Odbor 9.2.1 Informatika
Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Vlastnosti nekonečných slov generovaných pomocou DGSM

(diplomová práca)

Marián Sládek

Školiteľ: doc. RNDr. Pavol Ďuriš, CSc.

Bratislava, 2009

Abstrakt

SLÁDEK, Marián. Vlastnosti nekonečných slov generovaných pomocou DGSM [diplomová práca]. Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky. Katedra informatiky. Školiteľ: doc. RNDr. Pavol Ďuriš, CSc. Bratislava: FMFI UK, 2009. 53s.

Táto práca sa zaoberá nekonečnými slovami generovanými pomocou systému DGSM TAG, kde sa snaží hľadať nové spôsoby riešenia otvorených problémov súvisiacich s týmto systémom. Definujeme v nej základné mechanizmy a princípy TAG systémov. Hľadáme v nej rôzne druhy zjednodušení a abstrakcií pre jednoduchšie rozvíjanie tejto oblasti. Poukazujeme na ďalšie smerovanie výskumu. S tým je úzko spätý dôkaz o existencii pamäťovo optimálneho Turingovho stroja, ktorý generuje ľubovoľné DGSM slovo v optimálnej pamäti.

Kľúčové slová: DGSM, nekonečné slová, TAG systémy, vlastnosti, GSPACE

Predhovor

Jednou z mnohých oblastí, ktorou sa zaoberá informatika, je štúdium zložitosti jazykov a slov, či už konečných alebo nie. Pri výpočtovej zložitosti nad nekonečnými slovami sa meria, koľko času alebo pamäte potrebujeme na vygenerovanie n-tého písmena v nekonečnom slove. Bolo dosiahnutých mnoho výsledkov, či už v oblasti podslov, slov samotných alebo doplnkových faktorov, ktoré su naviazané na daný problém.

V tejto práci sa zameriame na pamäťovú zložitosť stroja DGSM, konkrétne ukážeme nové možnosti smerovania a porovnáme si vplyv mojich výsledkov na niektoré už dosiahnuté. Táto práca si dáva za cieľ vniesť trochu svetla do predmetnej oblasti a ukázať, že zaoberať sa týmito problémami má stále zmysel a je na čom pracovať.

Skôr než budeme pokračovať ďalej, musím poznamenať, že všetky výsledky v tejto práci sú moje, s výnimkou pár výsledkov, pri ktorých je explicitne uvedený pôvod. Rád by som sa taktiež poďakoval každému, kto túto prácu držal v ruke a dostal sa aspoň k týmto riadkom.

Obsah

KAPITOLA 1

Úvod

V tejto kapitole si vymedzíme základnú oblasť pôsobnosti, určíme, akú terminológiu budeme používať a ukážeme, akým smerom sa budeme uberať.

Definície

Stredobodom nášho záujmu budú nekonečné slová, stroje, ktoré ich generujú, a ich vlastnosti. Celá táto práca sa snaží napomôcť k vyriešeniu otvoreného problému, či všetky slová generované systémom DGSM TAG sú z množiny $GSPACE(\log n)$. Skôr než s čímkoľvek začneme pracovať, je potrebné si ujednotiť terminológiu:

DEFINÍCIA 1. `Abeceda` *je neprázdna konečná množina symbolov (písmen), ktorú budeme označovať* $\sum$.

DEFINÍCIA 2. `Konečné slovo` w *nad abecedou* $\sum$ *je n-tica* $(a_1, a_2, a_3...a_n)$, *kde n je nejaké prirodzené číslo a všetky symboly slova sú z množiny* $\sum$ *(všetky písmená sú z abecedy* $\sum$*). Pre jednoduchosť budeme písať:* $w = a_1a_2a_3...a_n$.

DEFINÍCIA 3. `Prázdne slovo` *je vlastne prázdna postupnosť. Toto slovo budeme označovať ako* ϵ.

DEFINÍCIA 4. `Nekonečné slovo` w *nad abecedou* $\sum$ *je zobrazenie* $f : N \to \sum$. $w = f(1)f(2)f(3)f(4)\ldots = a_1a_2a_3a_4\ldots$

DEFINÍCIA 5. `Dĺžka slova` *je dĺžka postupnosti, ktorá ho vytvára. Dĺžku slova* w *budeme označovať* $|w|$. *(Teda ak* $w = a_1a_2...a_n$, *potom* $|w| = n$.*)*

DEFINÍCIA 6. `Zreťazenie slov`. *Nech máme konečné slovo* $u = u_1u_2u_3\ldots u_n$ *a slovo* $v = v_1v_2v_3\ldots$, *potom* $u \cdot v = u_1u_2u_3\ldots u_nv_1v_2v_3\ldots$ *je operácia zreťazenia, ktorá pripojí druhé slovo na koniec prvého. Znak zreťazenia (bodku) budeme väčšinou vynechávať.*

DEFINÍCIA 7. `Mocnina slova` *je* $w^0 = \epsilon$, $\forall n \in N : w^n = w \cdot w^{n-1}$.

DEFINÍCIA 8. `Iterácia` *nad konečným slovom je* $w^* = www...$, *teda vytvorí periodické nekonečné slovo s periódou* w.

DEFINÍCIA 9. `Jazyk` *nad abecedou* $\sum$ *je ľubovoľná množina slov nad abecedou* $\sum$.

DEFINÍCIA 10. `Zreťazenie jazykov` M *a* N *je* $M \cdot N = \{ab|a \in M; b \in N;\}$.

DEFINÍCIA 11. `Mocnina jazyka` M *je* $M^0 = \{\}$, $\forall n \in N : M^n = M \cdot M^{n-1}$.

DEFINÍCIA 12. `Iterácia` *nad jazykom* M *je* $M^* = \bigcup_{i=0}^{\infty} M^i$.

DEFINÍCIA 13. `Kladná iterácia` *nad jazykom* M *je* $M^+ = \bigcup_{i=1}^{\infty} M^i$.

DEFINÍCIA 14. `Blok` *znakov* a *je slovo z množiny* $\{a\}^*$, *teda slovo, ktoré sa skladá zo súvislého radu rovnakých znakov.*

DEFINÍCIA 15. `Podslovo` *slova* w *je každé slovo* u *také, ku ktorému vieme nájsť slová* m *a* n *také, že platí* $w = mun$.

DEFINÍCIA 16. `Prefix` *slova* w *je každé slovo* u *také, ku ktorému vieme nájsť slovo* v *také, že platí* $w = uv$. *Tento vzťah budeme označovať* $u \prec w$.

DEFINÍCIA 17. `Sufix` *slova* w *je každé slovo* v *také, ku ktorému vieme nájsť slovo* u *také, že platí* $w = uv$. *Tento vzťah budeme označovať* $w \succ v$.

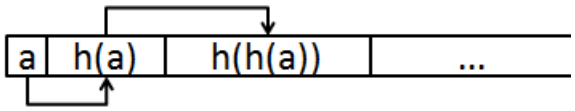
DEFINÍCIA 18. `Horná celá časť` *čísla* x *(označujeme* $\lceil x \rceil$*) je najmenšie prirodzené číslo také, ktoré je väčšie alebo rovné ako číslo* x.

DEFINÍCIA 19. `Dolná celá časť` *čísla* x *(označujeme* $\lfloor x \rfloor$*) je najväčšie prirodzené číslo také, ktoré je menšie alebo rovné ako číslo* x.

DEFINÍCIA 20. `Homomorfizmus` *je zobrazenie* h *z* $\sum_1$ *do* $\sum_2$ *také, že pre všetky slová* u *a* v *z* $\sum_1^*$ *platí* $h(u \cdot v) = h(u) \cdot h(v)$.
Ak platí $\forall x \in \sum_1^* : h(x) = \epsilon \Leftrightarrow x = \epsilon$, *potom* h *je nevymazávajúci homomorfizmus.*

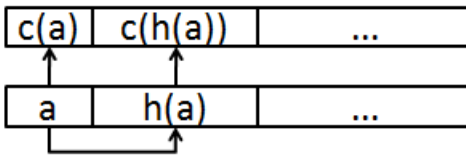
DEFINÍCIA 21. `GSPACE(`f`)` *je množina všetkých nekonečných slov* w *takých, že pre každé* $w \in GSPACE(f)$ *existuje Turingov stroj, ktorý generuje slovo* w, *pričom na vygenerovanie každého prefixu dĺžky* n *použije* $O(f(n))$ *pamäte.*

Týmto sme si špecifikovali základné prostriedky, ktoré budeme používať. Keďže táto práca pojednáva o nekonečných slovách generovaných strojom DGSM, je nevyhnutné pochopiť hierarchiu strojov generujúcich nekonečné slová.



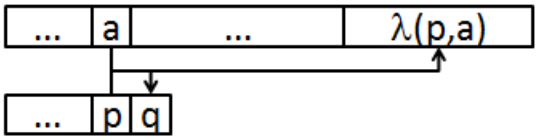
OBRÁZOK 1. D0L TAG

Najjednoduchším modelom v hierarchii je D0L TAG (obr. 1). Ide o metódu, kde máme nejaký nevymazávajúci homomorfizmus h a počiatočný symbol a. Ako naznačuje obrázok, tento model postupne číta zo vstupu znaky, pričom ich homomorfizmy zapisuje na koniec pásky, čím si pásku zároveň predlžuje. Tento systém generuje nekonečné slovo $w = a \cdot h(a) \cdot h^2(a) \cdot h^3(a) \ldots$ D0L TAG spomenul Post už v roku 1920, viac v [M67]. Tento systém dokáže generovať napr. slovo `FIBONACCI`.
$f_0 = a; f_1 = ab; f_{i+1} = f_i f_{i-1}; \forall i \in N : i > 1$
`FIBONACCI`$=\lim_{i \to \infty} f_i$
Dokážeme ho generovať pomocou homomorfizmu h: $a \to ab; \quad b \to a$.



Obrázok 2. CD0L TAG

Jedným z možných rozšírení systému D0L TAG je CD0L TAG (obr. 2). Ide o metódu, kde generujeme slovo na pomocnej (spodnej) páske presne tak, ako by išlo o D0L TAG, pričom ale toto slovo je ešte prekódované (pomocou nevymazávajúceho homomorfizmu c) na výstupnú (vrchnú) pásku. Tento model nám generuje nekonečné slovo $w = c(a{\cdot}h(a){\cdot}h^2(a){\cdot}h^3(a)\ldots) = c(a){\cdot}c(h(a)){\cdot}c(h^2(a)){\cdot}c(h^3(a))\ldots)$. Pomocou tohto stroja vieme generovať napr. slovo
PER=$Sa\#aa\#aa\#aaa\#aaa\#aaaa\#\ldots$, no D0L TAG už naň nestačí [K06].

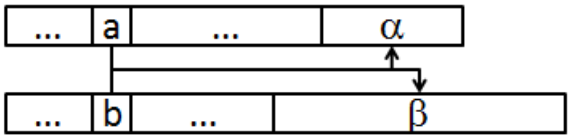


Obrázok 3. DGSM TAG

Druhým z možných rozšírení systému D0L TAG je DGSM TAG (obr. 3). Tu nahrádzame homomorfizmus deterministickým strojom GSM (generalized sequential machine) [M94], pre ktorý budeme používať funkciu $\lambda(p, a)$ vyjadrujúcu výstup tohto stroja pre vstupný stav p a vstupný symbol a. Tento systém potrebuje mať dve pásky. Jednu pomocnú (spodnú), na ktorej si evidujeme stavy pre GSM a druhú výstupnú (vrchnú), kam sa generuje slovo. Vždy keď stroj GSM vráti nejaký výstup, zapíšeme si ho na koniec výstupnej pásky, stav, v ktorom skončil, zapíšeme na koniec pomocnej (stavovej) pásky a prečítame ďalšiu dvojicu symbolov.

Tento stroj dokáže generovať napr. slovo RBIN=$S0\#1\#01\#11\#001\#\ldots$, s ktorým si CD0L TAG neporadí [K06]. Opačne ale slovo SQUARE=$Sa\#a^4\#a^9\#a^{16}\#\ldots$ sa dá generovať pomocou CD0L TAG, no nie pomocou DGSM TAG [K06].
Slovami generovanými pomocou DGSM TAG systému (skrátene DGSM slová) sa zaoberá práve táto diplomová práca. Preto si tu uvedieme iba ťažiskový otvorený problém, a to, či DGSM slová patria do množiny $GSPACE(\log n)$.

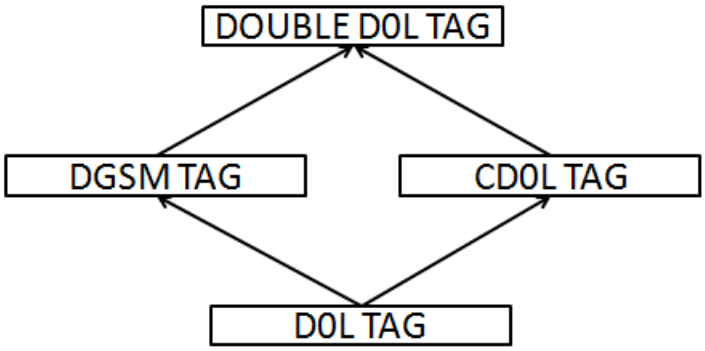
Ďalším systémom, ktorý zároveň tieto vetvy spája, je DOUBLE D0L TAG (obr. 4). Je to systém, ktorý si vieme predstaviť ako stroj, ktorý má dve pásky a prechodovú funkciu z $\sum_1 \times \sum_2 \to \sum_1^+ \times \sum_2^+$. Táto funkcia prečíta z každej pásky jeden symbol a na koniec každej pásky zapíše príslušné slovo. Výstupom stroja je však iba slovo z jednej pásky. Obdobne sa dá pokračovať pre stroje s viac



Obrázok 4. DOUBLE D0L TAG

páskami, no tieto konštrukcie už presahujú rámec tejto práce.

Všetky tieto systémy majú rôznu generatívnu silu, ktorá je zhrnutá v obr. 5.



Obrázok 5. Porovnanie generatívnej sily

Časť 1

Úvod do problematiky

KAPITOLA 2

DGSM

V tejto kapitole sa oboznámime s formálnym aparátom, ktorý sme si už čiastočne opísali a ktorý bude elementárnym prvkom na nasledujúcich stranách.

Definície

V literatúre sa tieto stroje spomínajú obdobne, ako sme ich spomenuli v kapitole 1. To, čo čitateľa zrejme zarazilo, bol stroj GSM, keďže sme si ho do tejto chvíle neopísali. Nie je to však vôbec potrebné, keďže GSM so vstupom jedného písmena a jedného stavu sa dá pokojne nahradiť riešením, kde ku každému stavu vytvoríme dva homomorfizmy také, že ak na páskach prečítame symboly a a p, potom na výstupnú (vrchnú) pásku zapíšeme $p_1(a)$ a na pomocnú (spodnú) pásku $p_2(a)$. Keďže táto práca pojednáva o DGSM slovách, je nevyhnutné si riadne zadefinovať stroj generujúci tieto slová, teda stroj DGSM, čo v našom prípade nebude deterministický GSM, ale stroj generujúci DGSM slová.

Definícia 22. *DGSM A je pätica $(\sum_1, \sum_2, H, \delta_1, \delta_2)$, kde $\sum_1$ je výstupná abeceda, $\sum_2$ je množina stavových symbolov, δ_1 je počiatočný výstupný symbol z abecedy $\sum_1$, δ_2 počiatočný stavový symbol z abecedy $\sum_2$ a $H \subseteq \sum_1 \times \sum_2 \times \sum_1^+ \times \sum_2$ je deterministická prechodová funkcia (štvorica $(a, p, w, q) \in H$ bude znamenať, že ak prečítame dvojicu symbolov a a p, potom na koniec výstupnej pásky zapíšeme slovo w a na koniec pomocnej(stavovej) pásky zapíšeme symbol q).*

Definícia 23. `Konfigurácia` *stroja DGSM je štvorica (u, v, w, p), kde $|u| = |v|$, u je prečítané slovo z výstupnej pásky, v je prečítané slovo z pomocnej pásky, w je zapísaná, ale ešte neprečítaná časť na výstupnej páske a q je neprečítaný symbol na pomocnej páske (pozícia hláv je v tejto konfigurácii nasledovná: čítacia hlava na výstupnej páske číta prvý symbol slova w, čítacia hlava na pomocnej páske číta symbol p, zapisovacia hlava na výstupnej resp. pomocnej páske je bezprostredne vpravo od w resp. p).*

Definícia 24. `Krok výpočtu` *DGSM je relácia $\mapsto$ definovaná pomocou konfigurácií takto: $(u, v, xw, p) \mapsto (ux, vp, wt, q) \Leftrightarrow (x, p, t, q) \in H$.*

Definícia 25. `Stav výpočtu` *DGSM je dvojica symbolov, ktoré budú čítané pri nasledujúcom kroku výpočtu. Ak sa stroj nachádza v konfigurácií (u, v, xw, p), potom stavom výpočtu je dvojica $\begin{pmatrix} x \\ p \end{pmatrix} \in \sum_1 \times \sum_2$.*

DEFINÍCIA 26. `Poradie/pozícia` *dvojice symbolov na páskach DGSM je počet krokov stroja, než sa práve táto dvojica stane stavom výpočtu. Počiatočná dvojica symbolov má teda pozíciu 0.*

DEFINÍCIA 27. `Pamäť` *stroja DGSM je neprečítaná časť výstupnej pásky. Ak sa stroj nachádza v konfigurácii* (u, v, w, p)*, potom pamäť je slovo* w*.*

DEFINÍCIA 28. `Stav` *stroja DGSM je neprečítaný symbol na pomocnej páske. Ak sa stroj nachádza v konfigurácii* (u, v, w, p)*, potom stav stroja DGSM je symbol* p*.*

DEFINÍCIA 29. `Výstup` *stroja DGSM* A *je nekonečné slovo* v*, ktoré daný stroj generuje. Výstup stroja reprezentujeme funkciou* W*:* $W(A) = v$*.*

DEFINÍCIA 30. `Jazyk` *množiny strojov je množina všetkých slov, ktoré dokážu byť vygenerované nejakým strojom z tejto množiny. Jazyk množiny strojov reprezentujeme funkciou* $L : L(M) = \{v | \exists A \in M : W(A) = v\}$*.*

Množstvo definícií často človeka mätie, preto si urobme drobnú rekapituláciu, ale menej formálne. Stroj DGSM si vieme predstaviť ako dve pásky, pričom každá páska má samostatnú zapisovaciu a samostatnú čítaciu hlavu. Všetky tieto hlavy, hneď ako niečo prečítajú resp. zapíšu, posunú sa na nasledujúce políčko. Čítacie hlavy v každom kroku prečítaju po jednom znaku, idú teda spolu, a zapisovacie zapisujú na koniec pásky vždy aspoň jeden znak. DGSM má ešte jedno drobné obmedzenie, zapisovacia hlava pre pomocnú pásku v jednom kroku zapíše vždy práve jeden symbol. To znamená, že koniec pomocnej pásky je vždy o krok pred čítacími hlavami. Preto sa táto páska dá chápať ako ekvivalent k stavom, z čoho vychádza aj alternatívny názov pre túto pomocnú pásku: stavová.

KAPITOLA 3

Obmedzenia

Keďže v ďalšom texte sa budeme zaoberať špecifickejšími problémami, bude vhodnejšie, aby sme si ujednotili a špecifikovali ďalšiu terminológiu, ktorú budeme používať.

Definície

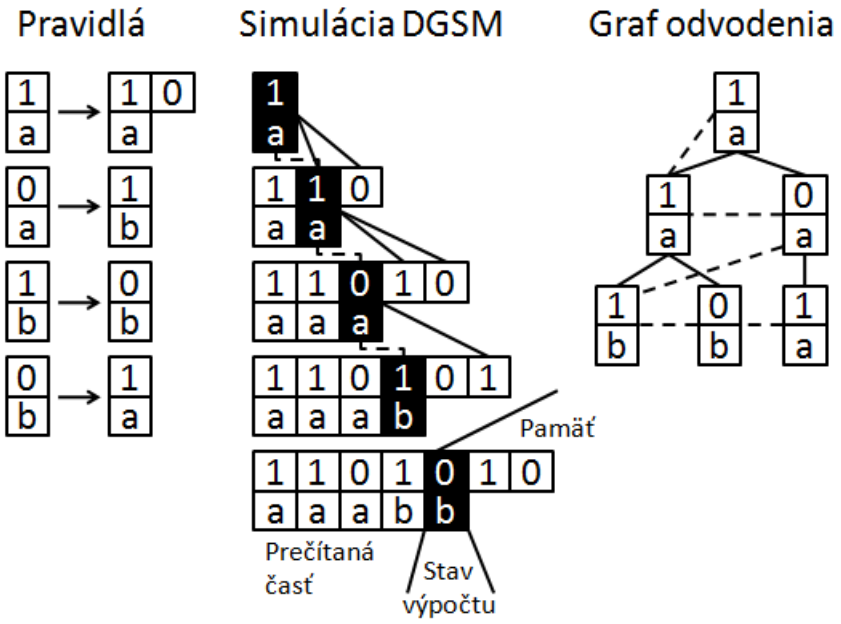
Najprv si zadefinujeme pojem strom odvodenia a graf odvodenia pre DGSM. Graf je presnejšie vyjadrenie, no ako bude zrejmé aj z neskoršieho príkladu, tento graf má nepopierateľnú stromovú podštruktúru (obr. 6 - plné hrany) a práve jej vlastnosti budeme často využívať. Pre jednoduchšie pochopenie sa preto sem-tam uchýlime k pojmu strom odvodenia, čím bude zrejmé, že využívame vlastnosti tejto podštruktúry, avšak stále budeme mať na pamäti, že sa jedná o graf odvodenia s jeho všetkými väzbami.

Definícia 31. `Triviálny` *DGSM A je každý stroj, pre ktorý existuje nejaký Turingov stroj, ktorý generuje rovnaké nekonečné slovo $W(A)$ s konštantnou pamäťou, t. j. $W(A) \in GSPACE(O(1))$.*

Definícia 32. `Netriviálny` *DGSM A je každý DGSM A, ktorý nie je triviálny.*

Definícia 33. `Stromom odvodenia` *DGSM $A = (\sum_1, \sum_2, H, \delta_1, \delta_2)$ je taký graf (strom) T, ktorého vrcholy si označíme ako $\begin{pmatrix} a \\ p \\ i \end{pmatrix} \in \sum_1 \times \sum_2 \times N$. Teda a je čítaný symbol na výstupnej páske, p je čítaný symbol na pomocnej (stavovej) páske a i je pozícia resp. poradie tejto dvojice na páskach DGSM. Hrana spája všetky tie vrcholy $\begin{pmatrix} a \\ p \\ i \end{pmatrix}$ a $\begin{pmatrix} b \\ q \\ j \end{pmatrix}$, pre ktoré sa pri simulácii stroja A použije pravidlo $(a, p, ubv, r) \in H$, pričom sa číta z i-tej pozície (oboch pások) a zapisuje sa na j-tu pozíciu výstupnej pásky.*

Definícia 34. `Grafom odvodenia` *DGSM $A = (\sum_1, \sum_2, H, \delta_1, \delta_2)$ je taký graf G, ktorého vrcholy si označíme ako $\begin{pmatrix} a \\ p \\ i \end{pmatrix} \in \sum_1 \times \sum_2 \times N$. Teda a je čítaný symbol na výstupnej páske, p je čítaný symbol na pomocnej páske a i je poradie tejto dvojice na páskach DGSM. Hrana spája všetky tie vrcholy $\begin{pmatrix} a \\ p \\ i \end{pmatrix}$ a $\begin{pmatrix} b \\ q \\ j \end{pmatrix}$,*



Obrázok 6. DGSM

pre ktoré sa pri simulácii stroja A použije jedno z pravidiel:
$(a, p, ubv, r) \in H$*, pričom sa číta z i-tej pozície a zapisuje sa na j-tu pozíciu výstupnej pásky;*
$(a, p, u, q) \in H$*, pričom sa číta z i-tej pozície a zapisuje sa na j-tu pozíciu pomocnej (stavovej) pásky. Tu platí, že* $j = i + 1$.

Čiže strom odvodenia zachytáva použitie pravidiel voči výstupnej páske (vytvára graf - strom), kým graf odvodenia voči obom páskam. Lepšie sa to dá vidieť na príklade (obr. 6), ale tam sa pre jednoduchosť dopustíme praktickej nepresnosti a poradové čísla z grafu a jeho vrcholov vynecháme. Miesto toho pri kreslení grafu radšej odlíšime hrany, aby bolo jasné, na základe ktorého pravidla vznikli. Toto nám jednoznačne určí poradie vrcholov na páskach a zároveň ukáže aj samotnú stromovú štruktúru.

Príklad 1. *(obr. 6)*
Pamäť DGSM je napravo od čiernej (čítanej) dvojice symbolov.
Stav DGSM je čierna dvojica symbolov.
Plné hrany spájajú tie vrcholy, pri ktorých sa uplatnilo nejaké pravidlo, a vzťah je na základe zápisu na výstupnej (vrchnej) páske.
Prerušované hrany spájajú tie vrcholy, pri ktorých sa uplatnilo nejaké pravidlo, a vzťah je na základe zápisu na pomocnej (spodnej) páske.

Ďalej si zadefinujeme doplnkové obmedzenia typu DGSM[X=x].

Definícia 35. `Abecedné obmedzenie` *[*A=x*] znamená, že stroj nepoužíva pre výstupnú pásku viac ako* x *rôznych symbolov.*
Formálne: pre každý stroj D *z DGSM[*A=x*] taký, že* $D = (\sum_1, \sum_2, H, \delta_1, \delta_2)$*, platí* $|\sum_1| \leq x$.

DEFINÍCIA 36. `Stavové obmedzenie` *[S=x] znamená, že stroj nepoužíva pre stavovú pásku viac ako x rôznych symbolov.*
Formálne: pre každý stroj D z DGSM[S=x] taký, že $D = (\sum_1, \sum_2, H, \delta_1, \delta_2)$, *platí* $|\sum_2| \leq x$.

DEFINÍCIA 37. `Dĺžkové obmedzenie` *[R=x] znamená, že stroj nezapisuje v žiadnom kroku odvodenia viac ako x symbolov na výstupnú pásku.*
Formálne: pre každý stroj D z DGSM[R=x] taký, že $D = (\sum_1, \sum_2, H, \delta_1, \delta_2)$, *platí* $\forall(a, p, w, q) \in H : |w| \leq x$.

Rovnako je možné tieto obmedzenia spájať, napr. DGSM[A=2][S=3][R=4] D. V takomto prípade pôjde o DGSM D taký, ktorý bude generovať slovo zložené z maximálne dvoch rôznych znakov, bude používať maximálne tri rôzne stavové symboly a v jednom kroku zapíše maximálne štyri znaky na výstupnú pásku.

Vety

Skôr než sa pustíme do prvých dôkazov, uvedomme si, ako sa správa DGSM voči svojmu grafu. Pokiaľ je stroj DGSM v nejakom stave výpočtu, ktorému je asociovaný nejaký vrchol v strome v hĺbke h, potom pamäť stroja DGSM je popísaná týmto vrcholom, všetkými vrcholmi z tejto hĺbky napravo od neho (obr. 6) a vrcholmi z hĺbky $h + 1$ naľavo od jeho synov. Práve z tejto úvahy sa dá ľahko usudzovať, že pre simulovanie DGSM nám stačí asymptoticky taká pamäť, aká je šírka stromu odvodenia pre daný DGSM a pre daný prefix nekonečného slova. Na simulovanie nám postačí stroj, ktorý si bude pamätať celú pamäť stroja DGSM ako frontu.

Rovnako nám pre simulovanie DGSM stačí asymptoticky taká veľká pamäť, aká je výška stromu odvodenia k danému DGSM pre daný prefix nekonečného slova. Je potrebné si všimnúť, že na vygenerovanie ľubovoľného vrchola v v strome odvodenia potrebujeme poznať rodičovský vrchol vrchola v, jeho poradie spomedzi jeho bratov a vrchol, ktorého pozícia na páskach DGSM je o 1 menšia. Na prvé dva typy údajov potrebujeme pamäť takej veľkosti, ako je výška stromu. A pokiaľ vrcholy generujeme v poradí, v akom sú na páskach, potom predchodca je vždy obsiahnutý v pamäti. Takže kedykoľvek potrebujeme vygenerovať nejaký vrchol, pozrieme sa na rodiča, jeho syna a poradie tohto syna, ktoré si pamätáme a vieme vygenerovať nasledujúceho brata. Ak neexistuje nasledujúci brat, potom nový vrchol nemá spoločného rodiča a najprv vygenerujeme rodiča... Takouto rekurziou vieme postupne vygenerovať všetky vrcholy grafu, a teda aj simulovať stroj DGSM.

TVRDENIE 1. *Pre simulovanie výpočtu DGSM pomocou Turingovho stroja mu musí stačiť pamäť asymptoticky rovná menšej z hodnôt výška/šírka stromu odvodenia.*

Rovnako jednoducho sa dá dokázať:

TVRDENIE 2. *Pre každé slovo z DGSM[A=1] existuje Turingov stroj, ktorý toto slovo generuje s použitím konštantnej pamäte.*

Ide vlastne o nekonečné slovo, ktoré je skonštruované iba z jedného znaku, teda na jeho generovanie postačí aj konečný automat.

Tvrdenie 3. *Pre každé slovo z DGSM[R=1] existuje Turingov stroj, ktorý toto slovo generuje s použitím konštantnej pamäte.*

Tu si stačí uvedomiť, že pamäť stroja DGSM obsahuje vždy práve jeden znak (ak uvážime, že čítanie aj zápis je jeden a ten istý krok), resp. šírka stromu odvodenia je vždy 1. Na simulovanie takéhoto stroja nám taktiež postačí konečný automat.

Tvrdenie 4. *(HKL94) Pre každé slovo z DGSM[S=1] existuje Turingov stroj, ktorý toto slovo generuje s použitím logaritmickej pamäte.*

Tieto stroje generujú tie isté slová ako systém D0L TAG, resp. systém D0L TAG získame z takéhoto DGSM vynechaním stavovej pásky (kde by bol aj tak vždy jeden a ten istý symbol). D0L TAG systém ale generuje iba slová z $GSPACE(\log n)$.

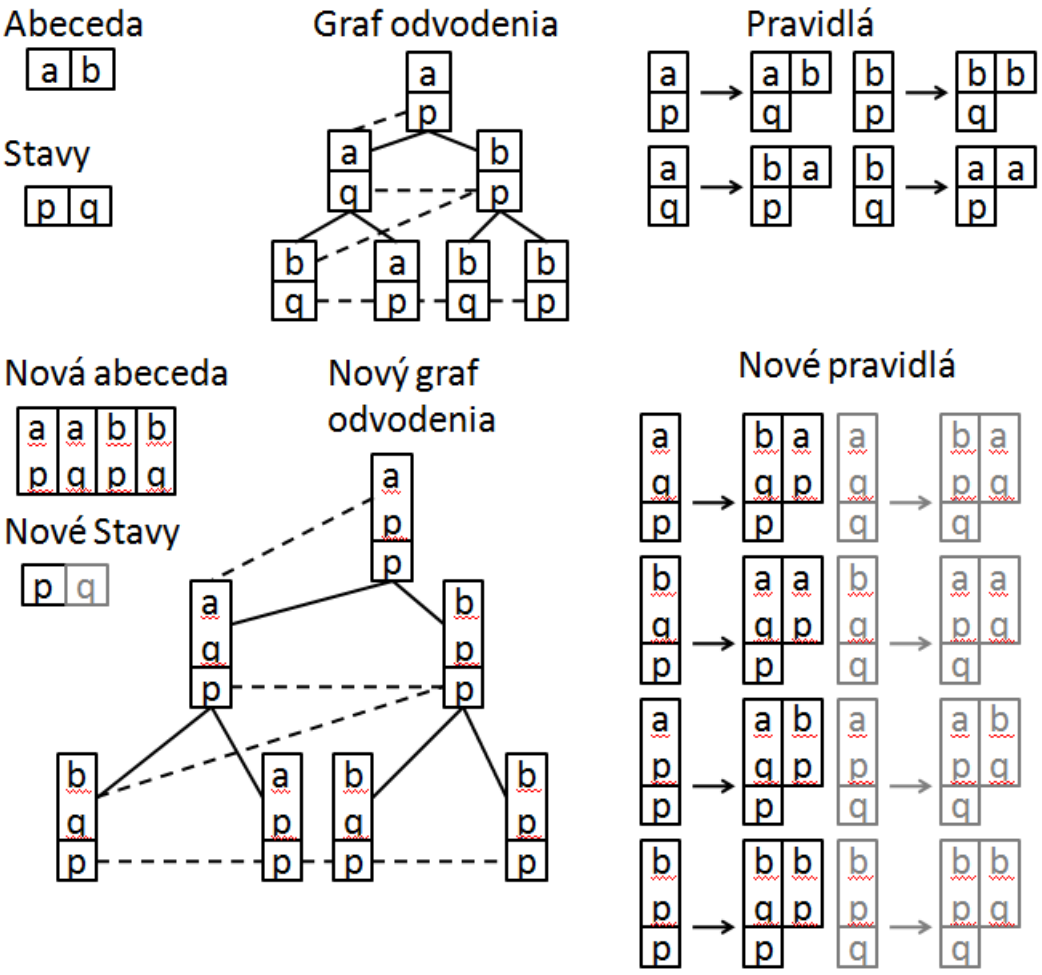
Mnohé grafy odvodenia skonštruované k náhodným DGSM strojom (desiatky tísíce strojov) vždy spadajú do jednej z troch kategórií:

- strom je veľmi úzky a vysoký;
- strom je značne košatý a nízky;
- šírka aj výška stromu sú približne rovnaké, resp. sa zdá, že rastú asymptoticky rovnako rýchlo.

Táto kategorizácia nemá žiadnu oporu vo formálnom aparáte, je skôr podnetom k hypotéze: $L(DGSM) \subseteq GSPACE(\sqrt{n})$.
Cesta, ktorou by sa dalo uberať pri dôkaze tejto hypotézy, je pre prípady, kde to pomôže, použiť tvrdenie o výške/šírke stromu. Pre prípady, kde to nepomôže, sa môžeme pokúšať dokázať, že pre všetky stromy odvodenia všetkých prefixov daného DGSM platí vzťah: $n+n \geq v \cdot s \geq n$, kde n je počet vrcholov, s je šírka stromu a v je výška stromu. Dôsledkom platnosti tohto vzťahu by bola platnosť spomenutého tvrdenia. Pričom pri dôkaze tohto vzťahu by sme sa mohli pokúšať skonštruovať dva totožné grafy odvodenia, jeden z nich horizontálne prevrátiť a položiť vedľa druhého. Takto dostaneme útvar, ktorý má zachovanú výšku a zdvojnásobí sa počet vrcholov, ale podstatné je, či sa zmení celková šírka maximálne o nejakú predom stanovenú konštantu.

Túto myšlienku ďalej rozvádzať nebudeme, keďže naše ambície a ciele sa uberajú iným smerom a rovnako nie je známe, či by išlo vôbec o nejaký zmysluplný posun v tejto téme. Miesto toho sa ešte zastavíme pri grafoch a ich ďalších vlastnostiach.

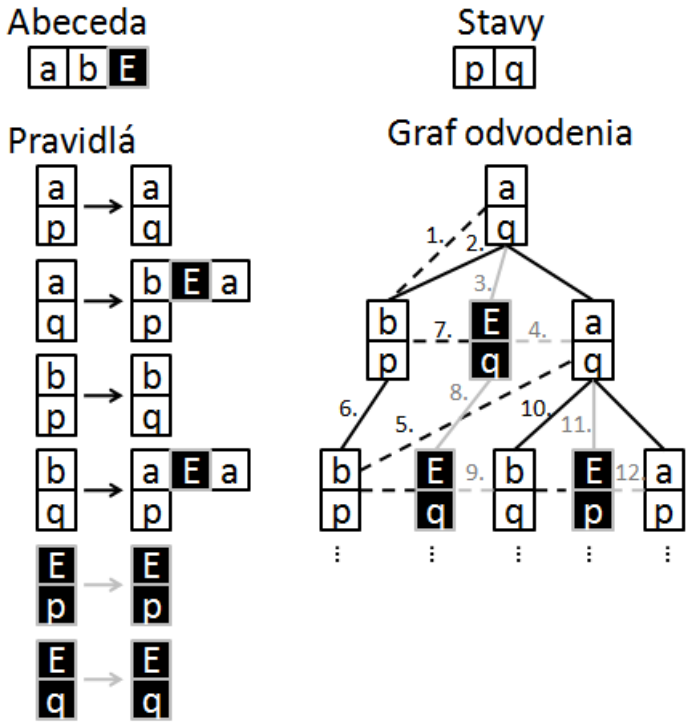
Dôvodov, prečo sledovať grafové štruktúry strojov DGSM, je hneď niekoľko, napr. že slová s „rôznou zložitosťou“ môžu mať identické grafy odvodenia. Ako príklad sa dá uviesť ľubovoľný stroj DGSM $A = (\sum_1, \sum_2, H, \delta_1, \delta_2)$, ku ktorému



Obrázok 7. Pohltenie stavov

vieme skonštruovať taký DGSM $B = \left(\sum_1 \times \sum_2, \sum_2, H_B, \begin{pmatrix} \delta_1 \\ \delta_2 \end{pmatrix}, \delta_2\right)$, že výstupom bude dvojúrovňové slovo, ktoré bude mať na prvej úrovni pôvodné slovo $W(A)$ a na druhej všetky stavy presne tak, ako by išli pri odvodzovaní pomocou stroja A. Teda výstupné slovo bude zložené zo stavov odvodenia predošlého DGSM. Ako stav budeme v novom stroji používať vždy symbol prislúchajúci k poslednému zapísanému znaku predchádzajúceho DGSM. Takto vždy vieme, aké pravidlo by použil predošlý stroj, a aby sme vedeli zapísať korektný výstup, využijeme tam práve posledný stav z predošlého stroja a ďalší upravíme na hodnotu po tomto kroku.

Takto skonštruované slovo môže mať menej stavov ako pôvodný stroj, generuje komplexnejší výstup, ale aj napriek tomu sú ich grafy identické. Dokonca si neskôr ukážeme, že tieto stroje sú ekvivalentné.



Obrázok 8. Eulerov ťah

Veta 1. *Ku každému DGSM A existuje asociovaný DGSM B taký, že nekonečný strom odvodenia vytvorí graf (pre nekonečné slovo), pre ktorý existuje Eulerov ťah, pričom B generuje také nekonečné slovo v, ktoré po vymazaní všetkých výskytov znaku E vytvorí ekvivalentné slovo k pôvodnemu. Zároveň sa v tomto slove nenachádzaju žiadne dva znaky E vedľa seba, a tak dĺžka slova po redukcii zostane väčšia alebo rovná polovici dĺžky slova pred redukciou.*

Dôkaz. Nech máme daný DGSM $A = (\sum_1, \sum_2, H, \delta_1, \delta_2)$ a $E \notin \sum_1$, potom si skonštruujeme nový DGSM $B = (\sum_1 \cup \{E\}, \sum_2, H_B, \delta_1, \delta_2)$. Už tu je zrejmé, že to podstatné bude schované výlučne v prechodovej funkcií (obr. 8).

$H_B = \{(a,p,w,q) | (a,p,w,q) \in H; |w| \text{ je nepárne}\} \cup$
$\cup \{(a,p,cEw,q) | (a,p,cw,q) \in H; c \in \sum_1; |cw| \text{ je párne}\} \cup$
$\cup \{(E,p,E,p) | p \in \sum_2\}$

Čiže sme zmenili iba tie pravidlá, ktorých dĺžka pravej strany je párna. Do takéhoto pravidla sme pridali znak E. Podstatné je, že pri každom pravidle je symbol E obklopený nejakými pôvodnými symbolmi. Z tohto dôvodu ani pri ďalšom prepisovaní tohto symbolu nemôže nastať situácia, že budú dva symboly E vedľa seba. Toto už samo o sebe spĺňa obe podmienky stanovené vo vete. To, čo chýba, je už len existencia Eulerovho ťahu, ale tá je daná faktom, že všetky pravidlá majú nepárny počet symbolov na pravej strane pravidiel. Uvedené ale v koncovom grafe spôsobí, že každý vrchol bude incidentný s párnym počtom hrán. To neplatí

pre poslednú úroveň grafu, no tú uvažovať nemusíme, pokiaľ uvažujeme nekonečné slovo. □

Očividný dôsledok tejto vety je ten, že všetky tieto „Euler-friendly“ DGSM generujú slová z $GSPACE(f)$ práve vtedy, keď všetky slová generované ľubovoľným DGSM sú taktiež z $GSPACE(f)$. Dokonca sa veta dá rozšíriť tak, že existuje taký Eulerov ťah, ktorý bude graf prehľadávať tak, aby sme vrcholy objavovali v poradí, v akom sú ich ekvivalenty na páske. Čo v spojení napr. s peblovaním môže viesť ku konkrétnejším výsledkom. Týmto zanecháme grafové vlastnosti ako otvorenú množinu problémov a v ďalšom texte sa budeme venovať porovnaniu DGSM a logaritmickej pamäte.

Časť 2

Jednoduchšie modely

KAPITOLA 4

Λ-DGSM

V nasledujúcich dvoch kapitolách sa budeme snažiť porovnať DGSM a logaritmickú pamäť, pričom začneme s jednoduchšími modelmi a postupne budeme zväčšovať ich generatívnu silu.

Definície

DEFINÍCIA 38.
Množina $\Lambda = \{0^{m_1}10^{m_2}10^{m_3}1\ldots10^{m_n} | n \in N; \forall i = 1, 2, 3, \ldots, n-2 : m_i \leq m_{i+1}\}$.

Voľne povedané ide o množinu slov, v ktorých sú každé dve susedné jednotky oddelené blokom núl a veľkosť týchto blokov má neklesajúci charakter s výnimkou posledného bloku núl.

DEFINÍCIA 39. *Λ-DGSM A je každý DGSM A, ktorý generuje nekonečné slovo také, že všetky jeho prefixy sú z množiny* Λ.

Vety

LEMA 1. *Ak pre daný Λ-DGSM A existuje konštanta l taká, že všetky bloky núl v slove* $W(A)$ *budú maximálne dĺžky l, potom A je triviálny DGSM. T. j. ak existuje nejaký blok núl, od ktorého nie je žiaden ďalší blok už väčší, potom nám stačí na generovanie tohto slova Turingov stroj s pamäťou* $O(1)$.

DÔKAZ. Nech sú splnené predpoklady lemy, teda existuje nejaký blok núl v taký, že žiaden ďalší blok núl už nie je väčší. Z definície množiny Λ je zrejmé, že žiaden nasledujúci blok nie je ani menší. Čiže slovo $W(A) = u1v1v1v...$, kde u je nejaké úvodné slovo. Teda hľadaný Turingov stroj bude fungovať tak, že priamo vygeneruje slovo u a ďalej bude cyklicky generovať $1v$. Takémuto Turingovmu stroju ale stačí $O(1)$ pamäť. □

DÔSLEDOK 1. *Pre každý netriviálny stroj z Λ-DGSM sa počet núl medzi dvoma 1 postupne zväčšuje.*

Tu sa určite dá uvažovať, ako pomaly sa môžu zväčšovať bloky núl pre slová generované pomocou Λ-DGSM. Riešenie problému príde, až bude správny čas, ale zopár bodov na zamyslenie si predostrieme už teraz. Ak sa pozrieme na konfiguráciu DGSM, na jeho pamäť, tak čo sa stane, keď prečíta celú pamäť? Bude v tom istom stave alebo v inom? Bude mať rovnakú pamäť alebo dlhšiu? Čo sa stane, ak sa to zopakuje dvakrát, trikrát...

LEMA 2. *Pre každý netriviálny Λ-DGSM A existujú konštanty k a l také, že A po l krokoch už nikdy nebude mať v pamäti viac ako k jednotiek.*

DÔKAZ. Zoberme si najdlhšiu pravú stranu z množiny pravidiel nášho DGSM A a nech táto pravá strana vygeneruje x symbolov. Ďalej nech náš DGSM A má y stavov. Dôsledkom netriviálnosti (lema 1) je, že v slove generovanom strojom A existuje blok núl dlhší ako $(x \cdot y + y + 1)$, teda $W(A) = u10^{(x \cdot y+y+1)} \dots$ Takže nech sa DGSM A nachádza v nejakej konfigurácií $(u, v, 1w, p)$, čo ale znamená, že po prečítaní jednotky za slovom u máme maximálne y krokov, aby sme zapísali na koniec výstupnej pásky maximálne jednu novú jednotku. Viac ich A zapísať nemôže, lebo by ich nedokázal oddeliť dostatočným počtom núl, aby vygenerovaný prefix patril do množiny Λ. Ale po absolvovaní spomínaných y krokov sa A začne už nevyhnutne cykliť a jediné, čo počas tohto cyklu môže A robiť, je zapisovať nuly až do prečítania ďalšej jednotky. V opačnom prípade by A počas každého cyklu zapísal aspoň 1 novú jednotku, no žiadne dve takéto jednotky by neboli oddelené dostatočne dlhým blokom núl, aby tento prefix patril do Λ. Čím sme ukázali, že počet jednotiek v pamäti už neprekročí počet jednotiek v slove $1w$, tento počet bude konštanta k. Konštanta l bude dĺžka slova u. □

VETA 2. *Každý Λ-DGSM A sa dá simulovať nejakým Turingovym strojom B s $O(\log n)$ pamäťou.*

DÔKAZ. Samostatne pre triviálne a samostatne pre netriviálne DGSM.

- Nech je A triviálny: Z definície triviálnosti platí, že existuje Turingov stroj B s pamäťou $O(1)$ simulujúci A.
- Nech je A netriviálny: Podľa lemy 2 existujú vhodné konštanty k a l také, že náš DGSM A po l krokoch už nikdy nebude mať v pamäti viac ako k jednotiek. Teraz nám na simulovanie výpočtu stačí skonštruovať taký Turingov stroj B, ktorý prvých l krokov odsimuluje priamo a následne bude mať DGSM A v pamäti maximálne $k + 1$ úsekov núl oddelených jednotkou. Takže na simulovanie DGSM A nám stačí evidovať počty núl pre každú z $k + 1$ skupín, preto budeme potrebovať pamäť veľkosti $O((k + 1) \cdot \log n) = O(\log n)$.

□

VETA 3. *Existuje taký TS A s pamäťou veľkosti $O(\log n)$ generujúci iba prefixové slová z množiny Λ, ku ktorému neexistuje ekvivalentný Λ-DGSM B.*

DÔKAZ. Takýto Turingov stroj je napr. stroj generujúci 1 každých $(2^n)^2$ krokov, teda $10^{2^{0^2}}10^{2^{1^2}}10^{2^{2^2}}10^{2^{3^2}}... = 10^{1}10^{4}10^{16}10^{256}...$ Konštrukcia takéhoto Turingovho stroja je pomerne priamočiara záležitosť a my sa zameriame len na neexistenciu DGSM.
Sporom: nech existuje DGSM B generujúci takéto slová a nech má k stavov. Teraz si uvedomme, ako sa bude takýto stroj správať. Jeho pamäť môže byť maximálne asymptoticky rovnako veľká ako celá prečítaná časť slova. V každom kroku totiž DGSM prečíta jeden symbol z pamäte a do pamäte pridá maximálne l nových symbolov, kde l je dĺžka najdlhšieho slova, ktoré vie zapísať pomocou svojich pravidiel.

No slovo, ktoré chce vygenerovať má za následok to, že nastane moment, keď B bude mať vo svojej pamäti iba samé nuly. Dokonca ich tam bude musieť mať aj po ďalších k krokoch, keďže počas tých vie vygenerovať maximálne $k \cdot l$ nových symbolov, čo ale nepostačuje, aby dostal do pamäte symbol 1. No po týchto k krokoch sa už začne cykliť a z cyklu sa už ani nikdy nedostane, keďže jednotku nikdy nevygeneruje a teda ani neprečíta. □

Idea použitá v tomto dôkaze sa dá jednoducho aplikovať a rozšíriť na ďalšie tvrdenie.

Dôsledok 2. *Pre každý Λ-DGSM A existuje konštanta k taká, že pre všetky podslová w slova $W(A)$ také, že $w = 10^i10^j1$, platí: $j \leq i \cdot k$.*

Toto tvrdenie je ekvivalentné tvrdeniu, že maximálny možný rast počtu núl v postupnosti blokov zodpovedá geometrickému radu. Ak nechceme, aby sa stroj začal cykliť, tak nemôžeme vygenerovať viac núl ako nejaký k-násobok dĺžky celého prečítaného slova. Naopak najmenší možný rast núl, pokiaľ uvažujeme netriviálne prípady, je vystihnutý v nasledujucej vete.

Veta 4. *Pre každý netriviálny Λ-DGSM A existuje konštanta $k \in N$ taká, že pre všetky podslová w nekonečného slova $W(A)$ také, že $w = 10^{i_1}10^{i_2}10^{i_3}1...10^{i_k}1$, platí: $i_1 + 1 \leq i_k$.*

Dôkaz. Možno konštatovať, že toto tvrdenie je ekvivalentné tvrdeniu, že minimálny možný rast počtu núl v postupnosti blokov zodpovedá lineárne rastúcemu radu. Nech $A = (\sum_1, \sum_2, H, \delta_1, \delta_2)$. S využitím lemy 2 je zrejmé, že existuje taký počet krokov m, po ktorom nám vznikne v pamäti stroja A n jednotiek a viac ich tam už nikdy nebude. Teda rovnako tam bude limitovaný aj počet blokov núl. Ak budeme predpokladať, že pamäť stroja začína symbolom 1 a budeme ignorovať posledný blok núl, lebo nemusí byť dokončený, potom sa môžeme pozrieť, či sú všetky ostatné rovnako dlhé. Ak nie, potom hľadaný blok je jeden z nasledovných n, teda toto by bola zároveň aj hľadaná konštanta pre tento prípad. Opačne nech sú všetky rovnako dlhé, potom zas nastáva prípad, keď náš stroj bude generovať stále rovnako veľké úseky, ale po prečítaní $|\sum_2|$ blokov už musí nastať zmena, ináč sa začne cykliť nad blokmi a už by nikdy žiaden väčší blok nevygeneroval. Teda bol by triviálny. Preto hľadaná konštanta je maximum z n, $|\sum_2|$ a m. □

Lema 3 (HK97). *Pre každú funkciu $f : N \to N, f(n) = o(\log n)$ platí: $GSPACE(O(1)) = GSPACE(f)$.*

Veta 5. *Pre každé slovo z L(Λ-DGSM) existuje Turingov stroj taký, že toto slovo generuje práve s použitím konštantnej pamäte alebo práve s použitím logaritmickej pamäte.*

Dôkaz. Pre každé slovo z L(Λ-DGSM) existuje stroj $A \in$ Λ-DGSM, ktorý toto slovo generuje. Ak je stroj A triviálny, potom na základe definície triviálnosti existuje Turingov stroj, ktorý generuje slovo $W(A)$ s použitím konštantnej pamäte. Pokiaľ je stroj A netriviálny, každý Turingov stroj generujúci slovo $W(A)$ musí používať pamäť väčšiu ako konštantnú, pričom podľa lemy 3 to musí byť pamäť aspoň logaritmická. Podľa lemy 2 nám ale postačuje práve logaritmická pamäť. □

V tejto kapitole sme jasne dokázali, že Λ-DGSM stroje sú nepomerne slabšie ako Turingov stroj s pamäťou $O(\log n)$. Preto sa v ďalšom texte zameriame na všeobecnejšie stroje, konkrétne binárne DGSM.

KAPITOLA 5

Binárny DGSM

Ako ďalší medzistupeň si zvolíme stroje DGSM, ktoré generujú slová nad abecedou $\{0,1\}$. A opäť sa pokúsime ukázať, aká je ich generatívna zložitosť vzhľadom na použitú pamäť.

Definície

DEFINÍCIA 40. `Binárny DGSM` *je každý DGSM, ktorý postupne generuje prefixové slová z množiny* $\{0,1\}^+$.
Ekvivalentne stačí povedať, že daný DGSM je binárny práve vtedy, keď spĺňa obmedzenie [A=2].

DEFINÍCIA 41. `Funkciu dĺžky prekladu` *nekonečného slova w pomocou deterministického A-prekladača M budeme označovať* $f_{M(w)}(x)$ *a bude vyjadrovať, aký dlhý prefix slova w potrebujeme, aby sme jeho preložením získali slovo, ktorého dĺžka bude aspoň x resp. 0, pokiaľ taký prefix neexistuje. Formálne ide o zobrazenie z* N_0 *do* N_0, *kde platí:*
$f_{M(w)}(x) = |ua| \qquad (\exists u, v \in \sum^*, \exists a \in \sum : w = uav, |M(u)| < x \leq |M(ua)|)$
$f_{M(w)}(x) = 0 \qquad (inak).$

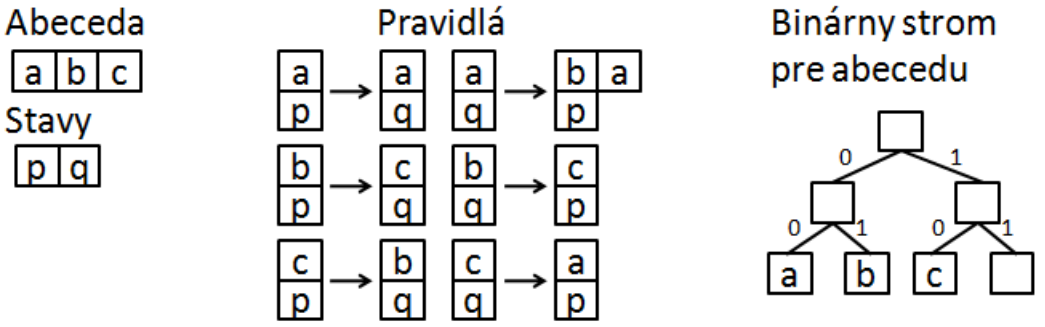
DEFINÍCIA 42. `Pamäťová redukcia` *je relácia* $\gtrsim$, *ktorá je definovaná nad nekonečnými slovami u a v takto:*
$v \gtrsim u \Leftrightarrow \exists$*(det. A-prekladač)*$M : (M(v) = u) \wedge (f_{M(v)} = \Theta(n))$.
Slovom: ...práve vtedy, keď existuje deterministický A-prekladač M taký, ktorý preloží slovo v na slovo u a zároveň funkcia dĺžky prekladu slova v A-prekladačom M je lineárna.

Obdobne sa touto reláciou vedia označovať stroje DGSM A a B:
$A \gtrsim B \Leftrightarrow W(A) \gtrsim W(B)$.
Slovom: ...práve vtedy, keď generujú slová, pre ktoré existuje táto relácia.

Je dôležité si všimnúť, že relácia $\gtrsim$ je reflexívna, tranzitívna, ale nie je symetrická.
Rovnako sa dá postrehnúť, že transformácia na „Euler-friendly“ DGSM bola práve pamäťová redukcia.

DEFINÍCIA 43. `Pamäťová ekvivalencia` *je relácia* $\approx$, *ktorá je definovaná nad nekonečnými slovami u a v takto:* $u \approx v \Leftrightarrow u \gtrsim v \wedge u \lesssim v$;
resp. pre DGSM A a B: $A \approx B \Leftrightarrow A \gtrsim B \wedge A \lesssim B$.

$\approx$ je už reláciou ekvivalencie.



Obrázok 9. DGSM pred binarizáciou

Teraz sa dá ukázať, že oba modely z obrázku 7 sú ekvivalentné. Jeden smer je triviálna záležitosť, iba vypustíme stavové symboly, opačne je to rovnako jednoduché, len si bude musieť A-prekladač pamätať a počítať stav z pôvodného DGSM. Toto nás však vedie k otvoreniu nového problému: či sú dva stroje DGSM pamäťovo ekvivalentné, ak majú identické grafy odvodenia.

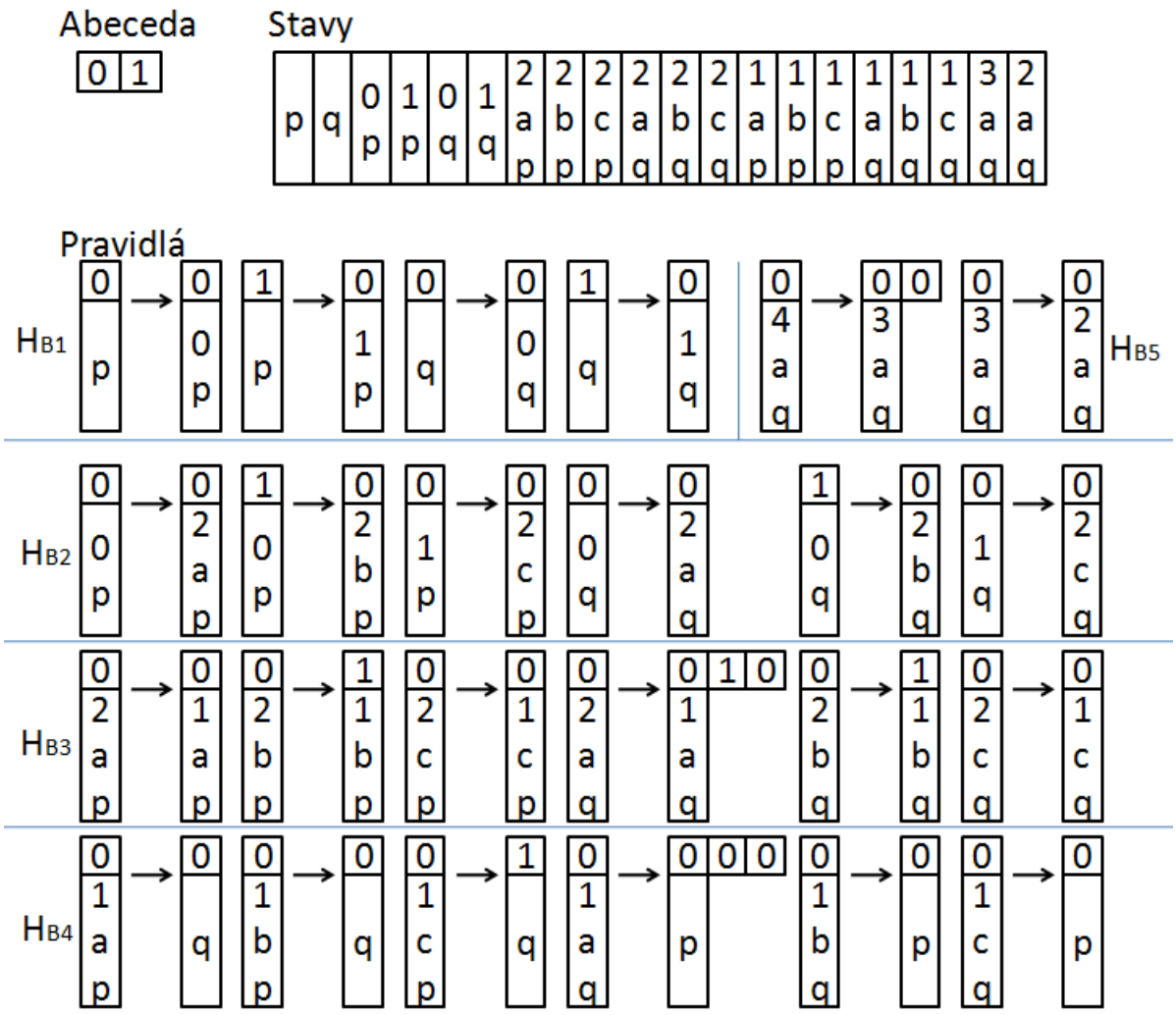
Vety

Veta 6. *Nech máme dva ľubovoľné DGSM: A taký, že $W(A) \in GSPACE(f)$ a B taký, že $A \gtrsim B$. Potom $W(B) \in GSPACE(f)$.*

Dôkaz. Nech máme splnené predpoklady tejto vety, potom na základe platnosti vzťahu $W(A) \in GSPACE(g)$ musí existovať Turingov stroj T taký, ktorý generuje slovo $W(A)$, pričom používa $O(g(n))$ pamäte. Na základe vzťahu $A \gtrsim B$ vieme, že existuje deterministický A-prekladač M taký, ktorý preloží slovo $W(A)$ na slovo $W(B)$. Preto nie je ťažké zostrojiť taký Turingov stroj, ktorý simuluje stroj T, pričom zároveň výstup tohto stroja prekladá tak, ako by to robil M. Takto skonštruovaný stroj potrebuje $O(g(f_{M(W(A))}(n)))$ pamäte. Ale na základe lineárnosti funkcie $f_{M(W(A))}$ je to asymptoticky rovné $O(g(n))$. □

Skôr než sa pustíme do ďalšieho dôkazu, vrelo odporúčam si pozrieť a pochopiť princíp použitý v nasledujúcom príklade, nasledovať bude totiž formálny opis týchto princípov, ktorých pochopenie je už podstatne ťažšie.

Príklad 2. *K danému DGSM A skonštruujeme DGSM B, ktorý bude postupne čítať symboly z abecedy 0,1 a v stave si bude skladať binárnu reprezentáciu pôvodného symbolu, ktorý by prečítal stroj A. Zatiaľ na koniec pásky zapisujeme nuly, kedže pravidlá musia byť nevymazávajúce. Až zistíme, aký symbol by prečítal stroj A, potom pri čítaní ďalších n-núl (ktoré boli zapísané práve takto aj za binárnu reprezentáciu tohoto symbolu) zapisujeme to, čo by zapísal pôvodný stroj (samozrejme binárnou reprezentáciou). Každý pôvodný symbol bude teda reprezentovaný binárnou reprezentáciou (n-symbolov) nasledovanou n nulami presne tak, ako ukazujú obrázky 9 a 10. Tu sa jedno pôvodné pravidlo nahradí štyrmi. Najprv sa použije nejaké pravidlo z H_{B1} a potom z H_{B2}. Takto už vieme, aký symbol vlastne spracovávame a v H_{B3} a H_{B4} zapisujeme binárnu reprezentáciu pravej*



Obrázok 10. DGSM po binarizácii

strany pôvodného pravidla okrem posledných n núl, ktoré by mali ísť za posledným symbolom pravidla, lebo tieto vzniknú pri čítaní ďalšieho symbolu.
Doposiaľ nespomenutou výnimkou je začiatok odvodzovania slova, kde máme v pamäti iba jeden znak pre obe pásky. Tam použijeme pravidlá z H_{B5}.
Je dobré si všimnúť, že maximálna dĺžka pravej strany pravidiel (l) ostáva asymptoticky totožná s dĺžkou pravidiel v pôvodnom stroji. Je to triviálny dôsledok tvorby nových pravidiel, kde sa generovanie celej pravej strany udeje počas čítania d núl, pričom potrebujeme vygenerovať $2 \cdot l \cdot d - d$ symbolov. Na jeden krok to vychádza $2 \cdot l - 1$. Takže nás vôbec nezaujíma, koľko symbolov spracovávame do binárneho zápisu!

Konštrukcia 1. *Nech máme DGSM $A = (\sum_1, \sum_2, H, \delta_1, \delta_2)$ a nech máme injektívnu funkciu $f : \sum_1 \to \{0,1\}^{\lceil \log |\sum_1| \rceil} \times \{0^{\lceil \log |\sum_1| \rceil}\}$ takú, ktorá každému*

znaku z $\sum_1$ priradí jeho jednoznačný binárny zápis fixnej dĺžky $\lceil \log |\sum_1| \rceil$ nasledovaný rovnakým počtom núl. Pre minimálnu dĺžku binárneho zápisu sa použilo jeho horné ohraničenie $\lceil \log |\sum_1| \rceil$, ktoré budeme označovať η. Teraz vieme skonštruovať binárny DGSM $B = (\{0,1\}, \sum_B, H_B, \delta_1^B, \delta_2^B)$, kde platí:

$$\delta_1^B = a_1 | f(\delta_1) = a_1 a_2 a_3 \dots a_\eta a_{\eta+1} \dots a_{2\cdot\eta}$$

$$\delta_2^B = \begin{pmatrix} 2\cdot\eta \\ \delta_1 \\ \delta_2 \end{pmatrix}$$

$$\sum_B = \sum_2 \cup \left\{ \begin{pmatrix} w \\ p \end{pmatrix} \;\middle|\; \begin{array}{l} w \in \{0,1\}^{\eta-1}; 1 \le |w|; \\ p \in \sum_2; \end{array} \right\} \cup$$
$$\cup \left\{ \begin{pmatrix} k \\ a \\ p \end{pmatrix} \;\middle|\; \begin{array}{l} k \in N; 1 \le k \le \eta; \\ a \in \sum_1; \\ p \in \sum_2; \end{array} \right\} \cup \left\{ \begin{pmatrix} k \\ \delta_1 \\ \delta_2 \end{pmatrix} \;\middle|\; \begin{array}{l} k \in N; \\ \eta < k \le 2\cdot\eta; \end{array} \right\}$$

$$H_B = H_{B1} \cup H_{B2} \cup H_{B3} \cup H_{B4} \cup H_{B5}$$

$$H_{B1} = \left\{ \left(a, p, 0, \begin{pmatrix} a \\ p \end{pmatrix}\right) \;\middle|\; \begin{array}{l} a \in \{0,1\}; \\ p \in \sum_2; \end{array} \right\} \cup$$
$$\cup \left\{ \left(a, \begin{pmatrix} w \\ p \end{pmatrix}, 0, \begin{pmatrix} wa \\ p \end{pmatrix}\right) \;\middle|\; \begin{array}{l} w \in \{0,1\}^{\eta}; 1 \le |w| < \eta; \\ p \in \sum_2; \end{array} \right\}$$

$$H_{B2} = \left\{ \left(a, \begin{pmatrix} w \\ p \end{pmatrix}, 0, \begin{pmatrix} \eta \\ b \\ p \end{pmatrix}\right) \;\middle|\; \begin{array}{l} \eta \in N; w \in \{0,1\}^*; \\ |w| = \eta - 1; f(b) = wa0^\eta; \\ p \in \sum_2; \end{array} \right\}$$

$$H_{B3} = \left\{ \left(0, \begin{pmatrix} k \\ a \\ p \end{pmatrix}, u_k, \begin{pmatrix} k-1 \\ a \\ p \end{pmatrix}\right) \;\middle|\; \begin{array}{l} (a,p,w,q) \in H; k \in N; 1 < k \le \eta; \\ f(w) = u_\eta u_{\eta-1} \dots u_2 u_1 0^\eta; \\ |u_\eta| = \dots = |u_2| \ge |u_1| > 0; \end{array} \right\}$$

$$H_{B4} = \left\{ \left(0, \begin{pmatrix} 1 \\ a \\ p \end{pmatrix}, u_1, q\right) \;\middle|\; \begin{array}{l} (a,p,w,q) \in H; \\ f(w) = u_\eta u_{\eta-1} \dots u_2 u_1 0^\eta; \\ |u_\eta| = \dots = |u_2| \ge |u_1| > 0; \end{array} \right\}$$

$$H_{B5} = \left\{ \left(a, \begin{pmatrix} k \\ \delta_1 \\ \delta_2 \end{pmatrix}, u_k, \begin{pmatrix} k-1 \\ \delta_1 \\ \delta_2 \end{pmatrix}\right) \;\middle|\; \begin{array}{l} k \in N; k > \eta; \\ f(w) = \delta_1^B u_{2\cdot\eta} u_{2\cdot\eta-1} \dots u_{\eta+1}; \\ |u_{2\cdot\eta}| = \dots = |u_{\eta+2}| \ge |u_{\eta+1}| > 0; \end{array} \right\}.$$

VETA 7. `O binarizácii.` *Pre každý DGSM A existuje pamäťovo ekvivalentný binárny stroj DGSM B. Formálne: $\forall$ DGSM $A : \exists$ DGSM[A=2] $B : A \approx B$.*

DÔKAZ. Najprv si ukážeme existenciu relácie v smere binarizácie:
Nech $A = (\sum_1, \sum_2, H, a, q)$. Použitím konštrukcie 1 dostaneme binárny DGSM $B = ([0,1], \sum_B, H_B, b, q_2^{start})$. Teraz potrebujeme ukázať, že platia podmienky redukcie:
A-prekladač M bude prekladať každý jeden symbol z $\sum_1$ na $2 \cdot \lceil \log |\sum_1| \rceil$ nových symbolov tak, že prvá polovica bude jednoznačný binárny zápis (viď funkcia f v konštrukcii 1) a druhá polovica budú iba nuly. Konštrukcia takéhoto prekladača je triviálna záležitosť. Ešte potrebujeme ukázať, že funkcia $f_{M(W(A))}$ je lineárna, čo je ale jednoduché, lebo funkcia $f_{M(W(A))}(x) = \lceil x/(2 \cdot \lceil \log |\sum_1| \rceil) \rceil$, kde $(2 \cdot \lceil \log |\sum_1| \rceil)$ je konštanta.

Pre opačný smer vieme skonštruovať rovnako jednoduchý A-prekladač N, ktorý prečíta $\lceil \log |\sum_1| \rceil$ znakov binárneho zápisu, ďalej prečíta rovnaký počet núl a podľa binárneho zápisu, ktorý si uchováva v stavoch, to celé prepíše na príslušný symbol z $\sum_1$. A funkcia $f_{N(W(B))}(x) = x \cdot 2 \cdot \lceil \log |\sum_1| \rceil$. □

DÔSLEDOK 3. *Všetky slová generované pomocou DGSM sú v* $GSPACE(f)$ *práve vtedy, ak sú v tejto množine všetky slová generované pomocou binárneho DGSM.*
Formálne: $L(DGSM[A{=}2]) \subseteq GSPACE(f) \Leftrightarrow L(DGSM) \subseteq GSPACE(f)$

Čiže sme zistili, že binarizácia nám uľahčí prácu vzhľadom na veľkosť výstupnej abecedy za cenu jemného predĺženia pravej strany pravidiel. Na druhú stranu sa takto dajú tvoriť rovnako komplikované slová ako pomocou celej DGSM. Čo je ale inšpiráciou k použitiu iného obmedzenia: [R=2], [S=2] a ich kombinácií.

Časť 3

Zložitejšie modely

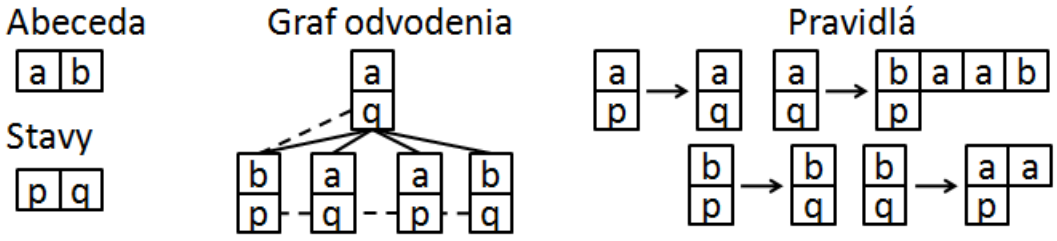
KAPITOLA 6

Prerozdelené DGSM

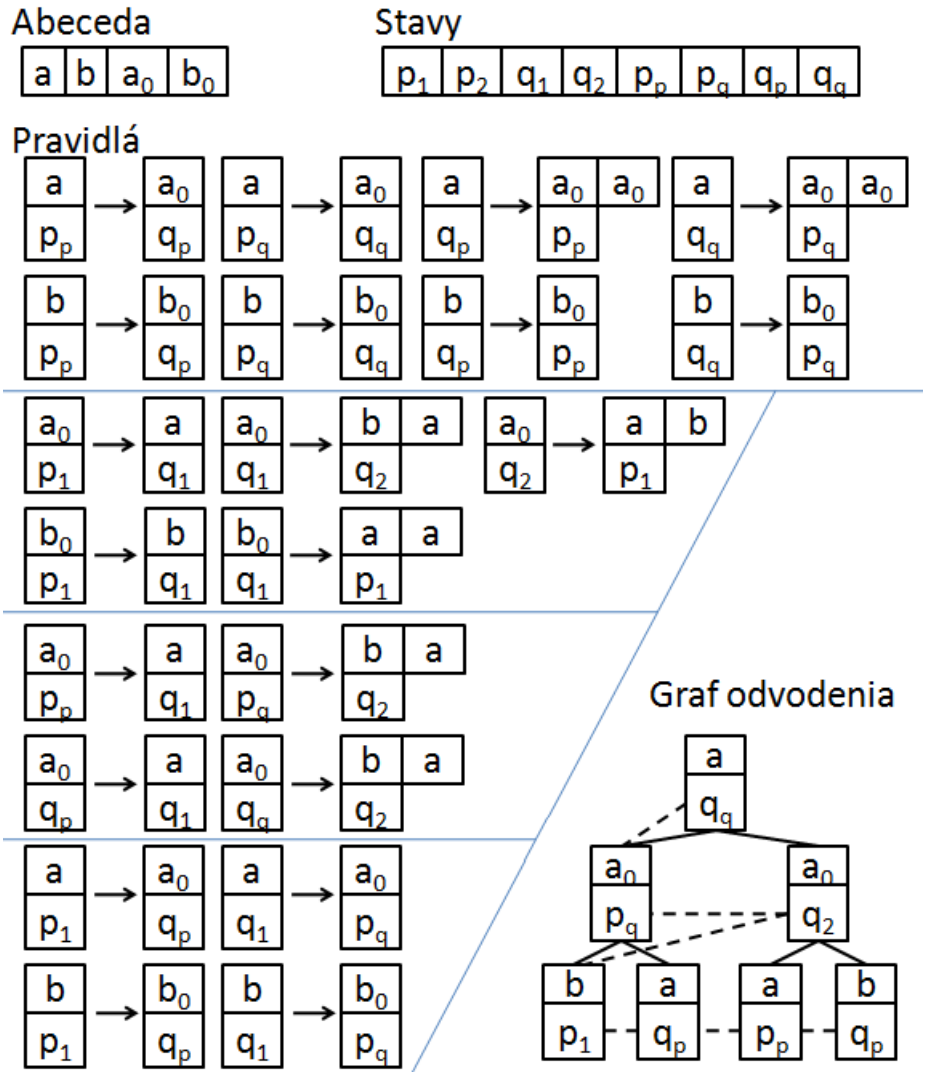
V tejto kapitole si ukážeme niektoré vlastnosti ďalších obmedzených DGSM a prerozdelíme si množinu strojov DGSM, na základe týchto obmedzení.

DGSM[R=2]

Príklad 3. *(Obr. 11 a 12)*
Princíp je založený na dvojfázovom prepise. Ak si to predstavíme ako strom, potom prečítame celú pamäť (celý riadok stromu) a stav meníme ako v štandardnom DGSM, pričom ale iba kopírujeme všetky výstupné znaky okrem prípadov, kedy by sme použili dlhé pravidlo, tam znak zdvojíme. Pri druhom prechode už budeme generovať to, čo by generoval pôvodný DGSM, potrebujeme len vedieť, v akom stave sme boli, keď sme začali len kopírovať a zdvojovať znaky. Na toto nám stačí dvojúrovňový stavový symbol. A ešte potrebujeme odlíšiť, kde sa končí úroveň stromu, pričom pre tento účel budeme mať dvojnásobnú abecedu. Práve tie zdvojené a skopírované symboly sa budú môcť A-prekladačom zmazať, aby sme získali pôvodné slovo.



Obrázok 11. DGSM pred redukciou dĺžky pravidiel



OBRÁZOK 12. DGSM po redukcii dĺžky pravidiel

KONŠTRUKCIA 2. *Nech máme daný DGSM* $A = (\sum_1, \sum_2, H, \delta_1, \delta_2)$, *potom skonštruujeme*
DGSM $B = (\sum_1 \cup \{a_0 | a \in \sum_1\}, \sum_2 \times \{1,2\} \cup \sum_2 \times \sum_2, H_B, \delta_1, \delta_2 \times \delta_2)$, *kde*
$H_B = \{(a, q \times p, a_0, r \times p) | p \in \sum_1; (a,q,w,r) \in H; |w| = 1\} \cup$
$\cup \{(a, q \times p, a_0 a_0, r \times p) | p \in \sum_1; a,q,w,r) \in H; |w| > 1\} \cup$
$\cup \{(a_0, q \times 1, w, r \times 1) | (a,q,w,r) \in H; |w| = 1\} \cup$
$\cup \{(a_0, q \times 1, w, q \times 2) | (a,q,wv,r) \in H; |v| > 1; 0 \leq |w| - |v| \leq 1\} \cup$
$\cup \{(a_0, q \times 2, v, r \times 1) | (a,q,wv,r) \in H; |v| > 1; 0 \leq |w| - |v| \leq 1\} \cup$
$\cup \{(a, q \times 1, a_0, r \times q) | (a,q,w,r) \in H; |w| = 1\} \cup$
$\cup \{(a, q \times 1, a_0 a_0, r \times q) | (a,q,w,r) \in H; |w| > 1\} \cup$
$\cup \{(a_0, q \times p, w, r \times 1) | (a,p,w,r) \in H; |w| = 1\} \cup$
$\cup \{(a_0, q \times p, w, p \times 2) | (a,p,wv,r) \in H; |v| > 1; 0 \leq |w| - |v| \leq 1\}.$

VETA 8. `O redukcii dĺžky pravidiel`. *Pre každý DGSM[R=x] A existuje pamäťovo redukovateľný stroj* $B \in DGSM[R=\lceil x/2 \rceil]$.

DÔKAZ. Podľa konštrukcie 2. A-prekladač bude mazať všetky symboly doplnené o dolný index 0, čím sa dĺžka slova skráti maximálne o polovicu. Teda okrem poslednej úrovne, ale tá je najviac k-krát väčšia ako predošlá (obdobne ako pre dôsledok 2). Preto asymptotická dĺžka ostáva zachovaná. □

VETA 9. $L(DGSM)$ *je podmnožinou* $GSPACE(f)$ *práve vtedy, ak* $L(DGSM[R=2])$ *je podmnožinou* $GSPACE(f)$.

DÔKAZ. Každý DGSM má nejakú maximálnu dĺžku pravej strany pravidiel, teda náš stroj musí patriť do nejakej triedy DGSM[R=x], a ak opakovane použijeme vetu o redukcii dĺžky pravidiel, konkrétne $\lceil \log x \rceil$-krát, potom získame stroj z triedy DGSM[R=2]. □

DGSM[S=x]

Pokiaľ sa chceme pokúsiť o stavovú redukciu, potrebujeme si najskôr zhodnotiť doterajšie výsledky. To podstatné sa dokázalo v diplomovej práci Miroslavy Kemeňovej [K06]. Tam ukázala, že existujú slová:
$Lin_k = Sc_1\#c_2\#c_3 \ldots$, kde $c_n = a^n\#a^{2n}\#\ldots\#a^{kn}$;
$Lin_k = Sa^1\#a^2\#\ldots\#a^k\#a^2\#a^4\#\ldots\#a^{2k}\#\ldots\#a^{nk}\#a^{n+1}$...,
ktoré sa dajú generovať pomocou DGSM, pričom tento DGSM musí mať aspoň k stavov, a zároveň mu stačí práve k stavov.

Toto na prvý pohľad znemožnuje ďalšiu prácu, no na druhú stranu pre našu pamäťovú redukciu potrebujeme DGSM s menším počtom stavov a A-prekladač, ktorý môže mať stavov, koľko chceme. Preto si vieme skonštruovať obdobné slovo:
$Lin^S_{k-1} = Sc^S_1\#c^S_2\#c^S_3$..., kde $c^S_n = a^n\#a^{2n}\#\ldots\#a_0^{(k-1)n}\#a^{(k-1)n}$,
na ktoré by malo stačiť $k-1$ stavov a M-prekladač prekladá tak, že všetko iba kopíruje, okrem momentu, keď sa dostane k symbolu a_0. Tam začne prepisovať a_0 na a. Po prečítaní najbližšieho znaku # začne prepisovať a^{k-1} na a^k až po prečítanie #. Opäť začne všetko kopírovať a čaká, pokiaľ sa znovu nedostane k znaku a_0.

Táto úvaha teda otvára dvere k potencinálnej stavovej redukcii, no vytvoriť jednotný postup, ako redukovať stavy, sa javí ako pomerne náročný problém. Necháme ho ako ďalší oriešok na zamyslenie. Takto sme sa dostali k triedam, kde začneme kombinovať jednotlivé obmedzenia.

VETA 10. $L(DGSM)$ *je podmnožinou* $GSPACE(f)$ *práve vtedy, keď je ňou aj* $L(DGSM[A=4][R=2])$. $L(DGSM)$ *je podmnožinou* $GSPACE(f)$ *práve vtedy, keď je ňou aj* $L(DGSM[A=2][R=3])$.

DÔKAZ. Podľa predchádzajúcej vety ku každému DGSM existuje redukovateľný DGSM[R=2], po aplikácií binarizácie získame stroj z DGSM[A=2][R=3] (dĺžka nových pravidiel bude $2 \cdot 2 - 1$). Po opätovnom použití redukcie dĺžky pravidiel získame stroj z DGSM[A=4][R=2]. □

Čím sa dostávame k otvoreným problémom:
$L(\text{DGSM}) \subseteq GSPACE(f) \Leftrightarrow L(\text{DGSM[A=3][R=2]}) \subseteq GSPACE(f)$;
$L(\text{DGSM}) \subseteq GSPACE(f) \Leftrightarrow L(\text{DGSM[A=2][R=2]}) \subseteq GSPACE(f)$.

V tejto kapitole sme si, ukázali ako veľmi vieme obmedziť stroj DGSM, hlavne na úkor stavov. A stanovili sme tri nové problémy.

KAPITOLA 7

DGSM vs. univerzálny TS

V tejto kapitole si ukážeme, že existuje univerzálny Turingov stroj, ktorý dokáže vygenerovať rovnaké slovo ako ktorýkoľvek DGSM TAG systém, pričom dostane na vstup práve kód stroja DGSM príslušného k danému DGSM TAG systému.

DGSM vs SPACE(x)

Je dôležité spomenúť jeden z otvorených problémov, a to rozhodnuteľnosť, či dané slovo patrí do $GSPACE(O(1))$. Ako si ukážeme, vieme v leme 4 čiastočne rozhodnúť, že tam toto slovo patrí, čo ale nepostačuje. Preto nasledujúca veta bude obsahovať dve alternatívy.

VETA 11. `Univerzálna`. *Ak existuje taký Turingov stroj, ktorý vie rozhodnúť, či daný DGSM je triviálny, potom existuje taký Turingov stroj U, ktorý ak dostane na vstup kód stroja DGSM C generujúceho slovo z $GSPACE(f)$, potom tento Turingov stroj stroj začne generovať to isté slovo $W(A)$, pričom použije pamäť veľkosti $O(f)$.*
Ak neexistuje Turingov stroj, ktorý vie rozhodnúť, či daný DGSM je triviálny, potom existuje taký Turingov stroj U, ktorý ak dostane na vstup kód stroja DGSM C generujúceho slovo z $GSPACE(f)$, potom tento Turingov stroj stroj začne generovať to isté slovo $W(A)$, pričom použije pamäť veľkosti väčšej z hodnôt $O(f)$ a $O(\log n)$.

Skôr ako sa pustíme do dôkazu tejto vety, bude lepšie, ak si dokážeme dve čiastočné tvrdenia:

LEMA 4. *Existuje Turingov stroj, ktorý keď dostane na vstup kód triviálneho DGSM A, potom vie čiastočne rozhodnúť, že ide o triviálny DGSM a následne vie generovať slovo $W(A)$ v konštantnej pamäti.*

DÔKAZ. Z definície vieme, že na generovanie $W(A)$ nám stačí konštantná pamäť resp. konečný automat. Takýto DGSM generuje periodické slovo [HKL94]. Čo znamená, že existujú slová u a v také, že $W(A) = uvvvvv\ldots$ Takže DGSM bude mať po chvíli v pamäti už iba niečo ako $vvv\ldots vw$, kde w je nejaký prefix slova v (presnejšie je $v^i w$, kde $w \prec v$). V momente ako DGSM A prečíta slovo u, pozrieme sa na to, v akom stave sa nachádza a aký úsek posledného slova v máme v pamäti. Tu máme maximálne $|\sum_2| \cdot |v|$ rôznych možností, ale po ich vyčerpaní nastane situácia, keď DGSM A bude v stave, v ktorom už bol a zároveň na konci výstupnej pásky bude rovnaký prefix slova v. Jediné, čo sa môže zmeniť, je počet podslov v v pamäti. Po ďalšej takejto iterácii sa pamäť stroja A zmení zas o rovnaký počet podslov v a keďže ich počet nemôže klesať (viď konštrukcia

pravidiel stroja DGSM), je zrejmé, že DGSM A sa začne cykliť. Toto vie Turingov stroj ľahko odsledovať a odhaliť. Postupne simuluje DGSM A a pre každý vygenerovaný prefix slova $W(A)$, vyskúša všetky možnosti, ako tento prefix zapísať pomocou slov u a v tak, aby nastalo zacyklenie stroja DGSM A. Ak nenájde, jednoducho použije dlhší prefix slova $W(A)$. Ak nájde zacyklenie, potom môžeme prestať simulovať stroj DGSM a miesto toho budeme na výstup už len zapisovať slová v, keďže náš DGSM A, už nič iné robiť nemôže.

□

LEMA 5. *Existuje Turingov stroj U, ktorý keď dostane na vstup kód netriviálneho DGSM A generujúci slovo $W(A)$ z $GSPACE(f)$ a $f \geq \Omega(\log)$, potom vie generovať slovo $W(A)$ v pamäti $O(f)$.*

DÔKAZ. Nech C je DGSM taký, že $W(C) \in GSPACE(f)$, teda existuje príslušný Turingov stroj A, ktorý vygeneruje každý prefix $v \prec W(C)$, pričom použije pamäť veľkosti $f(|v|)$. Kód tohto stroja je voči generovanému slovu konštantný. Tento kód si označíme ako w_A. Ako vstup stroja bude kód stroja DGSM C (w_C). $2 \cdot O(f)$ je príslušný počet pamäťových miest na simuláciu dvoch Turingových strojov (viď konštrukcia UTS z [F04]). $3 \cdot \log(|v|)$ bude pamäť, kde si uložíme dlžku slova v, ako aj pozíciu, z ktorej by mal DGSM čítať, ak by chcel generovať niečo na pozíciu $v + 1$. Čiže cieľom je skonštruovať Turingov stroj generujúci slovo $W(A)$, pričom bude potrebovať pamäť veľkosti $3+|w_C|+2 \cdot |w_A|+3 \cdot \log(|v|)+2 \cdot O(f(|v|))$. Čo je ekvivalentné k $O(f(|v|))$. Samotný stroj U bude generovať slovo $W(A)$ nasledovne:

1. Nakopíruje si w_A do pamäte. Dokonca sa dá stroj rozšíriť o kontrolu, kde sa preverí, či je slovo w_A korektne zadaný DGSM.
2. Vygeneruje si kód takého deterministického Turingovho stroja, ktorý vždy vygeneruje iba počiatočný symbol pre výstupnú pásku v stroji A a následne sa zasekne. Tento stroj si označíme ako kontrolný.
3. Nastaví si počítadlo pre dĺžku vygenerovaného slova na hodnotu 1 a počítadlo pre pozíciu čítacej hlavy na hodnotu 0.
4. Vygeneruje „nasledujúci“ kód deterministického Turingovho stroja (prechádza všetky binárne kódy od 0^k po 1^k, kde k je na začiatku dĺžka kódu stroja, ktorý sme si označili ako kontrolný, teda konštanta, neskôr sa táto hodnota bude zväčšovať), taktiež musí overiť, či ide o deterministický Turingov stroj. Tento stroj si označíme ako kandidáta.
5. Stroj U bude postupne simulovať oba stroje (kontrolný aj kandidáta) tak, že si kontroluje, čo by chceli zapisovať a porovnáva ich výstupy po jednom znaku. Ak zapíše jeden, čaká až zapíše aj druhý atď. Neustále porovnáva, či generujú to isté. Zároveň si pamätá stavový symbol, ktorý by DGSM A čítal, ak by čítacie hlavy čítali posledný vygenerovaný znak. Teda ak oba stroje zapíšu ten istý symbol, potom si ešte odsleduje, aký bude nasledovný stavový symbol a prepíše ním predošlý.
6. Ak sa dostanú až na pozíciu čítacej hlavy (viď počítadlo pre jej pozíciu), potom si do pamäte uloží celú dvojicu z tejto pozície, ktorú by mal DGSM A čítať a ďalej už stavový symbol sledovať nepotrebuje.

7. U pokračuje v simulovaní oboch Turingových strojov, až vygenerujú toľko symbolov, že sa dostanú na miesto, kde by mala byť zapisovacia hlava DGSM A (viď počítadlo dĺžky vygenerovaného prefixu).
8. Teraz U porovná to, čo by zapísal kandidát s tým, čo by zapísal DGSM A z čítanej dvojice symbolov a ak sa to zhoduje, kandidát vie vygenerovať o niečo dlhšie slovo ako ten kontrolný. Z kandidáta sa stane kontrolór, inkrementujú sa pozície čítacej aj zapisovacej hlavy a pokračujeme od bodu 5.
Toto bol len základný scenár. Všetko ostatné, čo sa môže stať (pokaziť):

- Kandidát nebude generovať na konci to isté, čo hovoria pravidlá DGSM, potom ideme na bod 4 a hľadáme nového lepšieho kandidáta.
- Počas výpočtu nebude kandidát generovať to isté, čo kontrolór, resp. sa zasekne alebo presiahne vyhradenú pamäť. Toto je novinka, ktorá doteraz nebola spomenutá. U musí evidovať pamäť, ktorú môže kandidát používať. Pre nás to bude ale tá istá hodnota, ktorá vymedzuje maximálnu veľkosť kódu pre kandidáta/kontrolóra (k). Teda ho to vráti na bod č. 4.
- Posledným problémom je, že ak U vygeneruje všetky prípustné kódy pre kandidátov a ani jeden nebude postačovať, potom je to správny čas na inkrementovanie hodnoty k, spolu s tým U reštartuje počítadlo vygenerovaných kandidátov a pokračuje od bodu 4.

Celý dôkaz je založený na tom, že U nikdy nezistí, či je kandidát skutočne ten hľadaný stroj, ale na druhú stranu nikdy nepoužijeme asymptoticky viac pamäte, ako by použil náš kandidát. Toto neplatí len do momentu, než sa vyhradené pamäťové miesto nezväčší na veľkosť kódu hľadaného Turingovho stroja, ale to je konštantná hodnota, teda asymptotické ohraničenie sa tým nijak nepokazí. □

A teraz sľúbený dôkaz k vete 11:

Dôkaz.

- Nech je príslušnosť k triede triviálnych DGSM rozhodnuteľný problém. Potom podľa lemy 3 vieme DGSM rozdeliť na dve základné skupiny. Tie, pre ktoré stačí konštantná pamäť a tie, pre ktoré potrebujeme aspoň logaritmickú pamäť. Pre každú skupinu vieme príhodne použiť príslušnú lemu (4 a 5).
- Ak tento problém rozhodnuteľný nie je, potom musíme postupovať podľa lemy 5 a tam je spodné ohraničenie $\Omega(\log n)$, keďže U si pamätá napr. dĺžku vygenerovaného slova.

□

Dôsledok 4. *Pre žiadny stroj DGSM A neexistuje viacero pamäťovo optimálnych Turingových strojov generujúcich slovo $W(A)$ takých, že funkcie popisujúce ich generatívnu pamäťovú zložitosť by neboli asymptoticky rovnaké.*

Dôkaz. Sporom: nech teda existujú 2 takéto stroje. Funkcie popisujúce ich generatívnu pamäťovú zložitosť sú monotónne. Konštrukciou z vety 11 vieme vytvoriť Turingov stroj taký, ktorý bude mať generatívnu pamäťovú zložitosť vždy

lokálne tak efektívnu ako lokálne efektívnejšia funkcia pre pôvodné stroje, čo ale odporuje existencii dvoch rôznych optimálnych funkcii. □

Časť 4

Výsledky

KAPITOLA 8

Záver

V tejto práci sme ukázali rôzne možnosti ďalšieho rozvoja poznatkov o pamäťovej náročnosti strojov DGSM. Táto práca zaviedla nový pojem redukcie a ekvivalencie, ktorá značným spôsobom umožňuje ohraničiť množinu strojov DGSM, ktorými sa potrebujeme zaoberať pri skúmaní ich generatívnej zložitosti. Ako ukážku vieme skonštruovať postupnosť ohraničených množín: DGSM[A=2][R=3][S=1], DGSM[A=2][R=3][S=2], DGSM[A=2][R=3][S=3]...
Ďalším krokom je skonštruovanie postupnosti jazykov týchto množín a posledným krokom ohraničenie množín slov, pre ktoré nevieme rozhodnúť, či sú z množiny $GSPACE(\log n)$. Toto je všeobecne nerozhodnuteľný problém, no väčšinou sa to rozhodnúť dá. Ako príklad vie poslúžiť mohutnosť množiny DGSM[A=2][R=3][S=2] pre fixne zadanú abecedu. Táto množina obsahuje státisíce strojov DGSM, pričom zhruba iba pre 1% generovaných slov sa nedá ľahko určiť príslušnosť do množiny $GSPACE(\log n)$. Jedným z pôvodných cieľov tejto práce bolo dokázanie vzťahu: $L(DGSM[A = 2][R = 3][S = 2]) \in GSPACE(\log n)$. Keďže mnoho prác poukazuje na značnú zložitosť obdobného dôkazu, táto snaha pomohla len ako inšpirácia pri iných častiach práce a nechávam ju ako otvorený problém pre ďalšie riešenie.
Ďalším zaujímavým výsledkom je znovuotvorenie problému redukcie stavov, ktorý sa javí ako silnejší a zaujímavejší spôsob ohraničovania strojov DGSM, keďže abeceda aj dĺžka pravej strany pravidiel sa dajú redukovať výlučne na úkor stavov.
Posledným výsledkom tejto práce je konštrukcia univerzálneho Turingovho stroja, ktorý dokáže generovať ľubovoľné slovo z $L(DGSM)$ v optimálnej pamäti, okrem prípadov keď ide o triviálny DGSM. Tieto prípady však podliehajú doposiaľ nepreukázanej hypotéze o rozhodnuteľnosti triviálnosti systému DGSM TAG.

Zoznam bibliografických odkazov

[M67] Minsky, M. L. *Computation: finite and infinite words.* [S. l.]: Prentice-Hall, 1967.

[CS82] Culik II, K., Salomaa, A. *On infinite words obtained by iterating morphisms.* In Theoretical Computer Science, 1982, iss. 19, p. 29-38.

[S88] Shallit, J. *A generalization of automatic sequences.* In Theoretical Computer Science, 1988, iss. 1-16.

[CKL92] Culik, K., Karhumäki, J. a Lepistö, A. *Alternating iteration of morphisms and the Kolakovski sequence.* In Lindermayer Systems (Rozenberg, G., Salomaa, A., eds.). Berlin: Springer, 1992, p. 93-106.

[M94] Mitrana, V. *On some classes of grammar systems.* In Developments in Theoretical Computer Science (Dassow, J., Kelemenova, A., eds.). [S. l.]: CSC, 1994, p. 153-162.

[L94] Lepistö, A. *Repetitions in Kolakoski sequence.* In Developments in language theory (Rozenberg, G., Salomaa, A., eds.). Singapore: World Scientific, 1994, p. 130-143.

[CK94] Culik II, K., Karhumäki, J. *Iterative devices generating infinite.* In International Journal of Foundations of Computer Science, 1994, vol. 5, no. 1, p. 69-97.

[HKL94] Hromkovič, J., Karhumäki, J., Lepistö, A. *Comparing descriptional and computational complexity of infinite words* In Results and Trends in Theoretical Computer Science (Karhumäki, J., Mauer, H., Rozenberg, G., eds.), LNCS 812. Berlin: Springer, 1994, p. 169-182

[L96] Lepistö, A. *On the computational complexity of infinte words.* In Developments in language theory II (Dassow, J., Rozenberg, G., Salomaa, A., eds.). Singapore: World Scientific, 1996, p. 350-359.

[HK97] Hromkovič, J., Karhumäki, J. *Two lower bounds on computational complexity of infinite words.* In New trends in formal languages, LNCS 1218. 1997, p. 366-376.

[M02] Maňuch, J. *Descriptional and computational complexities of infinite words: project of the dissertation thesis.* Bratislava: [s. n.], 2002. Projekt k dizertačnej práci.

[DM03] Ďuriš, P., Maňuch, J. *On the computational complexity of infinite words.* In Theoretical Computer Science, 2003, iss. 1-3.

[F04] Forišek, M. *Formálne jazyky a automaty.* Verzia zo dňa 23. mája 2004. 103 s. Rukopis.

[K06] Kemeňová, M. *On descriptional complexity of infinite words.* Bratislava: [s. n.], 2006. Diplomová práca.

[K08] Kemeňová, M. *Complexity of infinite words: project of the dissertation thesis.* Bratislava: [s. n.], 2008. Projekt k dizertačnej práci.

Resumé

Táto diplomová práca sa zaoberá nekonečnými slovami generovanými pomocou systému DGSM TAG. Poukazuje na niektoré zatiaľ nespracované vlastnosti a možnosti ich použitia, rozšírenia a ďalšieho rozvoja. Prvá časť vymädzuje oblasť problematiky a popisuje dosiahnuté aj otvorené problémy, na ktoré táto práca nadväzuje. Druhá časť ukazuje použitie niektorých konštrukcií a ich možné využitie. Poukazuje hlavne na spojenie týchto konštrukcií a teórie grafov. V tretej časti sa venujeme obmedzenej množine nekonečných slov, pre ktoré popisujeme ich generatívnu zložitosť. V štvrtej časti sa dostávame k tomu najpodstatnejšiemu z tejto práce:

- Redukovanie uvažovanej množiny strojov DGSM.
- Redukovanie problému na jediný Turingov stroj.

V závere práce je ukázané použitie predošlých výsledkov, ďalšie smerovanie a rozvoj práce.