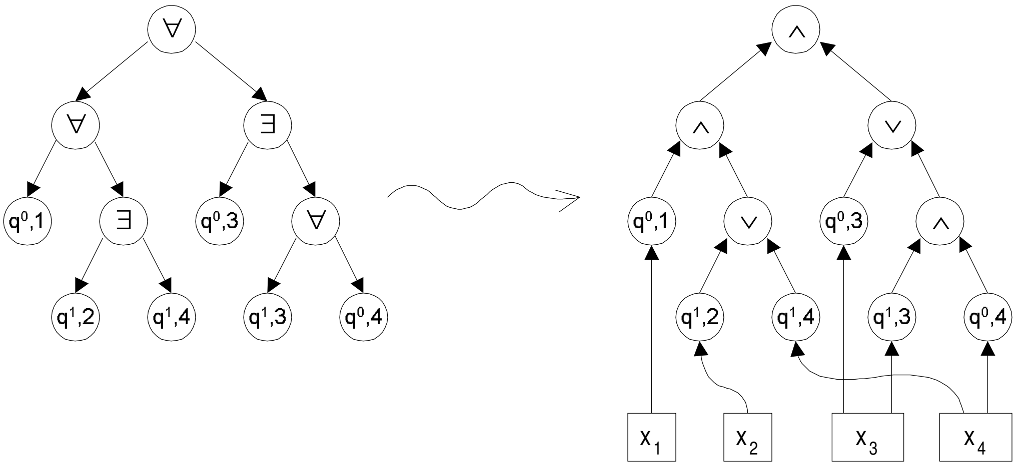
- ak c je univerzálna konfigurácia, tak vrchol $f(t, c)$ je typu $\wedge$
- ak c je existenná konfigurácia, tak vrchol $f(t, c)$ je typu $\vee$
- ak c je ítacia konfigurácia so stavom q^0 a na peciálnej páske je zapísané íslo j, tak vrchol $f(t, c)$ je typu[13] $\neg$ a jeho vstupom je j-ty bit vstupného slova w

[11] na jednej pozícii vstupu urenej íslom na peciálnej páske

[12] Uvaujeme tu konfiguráciu bez vstupu, lebo kadý FORKovaný výpoet pouíva len jeden symbol zo vstupu, take vstup v konfigurácii de facto nepotrebujeme. Na druhej strane vemi uitoná v konfigurácii je informácia o peciálnej páske urujúcej ktorý symbol sa má íta na konci výpotu.

[13] Ak A oakáva na vstupe 0 a naozaj tam 0 je, tak hradlo $\neg$ dostane na vstup 0 a na výstup pole 1, o signalizuje, e A oakával správne. Naopak ak je na vstupe 1, tak hradlo $\neg$ pole na výstup 0, o signalizuje, e A sa mýlil. Analogicky to funguje pre stav q^1 a hradlo I.

- ak c je ítacia konfigurácia so stavom q^1 a na peciálnej páske je zapísané íslo j, tak vrchol $f(t,c)$ je typu I a jeho vstupom je j-ty bit vstupného slova w
- ak c je akceptaná konfigurácia ale nie je ítacia, tak vrchol $f(t,c)$ je typu "1"
- ak c je odmietacia konfigurácia ale nie je ítacia, tak vrchol $f(t,c)$ je typu "0"



Obr. 6.10: Kontrukcia BO k ATS

Vstupnými vrcholmi pre vrchol $f(t,c)$ sú vrcholy s íslami $f(t+1,c_1)$ a $f(t+1,c_2)$ priom platí $c \underset{A}{\vdash} c_1$ a $c \underset{A}{\vdash} c_2$ (obr. 6.10).

Takto zostrojený obvod C_n pracuje presne tak ako výpoet ATS, ktorý sme opísali na zaiatku dôkazu. Teda príslun´a postupnos BO $\{C_n\}$ zrejme akceptuje jazyk $L(A)$.

Pozrime sa ete na miery zloitosti obvodu C_n. Obvod sme zostrojili "jedna k jednej" vzhadom na úplný strom konfigurácií ATS A. Z toho plynie, e $DEPTH(C_n) = T(n)$. $SIZE(C_n)$ zodpovedá potu vrcholov v strome konfigurácií. Kee do konfigurácií nezahame vstup, tak poet vetkých moných konfigurácií je $k^{S(n)}$ pre vhodné k. V zásade sa môe sta, e v úplnom strome konfigurácií máme v dvoch rôznych vetvách vrcholy reprezentujúce rovnakú konfiguráciu, o by mohlo spôsobi, e poet vrcholov, a teda aj vekos C_n by bola rádovo väia ako $k^{S(n)}$. Ak ale uvaujeme výpoet ATS ako FORKovanie procesov, tak nie je potrebné, aby boli spustené dva rovnaké procesy. Take ak chce ATS spusti proces, ktorého kópia u beí, tak tento duplicitný proces iba odkáeme na rovnaký u beiaci proces. To v booleovskom obvode znamená jedno prepojenie medzi hradlami. Take v konenom dôsledku môme uvaova opä akýsi normálový tvar ATS, ktorý bude ma v úplnom strome konfigurácií vetky konfigurácie disjunktné, a teda $SIZE(C_n) = O(k^{S(n)})$. $\square$

Kapitola 7

Parallel Random Access Machine ($PRAM$)

Model, ktorý predstavujeme v tejto kapitole, je najviac podobný reálnym paralelným architektúram. Ide o sústavu vea (teoreticky nekonene) samostatných výpotových jednotiek, ktoré nazývame procesory. Kadý procesor má svoju privátnu pamä a sadu intrukcií podobných tým, ktoré poznáme z asembleru. Jednotlivé procesory spolu komunikujú cez spolonú zdielanú pamä. Tento model je akousi automatovou analogiou k modelu $PCGS$.
Nebudeme sa zaobera porovnávaním generatívnej sily $PRAMu$ s modelmi Chomského hierarchie, pretoe u jeden samostatný procesor (RAM) má generatívnu silu Turingovho stroja. Zameriame sa skôr na vyuitie sily tohto modelu na rýchle paralelné rieenie problémov. V závere porovnáme model $PRAM$ s booleovskými obvodmi.

7.1 Definície a oznaenia

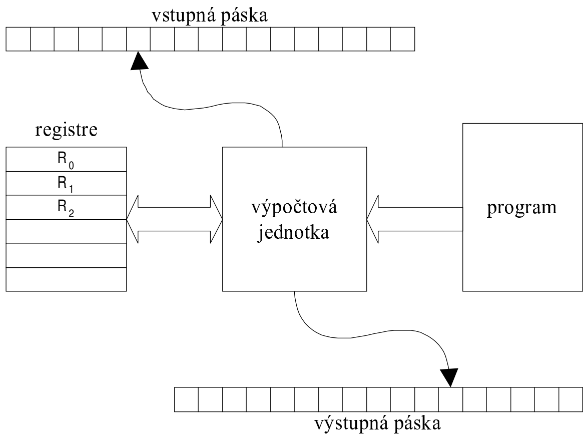
Skôr ako si zadefinujeme výpotový model $PRAM$, s ktorým budeme alej pracova, povieme si nieo o jeho základnej základnej výpotovej jednotke, ktorou je RAM.

7.1.1 RAM

Definícia 7.1.1. *RAM (Random Access Machine) je výpotový model pozostávajúci z výpotovej jednotky s pevne daným programom, jednej vstupnej a jednej výstupnej pásky a neobmedzeného potu registrov $R_0, R_1, R_2, \ldots,$ priom v jednom registri môe uchováva ubovone veké celé íslo (obr.7.1). Program výpotovej jednotky je postupnos jednoduchých[1] intrukcií, ktoré sú uvedené v tabuke 7.1.1. Výpoet zaína prvou intrukciou a koní intrukciou HALT.*

Model RAM sa dá zadefinova viacerými spôsobmi. Uvedenú definíciu môme, bez zmeny výpotovej sily a zloitosti, modifikova tak, e vstup nebude zadaný na vstupnej páske, ale v peciálnych registroch, priom v jednom z nich bude zadaná vekos vstupu n a v alich n registroch bude bit po bite zadaný samotný vstup.

[1] V sade intrukcií máme len jednoduché násobenie t.j. obsah registra krát nejaká kontanta. Nemáme tu násobenie medzi registrami, lebo to by dalo modelu RAM príli vekú silu.



Obr. 7.1: Model RAM

intrukcia	popis
$READ$	preítaj nasledujúci symbol zo vstupu a zapí ho do registra R_0
$WRITE$	obsah registra R_0 zapí na výstup
$STORE\ R_i$	obsah registra R_0 zapí do registra R_i
$COPY\ R_i$	skopíruj obsah registra R_i do registra R_0 ($R_0 \leftarrow [R_i]$)
$CONST\ c$	do registra R_0 zapí hodnotu c
$ADD\ R_i$	$R_0 \leftarrow [R_0] + [R_i]$
$SUB\ R_i$	$R_0 \leftarrow [R_0] - [R_i]$
$MULT\ c$	$R_0 \leftarrow [R_0].c$
$DIV\ c$	$R_0 \leftarrow [R_0]/c$
$IFZERO\ i$	ak register R_0 obsahuje 0, tak pokrauj intrukciou i
$GOTO\ i$	pokrauj intrukciou i
$HALTaccept$	ukoni výpoet a akceptuj
$HALTreject$	ukoni výpoet a neakceptuj

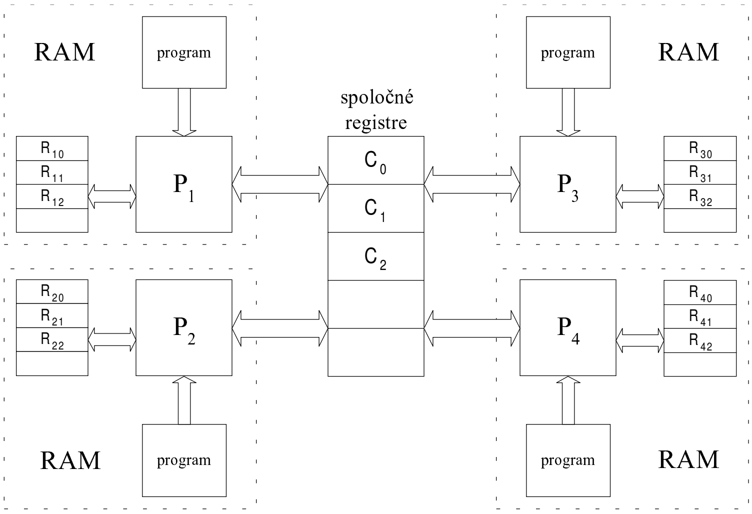
Tabuľka 7.1: Zoznam jednoduchých intrukcií $RAMu$

alou modifikáciou môe by rovnakým spôsobom upravený výstup, teda nie na výstupnej páske, ale opä v (na to urených) registroch.

7.1.2 *PRAM*

Prirodzeným rozírením modelu RAM v paralelných výpotoch je model $PRAM$, ktorý v sebe integruje viacero $RAMov$ komunikujúcich prostredníctvom zdieanej pamäte.

Definícia 7.1.2. *PRAM (Parallel Random Access Machine) je výpotový model pozostávajúci z neobmedzeného potu RAM procesorov oznaených $P_0, P_1, P_2, \ldots$ a neobmedzeného potu spolonýh (zdieaných) registrov $C_0, C_1, C_2, \ldots$ Kadý procesor P_i má svoje identifikané íslo (index), má svoju vlastnú pamä t.j. neobmedzenú sadu registrov $R_{i,0}, R_{i,1}, R_{i,2}, \ldots$ a intrukcie na priamy alebo nepriamy prístup (read/write) do spolonej pamäte. Základná sada intrukcii je zobrazená v tabuke 7.1.2. Procesory sú zosynchronizované poda globálnych hodín, teda intrukcie*



Obr. 7.2: Model $PRAM$

v jednotlivých procesoroch sú vykonávané v taktoch.
Vstup $x \in \{0,1\}^n$ *je zadaný nasledovne:*

- *v registri* C_0 *je uloená dka vstupu* n
- *v registri* C_i *je uloený i-ty bit vstupu* x_i

Výstup $y \in \{0,1\}^m$ *je uloený v rovnakej forme.*

intrukcia	popis
$R_i \leftarrow [R_j]$	skopíruj obsah registra R_j do registra R_i
$IDENT$	do registra R_0 zapí íslo procesora
$CONST\ c$	do registra R_0 zapí hodnotu c
$ADD\ R_i$	$R_0 \leftarrow [R_0] + [R_i]$
$SUB\ R_i$	$R_0 \leftarrow [R_0] - [R_i]$
$MULT\ c$	$R_0 \leftarrow [R_0].c$
$DIV\ c$	$R_0 \leftarrow [R_0]/c$
$IFZERO\ i$	ak register R_0 obsahuje 0, tak pokrauj intrukciou i
$GOTO\ i$	pokrauj intrukciou i
$HALT$	ukoni výpoet

Tabuľka 7.2: Zoznam jednoduchých intrukcií $PRAMu$

Pri práve zadefinovanom modeli sa ukazujú dva závané problémy:

1. Nemôeme predpoklada, e potenciálne nekonene vea procesorov sa bude podiea na danom výpote, a teda e budú vetky aktívne. Kadý výpoet potrebuje isté mnostvo procesorov, ktoré je závislé od vstupu resp. jeho dky. Preto jeden z procesorov, oznaený ako P_0, má význané postavenie, je to akýsi “generál”. P_0 zapíe do peciálneho registra C_{-1} maximálne

íslo aktívneho procesora t.j. vetky procesory s mením indexom sú aktívne poas výpotu. Teda najskôr je aktívny P_0 a ostatné akajú, kým zapíe hodnotu maximálneho indexu procesora.[2] Výpoet skoní, ke skoní procesor P_0 t.j. vykoná HALT.

2. Musíme vyriei konflikty pri viacnásobnom prístupe do spolonej pamäte. Kadá intrukcia je vykonávaná v troch fázach. V prvej fáze je povolený prístup (ak treba) do spolonej pamäte pre ítanie, potom sa vykoná prísluný výpoet pre danú intrukciu, a nakoniec je povolený prístup (ak treba) do spolonej pamäte pre zápis. Týmto sme oddelili prístup pre ítanie od prístupu pre zápis. Treba ete vyriei situáciu, ke viac procesorov naraz pristupuje do spolonej pamäte. Poznáme tri základné typy modelu $PRAM$:

   - $CRCW - PRAM$ (Concurrent Read Concurrent Write)
     Dovolíme procesorom súasné ítanie aj súasný zápis do spolonej pamäte (registra). Tento typ má tri verzie:
     - $PRIORITY$: Ak chce do jedného registra zapisova viacero procesorov, zápis vykoná len procesor s najmením íslom z procesorov, ktoré iadali o zápis.
     - $COMMON$: Ak chce do jedného registra zapisova viacero procesorov, zápis sa uskutoní len vtedy, ak vetky procesory chcú zapísa rovnakú hodnotu.
       V opanom prípade sa výpoet zasekne.
     - $ARBITRARY$: ubovoný[3] z procesorov iadajúcich o zápis zapíe svoju hodnotu do registra
   - $CREW - PRAM$ (Concurrent Read Exlusive Write)
     Dovolíme[4] procesorom súasné ítanie spoloného registra, ale zapisova do spoloného registra môe vdy len jeden procesor.
   - $EREW - PRAM$ (Exlusive Read Exlusive Write)
     íta a zapisova do spoloného registra môe vdy len jeden procesor.

7.2 Miery zloitosti

Pri modeli RAM uvaujeme tieto jenotkové miery zloitosti:

- $TIME\ T(n)$ = poet vykonaných intrukcií
- $SPACE\ S(n)$= poet pouitých registrov

Z definície týchto mier je zrejmé, e neuvaujú vekos dát (ísel), s ktorými intrukcie pracujú, teda pracova s vekými íslami je rovnako "drahé" ako pracova s malými íslami. Z toho plynie, e pouitie jednotkovej miery nie je vhodné napr. pri porovnávaní RAM s Turingovým strojom. V takýchto prípadoch uvaujeme tzv. logaritmickú mieru, ktorá zohaduje vekosti ísel pri jednotlivých operáciach. Logaritmická miera sa vak pouíva len zriedka, lebo v praxi máme aj tak obmedzenú vekos aj poet registrov.

[2] inou monosou rieenia tohoto problému by bolo zavedenie intrukcie FORK, ktorou uboný aktívny procesor môe aktivova alie, priom na zaiatku bude aktívny len procesor P_0

[3] je to jediný nedeterministický prístup v modeli $PRAM$ t.j. môe sa sta, e pri opakovanom výpote na rovnakom vstupe dostaneme rozdielne výstupy

[4] to znamená, e program (postupnos intrukcií) pre jednotlivé procesory musí spa túto podmienku

Pri modeli $PRAM$ uvaujeme tieto jenotkové miery zloitosti:

- $TIME\ T(n)$ = poet krokov (taktov) procesora P_0 pri výpote na vstupe dky n
- $PROCESSORS\ P(n)$ = maximálny poet aktívnych procesorov poas výpotu na vstupe dky n

Model $PRAM$ nemôe generova príli veké (superpolynomiálne) ísla. To znamená, e po $T(n) \leq \log n$ taktoch nebude ani v jednom registri zapísané íslo s viac ako $O(T(n))$ bitmi. Z toho plynie, e pri výpote s asovou zloitosou $T(n)$ na $P(n)$ procesoroch $PRAM$ nespotrebuje na uloenie dát viac ako $O(P(n).T^2(n))$ bitov. Máme teda adekvátnu mieru pre pamäové nároky ($SPACE$).

7.3 Výpotová sila modelu $PRAM$

Najskôr si ukáeme niekoko príkladov výpotov na modeli $PRAM$.

Príklad 7.3.1. Chceme vypoíta maximum z postupnosti n zadaných ísel $x_1, \ldots, x_n$ na modeli $COMMON\ CRCW - PRAM$.
Vstup bude zadaný nasledovne: $C_0 \leftarrow n$, $C_1 \leftarrow x_1, \ldots, C_n \leftarrow x_n$.
Chceme[5] výstup: $[C_0] = \max\{x_1, \ldots, x_n\}$.
Na výpoet pouijeme n^2 procesorov ozn. $P_{i,j}$, kde $i, j \in \{1, \ldots, n\}$ plus procesor P_0:

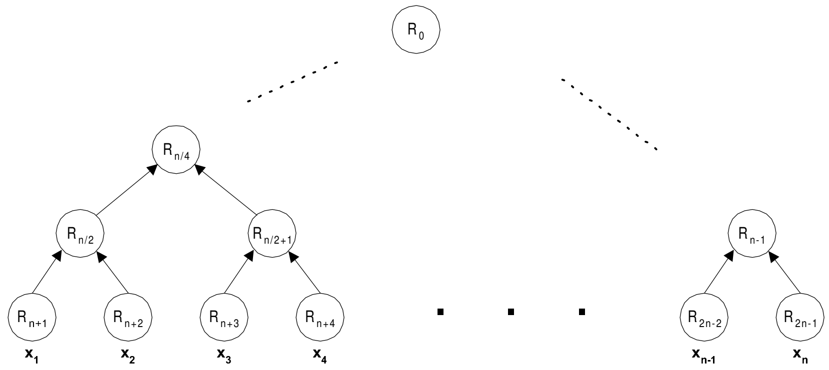
1. Kadý procesor $P_{i,1}$ vykoná: $C_{n+i} \leftarrow 0$
2. Kadý procesor $P_{i,j}$ vykoná[6]: $if\ [C_i] < [C_j]\ then\ C_{n+i} \leftarrow 1$
   Uvedomme si, e tu nenastane iadny konflikt pri súasnom zápise viacerých procesorov do toho istého registra, pretoe vetky procesory, ktoré budú chcie zapísova do registra C_{n+i}, budú chcie zapísa 1.
3. Kadý procesor $P_{i,1}$ vykoná: $if\ [C_{n+i}] = 0\ then\ C_0 \leftarrow [C_i]$
   Po kroku 2 je medzi registrami $C_{n+1}, \ldots, C_{2n}$ jediný s nulovou hodnotou a platí
   $[C_{n+i}] = 0$ práve vtedy, ke $x_i = \max\{x_1, \ldots, x_n\}$

Ako vidie, vaka pouitiu polynomiálneho potu procesorov, sa nám podarilo nájs maximum z íselnej postupnosti v kontantnom ase. Pri sekvenných modeloch na to potrebujeme lineárny as.

Príklad 7.3.2. Opä chceme vypoíta maximum ako v predchádzajúcom príklade, ale tentoraz na modeli $EREW - PRAM$.
Vstup je zadaný tak isto ako v predchádzajúcom príklade. Budeme[7] porovnáva dvojice registrov, priom výsledky zapíeme do nových registrov, ktoré budeme opä po dvojiciach porovnáva at. take dostaneme binárny porovnávací strom (obr.7.3). Kee pouívame stále nové a nové registre a kadý register má na starosti jeden procesor, tak procesory naraz neítajú ani nezapisujú do toho istého registra.

Procesorovú zloitos sme zmenili $P(n) = n$, ale asová zloitos vzrástla $T(n) = \log n$.

[5] $[C]$ oznauje obsah registra C, $C \leftarrow a$ oznauje zápis ísla a do regitra C
[6] procesor nemá k dispozícii takú silnú intrukciu, ale iste si vieme predstavi jej simuláciu pomocou koneného potu $PRAM$-intrukcií
[7] rovnakú mylienku mono poui pri úlohe síta n ísel



Obr. 7.3: Výpoet maxima na $EREW - PRAM$

Príklad 7.3.3. Ukáeme si, ako efektívne vieme triedi postupnos ísel $x_1, \ldots, x_n$ na modeli $CREW - PRAM$.
Vo vstupných registroch $C_1, \ldots, C_n$ sú hodnoty $x_1, \ldots, x_n$. Chceme, aby po skonení výpotu boli v registroch $C_1, \ldots, C_n$ ísla zo vstupu v utriedenom poradí.
Na výpoet pouijeme n^2 procesorov ozn. $P_{i,j}$.

1. Kadý procesor $P_{i,j}$ vykoná: $if\ [C_i] < [C_j]\ then\ C_{i,j} \leftarrow 1\ else\ C_{i,j} \leftarrow 0$

2. Kadých n procesorov $P_{i,1}, \ldots, P_{i,n}$ sa bude podiea na výpote[8]: $C_{n+i} \leftarrow 1 + \sum\limits_{j=1}^{n} [C_{i,j}]$

3. Kadý procesor $P_{i,1}$ vykoná: $C_{C_{n+i}} \leftarrow [C_i]$

asová zloitos $TIME$ je $T(n) = O(\log n)$, lebo prvý a tretí krok výpotu trvá kontantný as a na druhý potrebujeme logaritmický as, poet procesorov je $P(n) = n^2$.
Teraz ahko vidíme, e problém triedenia je v $\mathcal{NC}$, teda ho vieme efektívne paralelne riei.

Jednotlivé modely $PRAM$ sú medzi sebou relatívne ekvivalentné, take je v zásade jedno, ktorý pouívame. Tento poznatok vyuijeme pri porovnaní $PRAM$ s booleovskými obvodmi. Teraz si ukáeme porovnanie medzi modelmi $EREW$ a $CRCW$.

Veta 7.3.1. $\mathcal{L}(EREW - PRAM) = \mathcal{L}(CRCW - PRAM)$

Dôkaz: Inklúziu $\subseteq$ netreba dokazova, pretoe z definícií jednotlivých modelov vyplýva, e $EREW$ je peciálnym prípadom $CRCW$. Dokáeme opanú inklúziu.
Chceme simulova $CRCW$ na $EREW$. Treba vyriei situáciu, ke viacero procesorov chce naraz zapisova do toho istého registra t.j. treba z nich vybra jeden, ktorý svoju hodnotu do registra naozaj zapíe. Budeme z nich vybera procesor s najmením íslom (takto vyrieime naraz vetky tri verzie modelu $CRCW$).
Pod kadým registrom C_i budeme ma binárny strom pozostávajúci z nových registrov obsahujúcich ísla procesorov. V listoch sú ísla vetkých aktívnych procesorov. Na zaiatku kadý procesor pozná svoju pozíciu v tomto strome. Nie kadý aktívny procesor chce v danom okamihu zapisova do C_i. Budeme postupne po dvojiciach porovnáva obsahy registrov v tomto strome.
Ak je procesor avý (t.j. s mením íslom) a chce zapisova do C_i, tak postúpi o úrove vyie (t.j. zapíe

[8]síta n ísel je podobný problém ako nájs maximum z n ísel (príklad 7.3.2)

do prísluného registra svoje íslo).
Ak je procesor pravý, tak “skúsi”, i jeho avý sused postúpil.[9] Ak áno, tak v alom u nechce zapisova do C_i, ak nie, tak postúpi o úrove vyie. Toto sa vykonáva a ku koreu stromu, teda ku koreu sa dostane len jeden procesor (s najmením íslom z aktívnych procesorov, ktoré chceli zapisova do C_i) a ten zapíe svoju hodnotu do registra C_i.

ítanie bude fungova podobne, iba s tým rozdielom, e výpoet na strome pod kadým registrom sa bude vykonáva opaným smerom, aby kadý z procesorov, ktorý mal záujem íta, sa dostal k C_i.

Kee binárne stromy pod registrami majú výku $\log P(n)$, tak ítanie registra a zápis do registra sa z jedného kroku v modeli $CRCW$ predi na $O(\log P(n))$ krokov v modeli $EREW$, a teda as $EREW$ modelu sa oproti $CRCW$ zhorí $O(\log P(n))$ násobne. To je ale v celku dobrá simulácia.
□

7.4 Porovnanie modelov BO a $PRAM$

Veta 7.4.1. *Nech $\{C_n\}$ je BC-uniformná postupnos booleovských obvodov poítajúca funkciu $f_n : \{0,1\}^n \to \{0,1\}^m$ taká, e $DEPTH(C_n) = (\log n)^{O(1)}$ a $SIZE(C_n) = n^{O(1)}$. Potom existuje $CREW-PRAM$, ktorý poíta funkciu f_n v ase $T(n) = (\log n)^{O(1)}$ na $P(n) = n^{O(1)}$ procesoroch.*

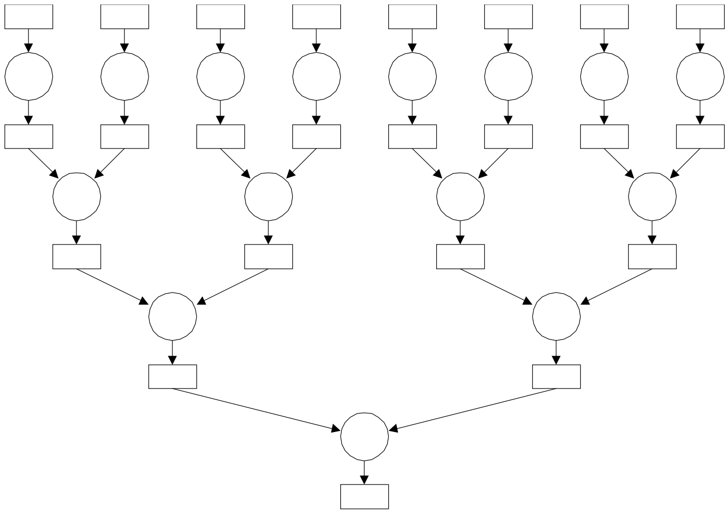
Dôkaz: Základná idea dôkazu spoíva v tom, e kadé hradlo booleovského obvodu (BO) bude simulované jedným procesorom a kadé spojenie medzi hradlami (t.j. vstup resp. výstup) v BO bude reprezentované jedným spoloným registrom. Potom kadý procesor v príslunom takte, poda hbky simulovaného hradla v BO, naíta obsahy registrov prislúchajúce k vstupom simulovaného hradla, vykoná daný výpoet poda typu hradla, a následne zapíe výsledok do registra prislúchajúceho k výstupu hradla (pozri obrázok 7.4, kde krunice predstavujú procesory a obdniky registre). Tento register je zárove vstupným registrom pre nejaký iný procesor, ktorý simuluje alie hradlo v BO.

Kadý procesor P_i, ktorý simuluje nejaké hradlo v BO, bude ma v spolonej pamäti vyhradené tyri registre $C_{i,IN1}, C_{i,IN2}, C_{i,OUT}, C_{i,TYP}$, priom v prvých dvoch budú zapísané adresy dvoch vstupných registrov, do tretieho procesor P_i zapíe výstup a vo tvrtom bude zapísaný typ simulovaného hradla. Procesor simulujúci vstupné hradlo samozrejme vystaí z jedným registrom pre adresu vstupu.

Obsahy registrov $C_{i,IN1}, C_{i,IN2}, C_{i,TYP}$ závisia od daného BO, a pred samotnou simuláciou ich poda neho musíme najskôr naplni. Pripomeme si, o to znamená, e máme daný BC-uniformný booleovský obvod. Znamená to, e poznáme DTS A pracujúci s jednou vstupnou, jednou pracovnou a jednou výstupnou páskou, ktorý na vstupe 1^n zapíe na výstupnú pásku kód BO C_n, priom pracovnú pásku má obmedzenú na priestor $\log SIZE(C_n)$.

Chceme zostroji $PRAM$ simulujúci postupnos BO $\{C_n\}$. To znamená, e pre nejaký vstup dky n potrebujeme najskôr odsimulova výpoet DTS A, ktorý vygeneruje kód BO, potom ho dekódova (t.j. správne naplni registre $C_{i,IN1}, C_{i,IN2}, C_{i,TYP}$), a nakoniec spusti samotnú simuláciu výpotu BO.

[9] To môeme zabezpei tak, e kadý postup o úrove vyie bude prebieha v troch krokoch. V prvom aví z dvojice bu nechcú zapisova a nepostupujú alebo postúpia, ak chcú zapisova, zatia o praví akajú t.j. vykonajú intrukciu NOP. V druhom kroku praví, ak chcú zapisova do C_i, tak ítajú obsah registra zodpovedajúcemu vyej úrovni. V treom kroku praví, ak zistili, e aví nezapísali svoje íslo do registra vyej úrovne, tak tam zapíu svoje íslo.



Obr. 7.4: Simulácia BO na modeli $PRAM$

- Simulácia výpotu DTS A:

  o vieme o $DTS \quad A$? Ak uvaujeme, e A pracuje na vstupe dky n v priestore $S(n) = \log SIZE(C_n)$, tak poet vetkých moných konfigurácií A je $k^{S(n)}.n.S(n)$ pre nejakú kontantu k, o je rádovo $(SIZE(C_n))^{O(1)}$ ie poda predpokladu $n^{O(1)}$. Oíslujme si jednotlivé konfigurácie $1, 2, \ldots r$. Nech[10] poiatoná konfigurácia má íslo 1 a koncová má íslo r. Poda predpokladu vety máme dostatok procesorov, aby sme kadej konfigurácii priradili jeden procesor. Kadému procesoru pridelíme dva registre zo spolonej pamäte, jeden[11] na výstup ($C_{j,OUT}$) a druhý na íslo registra ($C_{j,REG}$).

  V prvom kroku kadý procesor P_j prislúchajúci j-tej konfigurácii odsimuluje jeden krok výpotu DTS A z tejto konfigurácie, priom do prvého registra zapíe výstup zodpovedajúci tomuto kroku a do druhého zapíe íslo konfigurácie, do ktorej sa týmto krokom A dostal[12]. Kee sme odsimulovali jeden krok A z kadej konfigurácie, vyerpali sme tým vetky monosti δ-funkcie, take δ-funkciu u simulova nebudeme. V alom chceme jednotlivé kroky výpotu A vhodne "pospája", aby spolu tvorili celý výpoet A, ím by sme dostali výsledný kód $\langle C_n \rangle$.

  V kadom alom kroku procesor P_j vykoná:

  1. $C_{j,OUT} \leftarrow [C_{j,OUT}] \centerdot [C_{C_{j,REG},OUT}]$, kde $\centerdot$ je zreazenie[13] obsahov registrov
  2. $C_{j,REG} \leftarrow [C_{C_{j,REG},REG}]$

  Kadý procesor P naíta registre procesora prislúchajúceho ku konfigurácii, ktorej íslo má procesor P vo svojom druhom registri. Prvé (výstupné) registre zreazí a toto zreazenie zapíe ako novú hodnotu výstupného registra. Do druhého registra zapíe íslo konfigurácie, ktoré naítal (obr.7.5). Kadý procesor P_j teraz reprezentuje dva kroky výpotu

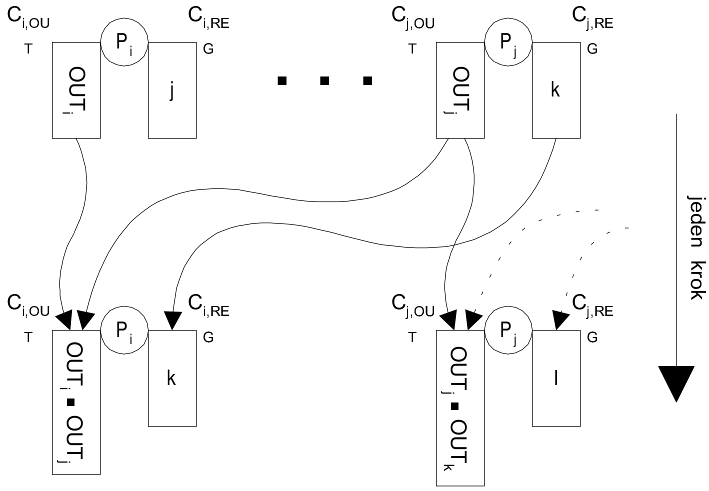
[10]takýto predpoklad môeme urobi, lebo pre daný vstup poznáme poiatonú konfiguráciu a môeme sa dohodnú na jednej koncovej konfigurácií pre vetky výpoty, z ktorej sa u nedá dosta do inej konfigurácie

[11]POZOR: Nemýli si s vyie uvedeným rovnako nazvaným registrom, pouitým pri samotnej simulácii BO

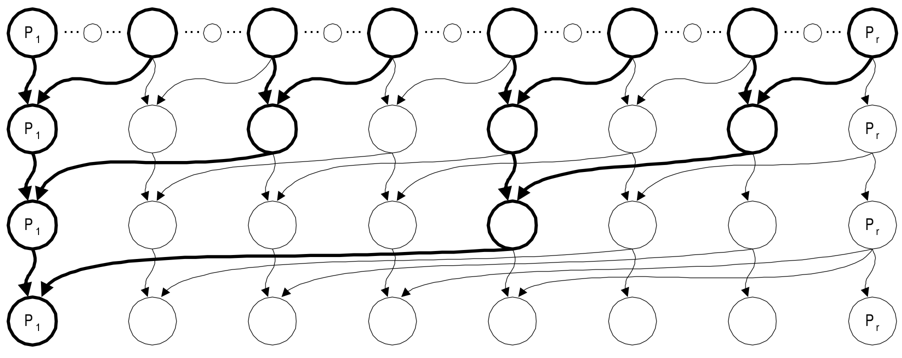
[12]tieto zápisy sú jednoznané, kee simulujeme deterministický stroj

[13]obsahmi registrov sú ísla v binárnom zápise, take zreazenie znamená nasledujúcu aritmetickú operáciu nad registrami: $C_{j,OUT} \leftarrow C_{j,OUT} + 2^{|C_{C_{j,REG},OUT}|}.C_{C_{j,REG},OUT}$

A zaínajúci konfiguráciou íslo j. Po alom kroku $PRAMu$ bude kadý procesor P_j reprezentova tyri kroky výpotu A s výnimkou tých, ktoré u vo svojom druhom registri majú íslo koncovej konfigurácie, z ktorej sa u nedá dosta. Toto sa opakuje a kým procesor P_1 nemá vo svojom druhom registri hodnotu r. Vtedy sa simulácia generovania kódu zastaví[14] a P_1 bude ma vo svojom prvom registri celý výstup A, teda kód BO C_n. Na obrázku 7.6 sú vyznaené procesory prislúchajúce ku konfiguráciám, v ktorých sa nachádza DTS pri generovaní daného kódu. Hrubo vyznaené procesory sú pre nás významné z pohadu generovania kódu. Tenko vyznaené procesory nám u nepomôu k vygenerovaniu kódu, napriek tomu stále vykonávajú svoj program, pretoe dopredu nevieme poveda, ktoré procesory budú významné a ktoré nie. Podobne nevieme dopredu uri, ktoré procesory (konfigurácie) sa budú podiela na výpote, a preto aj procesory, ktoré sa v konenom dôsledku na výpote podiela nebudú (na obrázku znázornené malými krunicami v prvom riadku), vykonávajú svoj program.



Obr. 7.5: Jeden krok simulácie DTS



Obr. 7.6: Generovanie kódu BO na modeli $PRAM$

[14] realizáciu si môeme predstavi nasledovne: P_1 zapíe do nejakého peciálneho booleovského registra, ktorý bol inicializovaný na TRUE, hodnotu FALSE a program pre procesory upravíme tak, e dané intrukcie budú vykonáva len vtedy, ke v tomto registri je hodnota TRUE

Kee po kadom kroku $PRAMu$ sa dka odsimulovaného výpotu A zdvojnásobí, tak na odsimulovanie celého výpotu vystaíme s asom logaritmus z potu konfigurácií, o je $O(\log SIZE(C_n))$.

- Dekódovanie:

  Predpokladajme, e u máme v nejakom registri kód $\langle C_n \rangle$. Ten je v tvare:

  $(\langle$ íslo hradla,typ hradla,íslo avého vstupu, íslo pravého vstupu $\rangle)^*$

  Zrejme dka kódu je $|\langle C_n \rangle| = O(SIZE(C_n).\log SIZE(C_n))$. Opä máme dostatok procesorov, aby sme nimi "pokryli" kadý symbol výstupu. Kadý procesor P_j naíta register s kódom $\langle C_n \rangle$ a akoby nastaví svoju ítaciu hlavu na j-ty symbol kódu $\langle C_n \rangle$. Procesory samozrejme iadne ítacie hlavy nemajú, ale ahko si vieme predstavi, e $PRAM$ by vedel na jeden krok "rozloi" obsah registra na vea registrov obsahujúcich po jednom symbole, a potom by u procesory pracovali nad registrami ako je to v modeli $PRAM$ zvykom a nie s ítacou hlavou ako prezentujeme tu.

  Po prvom kroku prestanú pracova vetky procesory, ktoré nepreítali na vstupe symbol[15] "(". Tie, ktoré "(" preítali (t.j. boli nastavené na zaiatok podslova v kóde $\langle C_n \rangle$ reprezentujúce kód nejakého hradla), v kadom alom kroku preítajú jeden symbol, a kým nepreítajú ")" (t.j. koniec kódu hradla). Kadý takýto procesor "zistí" informácie o jednom hradle BO, teda íslo hradla, typ hradla a ísla vstupov, a zapíe ich do prísluných registrov $C_{i,IN1}, C_{i,IN2}, C_{i,TYP}$.

  Kee ísla hradiel sú v kóde zapísané v binárnom tvare, tak dka kódu jedného hradla je $O(\log SIZE(C_n))$. Kadý procesor potrebuje preíta kód jedného hradla a zapísa získané informácie do registrov. Preto as, ktorý na dekódovanie potrebujeme, je $O(\log SIZE(C_n))$.

- Simulácia výpotu BO:

  Teraz u máme vetko pripravené na simuláciu výpotu BO, teda v registroch $C_{i,IN1}$, $C_{i,IN2}$, $C_{i,TYP}$ sú zapísané správne hodnoty prislúchajúce simulovanému BO. Predpokladajme, e vetky registre $C_{i,OUT}$ sú inicializované na nejakú NILovú hodnotu[16]. Procesor P_i bude vykonáva program:

  1. naítaj obsahy registrov $C_{i,IN1}, C_{i,IN2}, C_{i,TYP}$
  2. akaj kým platí: $[C_{C_{i,IN1}}] = NIL$ a $[C_{C_{i,IN2}}] = NIL$
  3. vykonaj operáciu poda typu hradla $[C_{i,TYP}]$ so vstupmi $[C_{C_{i,IN1}}], [C_{C_{i,IN2}}]$
  4. zapí výsledok operácie do registra $C_{i,OUT}$

  Na vykonanie tohto programu potrebuje kadý procesor as najviac $(\log n)^{O(1)}$, pretoe na druhom riadku bude procesor aka toko krokov koko je hbka hradla v BO a z predpokladu vieme, e $DEPTH(C_n) = (\log n)^{O(1)}$, ostatné riadky programu sa vykonajú v kontantnom ase.

Z dôkazu ahko vidie, e sme splnili poiadavky kladené na asovú a procesorovú zloitos, ako aj na model $CREW - PRAM$. $\square$

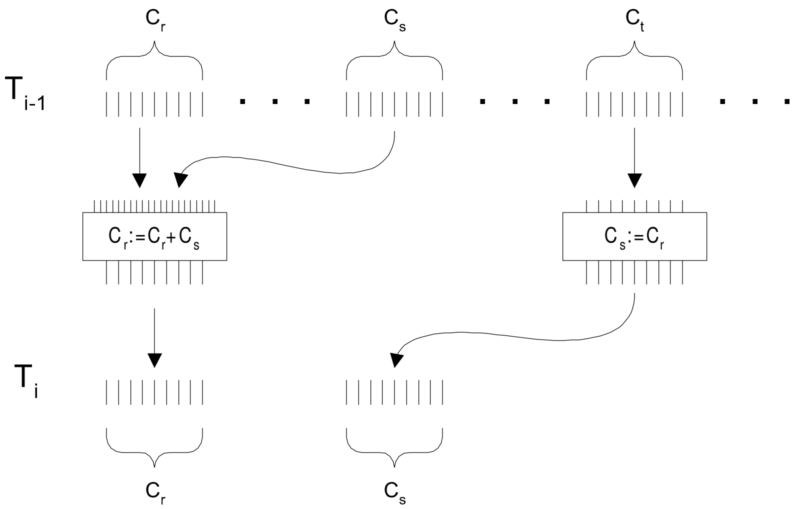
Veta 7.4.2. *Nech M je $CREW - PRAM$, ktorý v ase $T(n) = (\log n)^{O(1)}$ a s potom procesorov $P(n) = n^{O(1)}$ poíta funkciu f. Potom existuje kontanta k a BC-uniformná postupnos booleovských obvodov $\{C_n\}$ taká, e C_n poíta na vstupe $x_1 \ldots x_n$ výstup $y_{11}y_{12} \ldots y_{ij} \ldots$ kde y_{ij} je hodnota j-teho bitu spoloného registra C_i v ase $T(n)$ pre $1 \leq i \leq P(n).T(n)$ a $1 \leq j \leq k.T(n)$, priom $DEPTH(C_n) = (\log n)^{O(1)}$ a $SIZE(C_n) = n^{O(1)}$.*

[15] vetky symboly sú zakódované binárne, take procesory prestanú pracova po tom ako rozpoznajú symbol "("

[16] vzhadom na to, e vo výstupných registroch budú len hodnoty 0 alebo 1, hodnotu NIL môe reprezentova akákovek iná hodnota

Dôkaz: Chceme zostroji BO simulujúci $CREW-PRAM$. Vstupom pre M je prvých n spoloných registrov $C_1, \ldots, C_n$. Vstupom $x_1 \ldots x_n$ pre BO C_n sú obsahy týchto registrov v ase T_0.

Výpoet M závisí od postupnosti intrukcií jenotlivých procesorov. Pre kadú intrukciu zostrojíme elementárny obvod simulujúci túto intrukciu. Vstupom pre bude obsah registra (registrov), s ktorým prísluná intrukcia pracuje a výstupom bude nový obsah výstupného registra. Uvedomme si, e za náho predpokladu existencie kontanty k takej, e $1 \leq j \leq k.T(n)$ (t.j. vieme ohranii maximálnu vekos registra poas celého výpotu), kadú intrukciu vieme simulova elementárnym obvodom obsahujúcim konený poet hradiel.



Obr. 7.7: Simulácia výpotu $PRAM$ na BO

C_n bude ma $T(n)$ úrovní, priom v i-tej úrovni budeme ma obsahy vetkých registrov (bit po bite) v ase T_i. Dosiahneme to tak, e medzi (i-1). úrove a i-tu úrove umiestnime (a vhodne pospájame) elementárne obvody prislúchajúce k i-tym intrukciám vetkých aktívnych procesorov (obr.7.7). Take na poslednej úrovni budú obsahy registrov v ase $T(n)$, o zodpovedá výstupu M.

Vaka predpokladom o pote ($1 \leq i \leq P(n).T(n)$), vekosti ($1 \leq j \leq k.T(n)$) registrov a z toho vyplývajúcej konenej vekosti elementárnych obvodov dostávame zloitosti pre C_n: $DEPTH(C_n) = (\log n)^{O(1)}$ a $SIZE(C_n) = n^{O(1)}$.

Teda dokázali sme ekvivalentnos mier zloitosti medzi týmito dvoma modelmi t.j. as na modeli $PRAM$ zodpovedá hbke na BO a poet procesorov na $PRAM$ zodpovedá vekosti BO. □

Dôsledok 7.4.1. Model $PRAM$ je v druhej poítaovej triede.

Dôsledok 7.4.2. Triedu $\mathcal{NC}$ môeme pomocou modelu $PRAM$ zadefinova nasledovne:

$$\mathcal{NC} = TIMEPROCESSORS((\log n)^{O(1)}, n^{O(1)})$$

Literatúra

[1] John E. Hopcroft, Jeffrey D. Ullman: Formálne jazyky a automaty (Alfa SNTL, 1978)

[2] Gabor T. Herman: Closure Properties of Some Families of Languages Associated with Biological Systems (Information and Control, 24, 101-121, 1974)

[3] Peter Gvozdjak, Branislav Rovan: Time-Bounded Parallel Rewriting

[4] A. Salomaa: Formal Languages (Academic Press, New York, 1973)

[5] Lila Santean, Jarkko Kari: The impact of the number of cooperating grammars on the generative power (Theoretical Computer Science 98, 249-262, 1992)

[6] Juraj Hromkovi, Jarkko Kari, Lila Kari: Some hierarchies for the communication complexity measures of cooperating grammar systems (Theoretical Computer Science 127, 123-147, 1994)