UNIVERZITA KOMENSKÉHO, BRATISLAVA
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

Súvis generatívnych systémov a alternujúcich Turingových strojov

DIPLOMOVÁ PRÁCA

Mojmír Fendek **2008**

Súvis generatívnych systémov a alternujúcich Turingových strojov

Porovnanie tried zložitosti

DIPLOMOVÁ PRÁCA

Mojmír Fendek

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

Informatika

Vedúci diplomovej práce
Prof. RNDr. Branislav Rovan, PhD

BRATISLAVA APRÍL 2008

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry a s odbornou pomocou vedúceho práce.

Bratislava, apríl 2008 Mojmír Fendek

Poďakovanie

Chcel by som poďakovať svojmu školiteľovi prof. RNDr. Branislavovi Rovanovi, PhD za vedenie, konzultácie a motiváciu, taktiež veľká vďaka patrí mojej rodine za podporu a strpenie.

Abstrakt

Analyzujeme dva paralelné modely - Generatívny systém (ďalej už len G-systém) a alternujúci Turingov stroj (ďalej už len ATS). Táto práca sa zaoberá porovnaním týchto dvoch paralelných modelov, resp. porovnaním ich tried zložitosti, vzhľadom na dve miery zložitosti - čas a priestor.

Najskôr prinášame základné definície modelov, tried zložitosti a notácie, ktorú budeme používať. Nasleduje prehľad už dokázaných tvrdení, prípadne triviálne vyplývajúcich tvrdení, z už dokázaných.

Ďalej v práci porovnávame triedy zložitosti oboch modelov sformulovaním a dokázaním horných a dolných odhadov. Pri horných odhadoch popíšeme simuláciu jedného modelu druhým a dokážeme jej správnosť. Dolné odhady dokážeme nájdením "kontrapríkladových" jazykov a ukážeme, že, za určitých okolností, jeden model "kontrapríkladový" jazyk vie akceptovať, resp. generovať a druhý nie.

Kľúčové slová: G-systém, ATS, triedy zložitosti, dolný odhad, formálne jazyky

Predhovor

Cieľom tejto práce bolo v prvom rade analyzovať G-systém a ATS - identifikovať ich slabé a silné miesta. Na základe týchto informácií potom porovnať oba modely, t.j. preskúmať vzťahy medzi ich triedami zložitosti.

Oblasť je rozsiahla a len v malej miere preskúmaná, z čoho aj vyplýva hlavná motivácia tejto práce. Nové výsledky z tejto oblasti sa dajú zužitkovať na porovnanie iných, nielen paralelných modelov.

Podarilo sa nám nájsť slabé, resp. silné miesto G-systému - pomalá vnútorná komunikácia, resp. efektívne generovanie niektorých rýchlo generovateľných jazykov. Taktiež pri ATS sme našli slabé, resp. silné miesto - problém správneho rozdelenia vstupu medzi jednotlivé procesy, resp. efektívna práca vo veľmi malom priestore.

Druhý cieľ sa nám podarilo dosiahnuť len čiastočne, z dôvodu veľkej časovej náročnosti úlohy. Podarilo sa nám dokázať horný odhad - skonštruovať simuláciu ATS pomocou G-systému, ktorá je tesná, t.j. simulujúci G-systém pracuje v rovnakom čase ako ATS. Ďalej sa nám podarilo dokázať dolný odhad - ukázať, že existuje jazyk, ktorý vie ATS akceptovať v logaritmickom čase a zároveň ho G-systém nevie vygenerovať v logaritmickom priestore. Nakoniec uvádzame dve hypotézy, ktoré zrejme platia, nepodarilo sa ich však formálne dokázať. Dôkazy oboch hypotéz nechávame na budúce práce.

Vychádzali sme z referencovanej literatúry, z poznatkov z oblasti formálnych jazykov a teórie paralelných výpočtov.

Kľúčové slová: G-systém, ATS, triedy zložitosti, dolný odhad, formálne jazyky

OBSAH

ÚVOD

V ďalšom si vysvetlíme motiváciu práce, popíšeme v krátkosti oblasť, do ktorej práca patrí, cieľ práce a štruktúru práce.

Paralelizmus je oblasť, ktorá je skúmaná už roky. Napriek tomu nepoznáme odpovede na množstvo otázok. V snahe nejak charakterizovať, zaradiť a popísať rôzne druhy paralelizmu, vzniklo viacero paralelných modelov - ruské a indické paralelné gramatiky, PCGS, CDGS, Lindenmayerove systémy, Generatívne systémy, alternujúce Turingove stroje, Boolovské obvody a mnohé ďalšie. Po zavedení týchto modelov sa začalo so skúmaním generatívnej sily týchto modelov a zaraďovaním do Chomského hierarchie. Ukázalo sa, že generatívna sila G-systémov a ATS je rovnaká - $\mathcal{L}_{\mathrm{RE}}$.

Pozrime sa detailnejšie na tieto dva modely. G-systém je v podstate paralelná gramatika, ktorá nepoužíva pravidlá pri odvodení, ale 1-a prekladač (konečný automat s výstupom). Začína s počiatočným neterminálom a krok odvodenia realizuje transformáciou vetnej formy pomocou 1-a prekladača. ATS pracuje na úplne inom princípe. Ide o rozšírenie nedeterministického Turingovho stroja (ďalej už len NTS). Rozdiel je v tom, že ATS má množinu stavov rozdelenú na dve disjunktné podmnožiny. Jedna podmnožina je množina obyčajných stavov NTS, voláme ju množina existenčných stavov. Druhá podmnožina sa nazýva množina univerzálnych stavov. Keď je ATS v existenčnom stave, správa sa ako NTS. Keď je však v univerzálnom stave, tak rozvetví (rozdelí) proces na viacero už nezávislých (nekomunikujúcich) procesov so separátnymi páskami. Podrobné definície oboch modelov uvádzame v kapitole 1.

Ako vidíme, ide o dva veľmi rozdielne modely. Zistenie rovnosti generatívnej sily oboch modelov vyvolalo ďalšie otázky. Motiváciou tejto práce je objasniť dosiaľ neznáme poznatky o vzťahoch týchto dvoch modelov.

Pre oba modely definujeme miery zložitosti: časovú a priestorovú zložitosť. Pre G-systém časová, resp. priestorová zložitosť je taký počet krokov, koľko potrebuje na odvodenie slova, resp. maximálna dĺžka vetnej formy, ktorú potrebuje na odvodenie slova. Pre ATS sú miery zložitosti obdobné: časovú zložitosť definujeme ako počet krokov potrebný na akceptáciu a priestorovú zložitosť ako maximálnu veľkosť použitej pásky pri výpočte.

Na základe mier zložitosti definujeme tiedy zložitosti ATIME, ASPACE, GTIME, GSPACE. Formálne definície uvedieme v kapitole 1. Hlavný cieľ práce je porovnať triedy zložitosti ATIME, ASPACE, GTIME, GSPACE za rôznych podmienok, skonštruovať simulácie (horné odhady) a nájsť "kontrapríkladové" jazyky, t.j. dokázať dolné odhady. Týmto sa zaoberáme v kapitole 2. Kapitola 2 sa skladá z troch sekcií, v každej sekcii skúmame triedy zložitosti pre iný charakter funkcií. Na začiatku každej sekcie uvádzame prehľad tvrdení, ktoré sekcia obsahuje, pričom hlavné tvrdenia tejto práce sú od ostatných tvrdení odlíšené podčiarknutím.

1. ZÁKLADNÉ DEFINÍCIE A POPIS NOTÁCIE

V tejto práci budeme používať notáciu a niektoré výsledky pre alternujúci Turingov stroj a generatívny systém, prevzaté z [1]. Nasledujú potrebné definície a tvrdenia.

1.1 Generatívny systém

Definícia 1.1.1: 1-a prekladač je šestica $M = (K, \Sigma_1, \Sigma_2, H, q_0, F)$, kde K je konečná množina stavov, Σ_1, Σ_2 sú vstupná, resp. výstupná abeceda, $q_0 \in K$ je počiatočný stav, $F \subseteq K$ je množina akceptačných stavov a H konečná množina štvoríc $H \subseteq K \times \Sigma_1 \times \Sigma_2^* \times K$.

Výpočet 1-a prekladača sa dá zapísať ako slovo $v_1 \dots v_n \in H^*$, pričom platí, že 1-a prekladač je špeciálnym prípadom a-prekladača. 1-a prekladač má obmedzený počet čítaných symbolov, v jednej štvorici (v prechode po jednej hrane), na práve jeden.

Notácia pre grafické zakreslenie 1-a prekladača je nasledovná. Množinu stavov zakreslíme ako množinu vrcholov. Dva vrcholy p a q sú spojené orientovanou hranou z p do q práve vtedy, keď štvorica $(p, a, w, q) \in H$. Transformácia časti vstupu, realizovaná touto hranou, je nakreslená nad hranou, vo formáte a, w (a sa prepisuje na w). Detailnejšie informácie o a-prekladačoch sú v [2].

Definícia 1.1.2: Generatívny systém (ďalej už iba G-systém) je štvorica $G = (N, T, M, S)$, kde N a T sú konečné množiny neterminálnych a terminálnych symbolov, M je zobrazenie 1-a prekladačom a $S \in N$ je počiatočný neterminál.

Definícia 1.1.3: Krok odvodenia G-systému definujeme ako reláciu $a \Rightarrow w$ práve vtedy, keď $w \in M(a)$.

Definícia 1.1.4: Jazyk generovaný G-systémom $G = (N, T, M, S)$ je jazyk $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$. Kde $\Rightarrow^*$ je reflexívnym a tranzitívnym uzáverom relácie kroku odvodenia.

Poznámka 1.1.1: G-systém môže na prvý pohľad vyzerať ako sekvenčný model. Treba si však dôkladne prezrieť, ako je definovaná množina H v 1-a prekladači. Z tejto definície vyplýva, že symboly vo vetnej forme 1-a prekladač spracúvava po jednom. V jednom prechode vetnou formou, 1-a prekladač pracuje s každým symbolom, a teda každý symbol sa môže zmeniť. Kedže jeden prechod 1-a prekladača je jeden krok odvodenia, celá vetná forma je prepísaná naraz, teda paralelne.

1.2 Alternujúci Turingov stroj

Všetky definície týkajúce sa Turingových strojov sa dajú nájsť v [2].

Definícia 1.2.1: Alternujúci Turingov stroj (ďalej už len ATS) je šestica $A = (K, \Sigma, \Gamma, \delta, q_0, F)$, kde K je konečná množina stavov, skladajúca sa z dvoch disjunktných častí $K = K_\forall \cup K_\exists$, kde $K_\forall$ je množina univerzálnych stavov, $K_\exists$ je množina existenčných stavov, Σ je vstupná abeceda, Γ je pracovná abeceda, q_0 je počiatočný stav, F je množina akceptačných stavov a δ je prechodová funkcia

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

ATS predstavuje jeden z najprirodzenejších spôsobov paralelného riešenia problémov. V ľubovoľnom kroku sa na základe δ-funkcie môže proces rozvetviť na niekoľko ďalej paralelne pracujúcich procesov. Takto možno riešiť problém paralelne, a tým ušetriť čas v porovnaní s Turingovým strojom. Je to analógia operácie $FORK$ z programovacích jazykov. V tejto práci sa budeme zaoberať jednopáskovými Turingovými strojmi, pokiaľ nebude povedané inak.

Definícia 1.2.2: Konfiguráciu ATS definujeme analogicky ako pre nedeterministický Turingov stroj (ďalej už len NTS).

Poznámka 1.2.1: V ďalšom budeme potrebovať pracovať s ATS tak, že budeme sledovať "globálny stav", t.j. konfigurácie všetkých procesov.

Definícia 1.2.3: Ak má ATS v danom okamihu p bežiacich procesov, tak jeho globálny stav definujeme ako p-ticu $(u_1, u_2, \ldots, u_p)$, kde u_i je konfigurácia i-teho procesu. Globálny stav ATS je rozšírenie konfigurácie NTS. ATS môže mať naraz viacero bežiacich procesov, teda globálny stav ATS bude zložený z viacero konfigurácií (pre každý proces jedna konfigurácia).

Definícia 1.2.4: Krok výpočtu ATS je relácia $\vdash$ na konfiguráciách definovaná analogicky ako pre NTS.

Definícia 1.2.5: Úplný strom konfigurácií pre ATS A a vstupné slovo w je strom, ktorého vrcholy sú označené konfiguráciami, taký, že:

- koreň je označený počiatočnou konfiguráciou na slove w
- každý vrchol má za priamych (bezprostredných) nasledovníkov toľko vrcholov, koľko konfigurácií je v relácii $\vdash$ s konfiguráciou v danom vrchole, pričom každý je označený príslušnou konfiguráciou.

Definícia 1.2.6: Výpočet ATS A na slove w je podstrom úplného stromu konfigurácií na slove w taký, že:

- obsahuje koreň

- každý univerzálny vrchol obsahuje všetkých jeho priamych nasledovníkov
- každý existenčný vrchol obsahuje práve jedného jeho nasledovníka, ak existuje

Definícia 1.2.7: Akceptačný výpočet ATS A na slove w je taký výpočet na w, ktorý je konečný a každý list je označený akceptačnou konfiguráciou (t.j. obsahuje stav z F).

Definícia 1.2.8: Jazyk akceptovaný ATS A je $L(A) = \{w \in \Sigma^* \mid$ existuje akceptačný výpočet A na $w\}$.

Definícia 1.2.9: ATS je v binárnom normálnom tvare, ak je stupeň vetvenia každého vrcholu výpočtu ATS najviac dva.

Poznámka 1.2.2: Ľahko vidieť, že ku každému ATS sa dá nájsť ekvivalentný v binárnom normálnom tvare. Dôkaz sa dá nájsť v [1].

Pri paralelných výpočtoch môže dôjsť k situácii, že výpočet by mohol prebehnúť rýchlejšie ako v lineárnom čase. Aby náš model toto umožnil, budeme v týchto prípadoch používať variant ATS "s rýchlym prístupom na vstup".

Definícia 1.2.10: ATS s rýchlym prístupom na vstup je model ATS, ktorý je rozšírený o register. Písať do registra sa môžu iba symboly 0 a 1, teda obsah registra, t.j. adresa políčka pozostáva iba z týchto dvoch symbolov. Adresa k-teho políčka (číslované od 0, zľava) je k binárne zakódované. Naviac, môže ATS vrámci δ-funkcie použiť špeciálnu inštrukciu $JUMP$, ktorá v konštantnom čase premiestni hlavu vstupnej pásky na políčko, ktorého adresa je práve v registri. Pokiaľ je adresa nekorektná, dôjde k zaseknutiu. Formálnejšie:

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

δ-funkcia bola rozšírená o komponenty potrebné na riadenie hlavy v registri. Symbol J reprezentuje inštrukciu $JUMP$. Konfigurácia ATS je rozšírená o príslušný obsah registra. Počiatočná konfigurácia má v obsahu registra adresu prvého (najľavejšieho) políčka, t.j. $0^{log(n)}$. Hlava registra je na začiatku výpočtu nastavená na prvom políčku registra. Obsah registra môžeme zapísať veľmi podobne ako konfiguráciu LBA, t.j. $@u\natural v\$$, kde $u, v \in \{0,1\}^*$ a $\mid u \mid + \mid v \mid = log(n)$. Aby nedošlo k zneužitiu registra na pracovnú pásku, zavedieme obmedzenia na prácu s registrom. Čítať z registra sa môže ľubovoľne, ale zapisovať sa môže len vtedy, ak sa ide naraz zapísať adresa nejakého políčka a ihneď po vyplnení registra adresou sa vykoná $JUMP$. Po premiestnení čítacej hlavy na políčko sa hlava registra vráti opäť na začiatok registra. Tento model používame iba v prípade, ak skúmame časovú zložitostnú mieru pre funkcie $f(n) = o(n)$.

Poznámka 1.2.3: ATS s rýchlym prístupom na vstup sa dá zadefinovať aj inak. Namiesto registra model rozšírime o binárny strom, ktorý bude umiestnený nad vstupnou páskou, pričom listy budú políčka vstupnej pásky. Tento model má možnosť postupne presúvať čítaciu hlavu od koreňa k niektorému listu na základe riadiacej informácie. V každom kroku je treba špecifikovať, do ktorého z dvoch nasledovníkov aktuálneho uzla sa má hlava pohnúť, t.j. 0 (prvý nasledovník) alebo 1 (druhý nasledovník). δ-funkciu rozšírime o komponent, obsahujúci riadiacu informáciu. V tomto prípade konfiguráciu neobohatíme o obsah registra, ale o pointer do stromovej štruktúry nad vstupnou páskou.

Poznámka 1.2.4: Je ľahko vidieť, že oba modely ATS s rýchlym prístupom na vstup sú ekvivalentné. V tejto práci budeme používať verziu s registrom.

1.3 Triedy zložitosti

V tejto časti budeme definovať triedy zložitosti postupne pre oba modely. Vždy najprv definujeme miery a potom samotné triedy zložitosti.

1.3.1 Triedy zložitosti pre G-systém

Najprv definujeme miery (čas a priestor), ktoré budeme potrebovať na definovanie tried zložitosti G-systému.

Definícia 1.3.1.1: $TIME(G,w) = min\{l \mid S \Rightarrow^l w\}$, t.j. minimálny počet krokov na odvodenie w v G.

Definícia 1.3.1.2: $TIME(G,n) = max\{TIME(G,w) \mid w \in L(G), \mid w \mid \leq n\}$.

Definícia 1.3.1.3: $SPACE(G,w) = min\{l \mid l = max\{\mid w_i \mid \mid S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_k \equiv w\}\}$, t.j. minimálny priestor potrebný na odvodenie w v G.

Definícia 1.3.1.4: $SPACE(G,n) = max\{SPACE(G,w) \mid w \in L(G), \mid w \mid \leq n\}$.

Teraz sme pripravení definovať triedy zložitosti pre G-systémy. Nech $\mathcal{U}$ je trieda G-systémov.

Definícia 1.3.1.5: $GTIME_{\mathcal{U}}(f(n)) = \{L \mid \exists G \in \mathcal{U}, L = L(G), TIME(G,n) = O(f(n))\}$.

Definícia 1.3.1.6: $GSPACE_{\mathcal{U}}(f(n)) = \{L \mid \exists G \in \mathcal{U}, L = L(G), SPACE(G,n) = O(f(n))\}$.

Triedy zložitosti, sú v niektorých prípadoch prázdne, prípadne obsahujú iba konečné jazyky.

Poznámka 1.3.1.1: $GSPACE(f(n)) = \emptyset$, pre $f(n) = o(n)$.

Poznámka 1.3.1.2: $GTIME(f(n)) = \mathcal{F}$, pre $f(n) = o(log(n))$, kde $\mathcal{F}$ je trieda konečných jazykov.

1.3.2 Triedy zložitosti pre ATS

Ešte pred samotnými definíciami tried zložitosti definujeme časové a priestorové ohraničenie pre ATS.

Definícia 1.3.2.1: Hovoríme, že ATS A je $f(n)$ časovo ohraničený, ak pre každé slovo $w \in L(A)$ dĺžky n existuje výpočet, ktorý urobí najviac $f(n)$ krokov.

Definícia 1.3.2.2: Hovoríme, že ATS A je $f(n)$ priestorovo ohraničený, ak pre každé slovo $w \in L(A)$ dĺžky n existuje výpočet, ktorý použije maximálne $f(n)$ políčok na každej z pracovných pások.

Teraz sme pripravení definovať triedy zložitosti pre ATS.

Definícia 1.3.2.3: $ATIME(f(n)) = \{L \mid \exists$ ATS A taký, že $L = L(A)$ a zároveň je $f(n)$ časovo ohraničený$\}$.

Definícia 1.3.2.4: $ASPACE(f(n)) = \{L \mid \exists$ ATS A taký, že $L = L(A)$ a zároveň je $f(n)$ priestorovo ohraničený$\}$.

Triviálne prípady tried zložitosti

Poznámka 1.3.2.1: $ATIME(f(n)) = \mathcal{F}$, pre $f(n) = o(log(n))$. Toto tvrdenie predpokladá, že Turingov stroj musí prečítať celý vstup. Ak by sme zmenili definíciu Turingovho stroja a nepožadovali, aby musel prečítať celý vstup, l'ahko vieme ukázať, že $ATIME(f(n)) \supset \mathcal{F}$

Poznámka 1.3.2.2: $ASPACE(f(n)) = \mathcal{R}$, pre $f(n) = O(c)$.

2. POROVNANIE TRIED ZLOŽITOSTI

Naším cieľom je porovnať triedy zložitosti $ATIME(f(n))$, $ASPACE(f(n))$, $GTIME(f(n))$ a $GSPACE(f(n))$. Nakoľko tieto množiny veľmi výrazne menia charakter vzhľadom na typ funkcie f a taktiež sa menia podmienky pre používanie už dokázaných tvrdení, rozdelíme tvrdenia na viacero separátnych častí, podľa typu funkcií. Základne rozdelíme funkcie na $f(n) = \Omega(n)$, $f(n) = o(n)$ a $f(n) = O(c)$. Jemnejšie delenie budeme rozoberať v rámci konkrétnej sekcie. Na začiatok stručne prejdeme už dokázané tvrdenia, prípadne tvrdenia, ktoré triviálne vyplývajú z už dokázaných tvrdení. Potom pre každé možné (netriviálne) porovnanie tried zložitosti (okrem porovnania v rámci jedného modelu) sa pokúsime urobiť najprv horný odhad, teda simuláciu jedného modelu druhým, vzhľadom na príslušné miery zložitosti. V rámci týchto simulácií identifikujeme slabé a silné miesta oboch modelov a na ich základe sa pokúsime spraviť dolné odhady, t.j. nájsť kontrapríkladové jazyky, dokazujúce naše tvrdenia, t.j. vylučujúce existenciu tesnej simulácie.

Na začiatku každej sekcie uvádzame prehľad tvrdení, ktoré sekcia obsahuje, pričom hlavné tvrdenia tejto práce sú od ostatných tvrdení odlíšené podčiarknutím.

2.1 Porovnanie tried zložitosti pre $f(n) = \Omega(n)$

Intuitívne by sme očakávali, že trieda $ASPACE(f(n))$ bude asi najmohutnejšia, preto hneď na začiatok uvedieme dve triviálne tvrdenia týkajúce sa tejto triedy zložitosti.

Veta 2.1.1: $GTIME(f(n)) \subseteq ASPACE(f(n))$, pre $f(n) = \Omega(n)$.

Dôkaz: Vieme, že $GTIME(f(n)) = NSPACE(f(n))$ pre $f(n) = \Omega(n)$ z [1]. Ďalej z definície ATS triviálne vyplýva $NSPACE(f(n)) \subseteq ASPACE(f(n))$, pre $f(n) = \Omega(n)$. Naše tvrdenie vyplýva priamo z týchto dvoch tvrdení.

Veta 2.1.2: $GSPACE(f(n)) \subseteq ASPACE(f(n))$, pre $f(n) = \Omega(n)$.

Dôkaz: Z [1] vieme, že $GSPACE(f(n)) = GTIME(f(n))$ pre $f(n) = \Omega(n)$. Z tohto a predchádzajúceho tvrdenia vyplýva naše tvrdenie.

Nasledujúce tvrdenie skrýva viac, ako sa môže na prvý pohľad zdať. Výsledok tohto tvrdenia sa dá totiž použiť na porovnanie dvoch modelov Turingových strojov: NTS a ATS. Najprv ukážeme, kde stroskotal pokus o simuláciu, potom uvedieme myšlienky pre dolný odhad.

Veta 2.1.3: $ASPACE(f(n)) \subseteq \bigcup_c GTIME(c^{f(n)})$, pre $f(n) = \Omega(n)$.

Dôkaz: Triviálny horný odhad. Postupným prehľadávaním (napr. prehľadávaním do hĺbky) výpočtu ATS zistíme, či ide o akceptačný výpočet. Musíme prejsť každým vrcholom stromu, ktorých je exponenciálne veľa. V jednom kroku prejdeme jeden vrchol, preto je čas výpočtu exponenciálny.

Núka sa použiť myšlienku dôkazu $1NSPACE(f(n)) \subseteq GTIME(f(n))$ z [1]. Išlo o päťposchodovú vetnú formu, ktorá reprezentovala zmeny, ktoré sa diali na konkrétnom políčku pásky počas celého výpočtu. Keby sme vedeli urobiť paralelnú verziu tejto konštrukcie, dokázali by sme naše tvrdenie. Opäť by išlo o sledovanie zmien, ktoré sa diali na konkrétnom políčku pásky, ale tentokrát počas akceptačného výpočtu ATS, čo je strom. Vetná forma by sa skladala z viacerých častí, oddelených od seba špeciálnymi symbolmi. Počet častí by bol rovnaký ako počet listov v akceptačnom výpočte ATS. Celá vetná forma by sledovala zmeny konkrétneho políčka pásky, pričom každá časť vetnej formy by sledovala unikátnu cestu od koreňa do niektorého listu v akceptačnom výpočte ATS. Sú tu ale dva problémy.

Prvý je zabezpečiť, aby každá časť vetnej formy sledovala svoju unikátnu cestu, t.j. aby sa nestalo, že dve časti vetnej formy budú sledovať tú istú cestu od koreňa k listu v akceptačnom výpočte ATS. Tento problém sa dá vyriešiť pridaním šiestej stopy, v ktorej bude informácia, ktoré časti δ-fukcie sú, v určitom kroku simulácie, pre danú časť vetnej formy legálne, a ktoré nie. Pre vygenerovanie šiestej stopy je kľúčové vygenerovať jazyk, ktorého podslová (oddelené symbolom $\#$), pozostávajúce iba z 0 a 1, sú všetky binárne čísla rovnakej veľkosti, zoradené zľava doprava podľa ich hodnoty.

$$L = \{0^k\#0^{k-1}1\#0^{k-2}10\#\ldots\#1^{k-1}0\#1^k \mid k > 0\}$$

Tento jazyk sa dá vygenerovať G-systémom v čase $O(log(n))$. Keďže je podslovo zo šiestej stopy exponenciálne dlhé vzhľadom ku výstupnému slovu, čas generovania je v tomto prípade $O(n)$. Teda, vygenerovanie šiestej stopy sa dá zrealizovať v lineárnom čase.

Druhý, omnoho závažnejší problém, nastáva pri synchronizácii prvého poschodia vetnej formy. Prirodzene, požadujeme, aby každá časť vetnej formy pracovala na rovnakom slove.

Dĺžka prvého poschodia je však exponenciálna vzhľadom ku výstupnému slovu. Podľa "lemy o bariére" (rozoberieme neskôr) je potrebný exponenciálne dlhý čas na synchronizované vygenerovanie takejto vetnej formy. Toto je hlavný problém našej simulácie.

Sústreďme sa na dolný odhad. Keďže platí $GTIME(f(n)) = NSPACE(f(n))$, pre $f(n) = \Omega(n)$, naše tvrdenie sa dá previesť na $ASPACE(f(n)) \subseteq NSPACE(f(n))$, pre $f(n) = \Omega(n)$, čo je vlastne otázka, do akej miery pomáha alternovanie pre priestor. ATS môže odvetviť až exponeniálne veľa procesov a zdanlivo má k dispozícii až exponenciálne veľa priestoru. Po dôkladnejšom preskúmaní sa zdá, že ten na prvý pohľad veľký priestor, je rozkúskovaný medzi veľa procesov, ktoré nemôžu komunikovať. Taktiež vieme, že alternovanie pomáha najmä pri ušetrení času v porovnaní so sekvenčnými modelmi. Keď skúmame iba priestor, tak máme k dispozícii aj exponenciálne veľa času, a preto je možné, že alternovanie v tomto prípade nepomáha tak, ako by sme očakávali. Je možné, že táto simulácia je zvládnuteľná v polynomiálnom priestore.

Nakoľko rozhodnúť do akej miery pomáha alternovanie, sa zdá byť príliš ťažké, skúsime trochu zjednodušiť problém - obmedziť alternovanie. Zavedieme mieru ALT, čo je maximálny počet univerzálnych stavov po ceste od koreňa k listu v najkratšom akceptačnom výpočte na danom slove. Tvrdíme, že

$$ASPACE\text{-}ALT(f(n), g(n)) \subseteq GTIME(max(f(n), g(n)))$$

Zdôvodníme naše tvrdenie. k-páskový ATS budeme simulovať pomocou $k+1$ páskového NTS. NTS bude realizovať prehľadávanie do hĺbky na akceptačnom výpočte. Bude mať jednu pásku navyše, kde bude mať uloženú adresu aktuálnej cesty v akceptačnom výpočte. V adrese cesty stačí mať iba uložené, aké boli rozhodnutia pri univerzálnom vetvení. Na zvyšných páskach udržujeme konfigurácie ATS a simulujeme prácu ATS na jednotlivých páskach. G-systém bude simulovať tento NTS. Ako neskôr uvidíme, G-systém dokáže simulovať aj k-páskový NTS.

Veta 2.1.4: $ASPACE(f(n)) \subseteq \bigcup_c GSPACE(c^{f(n)})$, pre $f(n) = \Omega(n)$.

Dôkaz: Triviálny horný odhad. Postupným prehľadávaním výpočtu ATS zistíme, či ide o akceptačný výpočet. Musíme prejsť každým vrcholom stromu, ktorých je exponenciálne veľa. Musíme si pritom pamätať konkrétnu vetvu v strome, v ktorej sa práve nachádzame. Táto informácia môže byť až exponenciálne dlhá. Preto priestor, ktorý potrebujeme, je exponenciálny.

Pomocou tvrdenia $GSPACE(f(n)) = GTIME(f(n))$, pre $f(n) = \Omega(n)$, vieme previesť dolný odhad tohto tvrdenia na dolný odhad predchádzajúcej vety. Teda, ak sa podarí spraviť dolný odhad predchádzajúcej vety, dá sa použiť na dolný odhad tejto vety. Platí to aj opačne.

V nasledujúcom tvrdení urobíme tesnú simuláciu ATS pomocou G-systému vzhľadom na časovú zložitostnú mieru.

Veta 2.1.5: $ATIME(f(n)) \subseteq GTIME(f(n))$, pre $f(n) = \Omega(n)$.

Máme daný alternujúci Turingov stroj, bez ujmy na všeobecnosti predpokladáme, že je v binárnom normálnom tvare (stupeň vetvenia je najviac dva), s jedinou páskou. Skonštruujeme

k nemu G-systém, ktorý bude priamo simulovať ATS, krok po kroku, resp. vrstvu po vrstve v akceptačnom výpočte ATS. Teda ku každému globálnemu stavu ATS počas akceptačného výpočtu vytvoríme "korešpondujúcu" vetnú formu. Neskôr ukážeme, ako sa dá táto konštrukcia rozšíriť na simuláciu k-páskového ATS. Najprv načrtneme myšlienku dôkazu, potom formálne popíšeme konštrukciu G-systému a nakoniec dokážeme správnosť konštrukcie.

Neformálny popis konštrukcie

Prácu G-systému rozdelíme na tri hlavné fázy, tri medzifázy, inicializáciu a finalizáciu. Inicializácia predpripraví vetnú formu na ďalšiu prácu. V prvej fáze nastáva množenie neterminálu S na ľubovoľný počet (aj exponenciálne veľa, ak treba). G-systém musí vyrobiť o jeden viac neterminálov S ako je listov v akceptačnom výpočte ATS. Kedy skončiť s generovaním, hádame nedeterministicky, ak neuhádneme správne, dôjde k zaseknutiu (ako neskôr uvidíme). Prvá medzifáza vloží medzi jednotlivé neterminály S rôzne oddeľovače - $\#$ označuje začiatok úseku, $\Diamond$ označuje koniec úseku a [] označuje koniec pracovnej časti vetnej formy. $\Diamond$ tiež reprezentuje symbol $BLANK$ (zo sémantického hľadiska). Úseky budú neskôr obsahovať konfigurácie ATS, zatiaľčo časť vetnej formy za oddeľovačom [] sa na konci prepíše na výstup.

V druhej fáze synchronizovane generujeme slovo, na ktorom bude neskôr prebiehať výpočet, na všetkých úsekoch vetnej formy, aj na poslednom úseku za oddeľovačom []. Myšlienka generovania je jednoduchá - prvý úsek si vyberie symbol zo vstupnej abecedy ATS, pridá ho pred neterminál S a uloží vybraný symbol do stavu. Ostatné úseky pridajú pred S ten symbol, ktorý je uložený v stave, pričom ponechajú uložený symbol v stave pre ďalšie úseky. V jednom kroku synchronizovane vytvoríme jeden symbol. Toto sa opakuje, kým nevygenerujeme celé slovo, na ktorom bude prebiehať výpočet (nedeterministicky uhádneme). Po skončení generovania prejdeme do druhej medzifázy.

Počas druhej medzifázy G-systém predpripraví úseky na simuláciu ATS tým, že vygenerované slová v jednotlivých úsekoch upraví na počiatočné konfigurácie ATS. Vo všetkých úsekoch za symbol $\#$ pribudne symbol počiatočného stavu ATS a symbol (B), ktorý reprezentuje $BLANK$. Symbol začiatku prvého (najľavejšieho) úseku $\#$, prepíše na $[\#]$ - čo znamená, že úsek bol aktivovaný (viac o tom v ďalšej fáze). Úprava na konfigurácie sa netýka časti vetnej formy za oddeľovačom []. V celej vetnej forme sa zmažú neterminály S.

Tretia fáza je jadrom celej konštrukcie - samotná simulácia. Vetná forma sa pred simuláciou skladá z viacerých úsekov, ktoré obsahujú počiatočnú konfiguráciu ATS (na začiatku majú symboly $\#(B)$ a na konci symbol $\Diamond$). Najľavejší úsek má na začiatku symbol $[\#]$. Posledný úsek (najpravejší), oddelený [] od zvyšku vetnej formy, obsahuje iba slovo, ktoré sa neskôr (možno) prepíše na výstupné slovo. Počas celej tejto fázy ho nebudeme meniť, a teda vôbec nebudeme o ňom ani hovoriť. Vysvetlime si teraz sémantiku symbolov, ktoré sa nachádzajú na začiatku úsekov. $\#$ označuje neaktivovaný úsek, $[\#]$ označuje aktivovaný úsek a $(\#)$ označuje úsek, ktorý skončil simuláciu v akceptačnom stave (jeho konfigurácia sa už ďalej nebude meniť). Na začiatku je aktivovaný iba jeden (najľavejší) úsek - reprezentuje koreň akceptačného výpočtu ATS. Ostatné úseky sú zatiaľ neaktivované.

Pokiaľ je stav v konfigurácii (aktivovaného úseku) existenčný, tak nedeterministicky uhádneme príslušnú časť δ-funkcie a podľa nej zmeníme konfiguráciu aktuálne spracovávaného úseku. Ak je bičík na jednom z krajov konfigurácie, t.j. je na ľavom kraji (tesne za symbolom (B)),

alebo je na pravom kraji (číta symbol $\Diamond$), δ-funkcia sa simuluje tak, ako keby čítala $BLANK$. V prípade, že sa posúvame z ľavého kraja konfigurácie doľava, neprepisujeme špeciálny symbol (B). Vygenerujeme nový symbol, podľa δ-funkcie, medzi (B) a konfiguráciu príslušného úseku, a teda zväčšíme veľkosť konfigurácie. Obdobne ošetríme prípad na pravom kraji konfigurácie.

Každý neaktivovaný úsek bude modifikovať svoju konfiguráciu podľa vzoru úseku, ktorý sa nachádza bezprostredne naľavo od neho. Realizujeme to uložením vybranej časti δ-funkcie do stavu aktivovaným úsekom. Neaktivované úseky nehádajú časti δ-funkcie ako aktivované úseky, ale používajú časti δ-funkcie, ktoré majú uložené v stave, pričom uloženú časť δ-funkcie nemenia. V prípade univerzálneho stavu sa deje niečo obdobné. Aktivovaný úsek si vyberie časť δ-funkcie, zmení si podľa nej konfiguráciu a uloží vybranú časť δ-funkcie do stavu. Potom postupne spracúvame neaktivované úseky, až kým sa nerozhodneme (nedeterministicky, na ktorom) na niektorom neaktivovanom úseku, že ho aktivujeme. Prepíšeme začiatočný symbol z $\#$ na $[\#]$ a znovu vyberáme časť δ-funkcie, ale rôznu od tej časti, ktorú máme uloženú v stave. Po úprave konfigurácie vymažeme starú časť δ-funkcie a nahradíme ju novou - práve vybranou časťou δ-funkcie. Do stavu uložíme aj informáciu, že sme už aktivovali úsek, aby sme predišli ďalšej aktivácii. Pokračujeme spracovávaním ďalších neaktivovaných úsekov, ktoré sa riadia práve uloženou časťou δ-funkcie. Ak prídeme na ďalší aktívny úsek, celá procedúra sa zopakuje. Aktivácia úseku zodpovedá odvetveniu procesu v ATS.

Pri každom aktívnom úseku sa môžeme nedeterministicky rozhodnúť, že je už skončený. Prepíšeme symbol $[\#]$ na $(\#)$ a overíme, či jeho konfigurácia je akceptačná. Ak áno, pokračujeme, ak nie, dôjde k zaseknutiu. V prípade, že prídeme na už skončený úsek (majúci akceptačnú konfiguráciu), preskočíme ho (nemeníme jeho konfiguráciu). Skončené úseky nedovoľujeme mať na ľubovoľnom mieste vo vetnej forme. Skončený úsek nemôže byť priamo nasledovaný neaktívnym úsekom (ak je to tak - dôjde k zaseknutiu).

Simulujeme ATS, až kým všetky úseky nie sú skončené, t.j. majú akceptačnú konfiguráciu. Keď tento prípad nastane, ukončíme fázu simulácie a prejdeme do tretej medzifázy.

Úlohou tretej medzifázy je skontrolovať, či všetky úseky sú riadne skončené, teda, či ich konfigurácie sú akceptačné. Ak nie, dôjde k zaseknutiu, ak áno, prejdeme do poslednej fázy - finalizácie.

Finalizácia upraví vetnú formu do tvaru vhodnú na výstup. Zmažeme celú pracovnú časť vetnej formy, teda všetko až po symbol [] (vrátane). Nezmazanú časť vetnej formy už môžeme prehlásiť za vygenerované slovo.

Technické detaily ako je zabezpečené uhádnutie správneho počtu úsekov a správneho hádania aktivácií, sú detailne rozobraté neskôr.

Formálny popis konštrukcie G-systému, simulujúci ATS

Nech A je jednopáskový Alternujúci Turingov stroj v binárnom normálnom tvare $A = (K, \Sigma, \Gamma, \delta, q_0, F)$, pričom $K = \{q_1, q_2, \ldots, q_z\}$ je množina stavov, $\Sigma = \{a_1, a_2, \ldots, a_r\}$ je vstupná abeceda, $\Gamma = \{b_1, b_2, \ldots, b_v\}$ je pracovná abeceda, F je množina akceptačných stavov a δ je prechodová funkcia. Prechodovú funkciu najprv upravíme. Keďže ATS je v binárnom normálnom tvare, pravá strana δ-funkcie môže mať jeden alebo dva prvky. V prípade, že obsahuje dva prvky, vytvoríme kópiu riadku δ-funkcie, pričom z originálu odstránime druhý prvok pravej strany a z kópie odstránime prvý prvok pravej strany. Takto dostaneme rozvinutý

tvar δ-funkcie, ktorý má na pravej strane vždy iba jeden prvok. Touto úpravou sa počet riadkov δ-funkcie zvýšil o najviac $c \cdot \mid K_\forall \mid$, kde c je najväčší počet výskytov univerzálneho stavu na ľavej strane pravidla δ-funkcie, t.j. o konečný počet. Ďalej pre každý riadok

$$\delta(q_{p_j}, B) = \{(q_{r_j}, b_{t_j}, d_j)\}$$

pridáme riadok

$$\delta(q_{p_j}, \Diamond) = \{(q_{r_j}, b_{t_j}, d_j)\}$$

Dôsledok tejto úpravy je to, že symbol $\Diamond$ reprezentuje $BLANK$ z pohľadu δ-funkcie. Počet riadkov δ-funkcie sa opäť asymptoticky nezmenil. Po úpravách vyzerá δ-funkcia nasledovne:

$$\begin{aligned}
\delta(q_{p_1}, b_{s_1}) &= \{(q_{r_1}, b_{t_1}, d_1)\} \\
\delta(q_{p_2}, b_{s_2}) &= \{(q_{r_2}, b_{t_2}, d_2)\} \\
&\vdots \\
\delta(q_{p_g}, b_{s_g}) &= \{(q_{r_g}, b_{t_g}, d_g)\}
\end{aligned}$$

K alternujúcemu Turingovmu stroju A skonštruujeme ekvivalentný G-systém, pracujúci v rovnakom čase. $G = (N, T, M, S)$, kde $N = \{S, (1), (2), (3), (A), (B), \#, [\#], (\#), [], \Diamond\} \cup \{q_i \mid q_i \in K\}$, $T = \Gamma \cup \{B\}$, $M = (K_M, N \cup T, N \cup T, H, q_0, F_M)$.

$K_M = \{q_0, q_{(1)}, q_{1,2}, q_{(2)}, q_{2,3}, q_{(3)}, q_{(A)}, q_F, q_{(F)}, q_a, q_{(a)}, q_\#, q_{[\#]}, q_{(\#)}, q_{[]}, q_{[][]}\} \cup \bigcup_{i=1}^{14} K_{M_i}$

$K_{M_1} = \{q_{a_i} \mid a_i \in \Sigma\}$

$K_{M_2} = \{q_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)} \mid 1 \leq j \leq g\}$

$K_{M_3} = \{\bar{q}_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)} \mid 1 \leq j \leq g\}$

$K_{M_4} = \{q_{b_i}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)} \mid 1 \leq j \leq g, b_i \in T \cup \{(B)\}\}$

$K_{M_5} = \{\bar{q}_{b_i}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)} \mid 1 \leq j \leq g, b_i \in T \cup \{(B)\}\}$

$K_{M_6} = \{q_\forall^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)} \mid 1 \leq j \leq g\}$

$K_{M_7} = \{\bar{q}_\forall^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)} \mid 1 \leq j \leq g\}$

$K_{M_8} = \{q_\exists^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)} \mid 1 \leq j \leq g\}$

$K_{M_9} = \{q_{d_j=-1}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)} \mid 1 \leq j \leq g\}$

$K_{M_{10}} = \{\bar{q}_{d_j=-1}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)} \mid 1 \leq j \leq g\}$

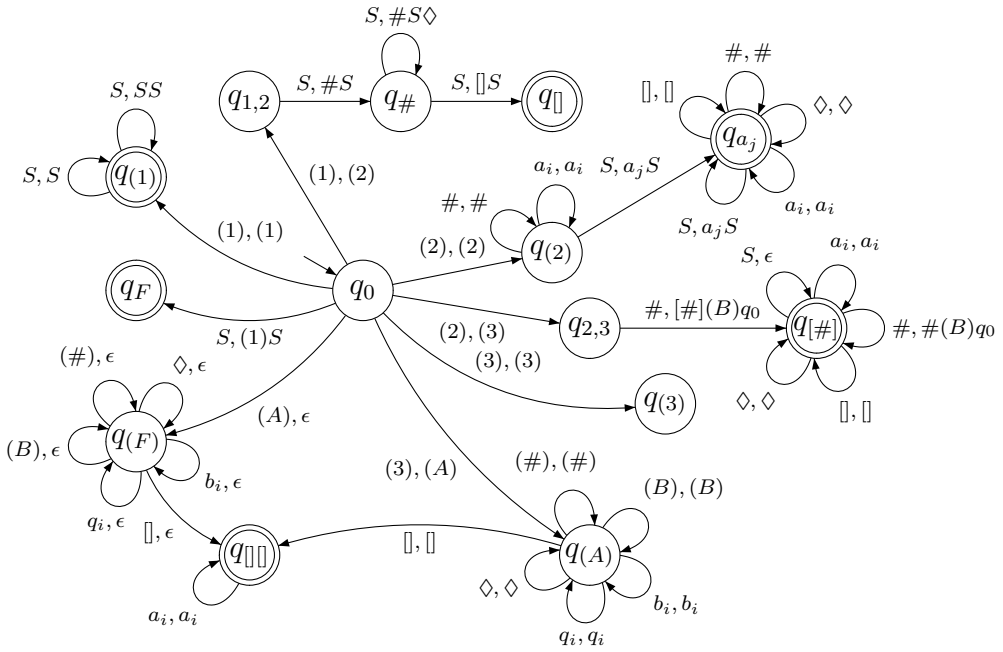


Fig. 2.1: Kontrolná časť G-systému

$$K_{M_{11}} = \{q_{d_j=0}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 0)} \mid 1 \leq j \leq g\}$$

$$K_{M_{12}} = \{\bar{q}_{d_j=0}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 0)} \mid 1 \leq j \leq g\}$$

$$K_{M_{13}} = \{q_{d_j=1}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 1)} \mid 1 \leq j \leq g\}$$

$$K_{M_{14}} = \{\bar{q}_{d_j=1}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 1)} \mid 1 \leq j \leq g\}$$

$$F_M = \{q_{(1)}, q_F, q_{[\#]}, q_{[]}, q_{[][]}\} \cup \{q_{a_i} \mid a_i \in \Sigma\}$$

Notácia: štvorice množiny H sú kvôli prehľadnosti rozdelené do skupín, podľa príslušnosti k fáze. Skupina 1 patrí do inicializácie, skupina 2 do prvej hlavnej fázy, skupina 3 do prvej medzifázy, skupina 4 do druhej hlavnej fázy, skupina 5 do druhej medzifázy, skupina 6 do tretej hlavnej fázy (táto fáza je natoľko rozsiahla, že je ďalej rozdelená do menších logických celkov), skupina 7 do tretej medzifázy a skupina 8 do finalizácie.

Notácia: zavedieme novú notáciu pre grafickú vizualizáciu 1-a prekladača. Podmienený prechod značíme prerušovanou šípkou, ktorá okrem transformačnej informácie (vstupný symbol, výstupné symboly) nesie aj podmienku zaznačenú v hranatej zátvorke (napr. $[d_j = 0]$). Sémantika tejto prerušovanej šípky (podmieneného prechodu) je, ak je podmienka splnená, tak je na jej mieste normálna šípka (normálny prechod), ak nie, tak tam šípka nie je.

Následne formálne popíšeme množinu H. Význam každej štvorice množiny H je obsiahnutý v dôkaze správnosti konštrukcie, ktorý nasleduje hneď za formálnym popisom množiny H. Pre lepšie pochopenie popisujeme množinu H aj formou obrázkov (obrázky 2.1 - 2.7).

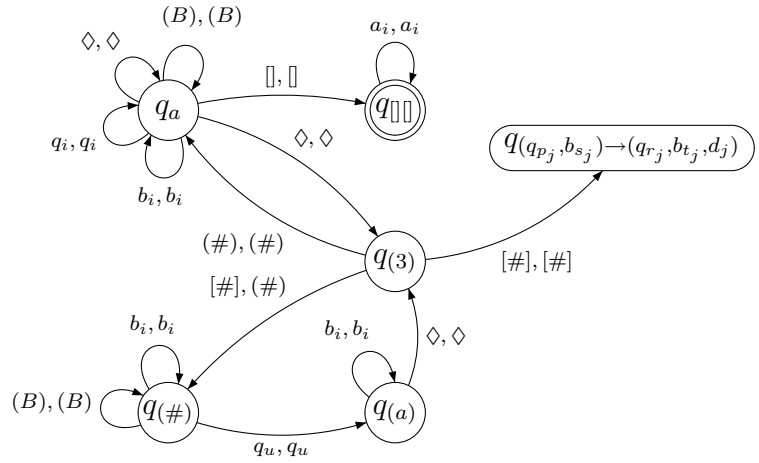


Fig. 2.2: Simulačná časť G-systému

$H = \{$

1. Inicializácia - predpríprava vetnej formy
   (a) $(q_0, S, (1)S, q_F)$
2. Prvá hlavná fáza - množenie neterminálu S
   (a) $(q_0, (1), (1), q_{(1)})$
   (b) $(q_{(1)}, S, S, q_{(1)})$
   (c) $(q_{(1)}, S, SS, q_{(1)})$
3. Prvá medzifáza - vloženie oddeľovačov
   (a) $(q_0, (1), (2), q_{1,2})$
   (b) $(q_{1,2}, S, \#S, q_\#)$
   (c) $(q_\#, S, \#S\Diamond, q_\#)$
   (d) $(q_\#, S, []S, q_{[]})$
4. Druhá hlavná fáza - synchronizovaná generácia slova
   (a) $(q_0, (2), (2), q_{(2)})$
   (b) $(q_{(2)}, \#, \#, q_{(2)})$
   (c) $(q_{(2)}, a_i, a_i, q_{(2)}) \mid a_i \in \Sigma$
   (d) $(q_{(2)}, S, a_iS, q_{a_i}) \mid a_i \in \Sigma$
   (e) $(q_{a_i}, [], [], q_{a_i}) \mid a_i \in \Sigma$
   (f) $(q_{a_i}, \#, \#, q_{a_i}) \mid a_i \in \Sigma$

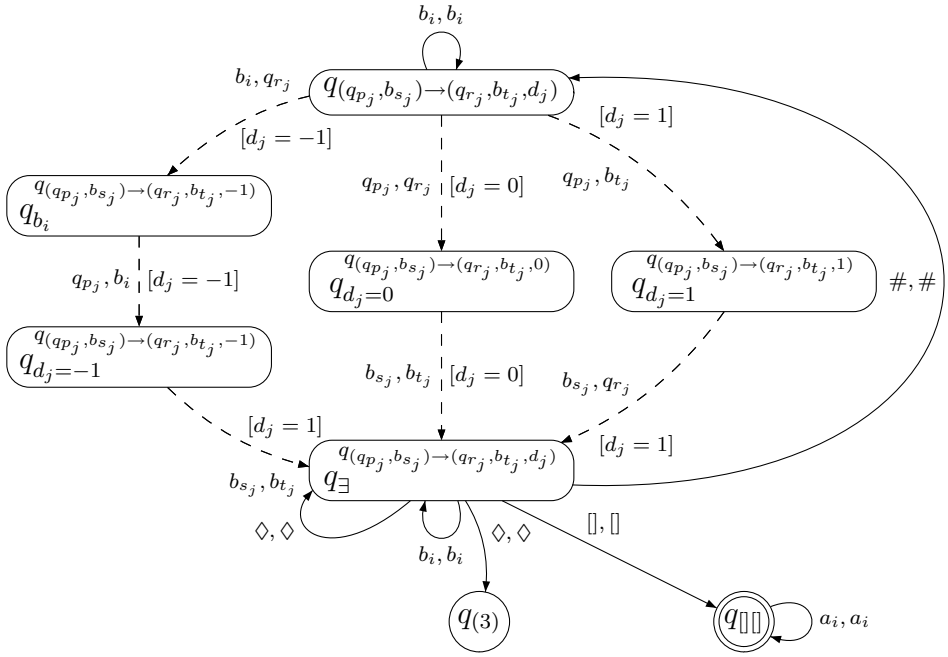


Fig. 2.3: Časť G-systému realizujúca δ-funkciu pre existenčné stavy

(g) $(q_{a_i}, \Diamond, \Diamond, q_{a_i}) \mid a_i \in \Sigma$

(h) $(q_{a_i}, a_j, a_j, q_{a_i}) \mid a_i \in \Sigma, a_j \in \Sigma$

(i) $(q_{a_i}, S, a_i S, q_{a_i}) \mid a_i \in \Sigma$

5. Druhá medzifáza - vytvorenie konfigurácií

(a) $(q_0, (2), (3), q_{2,3})$

(b) $(q_{2,3}, \#, [\#](B)q_0, q_{[\#]})$

(c) $(q_{[\#]}, S, \epsilon, q_{[\#]})$

(d) $(q_{[\#]}, a_i, a_i, q_{[\#]})$

(e) $(q_{[\#]}, \#, \#(B)q_0, q_{[\#]})$

(f) $(q_{[\#]}, [], [], q_{[\#]})$

(g) $(q_{[\#]}, \Diamond, \Diamond, q_{[\#]})$

6. Tretia hlavná fáza - simulácia

(a) $(q_0, (3), (3), q_{(3)})$

(b) Výber δ - funkcie, uloženie do stavu, simulácia δ - funkcie (spoločná časť pre existenčný aj univerzálny stav)

i. $(q_{(3)}, [\#], [\#], q_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g, \exists\delta(q_{p_j}, b_{s_j}) = \{(q_{r_j}, b_{t_j}, d_j)\}$

ii. $(q_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, b_i, b_i, q_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g, b_i \in T \cup \{(B)\}$

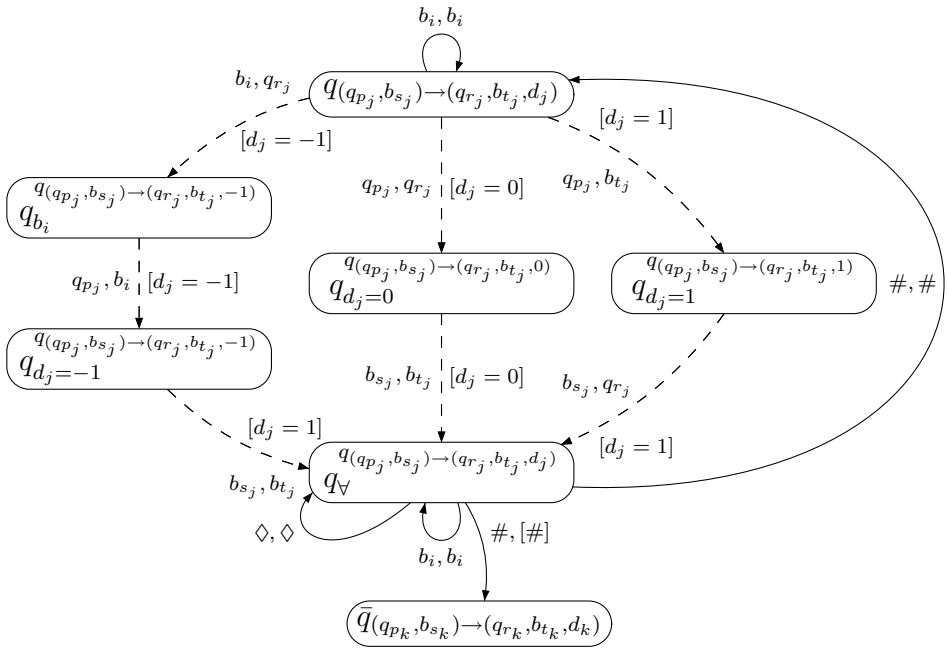


Fig. 2.4: Časť G-systému realizujúca δ-funkciu pre univerzálne stavy, časť prvá

iii. $(q_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, b_i, q_{r_j}, q_{b_i}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \leq j \leq g, b_i \in T$

iv. $(q_{b_i}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, q_{p_j}, b_i, q_{d_j=-1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \leq j \leq g, b_i \in T$

v. $(q_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}, q_{p_j}, q_{r_j}, q_{d_j=0}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}) \mid 1 \leq j \leq g$

vi. $(q_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, q_{p_j}, b_{t_j}, q_{d_j=1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g, b_{s_j} \notin \{\Diamond\}$

vii. $(q_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, (B), (B)q_{r_j}, q_{(B)}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \leq j \leq g$

viii. $(q_{(B)}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, q_{p_j}, B, q_{d_j=-1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \leq j \leq g$

ix. $(q_{(q_{p_j},\Diamond)\to(q_{r_j},b_{t_j},1)}, q_{p_j}, b_{t_j}, q_{d_j=1}^{(q_{p_j},\Diamond)\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g$

(c) Simulácia δ - funkcie (časť špecifická pre existenčný stav)

i. $(q_{d_j=-1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, b_{s_j}, b_{t_j}, q_\exists^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_\exists$

ii. $(q_{d_j=0}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}, b_{s_j}, b_{t_j}, q_\exists^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}) \mid 1 \leq j \leq g, q_{p_j} \in K_\exists$

iii. $(q_{d_j=1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, b_{s_j}, q_{r_j}, q_\exists^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_\exists$

iv. $(q_{d_j=1}^{(q_{p_j},\Diamond)\to(q_{r_j},b_{t_j},1)}, \Diamond, q_{r_j}\Diamond, q_\exists^{(q_{p_j},\Diamond)\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_\exists$

v. $(q_\exists^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, b_i, b_i, q_\exists^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g, b_i \in T$

vi. $(q_\exists^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \Diamond, \Diamond, q_\exists^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g$

vii. $(q_\exists^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \#, \#, q_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g$

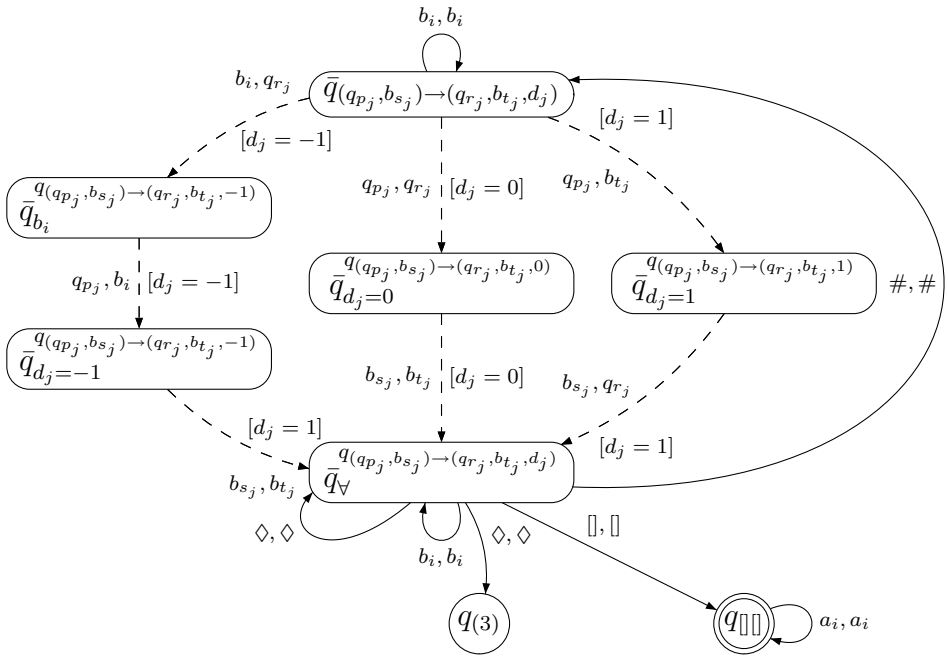


Fig. 2.5: Časť G-systému realizujúca δ-funkciu pre univerzálne stavy, časť druhá

viii. $(q_{\exists}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}, [], [], q_{[][]}) \mid 1 \leq j \leq g$

ix. $(q_{\exists}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}, \Diamond, \Diamond, q_{(3)}) \mid 1 \leq j \leq g$

(d) Simulácia δ - funkcie (časť špecifická pre univerzálny stav, pred aktivovaním úseku)

i. $(q_{d_j=-1}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}, b_{s_j}, b_{t_j}, q_{\forall}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\forall}$

ii. $(q_{d_j=0}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 0)}, b_{s_j}, b_{t_j}, q_{\forall}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 0)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\forall}$

iii. $(q_{d_j=1}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 1)}, b_{s_j}, q_{r_j}, q_{\forall}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\forall}$

iv. $(q_{d_j=1}^{(q_{p_j}, \Diamond) \to (q_{r_j}, b_{t_j}, 1)}, \Diamond, q_{r_j} \Diamond, q_{\forall}^{(q_{p_j}, \Diamond) \to (q_{r_j}, b_{t_j}, 1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\forall}$

v. $(q_{\forall}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}, b_i, b_i, q_{\forall}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}) \mid 1 \leq j \leq g, b_i \in T$

vi. $(q_{\forall}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}, \Diamond, \Diamond, q_{\forall}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}) \mid 1 \leq j \leq g$

vii. $(q_{\forall}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}, \#, \#, q_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}) \mid 1 \leq j \leq g$

(e) Simulácia δ - funkcie (časť špecifická pre univerzálny stav, po aktivovaní úseku)

i. $(q_{\forall}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}, \#, [\#], \bar{q}_{(q_{p_k}, b_{s_k}) \to (q_{r_k}, b_{t_k}, d_k)}) \mid 1 \leq j \leq g, 1 \leq k \leq g, j \neq k$

ii. $(\bar{q}_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}, b_i, b_i, \bar{q}_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}) \mid 1 \leq j \leq g, b_i \in T \cup \{(B)\}$

iii. $(\bar{q}_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}, b_i, q_{r_j}, \bar{q}_{b_i}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}) \mid 1 \leq j \leq g, b_i \in T$

iv. $(\bar{q}_{b_i}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}, q_{p_j}, b_i, \bar{q}_{d_j=-1}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}) \mid 1 \leq j \leq g, b_i \in T$

v. $(\bar{q}_{d_j=-1}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}, b_{s_j}, b_{t_j}, \bar{q}_{\forall}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\forall}$

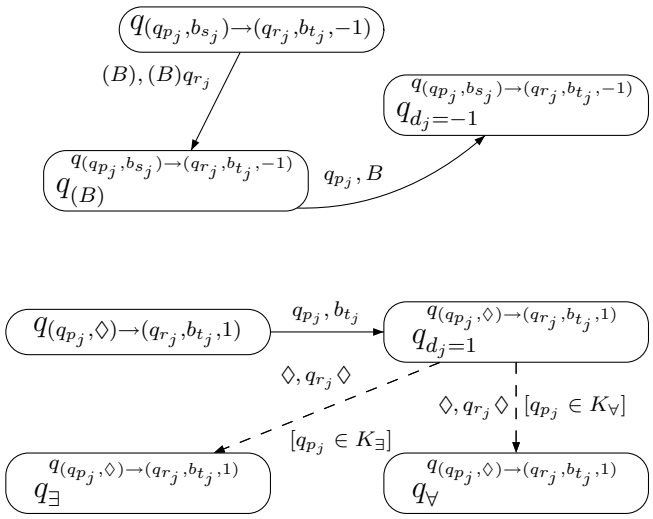


Fig. 2.6: Časť G-systému realizujúca δ-funkciu pre krajné prípady konfigurácie, časť prvá

vi. $(\bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}, q_{p_j}, q_{r_j}, \bar{q}_{d_j=0}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}) \mid 1 \leq j \leq g$

vii. $(\bar{q}_{d_j=0}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}, b_{s_j}, b_{t_j}, \bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}) \mid 1 \leq j \leq g, q_{p_j} \in K_\forall$

viii. $(\bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, q_{p_j}, b_{t_j}, \bar{q}_{d_j=1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g$

ix. $(\bar{q}_{d_j=1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, b_{s_j}, q_{r_j}, \bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_\forall,$
$b_{s_j} \notin \{\Diamond\}$

x. $(\bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, (B), (B)q_{r_j}, \bar{q}_{(B)}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \leq j \leq g$

xi. $(\bar{q}_{(B)}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, q_{p_j}, B, \bar{q}_{d_j=-1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \leq j \leq g$

xii. $(\bar{q}_{(q_{p_j},\Diamond)\to(q_{r_j},b_{t_j},1)}, q_{p_j}, b_{t_j}, \bar{q}_{d_j=1}^{(q_{p_j},\Diamond)\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g$

xiii. $(\bar{q}_{d_j=1}^{(q_{p_j},\Diamond)\to(q_{r_j},b_{t_j},1)}, \Diamond, q_{r_j}\Diamond, \bar{q}_{\forall}^{(q_{p_j},\Diamond)\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_\forall$

xiv. $(\bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, b_i, b_i, \bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g, b_i \in T$

xv. $(\bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \Diamond, \Diamond, \bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g$

xvi. $(\bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \#, \#, \bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g$

xvii. $(\bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, [], [], q_{[][]}) \mid 1 \leq j \leq g$

xviii. $(\bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \Diamond, \Diamond, q_{(3)}) \mid 1 \leq j \leq g$

(f) $(q_{[][]}, a_i, a_i, q_{[][]}) \mid a_i \in \Sigma$

(g) Preskočenie akceptačných úsekov (kopírovacie cykly)

i. $(q_{(3)}, (\#), (\#), q_a)$

ii. $(q_a, b_i, b_i, q_a) \mid b_i \in T$

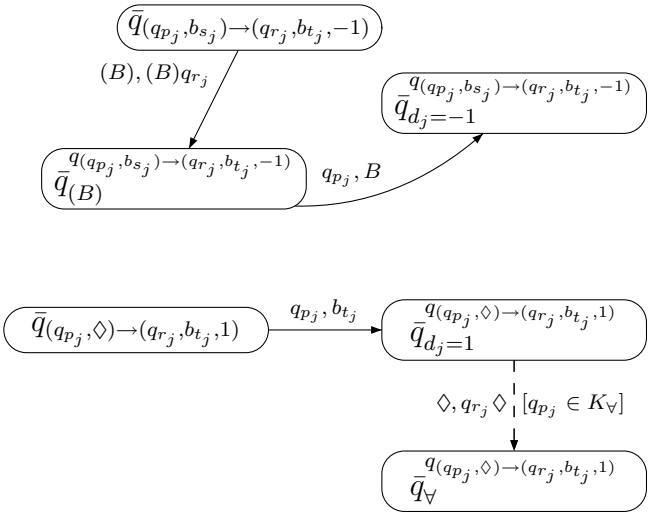


Fig. 2.7: Časť G-systému realizujúca δ-funkciu pre krajné prípady konfigurácie, časť druhá

iii. $(q_a, q_i, q_i, q_a) \mid q_i \in K$
iv. $(q_a, \Diamond, \Diamond, q_a)$
v. $(q_a, (B), (B), q_a)$
vi. $(q_a, [], [], q_{[][]})$
vii. $(q_a, \Diamond, \Diamond, q_{(3)})$

(h) Transformácia aktívneho úseku na akceptačný

i. $(q_{(3)}, [\#], (\#), q_{(\#)})$
ii. $(q_{(\#)}, b_i, b_i, q_{(\#)}) \mid b_i \in T$
iii. $(q_{(\#)}, (B), (B), q_{(\#)})$
iv. $(q_{(\#)}, q_u, q_u, q_{(a)}) \mid q_u \in F_M$
v. $(q_{(a)}, b_i, b_i, q_{(a)}) \mid b_i \in T$
vi. $(q_{(a)}, \Diamond, \Diamond, q_{(3)})$

7. Tretia medzifáza - kontrola spávnosti akceptácie úsekov

(a) $(q_0, (3), (A), q_{(A)})$
(b) $(q_{(A)}, (\#), (\#), q_{(A)})$
(c) $(q_{(A)}, (B), (B), q_{(A)})$
(d) $(q_{(A)}, b_i, b_i, q_{(A)}) \mid b_i \in T$
(e) $(q_{(A)}, q_i, q_i, q_{(A)}) \mid q_i \in K$
(f) $(q_{(A)}, \Diamond, \Diamond, q_{(A)})$
(g) $(q_{(A)}, [], [], q_{[][]})$

8. Finalizácia - vymazanie pracovnej časti vetnej formy

   (a) $(q_0, (A), \epsilon, q_{(F)})$

   (b) $(q_{(F)}, (\#), \epsilon, q_{(F)})$

   (c) $(q_{(F)}, b_i, \epsilon, q_{(F)}) \mid b_i \in T$

   (d) $(q_{(F)}, q_i, \epsilon, q_{(F)}) \mid q_i \in K$

   (e) $(q_{(F)}, (B), \epsilon, q_{(F)})$

   (f) $(q_{(F)}, \Diamond, \epsilon, q_{(F)})$

   (g) $(q_{(F)}, [], \epsilon, q_{[][]})$

}

Dôkaz správnosti konštrukcie

Lema 2.1.1: Fázy generovania G-systému prebehnú v tomto poradí: inicializácia, prvá hlavná fáza, prvá medzifáza, druhá hlavná fáza, druhá medzifáza, tretia hlavná fáza, tretia medzifáza a finalizácia. Žiadne iné poradie nie je možné.

Dôkaz: Na začiatku generovania sa vetná forma skladá iba z neterminálu S a začíname v stave q_0. V množine H je viacero štvoríc, ktoré majú v prvej komponente stav q_0. Sú to štvorice 1(a), 2(a), 3(a), 4(a), 5(a), 6(a), 7(a) a 8(a). Vstup sa skladá iba z neterminálu S a iba jediná štvorica má tento neterminál v druhej komponente - štvorica 1(a). Teda G-systém nemá na výber inú štvoricu ako 1(a), ktorá reprezentuje inicializáciu. V inicializácii vytvoríme špeciálny symbol (1) na začiatku vetnej formy, ktorý bude udržovať informáciu o aktuálnej fáze a zablokuje zopakovanie inicializácie. Tento údaj sa môže meniť len počas medzifáz, nemôže sa meniť počas hlavných fáz. Po prebehnutí inicializácie sa aj skončí prvý prechod vetnou formou, nakoľko vetná forma pozostávala len z jediného symbolu.

Tvar vetnej formy po inicializácii:

$$(1)S$$

V druhom prechode máme opäť na výber štvorice 1(a), 2(a), 3(a), 4(a), 5(a), 6(a), 7(a) a 8(a). Tentokrát na začiatku vetnej formy máme informáciu o hlavnej fáze - špeciálny symbol (1). Do úvahy prichádzajú štvorice 2(a) a 3(a). G-systém musí vybrať ako prvú štvoricu 2(a), v ďalších prechodoch má na výber. Iné poradie vedie k zaseknutiu, ako sa ukáže v ďalšej medzifáze. Medzifázy slúžia nielen na predprípravu vetnej formy, ale majú aj kontrolný význam. G-systém spraví niekoľko prechodov vetnou formou, vyberajúc si štvoricu 2(a). Toto reprezentuje prvú hlavnú fázu a množenie neterminálu S na ľubovoľný počet. Výberom štvorice 3(a) ukončí prvú hlavnú fázu a rozhodne sa prejsť do prvej medzifázy. Výber štvorice 3(a) musí raz nastať, v opačnom prípade nevedie odvodenie k terminálnej vetnej forme, nakoľko štvorica 2(a) a na ňu nadväzujúce štvorice 2(b) a 2(c) neobsahujú terminály.

Tvar vetnej formy po prvej hlavnej fáze:

$$(1)S^+$$

Štvorica 3(a) prepíše symbol (1) na (2), teda vojde do prvej medzifázy (zablokuje návrat do prvej hlavnej fázy). Naväzujúce štvorice 3(b),3(c) a 3(d) povkladajú ďalšie špeciálne symboly medzi neterminály S a skontrolujú, či sa vo vetnej forme nachádzajú aspoň dva neterminály S. Ak je vo vetnej forme iba jeden S, tak vstup sa dočíta v stave $q_{\#}$, ktorý ale nie je akceptačný, teda dôjde k zaseknutiu. Nesprávne uhádnutie posledného S, na ktorý treba aplikovať štvoricu 3(d), vedie tiež k zaseknutiu.

Tvar vetnej formy po prvej medzifáze:

$$(2)(\#S\Diamond)^+[]S$$

Definujeme "úsek", v ďalšom ho budeme často používať. Úsek je časť vetnej formy začínajúci symbolom $\#$, $[\#]$ alebo $(\#)$, za ktorým hneď nasleduje symbol (B) a končiaci symbolom $\Diamond$, neobsahujúci ďalšie špeciálne symboly. Špeciálny úsek je časť vetnej formy začínajúci symbolom $[]$ a siahajúcim až po koniec vetnej formy.

Pred druhou fázou má G-systém opäť dve možnosti výberu. Štvorica 4(a) a 5(a) vyhovujú stavu a vstupu. V prípade štvorice 4(a) sa prejde do druhej hlavnej fázy, ktorá má za úlohu synchronizovane vygenerovať rovnaké slovo, na ktorom bude prebiehať výpočet, na každom úseku (aj na špeciálnom úseku). Po vygenerovaní tohoto slova sa niekedy G-systém musí rozhodnúť použiť štvoricu 5(a), a tým ukončiť túto fázu. Ak tak neurobí, nevygeneruje sa terminálny výstup, nakoľko vetná forma obsahuje neterminál S a štvorica 4(a) a naväzujúce štvorice (4(b)-4(i)) nevedia S vymazať. Je prípustné vybrať hneď štvoricu 5(a) a štvoricu 4(a) vôbec nevybrať, teda celú druhú hlavnú fázu preskočiť. V tomto prípade je pracovné slovo ϵ.

Tvar vetnej formy po druhej hlavnej fáze:

$$(2)(\#wS\Diamond)^+[]wS \mid w \in \Sigma^*$$

Druhá medzifáza začne vybratím štvorice 5(a), t.j. prepíše symbol (2) na (3) (zablokuje návrat do druhej hlavnej fázy). Naväzujúce štvorice (5(b)-5(g)) vymažú zo všetkých úsekov (aj špeciálneho úseku) neterminál S, naviac za každý symbol $\#$ vložia symbol q_0, čo reprezentuje stav a bičík konfigurácie ATS. Špeciálne aktivuje prvý úsek prepísaním jeho začiatočného symbolu $\#$ na $[\#]$ (štvorica 5(b)). O aktiváciách úsekov si povieme viac neskôr. Úlohou druhej medzifázy je predpripraviť vetnú formu na simuláciu výpočtu ATS, tým že obsah úsekov zmení na konfigurácie. Do špeciálneho úseku na konci vetnej formy nie je vložená konfigurácia.

Tvar vetnej formy po druhej medzifáze:

$$(3)[\#](B)q_0w\Diamond(\#(B)q_0w\Diamond)^*[]w \mid w \in \Sigma^*$$

Pred treťou fázou je na začiatku vetnej formy symbol (3). Vyhovujú dve štvorice: 6(a) a 7(a). Uvažujme prípad, kedy si vyberieme 6(a), t.j. prejdeme do tretej fázy. Jeden prechod treťou fázou zodpovedá jednému kroku výpočtu ATS. V tejto fáze sa simuluje δ-funkcia, odvetvovanie a akceptácie procesov. Bližšie sa k nej vrátime neskôr. Dôležité je, že v tretej fáze musí G-systém zotrvať, kým neodsimuluje celý akceptačný výpočet ATS, t.j. všetky úseky skončia v akceptačných konfiguráciách. Ak by skončil predčasne prechodom pomocou štvorice 7(a), dostal by sa do stavu $q_{(A)}$. V tomto stave sa prečíta celá vetná forma (štvorice 7(b)-7(f)), ak tam existuje aktívny alebo neaktívny úsek (symboly $\#$ alebo $[\#]$), dôjde k zaseknutiu. V prípade, že si hneď v prvom prechode zvolí štvoricu 7(a), dôjde určite k zaseknutiu, lebo vetná forma určite obsahuje práve jeden aktívny úsek a (možno) veľa neaktívnych úsekov. V prípade, že G-systém háda, že všetky úseky už akceptujú, vyberie štvoricu 7(a) a ukončí fázu. G-systém musí tak raz urobiť, inak nikdy nevygeneruje terminálne slovo (nezbaví sa špeciálnych symbolov $\#$, $[\#]$, $(\#)$, (B), $\Diamond$ a $[]$).

Tvar vetnej formy po tretej hlavnej fáze:

$$(3)(\#)(B)w^1_{p_1}q_{p_1}w^2_{p_1}\Diamond(\#)(B)w^1_{p_2}q_{p_2}w^2_{p_2}\Diamond\ldots(\#)(B)w^1_{p_s}q_{p_s}w^2_{p_s}\Diamond[]w$$

$$\mid w^j_{p_i} \in \Gamma^*, 1 \leq i \leq s, 1 \leq j \leq 2, w \in L(A)$$

Štvorica 7(a) prepíše symbol (3) na (A), t.j. ukončí tretiu hlavnú fázu (zablokuje návrat do tretej hlavnej fázy) a začne tretia medzifáza - kontrola akceptácie. V tejto fáze, pomocou štvoríc 7(b)-7(f) prejdeme celú vetnú formu až po symbol $[]$. Pokiaľ nie sú všetky úseky akceptujúce, tak dôjde k zaseknutiu. Zvyšok vetnej formy (po symbole $[]$) len dočítame (štvorice 7(g) a 7(f)).

Tvar vetnej formy po tretej medzifáze:

$$(A)(\#)(B)w^1_{p_1}q_{p_1}w^2_{p_1}\Diamond(\#)(B)w^1_{p_2}q_{p_2}w^2_{p_2}\Diamond\ldots(\#)(B)w^1_{p_s}q_{p_s}w^2_{p_s}\Diamond[]w$$

$$\mid w^j_{p_i} \in \Gamma^*, 1 \leq i \leq s, 1 \leq j \leq 2, w \in L(A)$$

Špeciálny symbol (A) na začiatku vetnej formy nám umožňuje vybrať iba štvoricu 8(a), pomocou ktorej sa dostaneme do stavu $q_{(F)}$. V tomto stave, pomocou štvoríc 8(b)-8(f), zmažeme pracovnú časť vetnej formy (až po symbol $[]$). Štvorica 8(g) dostane G-systém do kopírovacieho stavu, ktorý už zvyšok vetnej formy nezmení, pričom ale zmaže symbol $[]$. Takto dostaneme terminálne výstupné slovo. Úlohou finalizácie je iba upraviť slovo na výstup.

Tvar vetnej formy po finalizácii:

$$w \mid w \in L(A)$$

G-systém nemôže vykonať ďalší prechod vetnou formou, lebo na vstupe má iba terminály a žiadna zo štvoríc, majúca q_0 v prvej komponente, nemá v druhej komponente terminál. Neformálne povedané: z q_0 nemáme prechod na terminál a dôjde k zaseknutiu. Teraz je už zrejmé, že pri generovaní slova musí G-systém dodržať predpísané poradie fáz.

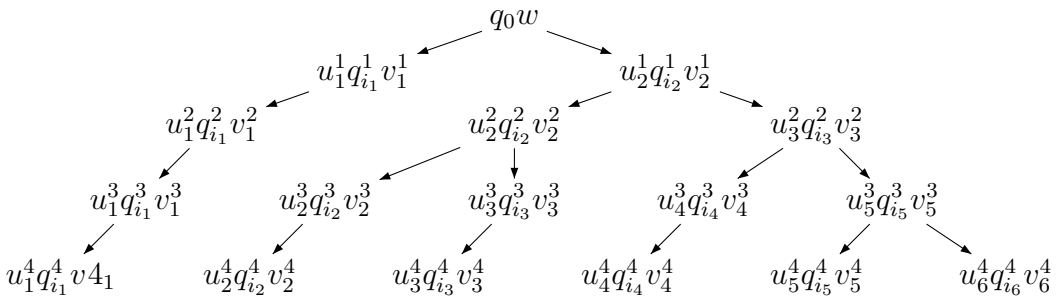
Definícia 2.1.1: Hovoríme, že vetná forma $u \cong (u_1, u_2, \ldots, u_n)$ (u korešponduje s globálnym stavom ATS $(u_1, u_2, \ldots, u_n)$) práve vtedy, keď $Trans(u) = \triangle(B)u_1\Diamond\triangle(B)u_2\Diamond\ldots\triangle(B)u_n\Diamond$, kde $\triangle \in \{[\#], (\#)\}$.

$Trans$ je zobrazenie, ktoré zobrazí niektoré časti vetnej formy do ϵ.

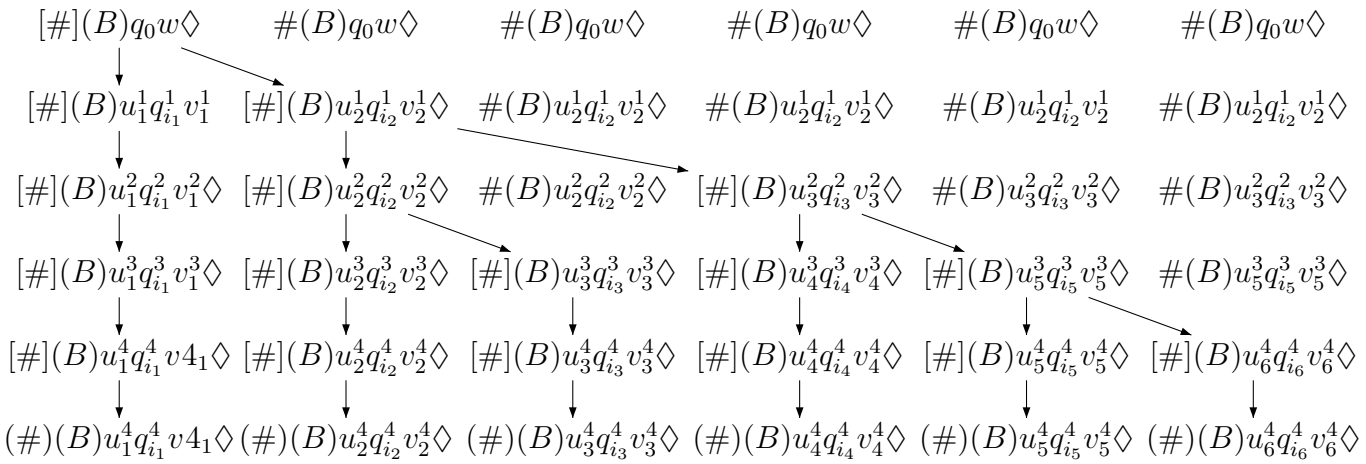
- $(3) \rightarrow \epsilon$ (zmažeme začiatočný symbol vetnej formy)
- $[]w \rightarrow \epsilon \mid w \in \Sigma^*$ (zmažeme výstupnú časť vetnej formy)
- $\#(B)w^1_{p_i}q_{p_i}w^2_{p_i}\Diamond \rightarrow \epsilon \mid w^j_{p_i} \in \Gamma^* 1 \leq i \leq s, 1 \leq j \leq 2$ (zmažeme neaktívne úseky, nakoľko zatiaľ nenesú žiadnu informáciu, dôležitú pre globálny stav ATS)

Neformálne povedané, vetná forma G-systému obsahuje celý globálny stav ATS a naviac ho vieme ľahko zrekonštruovať. Stačí vymazať niektoré časti vetnej formy; na toto nám slúži zobrazenie $Trans$, ktoré sa dá ľahko realizovať jednoduchým a-prekladačom a jediným prechodom vetnou formou. Jeden krok odvodenia v tretej hlavnej fáze reprezentuje jeden krok výpočtu ATS. Teda vetná forma krok po kroku korešponduje s globálnym stavom ATS. Naším ďalším snažením bude dokázať toto tvrdenie.

V ďalšom ilustrujeme korešpondenciu medzi vetnou formou G-systému a globálnym stavom ATS počas akceptačného výpočtu. Uvažujme akceptačný výpočet ATS nad slovom w. Prepokladáme, že ak niektorý proces skončí a akceptuje, ďalej už nemení svoju konfiguráciu. Pre jednoduchosť, v našom výpočte dôjde k akceptácii naraz, v konštrukcii túto vlastnosť nepredpokladáme. Stavy $q^4_{i_j}, 1 \leq j \leq 6$ sú akceptačné.



Teraz si ukážeme ako vyzerajú vetné formy G-systému v tretej hlavnej fáze, v ktorej sa simuluje tento akceptačný výpočet ATS. Pre jasnejšie pochopenie je simulácia odvetvovania procesov zvýraznená, ľahko vidno, že odvetvovanie má rovnakú stromovú štruktúru ako pri výpočte ATS.



Definujeme "postupnosť úsekov". Postupnosť úsekov sa skladá aspoň z jedného aktívneho úseku, ktorý je vždy ako prvý (najľavejší) a (možno) z viacerých neaktívnych úsekov. Táto postupnosť neobsahuje akceptujúce úseky.

Lema 2.1.2: Ak existuje akceptačný výpočet ATS na slove w, potom existuje aj odvodenie G-systému, ktorého vetná forma má na začiatku tretej hlavnej fázy tvar

$$(3)[\#](B)q_0w\Diamond(\#(B)q_0w\Diamond)^*[]w$$

a naviac počas trvania tretej hlavnej fázy platí, že sa skladá z viacerých (aspoň jednej) postupností úsekov, medzi ktorými môžu byť akceptujúce úseky.

Dôkaz: Indukciou vzhľadom na počet krokov odvodenia od začiatku tretej hlavnej fázy.

1. Báza indukcie triviálne platí, nakoľko na začiatku simulácie (tretej hlavnej fázy) je vetná forma tvaru $(3)[\#](B)q_0w\Diamond(\#(B)q_0w\Diamond)^*[]w \mid w \in \Sigma^*$ podľa lemy 2.1.1. Teda celá vetná forma je jediná postupnosť úsekov.

2. Predpokladajme, že pre prvých k krokov odvodenia tvrdenie platí. Dokážeme tvrdenie pre $k+1$-vý krok.
Po k krokoch odvodenia vetná forma spĺňa naše tvrdenie. Uvažujme teraz pre jednu postupnosť úsekov všetky prípady. Pre ostatné postupnosti úsekov je dôkaz obdobný.

Najprv si všimnime, že keďže vetná forma spĺňa podmienku nášho tvrdenia, nemôže sa stať, že nejaké akceptujúce úseky sa nachádzajú vnútri našej postupnosti úsekov. Nakoľko sa poradie úsekov počas celej simulácie nemení, nemôže sa stať, že by sa zámenou poradia úsekov dostal akceptačný úsek do postupnosti úsekov. Jediný spôsob, akým by sa tam mohol akceptačný úsek dostať, je prípad, keď by sme nejaký úsek už patriaci do postupnosti, zmenili na akceptačný.

Po prečítaní znaku (3) sa G-systém dostane do stavu $q_{(3)}$ (štvorica 6(a)). Do úvahy prichádza viacero možností.

Najjednoduchšia z nich je preskočenie akceptujúceho úseku (štvorice 6(g)[i-vii]), v prípade, že práve čítaný úsek je akceptujúci (začína znakom $(\#)$). Vetná forma sa nemení, teda táto možnosť podmienku nenarúša.

V prípade, že ide o aktívny úsek (začína symbolom $[\#]$), sú dve možnosti.

Prvá možnosť je vybrať si časť δ-funkcie a simulovať ju (štvorica 6(b)[i-ix]). Skupina štvoríc 6(b) obsahuje len spoločnú časť simulácie δ-funkcie pre existenčné a aj univerzálne stavy. Tu rozlišujeme dva prípady, každý má separátnu skupinu štvoríc, ktoré dokončia simuláciu δ-funkcie.

Ak je stav v aktívnom úseku existenčný, tak sa pomocou štvoríc 6(c)[i-ix] simuluje δ-funkcia a ostatné úseky postupnosti úsekov tiež len simulujú δ-funkciu. Žiadna z týchto štvoríc neaktivuje úsek, pretože nemení symboly začiatkov úsekov - $\#$ a $[\#]$, iba ich kopíruje. Teda podmienka zo znenia lemy sa zachováva.

V prípade, že stav je univerzálny, deje sa čosi podobné ako pri existenčnom stave (štvorice 6(d)[i-vii]), ale rozdiel je v tom, že jeden z neaktívnych úsekov sa určite aktivuje (štvorice 6(e)[i-xviii]), inak dôjde k zaseknutiu, a teda toto nemohlo byť odvodenie. Teda naša postupnosť

úsekov sa rozdelila na dve nové postupnosti v bode novoaktivovaného úseku. Pôvodná postupnosť až po novoaktivovaný úsek je prvá nová postupnosť a zvyšok pôvodnej postupnosti začínajúcej novoaktivovaným úsekom sa stane druhou novou postupnosťou úsekov. Nové postupnosti spĺňajú podmienku zo znenia lemy, lebo neboli vytvorené žiadne akceptujúce úseky.

Druhá možnosť je zmeniť aktívny úsek na akceptačný a skontrolovať oprávnenosť jeho akceptácie (štvorica 6(h)[i]). Ak úsek pri kontrole neobsahuje akceptačný stav, tak dôjde k zaseknutiu (štvorice 6(h)[ii-vi]). Ak sa stane, že touto zmenou spôsobíme porušenie podmienky zo znenia lemy, t.j. neaktívny úsek sa ocitne priamo za akceptujúcim, dôjde pri nasledujúcom prechode vetnou formou k zaseknutiu (zo stavu $q_{(3)}$ neexistuje prechod na $\#$), a teda toto nemohlo byť odvodenie. Znamená to, že sme vytvorili príliš veľa úsekov alebo boli zle uhádnuté aktivácie úsekov. Ale keďže existuje akceptačný výpočet ATS na w, existuje aj odvodenie G-systému, ktoré uhádne správny počet úsekov a správne aktivácie úsekov, nakoľko vieme vygenerovať ľubovoľný počet úsekov a aktivácie vieme spraviť v ľubovoľnom mieste, kde je to legálne.

Dokázali sme, že pri každom prípade, buď podmienka ostáva splnená, alebo nevyhnutne dôjde k zaseknutiu.

Lema 2.1.3: Počas simulácie (tretia hlavná fáza) je δ-funkcia ATS, korektne simulovaná G-systémom na konfiguráciách, pri existenčných aj univerzálnych stavoch.

Dôkaz: Vyplýva z konštrukcie a-prekladača.

- Ak $d_j = -1$, teda $\delta(q_{p_j}, b_{s_j}) = \{(q_{r_j}, b_{t_j}, -1)\}$, simulujeme pohyb vľavo tým, že uhádneme symbol, ktorý je tesne pred bičíkom q_{p_j}. Tento symbol uložíme do stavu a na jeho miesto napíšeme cieľový stav q_{r_j} δ-funkcie. Namiesto bičíka napíšeme uložený symbol a nasledujúci symbol je ten, ktorý bol δ-funkciou práve čítaný (b_{s_j}). Ten prepíšeme na cieľový symbol b_{t_j}.

- Ak $d_j = 0$, teda $\delta(q_{p_j}, b_{s_j}) = \{(q_{r_j}, b_{t_j}, 0)\}$, jednoducho prepíšeme stav na cieľový stav (q_{p_j} na q_{r_j}) a čítaný symbol na cieľový (b_{s_j} na b_{t_j}).

- Ak $d_j = 1$, teda $\delta(q_{p_j}, b_{s_j}) = \{(q_{r_j}, b_{t_j}, 1)\}$, postupujeme podobne ako v predchádzajúcom prípade, iba vymeníme pozíciu stavu a symbolu. Prepíšeme stav na cieľový symbol (q_{p_j} na b_{t_j}) a čítaný symbol na cieľový stav (b_{s_j} na q_{r_j}).

Na krajoch úseku je simulácia δ-funkcie trochu odlišná. Symboly (B) a $\Diamond$ sa správajú ako $BLANK$-y. Nechceme ich prepísať alebo vymazať, preto sa na ľavom kraji vytvorí symbol B a dôjde k posunutiu symbolu (B), a teda aj zväčšeniu vetnej formy. Na pravom kraji sa symbol $\Diamond$ tiež iba posunie a simuluje sa δ-funkcia, ako keby čítala $BLANK$.

Lema 2.1.4: δ-funkcia je korektne simulovaná na celej postupnosti úsekov, začínajúca aktívnym úsekom, ktorý obsahuje existenčný stav.

Dôkaz: Vyplýva z konštrukcie a-prekladača.

G-systém je v stave $q_{(3)}$, keď začne čítať postupnosť úsekov - najľavejší aktívny úsek, začínajúci symbolom $[\#]$. V tomto prechode si vyberie časť δ-funkcie a uloží ju do stavu

(štvorica 6(b)[i]). Z predchádzajúcej lemy vyplýva, že δ-funkcia je korektne simulovaná na aktívnom úseku, podľa uloženej informácie. Na ostatných (neaktívnych) úsekoch postupnosti úsekov je tiež korektne simulovaná δ-funkcia podľa uloženej informácie v stave, nakoľko ju tieto úseky nemenia (štvorica 6(c)[vii]). Po prečítaní všetkých úsekov postupnosti sa vrátime opäť do stavu $q_{(3)}$ (štvorica 6(c)[ix]).

Lema 2.1.5: δ-funkcia je korektne simulovaná na celej postupnosti úsekov, začínajúca aktívnym úsekom, ktorý obsahuje univerzálny stav. Taktiež je korektne simulovaná aktivácia niektorého z neaktívnych úsekov postupnosti úsekov.

Dôkaz: Vyplýva z konštrukcie a-prekladača a je veľmi podobný dôkazu z predcházajúcej lemy.

G-systém je v stave $q_{(3)}$, keď začne čítať postupnosť úsekov - najľavejší aktívny úsek, začínajúci symbolom $[\#]$. V tomto prechode si vyberie časť δ-funkcie a uloží ju do stavu (štvorica 6(b)[i]). Z lemy 2.1.3 vyplýva, že δ-funkcia je na aktívnom úseku korektne simulovaná podľa tejto uloženej informácie. Na ostatných (neaktívnych) úsekoch postupnosti úsekov je tiež korektne simulovaná δ-funkcia podľa uloženej informácie v stave, nakoľko ju tieto úseky nemenia (štvorica 6(d)[vii]). Na rozdiel od predchádzajúceho dôkazu, nemáme možnosť sa vrátiť do stavu $q_{(3)}$. Môžeme buď pokračovať prechodom neaktívnymi úsekmi alebo aktivovať nejaký úsek. Ak sa neaktivuje ani jeden úsek, dôjde k zaseknutiu (nemáme prechod zo stavu $q_{\forall}^{(q_{p_j},b_{s_j})\rightarrow(q_{r_j},b_{t_j},d_j)}$ do stavu $q_{(3)}$). Pri aktivácii zmeníme symbol aktuálne čítaného úseku z $\#$ na $[\#]$, vyberie sa nová časť δ-funkcie (štvorica 6(e)[i]), ktorá sa uloží do stavu namiesto starej. Taktiež do stavu uložíme informáciu o tom, že už prebehla jedna aktivácia (reprezentujeme pruhom nad stavom). Po aktivácii sa korektne simuluje δ-funkcia na novoaktivovanom úseku a zvyšné neaktívne úseky taktiež korektne simulujú δ-funkciu podľa nového predpisu (štvorice 6(e)[ii-xviii]). Pruhovaná verzia časti, ktorá realizuje δ-funkciu, nám nedovolí aktivovať ďalší úsek, umožňuje ale návrat do stavu $q_{(3)}$.

Lema 2.1.6: Uvažujme ľubovoľný úsek vetnej formy G-systému počas tretej hlavnej fázy. Nech je to $\triangle(B)u_i\Diamond$, kde $\triangle \in \{[\#],(\#)\}$. Pokiaľ nejde o počiatočnú konfiguráciu vieme na základe δ-funkcie skonštruovať a-prekladač, ktorý simuluje inverznú δ-funkciu a pomocou tohto a-prekladača pretransformovať úsek na podobu, ako vyzeral v predchádzajúcom kroku.

Dôkaz: Vyplýva z konštrukcie a-prekladača.

Všimnime si, že časť a-prekladača simulujúca δ-funkciu zobrazuje väčšinou iba jeden symbol na jeden symbol. Taktiež k δ-funkcii v upravenom formáte (pozri začiatok konštrukcie) je ľahko vytvorená inverzná δ-funkcia. Rozoberme si jednotlivé inverzné prípady.

- Najjednoduchší prípad je použitie inverznej δ-funkcie v nie krajových častiach úseku. Jednoducho simulujme inverznú δ-funkciu (štvorice 2(a) - 2(f)). Pre existenčný stav štvorice 3(a) - 3(i).
- Krajné prípady úseku sú ošetrené špeciálne, nakoľko v nich dochádza k skráteniu úseku (štvorice 2(g) - 2(m), ich pruhovaná kópia 5(j) - 5(p)).

- Pri univerzálnom stave je to o niečo zložitejšie. Zavedieme inverznú operáciu ku aktivácii - deaktiváciu (štvorica 5(a)). Deaktivácia núti vybrať deaktivovaný úsek inú časť inverznej δ-funkcie, ako je uložená v stave. Ak vykonanie tejto operácie vedie ku inej konfigurácii ako má úsek, ktorý uložil informáciu do stavu, určite niekedy dôjde k zaseknutiu (v ďalšom kroku je úsek už neaktívny a ak má inú konfiguráciu ako vedúci úsek, tak sa to časom prejaví).
- Akceptujúce úseky môžeme prejsť bez zmeny, avšak môžme sa aj nedeterministicky rozhodnúť, že aktivujeme akceptačný úsek.

Kvôli prehľadnosti zopakujeme definíciu G-systému, ktorý simuluje ATS. $G = (N, T, M, S)$, kde $N = \{S, (1), (2), (3), (A), (B), \#, [\#], (\#), [], \Diamond\} \cup \{q_i \mid q_i \in K\}$, $T = \Gamma \cup \{B\}$, $M = (K_M, N \cup T, N \cup T, H, q_0, F_M)$.

Skonštruujeme inverzný a-prekladač $\bar{M}$. $\bar{M} = (K_M, N \cup T, N \cup T, \bar{H}, q_0, F_M)$. Všimnime si, že sa od pôvodného a-prekladača líši len v množine H a v množine stavov. Definujeme inverzný smer pohybu hlavy na páske. Ak d_j je smer pohybu, tak $\bar{d_j}$ je inverzný smer pohybu. Predpis je intuitívny:

$\bar{d_j} =$

- 1 ak $d_j = -1$
- 0 ak $d_j = 0$
- -1 ak $d_j = 1$

$\bar{d_j}$ je inverzný pohyb ku d_j, a teda aj d_j je inverzný ku $\bar{d_j}$.

$H = \{$

1. $(q_0, (3), (3), q_{(3)})$
2. Časť simulujúca inverznú δ - funkciu (spoločná časť pre existenčný aj univerzálny stav)
   (a) $(q_{(3)}, [\#], [\#], q_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}) \mid 1 \leq j \leq g, \exists \delta(q_{r_j}, b_{t_j}) = \{(q_{p_j}, b_{s_j}, \bar{d_j})\}$
   (b) $(q_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}, b_i, b_i, q_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, d_j)}) \mid 1 \leq j \leq g, b_i \in T \cup \{(B)\}$
   (c) $(q_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}, b_i, q_{r_j}, q_{b_i}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}) \mid 1 \leq j \leq g, b_i \in T, b_{t_j} \notin \{\Diamond\}$
   (d) $(q_{b_i}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}, q_{p_j}, b_i, q_{d_j=-1}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, -1)}) \mid 1 \leq j \leq g, b_i \in T$
   (e) $(q_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 0)}, q_{p_j}, q_{r_j}, q_{d_j=0}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 0)}) \mid 1 \leq j \leq g$
   (f) $(q_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 1)}, q_{p_j}, b_{t_j}, q_{d_j=1}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 1)}) \mid 1 \leq j \leq g$
   (g) $(q_{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 1)}, (B), (B), q_{(B)}^{(q_{p_j}, b_{s_j}) \to (q_{r_j}, b_{t_j}, 1)}) \mid 1 \leq j \leq g$

(h) $(q_{(B)}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, q_{p_j}, q_{r_j}, (q_{[B]}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g$

(i) $(q_{[B]}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, B, \epsilon, q_{[\bar{B}]}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g$

(j) $(q_{[\bar{B}]}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, b_{s_j}, b_{t_j}, q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\exists}$

(k) $(q_{[\bar{B}]}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, b_{s_j}, b_{t_j}, q_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\forall}$

(l) $(q_{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}, b_{t_j}, \epsilon, q_{\Diamond}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}) \mid 1 \leq j \leq g$

(m) $(q_{\Diamond}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}, q_{p_j}, q_{r_j}, q_{d_j=-1}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}) \mid 1 \leq j \leq g$

3. Časť simulujúca inverznú δ - funkciu (časť špecifická pre existenčný stav)

(a) $(q_{d_j=-1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, b_{s_j}, b_{t_j}, q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\exists}$

(b) $(q_{d_j=0}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}, b_{s_j}, b_{t_j}, q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\exists}$

(c) $(q_{d_j=1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, b_{s_j}, q_{r_j}, q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\exists}$

(d) $(q_{d_j=-1}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}, \Diamond, \Diamond, q_{\exists}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\exists}$

(e) $(q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, b_i, b_i, q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g, b_i \in T$

(f) $(q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \Diamond, \Diamond, q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g$

(g) $(q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \#, \#, q_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g$

(h) $(q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, [], [], q_{[][]}) \mid 1 \leq j \leq g$

(i) $(q_{\exists}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \Diamond, \Diamond, q_{(3)}) \mid 1 \leq j \leq g$

4. Časť simulujúca inverznú δ - funkciu (časť špecifická pre univerzálny stav, pred deaktivovaním úseku)

(a) $(q_{d_j=-1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, b_{s_j}, b_{t_j}, q_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\forall}$

(b) $(q_{d_j=0}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}, b_{s_j}, b_{t_j}, q_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\forall}$

(c) $(q_{d_j=1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, b_{s_j}, q_{r_j}, q_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\forall}$

(d) $(q_{d_j=-1}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}, \Diamond, \Diamond, q_{\forall}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}) \mid 1 \leq j \leq g, q_{p_j} \in K_{\forall}$

(e) $(q_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, b_i, b_i, q_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g, b_i \in T$

(f) $(q_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \Diamond, \Diamond, q_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g$

(g) $(q_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \#, \#, q_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \leq j \leq g$

5. Časť simulujúca inverznú δ - funkciu (časť špecifická pre univerzálny stav, po deaktivovaní úseku)

   (a) $(q_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, [\#], \#, \bar{q}_{(q_{p_k},b_{s_k})\to(q_{r_k},b_{t_k},d_k)}) \mid 1 \le j \le g, 1 \le k \le g, j \neq k$

   (b) $(\bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, b_i, b_i, \bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \le j \le g, b_i \in T \cup \{(B)\}$

   (c) $(\bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, b_i, q_{r_j}, \bar{q}_{b_i}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \le j \le g, b_i \in T, b_{t_j} \notin \{\Diamond\}$

   (d) $(\bar{q}_{b_i}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, q_{p_j}, b_i, \bar{q}_{d_j=-1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \le j \le g, b_i \in T$

   (e) $(\bar{q}_{d_j=-1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}, b_{s_j}, b_{t_j}, \bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},-1)}) \mid 1 \le j \le g, q_{p_j} \in K_{\forall}$

   (f) $(\bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}, q_{p_j}, q_{r_j}, \bar{q}_{d_j=0}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}) \mid 1 \le j \le g$

   (g) $(\bar{q}_{d_j=0}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}, b_{s_j}, b_{t_j}, \bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},0)}) \mid 1 \le j \le g, q_{p_j} \in K_{\forall}$

   (h) $(\bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, q_{p_j}, b_{t_j}, \bar{q}_{d_j=1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \le j \le g$

   (i) $(\bar{q}_{d_j=1}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, b_{s_j}, q_{r_j}, \bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \le j \le g, q_{p_j} \in K_{\forall}$

   (j) $(\bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, (B), (B), \bar{q}_{(B)}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \le j \le g$

   (k) $(\bar{q}_{(B)}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, q_{p_j}, q_{r_j}, (\bar{q}_{[B]}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \le j \le g$

   (l) $(\bar{q}_{[B]}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, B, \epsilon, \bar{q}_{[\bar{B}]}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \le j \le g$

   (m) $(\bar{q}_{[\bar{B}]}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}, b_{s_j}, b_{t_j}, \bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},1)}) \mid 1 \le j \le g, q_{p_j} \in K_{\forall}$

   (n) $(\bar{q}_{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}, b_{t_j}, \epsilon, \bar{q}_{\Diamond}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}) \mid 1 \le j \le g$

   (o) $(\bar{q}_{\Diamond}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}, q_{p_j}, q_{r_j}, \bar{q}_{d_j=-1}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}) \mid 1 \le j \le g$

   (p) $(q_{d_j=-1}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}, \Diamond, \Diamond, \bar{q}_{\forall}^{(q_{p_j},b_{t_j})\to(q_{r_j},\Diamond,-1)}) \mid 1 \le j \le g, q_{p_j} \in K_{\forall}$

   (q) $(\bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, b_i, b_i, \bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \le j \le g, b_i \in T$

   (r) $(\bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \Diamond, \Diamond, \bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \le j \le g$

   (s) $(\bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \#, \#, \bar{q}_{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}) \mid 1 \le j \le g$

   (t) $(\bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, [], [], q_{[][]}) \mid 1 \le j \le g$

   (u) $(\bar{q}_{\forall}^{(q_{p_j},b_{s_j})\to(q_{r_j},b_{t_j},d_j)}, \Diamond, \Diamond, q_{(3)}) \mid 1 \le j \le g$

6. $(q_{[][]}, a_i, a_i, q_{[][]}) \mid a_i \in \Sigma$

7. Preskočenie akceptačných úsekov (kopírovacie cykly)

(a) $(q_{(3)}, (\#), (\#), q_a)$
(b) $(q_a, b_i, b_i, q_a) \mid b_i \in T$
(c) $(q_a, q_i, q_i, q_a) \mid q_i \in K$
(d) $(q_a, \Diamond, \Diamond, q_a)$
(e) $(q_a, (B), (B), q_a)$
(f) $(q_a, [], [], q_{[][]})$
(g) $(q_a, \Diamond, \Diamond, q_{(3)})$

8. Deakceptácia úseku (akceptačný úsek sa zmení na aktívny)

(a) $(q_{(3)}, (\#), [\#], q_{(\#)})$
(b) $(q_{(\#)}, b_i, b_i, q_{(\#)}) \mid b_i \in T$
(c) $(q_{(\#)}, (B), (B), q_{(\#)})$
(d) $(q_{(\#)}, q_u, q_u, q_{(a)}) \mid q_u \in F_M$
(e) $(q_{(a)}, b_i, b_i, q_{(a)}) \mid b_i \in T$
(f) $(q_{(a)}, \Diamond, \Diamond, q_{(3)})$

}

Vlastnosti inverzného a-prekladača:

- Pre naše účely nepotrebujeme púšťať inverzný a-prekladač na vetnú formu, korešpondujúcu s počiatočnou konfiguráciou. Preto tento prípad nebudeme ošetrovať. Nie je ale problém upraviť ATS do takého tvaru, aby sa nevrátil do počiatočného stavu. Po tejto úprave sa inverzný a-prekladač na vetnej forme, korešpondujúcej s počiatočnou konfiguráciou, zasekne.

- Ak sa v pôvodnej simulácii vetná forma predlžovala, tak v inverznej simulácii dochádza ku skracovaniu vetnej formy.

- Kvôli jednoznačnosti odvodenia požadujeme, aby G-systém vyberal vždy najkratšie odvodenie. Berieme vždy najkratšie odvodenie, teda neuvažujeme kroky, v ktorých sa neaplikuje žiadna δ-funkcia.

- V prípade, že existujú dve rôzne rovnako dlhé odvodenia tohto slova, môže nastať problém. Nazvime tieto odvodenia d_1 a d_2. $\bar{d_1}$ a $\bar{d_2}$ sú ich prislúchajúce inverzné odvodenia. Inverzný a-prekladač môže priradiť ku d_1 inverzné odvodenie $\bar{d_2}$ a naopak. Tento problém vyriešime tým, že jedno z týchto odvodení prehlásime za kratšie, a teda toho druhého sa zbavíme. Od začiatku porovnáme odvodenia. Keďže sú rôzne, musia sa v nejakom kroku líšiť. Uprednostíme to odvodenie, ktoré modifikuje vetnú formu zľava skôr ako to druhé odvodenie. V podstate je jedno, ktoré z odvodení uprednostíme, len to treba jednoznačne popísať. Týmto eliminujeme nejednoznačnosť odvodenia.

Pre potreby našich dôkazov nepotrebujeme takýto striktný spôsob riešenia. V dôkaze, kde používame toto tvrdenie potrebujeme ukázať, že k d_1 existuje inverzné odvodenie $\bar{d_1}$. Ak sme to neuhádli správne, hádali sme odvodenie $\bar{d_2}$; znamená to, že existuje odvodenie d_2. Teda ak existuje r rôznych odvodení, rovnakej dĺžky, jedného slova, tak nedeterministicky hádame pre d_1 jednu z r možností, pričom iba jedna je tá správna. Toto ale nič nemení na fakte, že inverzné odvodenie $\bar{d_1}$ existuje a vieme ho skonštruovať, čo postačuje pre potreby nášho dôkazu.

Pre lepšie porozumenie si ukážeme jednotlivé operácie, ktoré G-systém vykonáva. Zaujímavé časti vetnej formy sú orámované, kvôli lepšej viditeľnosti.

Štandarný prípad realizácie δ-funkcie

Uvažujme vetnú formu

$$(3)[\#](B)u_1 \boxed{\text{q a}}\, u_2\Diamond\ \#(B)u_1 \boxed{\text{q a}}\, u_2\Diamond\ \#(B)u_1 \boxed{\text{q a}}\, u_2\Diamond\ []w$$

Bez ujmy na všeobecnosti nech q je existenčný stav a u_1,$u_2 \in \Gamma^+$, teda nejde o kraj vetnej formy. Nech aktívny úsek háda časť δ-funkcie $\delta(q,a) = \{(p,b,1)\}$. Vetná forma v nasledujúcom kroku bude vyzerať takto:

$$(3)[\#](B)u_1 \boxed{\text{b p}}\, u_2\Diamond\ \#(B)u_1 \boxed{\text{b p}}\, u_2\Diamond\ \#(B)u_1 \boxed{\text{b p}}\, u_2\Diamond\ []w$$

Inverzná verzia štandardného prípadu je intuitívna. Uvažujme vetnú formu z ukážky štandardného prípadu.

$$(3)[\#](B)u_1 \boxed{\text{b p}}\, u_2\Diamond\ \#(B)u_1 \boxed{\text{b p}}\, u_2\Diamond\ \#(B)u_1 \boxed{\text{b p}}\, u_2\Diamond\ []w$$

Aktívny úsek uhádne časť δ-funkcie $\delta(q,a) = \{(p,b,1)\}$, ktorej inverznú podobu ($(p,b) \rightarrow (q,a,-1)$) uloží do stavu a odsimuluje ju. V ďalšom kroku bude vetná forma vyzerať takto:

$$(3)[\#](B)u_1 \boxed{\text{q a}}\, u_2\Diamond\ \#(B)u_1 \boxed{\text{q a}}\, u_2\Diamond\ \#(B)u_1 \boxed{\text{q a}}\, u_2\Diamond\ []w$$

Ostatné pohyby hlavy (-1 a 0) sú simulované obdobným spôsobom.

Aktivácia a deaktivácia

Najprv popíšeme aktiváciu úseku. Uvažujme vetnú formu: ($u_1, u_2 \in \Gamma^*$)

$$(3)[\#](B)u_1\ q_1\ a\ u_2\Diamond\ \#(B)u_1\ q_1\ a\ u_2\Diamond\ \boxed{\#(\text{B})}u_1\ q_1\ a\ u_2\Diamond\ []w$$

Bez ujmy na všeobecnosti nech G-systém uhádol, že aktivujeme tretí úsek. Prvý úsek si vybral časť δ-funkcie $\delta(q_1,a) = \{(q_2,b,0)\}$, druhý úsek si vybral $\delta(q_1,a) = \{(q_3,c,1)\}$. V ďalšom kroku bude vetná forma vyzrať takto:

$$(3)[\#](B)u_1\ q_2\ b\ u_2\Diamond\ \#(B)u_1\ q_2\ b\ u_2\Diamond\ \boxed{[\#](\text{B})}u_1\ c\ q_3\ u_2\Diamond\ []w$$

Opačná operácia k aktivácii je deaktivácia. Uvažujme vetnú formu z ukážky aktivácie.

$$(3)[\#](B)u_1\ q_2\ b\ u_2\Diamond\ \#(B)u_1\ q_2\ b\ u_2\Diamond\ \boxed{\text{[\#](B)}}u_1\ c\ q_3\ u_2\Diamond\ []w$$

Robíme inverzné odvodenie, teda hádame časť δ-funkcie, ktorú máme inverzne vykonať. Prvý úsek háda $\delta(q_1, a) = \{(q_2, b, 0)\}$. Túto informáciu v inverznej forme ($(q_2, b) \rightarrow (q_1, a, 0)$) uloží do stavu. Druhý úsek háda $\delta(q_1, a) = \{(q_3, c, 1)\}$, teda inverzná podoba bude $(q_3, c) \rightarrow (q_1, a, -1)$. Je nutné, aby obidva procesy uhádli rôznu časť δ-funkcie, inak dôjde k zaseknutiu. Druhý úsek sa deaktivuje, t.j. prepíše symbol z $[\#]$ na $\#$. Vetná forma bude v ďalšom kroku vyzerať takto:

$$(3)[\#](B)u_1\ q_1\ a\ u_2\Diamond\ \#(B)u_1\ q_1\ a\ u_2\Diamond\ \boxed{\text{\#(B)}}u_1\ q_1\ a\ u_2\Diamond\ []w$$

Krajné prípady vetnej formy

Krajné prípady treba ošetriť osobitne. Totiž úseky vetnej formy obsahujú konfiguráciu ATS obalenú dvomi špeciálnymi znakmi ((B) a $\Diamond$), ktoré reprezentujú nekonečne veľa $BLANK$-ov naľavo a napravo od konfigurácie ATS. Keď dôjde k tomu, že ATS ide prejsť hlavou na tieto BLANK-y, treba zväčšiť veľkosť konfigurácie v úseku. Ukážeme si oba prípady. Uvažujme vetnú formu ($u \in \Gamma^*$):

$$(3)[\#](B)\boxed{\text{q a}}\ u\Diamond\ \#(B)q\ a\ u\Diamond\ \#(B)q\ a\ u\Diamond\ []w$$

Nech aktívny úsek háda časť δ-funkcie $\delta(q, a) = \{(p, b, -1)\}$, teda chceme sa posunúť doľava. Tam je ale už koniec konfigurácie a špeciálny symbol (B). Urobíme to tak, že napravo od symbolu (B) vytvoríme symbol B, a tým zväčšíme veľkosť vetnej formy o jeden symbol. V nasledujúcom kroku bude vetná forma vyzerať takto:

$$(3)[\#](B)\boxed{\text{p B b}}\ u\Diamond\ \#(B)p\ B\ b\ u\Diamond\ \#(B)p\ B\ b\ u\Diamond\ []w$$

Teraz si ukážeme ako je ošetrený pravý kraj vetnej formy. Uvažujme vetnú formu ($u \in \Gamma^*$):

$$(3)[\#](B)u\ q\ \Diamond\ \#(B)u\ q\ \Diamond\ \#(B)u\ \boxed{\text{q}}\ \Diamond\ []w$$

Nech aktívny úsek háda časť δ-funkcie $\delta(q, \Diamond) = \{(p, b, 1)\}$. Na pravom kraji situáciu riešime odsunutím symbolu $\Diamond$ o jedno miesto doprava. V nasledujúcom kroku bude vetná forma vyzerať takto:

$$(3)[\#](B)u\ b\ p\ \Diamond\ \#(B)u\ b\ p\ \Diamond\ \#(B)u\ \boxed{\text{b p}}\ \Diamond\ []w$$

Teraz si ukážeme inverzné verzie krajných prípadov. Intuitívne, keď sme predlžovali úseky, tak teraz ich budeme skracovať. Najprv si vysvetlíme inverznú verziu ľavého prípadu. Uvažujme vetnú formu ($u \in \Gamma^*$):

$$(3)[\#](B)\boxed{\text{p B b}}\ u\Diamond\ \#(B)p\ B\ b\ u\Diamond\ \#(B)p\ B\ b\ u\Diamond\ []w$$

Treba si uvedomiť, že ak chceme skrátiť vetnú formu tak musí mať tento tvar, t.j. hlava musí čítať $BLANK$. Nech aktívny úsek háda časť δ-funkcie $\delta(q, a) = \{(p, b, -1)\}$. Do stavu uloží inverznú verziu ($(p, b) \rightarrow (q, a, 1)$). Skrátenie vetnej formy realizujeme vymazaním symbolu B. Vetná forma v nasledujúcom kroku bude vyzerať takto:

$$(3)[\#](B)\boxed{\text{q a}}\ u\Diamond\ \#(B)q\ a\ u\Diamond\ \#(B)q\ a\ u\Diamond\ []w$$

Inverzná verzia pravého kraju vetnej formy je obdobná. Namiesto $BLANK$-u, zmažeme symbol, ktorý bol vytvorený pri zväčšovaní vetnej formy. Uvažujme vetnú formu ($u \in \Gamma^*$):

$$(3)[\#](B)u\ b\ p\ \Diamond\ \#(B)u\ b\ p\ \Diamond\ \#(B)u\ \boxed{\text{b p}}\ \Diamond\ []w$$

Nech aktívny úsek háda časť δ-funkcie $\delta(q, \Diamond) = \{(p, b, 1)\}$, teda jej inverznú verziu uložíme do stavu ($(p, b) \rightarrow (q, \Diamond, -1)$). Vetná forma bude v nasledujúcom kroku vyzerať takto:

$$(3)[\#](B)u\ q\ \Diamond\ \#(B)u\ q\ \Diamond\ \#(B)u\ \boxed{\text{q}}\ \Diamond\ []w$$

Akceptácia a deakceptácia

V prípade, že sa aktívny úsek dostane do akceptačnej konfigurácie, má možnosť zmeniť sa z aktívneho úseku na akceptačný. Nech q je akceptačný stav a u_1,$u_2 \in \Gamma^*$. Uvažujme vetnú formu:

$$(3)\boxed{[\#]}(B)u_1\ q\ a\ u_2\Diamond\ [\#](B)v_1\ p\ v_2\Diamond\ \#(B)v_1\ p\ v_2\Diamond\ []w$$

Nech prvý aktívny úsek háda, že ide o akceptačnú konfiguráciu (ak uhádne zle, tak dôjde k zaseknutiu). Nasledujúca vetná forma bude vyzerať takto:

$$(3)\boxed{(\#)}(B)u_1\ q\ a\ u_2\Diamond\ [\#](B)\bar{v}_1\ \bar{p}\ \bar{v}_2\Diamond\ \#(B)\bar{v}_1\ \bar{p}\ \bar{v}_2\Diamond\ []w$$

Inverzná operácia ku akceptácii je deakceptácia. Uvažujme vetnú formu z ukážky aktivácie:

$$(3)\boxed{(\#)}(B)u_1\ q\ a\ u_2\Diamond\ [\#](B)\bar{v}_1\ \bar{p}\ \bar{v}_2\Diamond\ \#(B)\bar{v}_1\ \bar{p}\ \bar{v}_2\Diamond\ []w$$

Úsek má možnosť uhádnuť, že sa jedná o akceptačnú konfiguráciu. Toto skontroluje (ak zle hádal, dôjde k zaseknutiu) a aktivuje sa, t.j. prepíše symbol $(\#)$ na $[\#]$. Vetná forma bude v nasledujúcom kroku vyzerať takto:

$$(3)\boxed{[\#]}(B)u_1\ q\ a\ u_2\Diamond\ [\#](B)v_1\ p\ v_2\Diamond\ \#(B)v_1\ p\ v_2\Diamond\ []w$$

Môže sa zdať, že nastáva problém zlého poradia priebehu δ-funkcie; lebo hlava ATS najprv prečíta, potom zapíše a nakoniec sa pohne. Inverzná operácia sa intuitívne chápe v tomto poradí: najprv sa pohne, potom prečíta a nakoniec zapíše. G-systém však týmto problémom netrpí, nakoľko simuluje ATS tak, že realizuje tieto tri činnosti v jednom kroku, teda tento problém nenastáva.

Lema 2.1.7: Predpokladajme, že existuje výpočet ATS $(u_1, u_2, \ldots, u_n) \vdash_A^* (v_1, v_2, \ldots, v_m), m \geq n$. Nech $v \cong (v_1, v_2, \ldots, v_m)$, potom existuje u, také že platí $u \Rightarrow_G^* v$ a $u \cong (u_1, u_2, \ldots, u_n)$.

Dôkaz: Indukciou vzhľadom na dĺžku odvodenia.

$k = 0$. Báza indukcie triviálne platí, lebo u a v sú jedna a tá istá vetná forma.
$k = 1$. Existuje výpočet ATS $(w_1, w_2, \ldots, w_n) \vdash_A (v_1, v_2, \ldots, v_m), m \geq n$ a $v \cong (v_1, v_2, \ldots, v_m)$. Ukážeme, že existuje w také, že platí $w \Rightarrow_G v$ a $w \cong (w_1, w_2, \ldots, w_n)$. Existenciu slova w dokážeme jeho konštrukciou. Podľa lemy 2.1.6 vieme každý úsek vetnej formy v transformovať na podobu, akú mal v predchádzajúcom kroku odvodenia. Zvyšok vetnej formy sa nemení. Týmto spôsobom určite dostaneme slovo w, pre ktoré platí $w \Rightarrow_G v$. Vyplýva to z konštrukcie inverzného a-prekladača, z lemy 2.1.6. Podmienka $w \cong (w_1, w_2, \ldots, w_n)$ platí, nakoľko sme w vytvorili posladaním z úsekov w_i.
$k > 1$. Predpokladáme, že tvrdenie platí pre k krokov odvodenia. Ukážeme, že platí aj pre $k + 1$. Uvažujme výpočet ATS $(w_1, w_2, \ldots, w_n) \vdash_A^{k+1} (v_1, v_2, \ldots, v_m), m \geq n$. Je zrejmé, že existuje konfigurácia $(u_1, u_2, \ldots, u_l), m \geq l \geq n$, taká, že $(w_1, w_2, \ldots, w_n) \vdash_A (u_1, u_2, \ldots, u_l) \vdash_A^k (v_1, v_2, \ldots, v_m)$. Podľa indukčného predpokladu existuje slovo u, také, že platí $u \cong (u_1, u_2, \ldots, u_l)$ a $u \Rightarrow_G^k v$. Ukážeme, že existuje slovo w, také, že platí $w \cong (w_1, w_2, \ldots, w_n)$ a $w \Rightarrow_G u$. Postupovať budeme obdobne, ako v prípade $k = 1$.
Existenciu slova w dokážeme jeho konštrukciou. Podľa lemy 2.1.6 vieme každý úsek vetnej formy u transformovať na podobu, akú mal v predchádzajúcom kroku odvodenia. Zvyšok vetnej formy sa nemení. Týmto spôsobom určite dostaneme slovo w, pre ktoré platí $w \Rightarrow_G u$. Vyplýva to z konštrukcie inverzného a-prekladača z lemy 2.1.6. Podmienka $w \cong (w_1, w_2, \ldots, w_n)$ platí, nakoľko sme w vytvorili poskladaním z úsekov w_i.

Lema 2.1.8: Predpokladajme, že existuje odvodenie $w \Rightarrow_G^* v$ a $w \cong (w_1, w_2, \ldots, w_n)$. Potom existuje globálny stav ATS $(v_1, v_2, \ldots, v_m)$ taký, že existuje výpočet ATS $(w_1, w_2, \ldots, w_n) \vdash_A^* (v_1, v_2, \ldots, v_m)$, $m \geq n$ a platí $v \cong (v_1, v_2, \ldots, v_m)$.

Dôkaz: Indukciou vzhľadom na dĺžku odvodenia.

1. Báza indukcie triviálne platí, lebo w a v sú jedna a tá istá vetná forma.
2. Predpokladáme, že tvrdenie platí pre k krokov odvodenia. Ukážeme, že platí aj pre $k + 1$. Uvažujme odvodenie G-systému dĺžky $k + 1$, zrejme existuje vetná forma u taká, že platí $w \Rightarrow_G^k u$ a $u \Rightarrow_G v$. Po aplikovaní indukčného predpokladu dostaneme, že existuje globálny stav ATS $(u_1, u_2, \ldots, u_r)$, kde $n \leq r \leq m$, taký, že existuje výpočet ATS $(w_1, w_2, \ldots, w_n) \vdash_A^k (u_1, u_2, \ldots, u_r)$ a platí, že $u \cong (u_1, u_2, \ldots, u_r)$.

Platí, že $u \Rightarrow_G v$ a $u \cong (u_1, u_2, \ldots, u_r)$. Potrebujeme ukázať, že existuje výpočet ATS $(u_1, u_2, \ldots, u_r) \vdash_A (v_1, v_2, \ldots, v_m)$, kde $r \leq m$ a naviac platí, že $v \cong (v_1, v_2, \ldots, v_m)$.

Aplikovaním zobrazenia $Trans$ na vetnú formu u sa zbavíme nezaujímavých častí vetnej formy. Vetná forma $Trans(u)$ sa skladá z r úsekov. Toto vyplýva z predpokladu $u \cong (u_1, u_2, \ldots, u_r)$. Keďže $u \Rightarrow_G v$, existuje r výberov δ-funkcie, jeden na každom úseku vetnej formy $Trans(u)$. Keďže odvodenie G-systému sa presne riadi δ-funkciou ATS, musí existovať výpočet ATS $(u_1, u_2, \ldots, u_r) \vdash_A (v_1, v_2, \ldots, v_m)$. Skonštruujeme ho tak, že na každú konfiguráciu u_i aplikujeme δ-funkciu prislúchajúceho úseku. $r \leq m$ platí, lebo počet úsekov sa v simulačnej fáze nezmenšuje. $v \cong (v_1, v_2, \ldots, v_m)$ platí, lebo $(v_1, v_2, \ldots, v_m)$ sme vytvorili aplikovaním δ-funkcie vybratej príslušným (i-tym) úsekom na u_i.

Lema 2.1.9: Inicializácia, všetky medzifázy a finalizácia trvajú konštantný počet krokov.

Dôkaz: Vyplýva z konštrukcie a-prekladača.

Keďže tieto fázy menia kontrolnú informáciu na začiatku vetnej formy, štvorice 1(a), 3(a), 5(a), 7(a) a 8(a) sa použijú práve raz. Použitím ľubovoľnej z týchto štvoríc sa opakované použitie zablokuje. Použiť sa aspoň raz musia, inak sa nevygeneruje terminálne slovo (podľa lemy 2.1.1).

Lema 2.1.10: Všetky hlavné fázy trvajú $O(f(n))$ krokov.

Dôkaz:

- Prvá fáza generuje exponenciálnou rýchlosťou vetnú formu $(1)S^+$. Tento jazyk patrí medzi rýchlo generovateľné jazyky a je zrejmé, že čas generovania je dokonca $O(log(n))$.
- Druhá fáza generuje slovo, na ktorom bude prebiehať výpočet, t.j. slovo, ktoré bude na konci (možno) vygenerované. Toto slovo sa generuje symbol po symbole, t.j. pribudne jeden symbol v každom úseku, v každom prechode vetnou formou. Je zrejmé, že týmto spôsobom generovania, slovo dĺžky n potrebuje na vygenerovanie $O(n)$ krokov.
- Tretia fáza simuluje výpočet ATS. Jeden prechod touto fázou je ekvivalentný jednému kroku výpočtu ATS. Počet krokov výpočtu ATS je $O(f(n))$, teda aj počet krokov odvodenia, ktoré strávi G-systém v tretej fáze je $O(f(n))$.

Keďže $f(n) = \Omega(n)$, je už ľahko vidieť, že celkový počet krokov potrebný na všetky hlavné fázy je $O(log(n)) + O(n) + O(f(n)) = O(f(n))$.

Rozšírenie na simuláciu k-páskového ATS

Budeme mať k poschodovú vetnú formu. Počiatočná podoba úsekov bude vyzerať tak, že na najvyššom poschodí bude počiatočná konfigurácia ATS, na ostatných poschodiach budú $BLANK$-y. Namiesto hádania jednej časti δ-funkcie, hádame k častí δ-funkcie, pre každé poschodie jednu. Predlžovanie úseku nastáva vždy vtedy, keď aspoň jedna z pások potrebuje predĺžiť úsek. Na ostatné poschodia, ktoré nepotrebovali predĺžiť vetnú formu, pridáme na novovytvorené miesta $BLANK$-y. Konštrukcia sa mierne líši podľa štýlu akceptácie ATS. G-systém treba prispôsobiť tomu, či vyžadujeme akceptáciu iba jednej pásky, alebo všetkých pások.

Poznámka 2.1.1: Konštrukcia je nezávislá na tom, či model ATS, ktorý simuluje, môže, alebo nemôže zapisovať $BLANK$-y.

Detailne sme popísali konštrukciu simulácie a dokázali sme jej správnosť. Týmto je naše tvrdenie dokázané.

Veta 2.1.6: $ATIME(f(n)) \subseteq GSPACE(f(n))$, pre $f(n) = \Omega(n)$.

Dokázali sme, že $ATIME(f(n)) \subseteq GTIME(f(n))$, pre $f(n) = \Omega(n)$. Vieme, že platí $GSPACE(f(n)) = GTIME(f(n))$ pre $f(n) = \Omega(n)$ z [1]. Naše tvrdenie vyplýva priamo z týchto dvoch tvrdení.

Veta 2.1.7: $GTIME(f(n)) \subseteq ATIME(f^2(n))$, pre $f(n) = \Omega(n)$.

Dôkaz: Z [1] vieme, že $NSPACE(f(n)) \subseteq ATIME(f^2(n))$ pre $f(n) = \Omega(log(n))$. Ďalej vieme z [1], že $GTIME(f(n)) = NSPACE(f(n))$ pre $f(n) = \Omega(n)$. Dôsledkom týchto tvrdení je naše tvrdenie.

Ukázali sme, že existuje simulácia, ktorá ale nie je tesná. V ďalšom si prejdeme problémy, ktoré bránia konštrukcii tesnej simulácie.

Triviálne riešenie je simulovať 1-a prekladač na dvoch separátnych páskach. Problém je však v tom, že túto simuláciu by musel ATS zrealizovať v konštantnom čase. Naviac si treba uvedomiť, že výpočet 1-a prekladača môže mať až exponenciálne veľa krokov vzhľadom na dĺžku výstupného slova.

Lepšie riešenie je zrejme skontrolovať strom odvodenia G-systému a pokúsiť sa overiť, či sa vstupné slovo ním dá vygenerovať. Keď sa pozrieme na strom odvodenia, tak si môžeme všimnúť dva typy väzieb. Horizontálne - to sú susedné stavy 1-a prekladača, ktoré musia byť v pri sebe ležiacich štvoriciach zhodné. Vertikálne - to sú väzby následnosti generovania - časť vetnej formy, ktorú vygeneruje materská štvorica, musia brať postupne dcérske štvorice ako vstup. Myšlienka konštrukcie je spraviť paralelnú verziu simulácie NTS (prevzaté z [1]). Ku každej ceste od koreňa k listu v strome odvodenia a prislúchajúcej časti vygenerovaného slova priradíme jeden proces. Ten overí, či pre danú cestu je možné odvodiť danú časť výstupného slova.

Tu však narážame na problémy. Procesy medzi sebou nekomunikujú. Jediný spôsob, ako overiť následnosť stavov, je uhádnuť ich postupnosť ešte pred odvetvením. Táto postupnosť môže byť rovnako dlhá, ako je vysoký strom odvodenia G-systému. Takýto výpočet ATS by mohol mať až $O(f^2(n))$ krokov. Tento prístup veľmi efektívne overuje vertikálne väzby, ale má problém s horizontálnymi väzbami. Zdá sa, že G-systém má na rozdiel od ATS istú formu vnútornej komunikácie, ktorú ATS nemá.

Ďalší problém je samotné overenie časti vstupu. Proces musí uhádnuť správnu pozíciu na vstupnej páske, kde má začať porovnávať časť vstupu s výstupom, ktorý vypočítal na základe k nemu priradenej cesty v strome odvodenia.

Problémy by zrejme vyriešilo použitie iného modelu ATS, tzv. synchronizovaného, ktorý má istú formu vnútornej komunikácie. V našom tvrdení ale takýto predpoklad nemáme, a preto by sme museli synchronizovaný ATS simulovať. Dá sa ľahko nahliadnuť, že táto simulácia je reálna v kvadratickom čase. Teda čas, ktorý ušetríme použitím synchronizovaného ATS, stratíme pri jeho simulácii.

Kontrapríkladový jazyk, ktorý hľadáme, t.j. jazyk, ktorý by vylúčil možnosť existencie tesnej simulácie, patrí do $GTIME(f(n))$ a nepatrí do $ATIME(f(n))$, pre $f(n) = \Omega(n)$.

Veta 2.1.8: $GSPACE(f(n)) \subseteq ATIME(f^2(n))$, pre $f(n) = \Omega(n)$.

Dôkaz: Z [1] vieme, že $GSPACE(f(n)) = GTIME(f(n))$, pre $f(n) = \Omega(n)$. Z predchádzajúceho tvrdenia vieme, že platí $GTIME(f(n)) \subseteq ATIME(f^2(n))$, pre $f(n) = \Omega(n)$. Naše tvrdenie vyplýva z týchto dvoch tvrdení.

Podobne ako v predchádzajúcom tvrdení sme ukázali, že existuje simulácia, ktorá ale nie je tesná. V ďalšom si prejdeme problémy, ktoré bránia konštrukcii tesnej simulácie.

Triviálne riešenie je podobné ako v predchádzajúcom tvrdení, t.j. simulovať 1-a prekladač na dvoch separátnych páskach. Problém je však v tom, že počet krokov G-systému môže byť až exponenciálne veľa. Preto nech vieme akokoľvek rýchlo simulovať 1-a prekladač, simulácia môže stále trvať až exponenciálne veľa krokov.

Lepšie riešenie je zrejme skontrolovať strom odvodenia G-systému a pokúsiť sa overiť či sa vstupné slovo ním dá vygenerovať. Kotrola by však prebiehala odlišne ako v úvahách o predchádzajúcom tvrdení. Rozdiel je v tom, že počet štvoríc na jednej úrovni je vždy najviac $O(f(n))$, zatiaľ čo hĺbka stromu môže byť až exponenciálna. ATS by mohol simulovať G-systém tak, že by každý proces separátne overil jednu úroveň stromu odvodenia G-systému. Takto veľmi efektívne overíme horizontálne väzby, problém je však s vertikálnymi väzbami. Procesy medzi sebou nekomunikujú, a preto by sme museli vertikálne väzby pred odvetvením hádať, čo by spomalilo simuláciu. Opäť sa ukazuje, že G-systém má na rozdiel od ATS istú formu vnútornej komunikácie, ktorú ATS nemá.

Kontrapríkladový jazyk, ktorý hľadáme, t.j. jazyk, ktorý by vylúčil možnosť existencie tesnej simulácie, patrí do $GSPACE(f(n))$ a nepatrí do $ATIME(f(n))$, pre $f(n) = \Omega(n)$.

V ďalšom popíšeme jazyk, ktorý je vhodný kandidát na kontrapríkladový kazyk pre naše tvrdenie.

Jazyk $L = \{w_1\#w_2\#\ldots\#w_k \mid w_i \neq w_j$, pre $i \neq j, w_i \in \{0,1\}^k\}$ sa dá vygenerovať G-systémom v lineárnom priestore, t.j. $L \in GSPACE(n)$. Popíšeme konštrukciu G-systému pre jazyk L.

G-systém vygeneruje k rovnakých podslov oddelených mrežou. Podslová budú tvaru 0^k. Podslová reprezentujú binárne počítadlá a všetky začínajú s hodnotou 0. Jedným prechodom zvýšime hodnotu každého počítadla o jeden. Nedeterministicky sa môžeme rozhodnúť najviac jedno počítadlo počas jedného prechodu zastaviť. Zvyšujeme počítadlá, až kým sa všetky nezastavia. Ak dosiahneme podslovo 1^k a po tomto prechode existuje ešte aspoň jedno nezastavené počítadlo, dôjde k zaseknutiu. Dĺžka vetnej formy sa počas zvyšovania počítadiel nemení, a teda priestor potrebný na odvodenie je lineárny. Počet krokov odvodenia môže byť až exponenciálny.

Zvýraznime si problém akceptácie jazyka L pre ATS, tým, že vyriešime pár drobných zádrheľov, ktoré sa môžu na prvý pohľad zdať problematické. Prvá vec, čo musí ATS overiť, je správny tvar slova. Vstupné slovo sa musí skladať z k podslov a každé podslovo musí mať dĺžku práve k. Rozdelíme tento problém na dva podproblémy. Prvý podproblém je overiť, či je počet podslov rovný dĺžke nejakého podslova. Druhý podproblém je overenie, či všetky podslová majú rovnakú dĺžku.

Za predpokladu, že ATS vie simulovať synchronizovaný ATS v tom istom čase, vyriešime prvý podproblém synchronizovaným ATS a tento predpoklad dokážeme neskôr. Popíšeme synchronizovaný ATS, riešiaci náš problém. Odvetvíme dva procesy, jeden bude čítať prvé podslovo a druhý bude čítať znaky $\#$. Keďže znakov $\#$ je o jeden menej ako dĺžka podslova, rátame do počtu aj prvý $BLANK$, ktorý je napravo od konfigurácie. Ak je dĺžka prvého podslova rovná počtu znakov $\#$ ($+$ jeden $BLANK$), tak procesy synchronizujú a dôjde k akceptácii. Pokiaľ bol predpoklad simulácie synchronizovaného ATS nesprávny, tak potom vieme tento problém triviálne riešiť bez alternovania v čase $O(k^3)$, teda $O(n^{\frac{3}{2}})$.

Zdá sa, že ide o zle paralelizovateľný problém, nakoľko sa asi nedá využiť alternovanie.

Problém je v tom, že informácia, ktorú potrebujeme overiť, je rovnomerne rozdrobená po celom vstupnom slove. Keďže procesy nekomunikujú, rozdelenie vstupu medzi procesy je problematické.

Druhá časť problému je overiť, či sú všetky podslová rovnakej dĺžky. Na riešenie tohto problému bude stačiť ATS. Odvetvíme $k-1$ procesov, pričom každý proces bude mať na starosti skontrolovanie dvoch po sebe idúcich podslov. Keďže dĺžka podslova spĺňa podmienku tranzitivity (ak $i-1$-vé podslovo je rovnako dlhé ako i-te a i-te je rovnako dlhé ako $i+1$-vé, tak $i-1$-vé je rovnako dlhé ako $i+1$-vé), komunikácia medzi procesmi nie je nutná. Na takéto overenie treba $O(n)$ krokov (odvetvenie $O(n)$ a overenie $O(k)$).

Riešiť problém jazyka L nie je možné cez logický komplement jazyka

$$L_k = \{w_1\#w_2\#\ldots\#w_k \mid \exists w_i = w_j \text{ pre } i \neq j, w_i \in \{0,1\}^k\}$$

lebo ide o nedeterministický model. Nemôžeme pustiť ATS pre jazyk L_k a potom invertovať akceptáciu.

Preskúmajme ako simulovať synchronizovaný ATS pomocou obyčajného ATS. Triviálne riešenie je pred odvetvením procesov uhádnuť celú synchronizačnú postupnosť a zapísať ju na pásku. Potom odštartovať ATS, ktorý simulujeme, a pokiaľ sa nejaký proces dostane do synchronizačného stavu, tak overíme správnosť jeho výpočtu tým, že prerušíme výpočet, skontrolujeme stav v synchronizačnej postupnosti, vrátime sa na miesto výpočtu a pokračujeme ďalej vo výpočte. Miesto prerušenia výpočtu označíme, aby sme vedeli, kam sa vrátiť. Stavy, ktoré sme v synchronizačnej postupnosti už overili, tiež označíme, aby sme vedeli, ktorý z nich je aktuálne na rade.

Zhrňme časové nároky vo všeobecnosti. Simulujeme synchronizovaný ATS, pracujúci v čase $O(f(n))$. Dĺžka synchronizačnej postupnosti môže byť až $O(f(n))$. Overenie jediného synchronizačného stavu a návrat na miesto prerušenia môže trvať až $O(f(n))$ krokov. Týchto overení je treba až toľko, aká je dlhá synchronizačná postupnosť. Teda celkovo je to $O(f^2(n))$ krokov. Vo všeobecnosti je simulácia príliš pomalá.

Zhrňme časové nároky pre konkrétny synchronizovaný ATS, ktorý používame v našej konštrukcii. Čas výpočtu synchronizovaného ATS je $O(n)$. Dĺžka synchronizačnej postupnosti je $O(k)$ a počet prerušení je tiež iba $O(k)$. Teda celkové časové nároky sú $O(k^2)$. Vyjadrené vzhľadom na dĺžku vstupného slova je to $O(n)$. Takáto simulácia postačuje pre naše účely.

Jediný problém, ktorý ostáva preskúmať, je, či dokáže ATS v čase $O(n)$ overiť, či sú všetky podslová rôzne. Synchronizácia zrejme nepomôže, lebo nejde o kontrolu dĺžky, prípadne počtu symbolov, ale o kontrolu samotných symbolov. Triviálne riešenie problému je odvetviť proces pre každé podslovo, teda k procesov. Každý proces potom porovná svoje podslovo s každým iným podslovom. Toto trvá $O(k^4)$, teda $O(n^2)$ krokov.

Skúsime túto konštrukciu vylepšiť tým, že zapojíme viacej procesov, a tým zrýchlime celý výpočet. Odvetvíme najprv k procesov pre každé podslovo jeden. Potom sa každý z týchto procesov rozvetví na ďalších $k-1$ procesov. Teda máme $O(k^2)$ procesov, pre každú dvojicu podslov jeden. Teraz každý proces overí, či sú dve podslová rôzne. Toto overenie má $O(k.n)$ krokov (k je počet overovacích krokov a n je maximálna možná vzdialenosť týchto podslov). Celkovo je to $O(n^{\frac{3}{2}})$ krokov, čo je lepšie ako predchádzajúci výsledok.

Výsledok predchádzajúceho výpočtu stálne nie je dostatočne uspokojivý, a preto skúsime zapojiť ešte viac procesov. Problém veľkého počtu krokov potrebného na akceptáciu spočíva v neefektívnej metóde porovnania dvoch podslov. Zopakujeme konštrukciu z predchádzajúceho riešenia, ale vylepšíme porovnanie dvoch podslov. Toto porovnanie bolo realizované jediným procesom. Teraz odvetvíme proces pre každý symbol prvého podslova, a tieto kontrolné procesy budú mať za úlohu overiť, či symbol na pozícii, kde majú hlavu v prvom podslove, je rôzny od symbolu, ktorý je na korešpondujúcej pozícii v druhom podslove. Ak aspoň jeden z týchto kontrolných procesov nájde rôzne symboly, akceptuje. Samotný proces zodpovedný za porovnanie dvoch podslov akceptuje, ak akceptuje aspoň jeden z kontrolných procesov, ide teda o existenčné vetvenie. Toto vyplýva z faktu, že dve slová su určite rôzne, ak sa nezhodujú v aspoň jednom znaku. Týmto sme zrýchlili porovnanie dvoch podslov z $O(k.n)$ na $O(n)$. Teda celkový čas výpočtu je $O(n)$. Je tu ale jeden problém, ktorý spomaľuje výpočet. Keďže nemáme pásku s rýchlym prístupom na vstup, kontrolné procesy si nemajú kde pamätať, ako sa dostať na pozíciu v druhom podslove, pričom táto informácia má dĺžku $O(k)$. V ďalšom budeme túto informáciu nazývať offset. Jediná možnosť je postupovať po jednom symbole a vracať sa do prvého podslova po offset postupne. Toto však zaberie $O(k.n)$ krokov, a tým spomalí celý výpočet na $O(n^{\frac{3}{2}})$ krokov. Tento problém sa zrejme nedá odstrániť, lebo nemáme miesto, kam by sme offset uložili. Taktiež nemôžeme využiť nedeterminizmus, lebo overenie takéhoto hádania je rovnako náročné ako predchádzajúce riešenie (ukladanie offsetu).

Ďalšie zrýchlenie by sa dalo dosiahnuť skomprimovaním offsetu (dĺžky $O(k)$) na dĺžku $O(log(k))$. Vieme, že existuje synchronizovaný ATS, ktorý dokáže nájsť stred slova. Tento ATS vo všeobecnosti používa, na naše potreby, priveľa synchronizačných stavov, takže na vstupe dĺžky k ho vieme simulovať v čase $O(k^2)$, čo ale nevadí, lebo to je ešte stále iba $O(n)$. Idea je zmeniť unárnu podobu offsetu na binárnu, t.j. informáciu podobnú adrese políčka pásky s rýchlym prístupom na vstup. Najprv musíme vygenerovať túto adresu. To robíme tak, že používame ATS, ktorý hľadá stred slova. Nedeterministicky hádame, do ktorej polovice slova sa máme ďalej vnoriť. Informáciu, ktorú polovicu vyberieme ukladáme na pásku. Takto sa niekedy dopracujeme až k políčku, ktoré má proces skontrolovať. Týmto ukončíme komprimáciu offsetu a začneme hľadať cieľové políčko, ktoré treba skontrolovať, podľa adresy, ktorú sme práve vytvorili. Znova používame ATS, ktorý hľadá stred slova a berieme tú polovicu, ktorú nám predpisuje adresa. Takto nájdeme cieľové políčko a skontrolujeme ho.

Prejdeme časové nároky. Skomprimovať offset trvá $O(log(k).n)$ krokov. Nájsť cieľové políčko podľa adresy trvá tiež $O(log(k).n)$ krokov. Vieme, že $O(k^2) = O(n)$, a teda celkové časové nároky sú $O(n.log(n))$. Je možné, že v čase $O(n)$ nie je možné tento jazyk akceptovať jednopáskovým ATS.

Hypotéza: $L \notin ATIME(n)$, kde $L = \{w_1\#w_2\#\ldots\#w_k \mid w_i \neq w_j$, pre $i \neq j, w_i \in \{0,1\}^k\}$.

Poznámka 2.1.2: Je ľahko vidieť, že ATS akceptuje tento jazyk v čase $O(n)$, ak má aspoň dve pásky.

2.2 Porovnanie tried zložitosti pre $f(n) = o(n)$

Predtým, než začneme porovnávať triedy zložitosti, stručne prejdeme triviálnymi prípadmi. $GSPACE(f(n)) = \emptyset$ pre $f(n) = o(n)$. Preto inklúzie

$$GSPACE(f(n)) \subseteq ASPACE(f(n))$$

$$GSPACE(f(n)) \subseteq ATIME(f(n))$$

triviálne platia, nakoľko prázdna množina je podmnožinou každej množiny. Ďalšie dve tvrdenia

$$ASPACE(f(n)) \subseteq GSPACE(f(n))$$

$$ATIME(f(n)) \subseteq GSPACE(f(n))$$

sú nepravdivé, pretože žiadna neprázdna množina nie je podmnožinou prázdnej množiny.

Veta 2.2.1: $GTIME(f(n)) \subseteq ASPACE(f(n))$, pre $f(n) = o(n)$.

Dôkaz: G-systém pracujúci v čase $O(f(n))$ vieme simulovať jednosmerným NTS v priestore $O(f(n))$, teda triviálne ho vieme simulovať aj obyčajným NTS v priestore $O(f(n))$. ATS vie triviálne simulovať NTS, vieme teda simulovať G-systém pomocou ATS v priestore $O(f(n))$.

Definícia 2.2.1 (potomok štvorice): Štvorica (q_s, b, v, q_t) je potomkom štvorice (q_p, a, u, q_r) práve vtedy, keď $b \in u$ a v strome odvodenia sa štvorica (q_s, b, v, q_t) vyskytuje o úroveň nižšie ako štvorica (q_p, a, u, q_r). Intuitívne pod potomkom štvorice rozumieme štvoricu, ktorá v ďalšom kroku spracováva časť výstupu svojho predka. Predok je inverzná relácia ku potomkovi.

Lema 2.2.1 (o bariére): Pri odvodení slova z jazyka $L = \{w\#w \mid w \in \Sigma^+\}$ v čase $O(f(n))$ G-systémom, dôjde k preneseniu, z pravej časti vetnej formy do ľavej (rozdelené symbolom $\#$), najviac $O(f(n))$ symbolov.

Uvažujme generovanie slova z jazyka $L = \{w\#w \mid w \in \Sigma^+\}$ G-systémom v čase $O(f(n))$. Skonštruujeme stredovú postupnosť štvoríc. Po vygenerovaní je slovo tvaru $w\#w$. Existuje štvorica, ktorá vygenerovala symbol $\#$. Pridáme túto štvoricu do postupnosti. Ďalej pridáme predka práve pridanej štvorice. Potom predka predka tejto štvorice, a takto pridávame štvorice až raz pridáme štvoricu, ktorá je koreňom stromu odvodenia. Teraz transformujeme postupnosť štvoríc na postupnosť stavov zobrazením pr_4 (prevzaté z [1]), teda postupnosť štvoríc zmeníme

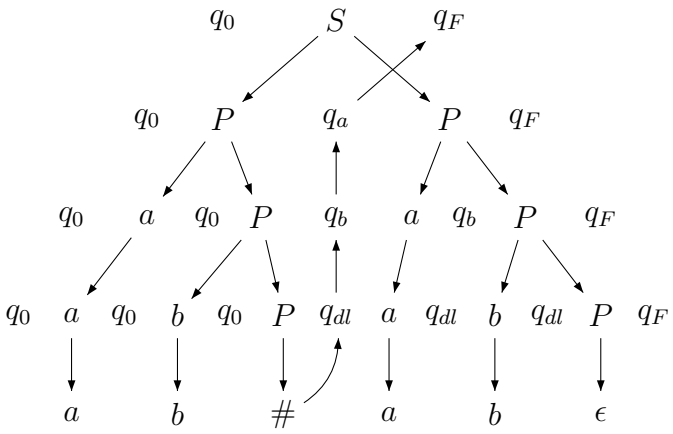


Fig. 2.8: Konštrukcia stredovej postupnosti stavov pri odvodení slova $abab$

na postupnosť stavov v ich štvrtej komponente. Dostali sme stredovú postupnosť stavov dĺžky $f(n)$. Keďže postupnosť je stredová, všetky prenesené symboly, smerujúce z prvého do druhého podslova, museli byť prenesené v stavoch tejto postupnosti. Nakoľko 1-a prekladač prepisuje vetnú formu zľava doprava, nemá zmysel merať prenesené symboly smerujúce z druhého podslova do prvého. V každom stave môžeme uložiť konštantné množstvo symbolov. Indukciou sa ľahko nahliadne, že pre stredovú postupnosť stavov dĺžky $f(n)$ sa počas odvodenia môže preniesť najviac $O(f(n))$ symbolov z prvého podslova do druhého.

Pre lepšie pochopenie ilustrujeme konštrukciu stredovej postupnosti na konkrétnom slove. Uvažujme slovo $abab$. Na obrázku 2.8 je zobrazené jeho odvodenie G-systémom. Jeden riadok reprezentuje jeden krok výpočtu G-systému. Stromová štruktúra reprezentuje prepisovanie terminálov a neterminálov v jednotlivých krokoch (šípky od hora dole). Symboly q reprezentujú stavy, v ktorých bol 1-a prekladač počas transformácie vetnej formy. Stredovú postupnosť stavov začneme konštruovať od symbolu $\#$, nakoľko rozdeľuje jednotlivé podslová. Konštrukcia pokračuje pridávaním stavov do postupnosti (šípky od dola nahor).

Na obrázku 2.8 sa stredová posupnosť drží prevažne v strede stromu odvodenia. Toto však závisí od konkrétneho stromu odvodenia. V prípade, že by sa v podstrome, ktorý odvodí ľavé podslovo, vyrábala dlhá časť vetnej formy, ktorá by sa neskôr zmazala, bola by stredová postupnosť výrazne vychýlená do pravej časti stromu odvodenia. Rovnako by to fungovalo aj pre pravé podslovo. Toto vychýlenie neovplyvňuje vlastnosť stredovej postupnosti - stále všetky prenesené symboly z ľavej časti vetnej formy do pravej, musia byť prenášané v stavoch tejto postupnosti. Je to z dôvodu, že dočasná časť vetnej formy, ktorá sa neskôr zmaže, nemá vplyv na počet prenesených symbolov medzi ľavou a pravou časťou vetnej formy.

Konštrukcia sa dá ľahko upraviť pre jazyk $L = \{ww \mid w \in \Sigma^+\}$. Stačí namiesto symbolu $\#$ vybrať posledný symbol prvého podslova w.

Veta 2.2.2: $ASPACE(f(n)) \subseteq \bigcup_c GTIME(c^{f(n)})$, pre $f(n) = o(n)$.

Dôkaz: Triviálny horný odhad. Spravíme dolný odhad.

Prvý nápad je využiť fakt (prevzaté z [5]), že pre jazyk

$$L = \{ww \mid w = 001^{\phi(k)}0 \ldots 01^{\phi(3k-1)}01^{2^{(3k-1)+2}-1},\ k \geq 2,\ \phi \text{ je permutácia } \}$$

platí $L \in GTIME(\Omega(log(n^{\frac{1}{3}})loglog(n^{\frac{1}{3}})))$. Ak by sa podarilo ukázať, že $L \in ASPACE(log(n))$, tak by sme tento jazyk mohli použiť ako základ pre dolný odhad. Ukážeme, že na akceptáciu tohto jazyka dokonca nepotrebujeme ani alternovanie, t.j. $L \in NSPACE(log(n))$. Z toho potom bude triviálne vyplývať, že $L \in ASPACE(log(n))$.

Pre zjednodušenie konštrukcie rozdelíme overenie vstupného slova na dve hlavné úlohy. Prvá je overiť, či vstupné slovo je tvaru ww a druhá je overiť, či prvé slovo je tvaru

$$w = 001^{\phi(k)}0 \ldots 01^{\phi(3k-1)}01^{2^{(3k-1)+2}-1}$$

Ak sú tieto dve vlastnosti overené, nie je treba nič viac overovať. Kvôli jednoduchosti si na začiatku odvetvíme dva procesy. Jeden overí zhodnosť oboch podslov ww. Druhý proces overí či je prvé (ľavé) podslovo tvaru $w = 001^{\phi(k)}0 \ldots 01^{\phi(3k-1)}01^{2^{(3k-1)+2}-1}$. Alternovanie v tomto prípade, nie je esenciálne, je použité len kvôli zjednodušeniu. Môžeme sa ho zbaviť aj tak, že to pustíme sekvenčne, teda najprv overíme prvú vlastnosť a potom druhú. Jediné, čo treba urobiť medzitým, je vyčistiť pracovnú pásku, ak máme dovolené zapisovať $BLANK$-y, ak nie, tak ju prepísať falošnými $BLANK$-ami a triviálne upraviť nasledovnú konštrukciu na prácu s falošnými $BLANK$-ami. Najprv popíšeme konštrukciu NTS akceptujúceho jazyk $L = \{ww \mid w \in \Sigma^+\}$.

ATS akceptujúci $L = \{ww \mid w \in \Sigma_L^+\}$ *v priestore* $O(log(n))$

Zostrojíme pre jazyk $L = \{ww \mid w \in \Sigma_L^+\}$ NTS so vstupnou páskou a jednou pracovnou páskou, ktorý akceptuje L v priestore $O(log(n))$. Myšlienka konštrukcie využíva fakt, že naraz si nepotrebujeme pamätať veľa informácie. Budeme postupne kontrolovať dvojice políčok, ktoré musia byť rovnaké. Využijeme rovnakú vzdialenosť týchto políčok. Túto vzdialenosť na začiatku vypočítame a v binárnej podobe uložíme na pracovnú pásku. Na pracovej páske budeme mať pracovnú adresu (vľavo od oddeľovača) a adresu stredového políčka (napravo od oddeľovača). Adresa stredového políčka je vzdialenosť medzi jednotlivými dvojicami políčok, ktoré kontrolujeme. Nasleduje formálna konštrukcia. Za formálnou konštrukciou nasleduje podrobný komentár významu každého riadku δ - funkcie.

NTS $A = (K, \Sigma, \Gamma, \delta, q_0, F)$, kde $\Sigma = \Sigma_L$, $\Gamma = \Sigma_L \cup \{\bar{0}, \bar{1}, \#_1\}$, $K = \{q_0, q_{addr}^{mid}, q_{inc}^{input}, q_{inc}^{mid}, q_{\rightarrow}^{mid}, q_{\leftarrow}^{store}, q_{read}^{store}, q_{inc}^{unmark}, q_{inc}^{move}, q_{dec}^{input}, q_{dec}^{mid}, q_{test}^{mid}, q_{\Rightarrow}^{mid}, q_{input}^{reset}, q_{store}^{input}, q_{\rightarrow}, q_{\leftarrow}, q_{unmark}^{\rightarrow}, q_{unmark}^{\leftarrow}, q_{\Rightarrow}, q_{dec}, q_{dec}^{\rightarrow}, q_{seek}^{final}, q_{dec}^{final}, q_{rewind}^{final}, q_F\ \} \cup \{q_{\rightarrow}^{a}, q_{\leftarrow}^{a}, q_{\Leftarrow}^{a|b}, q_{unmark}^{a|\rightarrow}, \bar{q}_{\Leftarrow}^{a|b}, q_{unmark}^{a|\leftarrow}, q_{\Rightarrow}^{a}, q_{dec}^{a}, q_{dec}^{a|\rightarrow}, q_{read}^{a}, q_{\Leftarrow}^{b}, \bar{q}_{\Leftarrow}^{b}, q_{b}^{store} \mid a, b \in \Sigma\}$, $F = \{q_F\}$ a δ je v tvare

δ(state,input,work) = {(state,work,input dir,work dir)}.

1. Vytvorenie adresy prvého políčka druhého podslova
   (a) $\delta(q_0, a, B) = \{(q_{addr}^{mid}, 0, 0, 1)\} \mid a \in \Sigma$
   (b) $\delta(q_{addr}^{mid}, a, B) = \{(q_{addr}^{mid}, 0, 0, 1), (q_{inc}^{input}, \#_1, 0, -1)\} \mid a \in \Sigma$
   (c) $\delta(q_{inc}^{input}, a, b) = \{(q_{inc}^{mid}, b, 1, 0)\} \mid a, b \in \Sigma$

(d) $\delta(q_{inc}^{mid}, a, 1) = \{(q_{inc}^{mid}, 0, 0, -1)\} \mid a \in \Sigma$

(e) $\delta(q_{inc}^{mid}, a, 0) = \{(q_{\rightarrow}^{mid}, 1, 0, 0)\} \mid a \in \Sigma$

(f) $\delta(q_{\rightarrow}^{mid}, a, b) = \{(q_{\rightarrow}^{mid}, b, 0, -1)\} \mid a \in \Sigma$

(g) $\delta(q_{\rightarrow}^{mid}, a, \#_1) = \{(q_{inc}^{input}, \#_1, 0, -1), (q_{\leftarrow}^{store}, \#_1, 0, -1)\}$

2. Skopírovanie adresy na trvalejšie miesto na páske

(a) $\delta(q_{\leftarrow}^{store}, a, b) = \{(q_{\leftarrow}^{store}, b, 0, -1)\} \mid a \in \Sigma, b \in \Sigma \cup \{\#_1\}$

(b) $\delta(q_{\leftarrow}^{store}, a, \bar{b}) = \{(q_{read}^{store}, \bar{b}, 0, 1)\} \mid a, b \in \Sigma$

(c) $\delta(q_{\leftarrow}^{store}, a, B) = \{(q_{read}^{store}, B, 0, 1)\} \mid a \in \Sigma$

(d) $\delta(q_{read}^{store}, a, b) = \{(q_{b}^{store}, \bar{b}, 0, 1)\} \mid a, b \in \Sigma$

(e) $\delta(q_{read}^{store}, a, \#_1) = \{(q_{inc}^{unmark}, \#_1, 0, -1)\} \mid a \in \Sigma$

(f) $\delta(q_{b}^{store}, a, c) = \{(q_{b}^{store}, c, 0, 1)\} \mid a, b \in \Sigma, c \in \Sigma \cup \{\#_1\}$

(g) $\delta(q_{b}^{store}, a, B) = \{(q_{\leftarrow}^{store}, b, 0, -1)\} \mid a, b \in \Sigma$

(h) $\delta(q_{inc}^{unmark}, a, \bar{b}) = \{(q_{inc}^{unmark}, b, 0, -1)\} \mid a, b \in \Sigma$

(i) $\delta(q_{inc}^{unmark}, a, B) = \{(q_{inc}^{move}, B, 0, 1)\} \mid a \in \Sigma$

(j) $\delta(q_{inc}^{move}, a, b) = \{(q_{inc}^{move}, b, 0, 1)\} \mid a, b \in \Sigma$

(k) $\delta(q_{inc}^{move}, a, \#_1) = \{(q_{dec}^{input}, \#_1, 0, -1)\} \mid a \in \Sigma$

3. Overenie správnosti adresy

(a) $\delta(q_{dec}^{input}, a, b) = \{(q_{dec}^{mid}, b, 1, 0)\} \mid a, b \in \Sigma$

(b) $\delta(q_{dec}^{input}, B, 0) = \{(q_{test}^{mid}, 0, 0, 0)\}$

(c) $\delta(q_{dec}^{mid}, a, 0) = \{(q_{dec}^{mid}, 1, 0, -1)\} \mid a \in \Sigma \cup \{B\}$

(d) $\delta(q_{dec}^{mid}, a, 1) = \{(q_{\Rightarrow}^{mid}, 0, 0, 0)\} \mid a \in \Sigma \cup \{B\}$

(e) $\delta(q_{\Rightarrow}^{mid}, a, b) = \{(q_{\Rightarrow}^{mid}, b, 0, -1)\} \mid a \in \Sigma \cup \{B\}$

(f) $\delta(q_{\Rightarrow}^{mid}, a, \#_1) = \{(q_{dec}^{input}, \#_1, 0, -1)\} \mid a \in \Sigma \cup \{B\}$

(g) $\delta(q_{test}^{mid}, B, 0) = \{(q_{test}^{mid}, 0, 0, -1)\}$

(h) $\delta(q_{test}^{mid}, B, B) = \{(q_{input}^{reset}, B, -1, 1)\}$

(i) $\delta(q_{input}^{reset}, a, b) = \{(q_{input}^{reset}, b, -1, 0)\} \mid a, b \in \Sigma$

(j) $\delta(q_{input}^{reset}, B, b) = \{(q_{store}^{input}, b, 1, 0)\} \mid b \in \Sigma$

4. Uloženie obsahu skúmaného políčka do stavu a prekopírovanie stredovej adresy do pracovnej adresy

(a) $\delta(q_{store}^{input}, a, b) = \{(q_{\rightarrow}^{a}, b, 0, 1)\} \mid a, b \in \Sigma$

(b) $\delta(q_{\rightarrow}^{a}, c, b) = \{(q_{\rightarrow}^{a}, b, 0, 1)\} \mid a, c \in \Sigma, b \in \Sigma \cup \{\#_1, \bar{0}, \bar{1}\}$

(c) $\delta(q^a_{\rightarrow}, c, B) = \{(q^a_{\leftarrow}, B, 0, -1)\} \mid a, c \in \Sigma$

(d) $\delta(q^a_{\leftarrow}, c, \bar{b}) = \{(q^a_{\leftarrow}, \bar{b}, 0, -1)\} \mid a, b, c \in \Sigma$

(e) $\delta(q^a_{\leftarrow}, c, b) = \{(q^{a|b}_{\Leftarrow}, \bar{b}, 0, -1)\} \mid a, b, c \in \Sigma$

(f) $\delta(q^a_{\leftarrow}, c, \#_1) = \{(q^{a|\rightarrow}_{unmark}, \#_1, 0, 1)\} \mid a, b, c \in \Sigma$

(g) $\delta(q^{a|b}_{\Leftarrow}, c, d) = \{(q^{a|b}_{\Leftarrow}, d, 0, -1)\} \mid a, b, c, d \in \Sigma$

(h) $\delta(q^{a|b}_{\Leftarrow}, c, \#_1) = \{(\bar{q}^{a|b}_{\Leftarrow}, \#_1, 0, -1)\} \mid a, b, c \in \Sigma$

(i) $\delta(\bar{q}^{a|b}_{\Leftarrow}, c, \bar{d}) = \{(\bar{q}^{a|b}_{\Leftarrow}, \bar{d}, 0, -1)\} \mid a, b, c, d \in \Sigma$

(j) $\delta(\bar{q}^{a|b}_{\Leftarrow}, c, d) = \{(q^a_{\rightarrow}, \bar{b}, 0, 1)\} \mid a, b, c, d \in \Sigma$

(k) $\delta(q^{a|\rightarrow}_{unmark}, c, \bar{b}) = \{(q^{a|\rightarrow}_{unmark}, \bar{b}, 0, 1)\} \mid a, b, c \in \Sigma$

(l) $\delta(q^{a|\rightarrow}_{unmark}, c, B) = \{(q^{a|\leftarrow}_{unmark}, B, 0, -1)\} \mid a, c \in \Sigma$

(m) $\delta(q^{a|\leftarrow}_{unmark}, c, \bar{b}) = \{(q^{a|\leftarrow}_{unmark}, b, 0, -1)\} \mid a, b, c \in \Sigma$

(n) $\delta(q^{a|\leftarrow}_{unmark}, c, \#_1) = \{(q^{a|\leftarrow}_{unmark}, \#_1, 0, -1)\} \mid a, c \in \Sigma$

(o) $\delta(q^{a|\leftarrow}_{unmark}, c, B) = \{(q^a_{\Rightarrow}, B, 0, 1)\} \mid a, c \in \Sigma$

(p) $\delta(q^a_{\Rightarrow}, c, b) = \{(q^a_{\Rightarrow}, b, 0, 1)\} \mid a, b, c \in \Sigma$

(q) $\delta(q^a_{\Rightarrow}, c, \#_1) = \{(q^a_{dec}, \#_1, 0, -1)\} \mid a, c \in \Sigma$

5. Presun hlavy na cieľové políčko a kontrola obsahu skúmaných políčok

(a) $\delta(q^a_{dec}, c, 0) = \{(q^a_{dec}, 1, 0, -1)\} \mid a, c \in \Sigma$

(b) $\delta(q^a_{dec}, c, 1) = \{(q^{a|\rightarrow}_{dec}, 0, 0, 0)\} \mid a, c \in \Sigma$

(c) $\delta(q^a_{dec}, c, B) = \{(q^a_{read}, B, 0, 1)\} \mid a, c \in \Sigma$

(d) $\delta(q^{a|\rightarrow}_{dec}, c, b) = \{(q^{a|\rightarrow}_{dec}, b, 0, -1)\} \mid a, b, c \in \Sigma$

(e) $\delta(q^{a|\rightarrow}_{dec}, c, \#_1) = \{(q^a_{dec}, \#_1, 1, -1)\} \mid a, c \in \Sigma$

(f) $\delta(q^a_{read}, a, 0) = \{(q_{\rightarrow}, 0, 1, 0)\} \mid a \in \Sigma$

6. Skopírovanie stredovej adresy do pracovnej adresy

(a) $\delta(q_{\rightarrow}, a, b) = \{(q_{\rightarrow}, b, 0, 1)\} \mid a \in \Sigma, b \in \Sigma \cup \{\#_1, \bar{0}, \bar{1}\}$

(b) $\delta(q_{\rightarrow}, a, B) = \{(q_{\leftarrow}, B, 0, -1)\} \mid a \in \Sigma$

(c) $\delta(q_{\leftarrow}, a, \bar{b}) = \{(q_{\leftarrow}, \bar{b}, 0, -1)\} \mid a, b \in \Sigma$

(d) $\delta(q_{\leftarrow}, a, b) = \{(q^b_{\Leftarrow}, \bar{b}, 0, -1)\} \mid a, b \in \Sigma$

(e) $\delta(q_{\leftarrow}, a, \#_1) = \{(q^{\rightarrow}_{unmark}, \#_1, 0, 1)\} \mid a, b \in \Sigma$

(f) $\delta(q^b_{\Leftarrow}, a, d) = \{(q^b_{\Leftarrow}, d, 0, -1)\} \mid a, b, d \in \Sigma$

(g) $\delta(q^b_{\Leftarrow}, a, \#_1) = \{(\bar{q}^b_{\Leftarrow}, \#_1, 0, -1)\} \mid a, b \in \Sigma$

(h) $\delta(\bar{q}^{b}_{\Leftarrow}, a, \bar{d}) = \{(\bar{q}^{b}_{\Leftarrow}, \bar{d}, 0, -1)\} \mid a, b, d \in \Sigma$

(i) $\delta(\bar{q}^{b}_{\Leftarrow}, a, d) = \{(q_{\rightarrow}, \bar{b}, 0, 1)\} \mid a, b, d \in \Sigma$

(j) $\delta(q^{\rightarrow}_{unmark}, a, \bar{b}) = \{(q^{\rightarrow}_{unmark}, \bar{b}, 0, 1)\} \mid a, b \in \Sigma$

(k) $\delta(q^{\rightarrow}_{unmark}, a, B) = \{(q^{\leftarrow}_{unmark}, B, 0, -1)\} \mid a \in \Sigma$

(l) $\delta(q^{\leftarrow}_{unmark}, a, \bar{b}) = \{(q^{\leftarrow}_{unmark}, b, 0, -1)\} \mid a, b \in \Sigma$

(m) $\delta(q^{\leftarrow}_{unmark}, a, \#_1) = \{(q^{\leftarrow}_{unmark}, \#_1, 0, -1)\} \mid a \in \Sigma$

(n) $\delta(q^{\leftarrow}_{unmark}, a, B) = \{(q_{\Rightarrow}, B, 0, 1)\} \mid a \in \Sigma$

(o) $\delta(q_{\Rightarrow}, a, b) = \{(q_{\Rightarrow}, b, 0, 1)\} \mid a, b \in \Sigma$

(p) $\delta(q_{\Rightarrow}, a, \#_1) = \{(q_{dec}, \#_1, 0, -1)\} \mid a \in \Sigma$

7. Presun hlavy na políčko pásky, ktoré bude skontrolované v nasledujúcom prechode

(a) $\delta(q_{dec}, a, 0) = \{(q_{dec}, 1, 0, -1)\} \mid a \in \Sigma$

(b) $\delta(q_{dec}, a, 1) = \{(q^{\rightarrow}_{dec}, 0, 0, 0)\} \mid a \in \Sigma$

(c) $\delta(q_{dec}, a, B) = \{(q^{input}_{store}, B, 0, 1)\} \mid a \in \Sigma$

(d) $\delta(q^{\rightarrow}_{dec}, a, b) = \{(q^{\rightarrow}_{dec}, b, 0, -1)\} \mid a, b \in \Sigma$

(e) $\delta(q^{\rightarrow}_{dec}, a, \#_1) = \{(q_{dec}, \#_1, -1, -1)\} \mid a \in \Sigma$

8. Ukončenie kontrolného cyklu

(a) $\delta(q^{input}_{store}, a, b) = \{(q^{final}_{seek}, b, 0, 1)\} \mid a, b \in \Sigma$

(b) $\delta(q^{final}_{seek}, a, b) = \{(q^{final}_{seek}, b, 0, 1)\} \mid a, b \in \Sigma$

(c) $\delta(q^{final}_{seek}, a, B) = \{(q^{final}_{dec}, B, 0, -1)\} \mid a \in \Sigma$

(d) $\delta(q^{final}_{dec}, a, 0) = \{(q^{final}_{dec}, 1, 0, -1)\} \mid a \in \Sigma$

(e) $\delta(q^{final}_{dec}, a, 1) = \{(q^{final}_{rewind}, 0, 0, 1)\} \mid a \in \Sigma$

(f) $\delta(q^{final}_{rewind}, a, b) = \{(q^{final}_{rewind}, b, 0, 1)\} \mid a, b \in \Sigma$

(g) $\delta(q^{final}_{rewind}, a, B) = \{(q^{final}_{dec}, B, 1, -1)\} \mid a \in \Sigma$

(h) $\delta(q^{final}_{dec}, B, 0) = \{(q^{final}_{dec}, 0, 0, -1)\}$

(i) $\delta(q^{final}_{dec}, B, \#_1) = \{(q_F, \#_1, 0, 0)\}$

Na pracovnej páske vytvoríme adresu (1(a), 1(b)). Pozostáva z $log(n)$ núl a je ukončená symbolom $\#_1$. Náš cieľ je zistiť adresu prvého políčka druhého podslova. Na začiatku je adresa nulová a čítacia hlava je na začiatku vstupného slova. Spustíme cyklus, v ktorého každom prechode posunieme hlavu na vstupe (1(c)), zvýšime adresu o jedna (1(d) a 1(e)) a presunieme hlavu pracovnej pásky na pravý kraj adresy (1(f)). Uhádneme, kedy sme na správnom políčku (overíme to neskôr) a ukončíme cyklus (1(g)).

Potrebujeme získanú adresu uložiť na trvalejšie miesto. Celú adresu skopírujeme za symbol $\#_1$. Kopírovanie pracuje v jednoduchom cykle. Presunieme hlavu pracovnej pásky na prvý nespracovaný symbol (2(a) - 2(c)), uložíme nespracovaný symbol do stavu a označíme ho (značíme pruhom) ako spracovaný (2(d)). Presunieme sa za symbol $\#_1$ (2(f)) a na príslušné miesto uložený symbol napíšeme (2(g)). Kopírovací cyklus skončíme, ak už nie je čo kopírovať (2(e)). Odznačíme pruhované symboly z prvej adresy (naľavo od $\#_1$) (2(h) a 2(i)). Presunieme sa na koniec prvej adresy (2(j) a 2(k)).

Skontrolujeme, či sme hádali stredové políčko správne. V cykle budeme čítať vstup (3(a)) a zároveň znižovať adresu (3(c) a 3(d)). Po znížení adresy presúvame hlavu pracovnej pásky na pravý koniec adresy (3(e) a 3(f)), kvôli ďalšiemu prechodu cyklu. Ak na vstupe prečítame celé slovo (3(b)), t.j. čítame $BLANK$, musí byť nutne adresa nulová (3(g) a 3(h)). V opačnom prípade sme hádali nesprávne a dôjde k zaseknutiu. Presunieme hlavu na vstupnej páske na začiatok (3(i) a 3(j)).

Uložíme obsah políčka zo vstupu do stavu (4(a)). Prekopírujeme adresu stredového políčka do pracovnej adresy. Nasleduje kopírovací cyklus. Presunieme hlavu pracovnej pásky na pravý koniec pracovnej pásky (4(b) a 4(c)). Nájdeme prvý nespracovaný symbol a uložíme ho do stavu (4(d) a 4(e)). Presunieme sa na najpravejší nespracovaný symbol pracovnej adresy (4(g) - 4(i)) a zapíšeme uložený symbol, ktorý práve kopírujeme (4(j)). Cyklus skončí, keď sa minú nespracované symboly (4(f)). Prejdeme celú pracovnú pásku a odznačíme všetky symboly (4(k) - 4(n)). Nastavíme hlavu pracovnej pásky na pravý koniec pracovnej adresy (4(o) - (q)).

Presunieme sa na cieľové políčko. Posúvací cyklus v jednom prechode zníži hodnotu adresy o jeden (5(a) a 5(b)), presunie hlavu pracovnej pásky na koniec pracovnej adresy (5(d)) a posunie hlavu na vstupe o jedno políčko doprava (5(e)). Cyklus skončí, keď je adresa nulová (5(c)), t.j. sme na cieľovom políčku. Skontrolujeme, či sa obsah skúmaných políčok zhoduje (5(f)) a posunieme sa na vstupe o jedno políčko doprava (to využijeme neskôr).

Opäť skopírujeme adresu stredového políčka do pracovnej adresy. Skupina 6 je takmer úplne identická so skupinou 4, jediný rozdiel je, že nemáme v stave uložený obsah políčka. Po skončení kopírovania nastavíme hlavu pracovnej pásky na pravý koniec pracovnej adresy.

Budeme opäť znižovať pracovnú adresu a posúvať hlavu na vstupe, tentokrát ale doľava. 7(a) a 7(b) zníži adresu, 7(d) presunie hlavu pracovnej pásky na pravý koniec pracovnej adresy a 7(e) posunie hlavu na vstupe. Cyklus skončí, ak je adresa nulová. Po skončení cyklu máme hlavu na vstupe presne tam, kde sme chceli: na políčku, ktoré je hneď napravo od práve skontrolovaného políčka. 7(c) uzatvára vonkajší cyklus.

Uhádneme, kedy ukončíme vonkajší cyklus. Bude to vtedy, ak hlava na vstupe číta prvý symbol druhého podslova. Naše hádanie overíme. Presunieme hlavu pracovnej pásky na koniec adresy stredového políčka (8(a) - 8(c)). Spustíme cyklus, v ktorého jednom prechode znížime adresu o jedna (8(d) a 8(e)), presunieme hlavu pracovnej pásky na pravý koniec adresy (8(f)) a posunieme hlavu na vstupe o jedna doprava (8(g)). Cyklus ukončíme v prípade, že sme už dočítali vstup (čítame $BLANK$) a zároveň adresa je nulová (8(h)). V tomto prípade akceptujeme (8(i)).

Veľkosť oboch adries na pracovnej páske je $O(log(n))$. Je zrejmé, že NTS používa $O(log(n))$ priestoru.

NTS akceptujúci jazyk $L = \{001^{\phi(k)}0 \ldots 01^{\phi(3k-1)}01^{2^{(3k-1)+2}-1} \mid k \geq 2\}$ *v priestore* $O(log(n))$

Bez ujmy na všeobecnosti budeme predpokladať, že na vstupe máme iba slovo w. Tento predpoklad nám veľmi uľahčí konštrukciu a nijako nemení charakter problému. Vďaka oddeľovaču 00, ktorý je na začiatku druhého (pravého) podslova, vieme zabezpečiť, aby sme neopustili prvé podslovo. Skrátenie vstupu na polovicu tiež nemá žiadny dopad, lebo asymptoticky je dĺžka rovnaká.

Uvažujme vstupné slovo w. Musíme overiť, či je slovo tvaru $w = uv$, kde

$$u = 001^{\phi(k)}0 \ldots 01^{\phi(3k-1)}0$$

a $v = 1^{2^{(3k-1)+2}-1}$. Podslovo u budeme nazývať permutačnou časťou slova w a v budeme nazývať vypchávkou.

Najprv overíme u - permutačnú časť slova w. Treba si všimnúť, že permutačná časť slova sa skladá z úsekov, ktoré obsahujú len jednotky a sú oddelené nulami. Uhádneme a napíšeme na pracovnú pásku k symbolov $\bar{1}$. Ak sme k zle uhádli, neskôr dôjde k zaseknutiu. Teraz odštartujeme kontrolný cyklus. Počet symbolov na pracovnej páske bude slúžiť ako počítadlo pre aktuálne skúmaný úsek. Na začiatku počítadlo obsahuje k symbolov, teda kontrolujeme úsek, ktorý obsahuje k jednotiek. Nedeterministicky uhádneme políčko vstupnej pásky, na ktorom začína ten správny úsek. To je ten úsek, ktorý obsahuje toľko jednotiek, koľko máme aktuálne v počítadle. Overíme toto hádanie, tým, že prečítame celý úsek a zároveň čítame aj počítadlo na pracovnej páske. Ak sa počet symbolov nezhoduje, tak dôjde k zaseknutiu, inak pokračujeme pridaním ďalšieho symbolu do počítadla a opakovaním kontrolného cyklu. Prvých k symbolov sú $\bar{1}$, tie ostatné, ktoré pridávame neskôr, sú 1. Potrebujeme odlíšiť prvých k symbolov, lebo si potrebujeme pamätať k.

Nedeterministicky uhádneme, kedy skončiť kontrolný cyklus. Po skončení overíme viacero vlastností. Najprv skontrolujeme, či je v počítadle presne $3k - 1$ symbolov. Toto vieme spraviť, lebo v počítadle máme odlíšených prvých k symbolov. Ak sa NTS nezasekol, znamená to, že pre každý počet jednotiek od k až po $3k - 1$, existuje nejaký úsek, ktorý má zhodný počet jednotiek. Ešte treba overiť, či počet úsekov je presne $2k$, t.j. že neexistujú dva zhodné úseky. Toto tiež overíme, lebo v počítadle máme uložené k. Týmto sme overili permutačnú časť slova. Môže sa zdať, že sme neošetrili možnosť, keby čítacia hlava zablúdila do vypchávky. V tomto prípade, však dôjde k zaseknutiu, lebo vypchávka nemá ako posledný symbol 0.

Nastavíme hlavu na vstupe na prvý symbol vypchávky, t.j. hneď napravo od poslednej nuly. Teraz naštartujeme kontrolný cyklus, ktorý overí, či je počet jednotiek správny. Na pracovnej páske máme počítadlo, ktoré obsahuje $3k - 1$ jednotiek. Na pracovnú pásku napíšeme dve adresy, ktoré budeme potrebovať. Adresa bude reprezentovať binárne zakódovanú vzdialenosť od začiatku vypchávky. Na začiatku tieto adresy obsahujú samé nuly a ich dĺžku uhádneme. Ich dĺžka je $O(log(n))$. Kontrolný cyklus bude vyzerať takto: za každú jednotku v počítadle urobíme nasledujúcu procedúru. Posunieme hlavu na vstupe toľkokrát doprava, koľko krokov trvá zníženie prvej adresy z aktuálnej hodnoty na nulu. Keďže v prvej adrese bola aktuálna vzdialenosť hlavy od začiatku vypchávky, týmto sme hlavu posunuli tak, že jej vzdialenosť od začiatku je teraz dvojnásobná. Teraz musíme aktualizovať obidve adresy. Druhá adresa si zachovala starú hodnotu adresy pozície hlavy na vstupe. Keďže vzdialenosť je teraz dvojnásobná,

stačí urobiť SHIFT LEFT (teda vynásobiť binárnu adresu dvomi). Túto novú adresu skopírujeme aj do prvej adresy.

Tento cyklus opakujeme kým sa neminú jednotky v počítadle. Posledný prechod cyklom treba separátne ošetriť, lebo počet jednotiek nemá byť $2^{(3k-1)}$, ale $2^{(3k-1)+2}-1$. Tieto konštanty ale nemenia podstatu kontrolného cyklu. Nasleduje formálna konštrukcia zjednodušenej verzie. Za konštrukciou je podrobný komentár ku každému riadku δ-funkcie.

Zostrojíme NTS A, ktorý akceptuje L v priestore $O(log(n))$. NTS má jednu vstupnú pásku z ktorej môže len čítať a jednu pracovnú pásku. $A = (K, \Sigma, \Gamma, \delta, q_0, F)$, kde $\Sigma = \{0,1\}$, $\Gamma = \{0, 1, \bar{1}, \tilde{1}, [1], (1), \bar{0}, \#_1, \#_2\}$,
$K = \{q_0, q_{init}, q_{find}, q_{found}, q^{\leftarrow}_{found}, q_{compare}, q_{\leftarrow}, q^{\bar{1}}_{select}, q^{\rightarrow}_{select}, q^{[1]}_{select}, q_{check}, q^{reset}_{input}, q^{1}_{prep}, q^{2}_{prep}, q^{count}_{zero_1}, q^{count}_{zero_2}, q_{\rightarrow}, q^{make}_{addr1}, q^{make}_{addr}, q^{make}_{addr2}, q_{rewind}, q^{comp}_{read}, q^{comp}_{\rightarrow}, q^{dec}, q^{dec}_{return}, q^{shift}_{move}, q^{shift}, q^{copy}_{\rightarrow}, q^{copy}_{read}, q^{unmark}_{\leftarrow}, q^{copy}_{\Rightarrow}, q^{unmark}_{\leftarrow}, q^{unmark}_{\rightarrow}, \bar{q}^{comp}_{\rightarrow}, \bar{q}^{dec}, \bar{q}^{dec}_{return}, q^{last}_{one}, q_F\} \cup \{q^{shift}_b, q^{\leftarrow}_b, q^{find}_b \mid b \in \{0,1\}\}$,
$F = \{q_F\}$ a δ je v tvare

δ(state,input,work) = {(state,work,input dir,work dir)}.

1. Inicializácia a porovnanie úseku zo vstupu s počítadlom na pracovnej páske
   (a) $\delta(q_0, 0, B) = \{(q_{init}, \bar{1}, 1, 1)\}$
   (b) $\delta(q_{init}, 0, B) = \{(q_{init}, \bar{1}, 0, 1), (q_{find}, \bar{1}, 0, 1)\}$
   (c) $\delta(q_{find}, a, B) = \{(q_{find}, B, b, 0), (q_{found}, B, 0, 0)\} \mid a \in \Sigma, b \in \{-1, 0, 1\}$
   (d) $\delta(q_{found}, a, B) = \{(q^{\leftarrow}_{found}, B, 0, -1)\} \mid a \in \Sigma$
   (e) $\delta(q^{\leftarrow}_{found}, a, c) = \{(q^{\leftarrow}_{found}, c, 0, -1)\} \mid a \in \Sigma, c \in \{1, \bar{1}\}$
   (f) $\delta(q^{\leftarrow}_{found}, a, B) = \{(q_{compare}, B, 0, 1)\} \mid a \in \Sigma$
   (g) $\delta(q_{compare}, 1, c) = \{(q_{compare}, c, 1, 1)\} \mid c \in \{1, \bar{1}\}$
   (h) $\delta(q_{compare}, 0, B) = \{(q_{find}, 1, 0, 1), (q_{\leftarrow}, B, 0, -1)\}$
2. Overenie správneho počtu symbolov na pracovnej páske po skončení porovnávacieho cyklu
   (a) $\delta(q_{\leftarrow}, 0, c) = \{(q_{\leftarrow}, c, 0, -1)\} \mid c \in \{1, \bar{1}, [1]\}$
   (b) $\delta(q_{\leftarrow}, 0, B) = \{(q^{\bar{1}}_{select}, B, 0, 1)\}$
   (c) $\delta(q_{\leftarrow}, 0, \tilde{1}) = \{(q^{\bar{1}}_{select}, \tilde{1}, 0, 1)\}$
   (d) $\delta(q^{\bar{1}}_{select}, 0, \bar{1}) = \{(q^{\rightarrow}_{select}, \tilde{1}, 0, 1)\}$
   (e) $\delta(q^{\rightarrow}_{select}, 0, \bar{1}) = \{(q^{\rightarrow}_{select}, \bar{1}, 0, 1)\}$
   (f) $\delta(q^{\rightarrow}_{select}, 0, [1]) = \{(q^{\rightarrow}_{select}, [1], 0, 1)\}$
   (g) $\delta(q^{\rightarrow}_{select}, 0, 1) = \{(q^{[1]}_{select}, [1], 0, 1)\}$
   (h) $\delta(q^{[1]}_{select}, 0, 1) = \{(q_{\leftarrow}, [1], 0, 0)\}$
   (i) $\delta(q^{[1]}_{select}, 0, B) = \{(q_{check}, [1], 0, -1)\}$

(j) $\delta(q_{check}, 0, d) = \{(q_{check}, d, 0, -1)\} \mid d \in \{\tilde{1}, [1]\}$

(k) $\delta(q_{check}, 0, B) = \{(q^{reset}_{input}, B, 0, 1)\}$

3. Overenie správneho počtu núl v permutačnej časti slova

(a) $\delta(q^{reset}_{input}, a, \tilde{1}) = \{(q^{reset}_{input}, \tilde{1}, -1, 0)\} \mid a \in \Sigma$

(b) $\delta(q^{reset}_{input}, B, \tilde{1}) = \{(q^{1}_{prep}, \tilde{1}, 1, 0)\}$

(c) $\delta(q^{1}_{prep}, 0, \tilde{1}) = \{(q^{2}_{prep}, \tilde{1}, 1, 0)\}$

(d) $\delta(q^{2}_{prep}, 0, \tilde{1}) = \{(q^{count}_{zero_1}, \tilde{1}, 1, 0)\}$

(e) $\delta(q^{count}_{zero_1}, 1, \tilde{1}) = \{(q^{count}_{zero_1}, \tilde{1}, 1, 0)\}$

(f) $\delta(q^{count}_{zero_1}, 0, \tilde{1}) = \{(q^{count}_{zero_2}, \tilde{1}, 1, 0)\}$

(g) $\delta(q^{count}_{zero_1}, 1, [1]) = \{(q_{\rightarrow}, [1], 0, -1)\}$

(h) $\delta(q^{count}_{zero_2}, 1, \tilde{1}) = \{(q^{count}_{zero_2}, \tilde{1}, 1, 0)\}$

(i) $\delta(q^{count}_{zero_2}, 0, \tilde{1}) = \{(q^{count}_{zero_1}, \tilde{1}, 1, 1)\}$

4. Inicializácia adries na pracovnej páske, potrebných na kontrolu vypchávky

(a) $\delta(q_{\rightarrow}, 1, c) = \{(q_{\rightarrow}, c, 0, 1)\} \mid c \in \{1, \bar{1}, [1]\}$

(b) $\delta(q_{\rightarrow}, 1, B) = \{(q^{make}_{addr1}, \#_1, 0, 0)\}$

(c) $\delta(q^{make}_{addr1}, 1, B) = \{(q^{make}_{addr1}, 0, 0, 1), (q^{make}_{addr}, 1, 0, 1)\}$

(d) $\delta(q^{make}_{addr}, 1, B) = \{(q^{make}_{addr2}, \#_2, 0, 1)\}$

(e) $\delta(q^{make}_{addr2}, 1, B) = \{(q^{make}_{addr2}, 0, 0, 1), (q_{rewind}, 1, 0, -1)\}$

(f) $\delta(q_{rewind}, 1, b) = \{(q_{rewind}, b, 0, -1)\} \mid b \in \Gamma - \{(1)\}$

(g) $\delta(q_{rewind}, 1, (1)) = \{(q^{comp}_{read}, (1), 0, 1)\}$

(h) $\delta(q_{rewind}, 1, B) = \{(q^{comp}_{read}, B, 0, 1)\}$

(i) $\delta(q^{comp}_{read}, 1, c) = \{(q^{comp}_{\rightarrow}, (1), 0, 1)\} \mid c \in \{\tilde{1}, [1]\}$

(j) $\delta(q^{comp}_{\rightarrow}, 1, c) = \{(q^{comp}_{\rightarrow}, c, 0, 1)\} \mid c \in \{\tilde{1}, [1], \#_1, 0, 1\}$

(k) $\delta(q^{comp}_{\rightarrow}, 1, \#_2) = \{(q^{dec}, \#_2, 0, -1)\}$

5. Vnútorný cyklus - posúvanie hlavy na vstupe a príslušná zmena adries

(a) $\delta(q^{dec}, 1, 0) = \{(q^{dec}, 1, 0, -1)\}$

(b) $\delta(q^{dec}, 1, 1) = \{(q^{dec}_{return}, 0, 0, -1)\}$

(c) $\delta(q^{dec}, 1, \#_1) = \{(q^{shift}_{move}, \#_1, 0, 1)\}$

(d) $\delta(q^{dec}_{return}, 1, c) = \{(q^{dec}_{return}, c, 0, 1)\} \mid c \in \{0, 1\}$

(e) $\delta(q^{dec}_{return}, 1, \#_2) = \{(q^{dec}, \#_2, 1, -1)\}$

(f) $\delta(q^{shift}_{move}, 1, c) = \{(q^{shift}_{move}, c, 0, 1)\} \mid c \in \{0, 1, \#_2\}$
(g) $\delta(q^{shift}_{move}, 1, B) = \{(q^{shift}, B, 0, -1)\}$
(h) $\delta(q^{shift}, 1, b) = \{(q^{shift}_{b}, 0, 0, -1)\} \mid b \in \{0, 1\}$
(i) $\delta(q^{shift}_{b}, 1, c) = \{(q^{shift}_{c}, b, 0, -1)\} \mid b, c \in \{0, 1\}$
(j) $\delta(q^{shift}_{0}, 1, \#_2) = \{(q^{copy}_{\rightarrow}, \#_2, 0, 1)\}$

6. Kopírovanie novovzniknutej adresy

(a) $\delta(q^{copy}_{\rightarrow}, 1, b) = \{(q^{copy}_{\rightarrow}, b, 0, 1)\} \mid b \in \{0, 1\}$
(b) $\delta(q^{copy}_{\rightarrow}, 1, \bar{b}) = \{(q^{copy}_{read}, \bar{b}, 0, -1)\} \mid b \in \{0, 1\}$
(c) $\delta(q^{copy}_{\rightarrow}, 1, B) = \{(q^{copy}_{read}, B, 0, -1)\}$
(d) $\delta(q^{copy}_{read}, 1, b) = \{(q^{\leftarrow}_{b}, \bar{b}, 0, -1)\} \mid b \in \{0, 1\}$
(e) $\delta(q^{copy}_{read}, 1, \#_2) = \{(q^{unmark}_{\leftarrow}, \#_2, 0, -1)\}$
(f) $\delta(q^{\leftarrow}_{b}, 1, c) = \{(q^{\leftarrow}_{b}, c, 0, -1)\} \mid b, c \in \{0, 1\}$
(g) $\delta(q^{\leftarrow}_{b}, 1, \#_2) = \{(q^{find}_{b}, \#_2, 0, -1)\} \mid b \in \{0, 1\}$
(h) $\delta(q^{find}_{b}, 1, \bar{c}) = \{(q^{find}_{b}, \bar{c}, 0, -1)\} \mid b, c \in \{0, 1\}$
(i) $\delta(q^{find}_{b}, 1, c) = \{(q^{copy}_{\Rightarrow}, \bar{b}, 0, 1)\} \mid b, c \in \{0, 1\}$
(j) $\delta(q^{copy}_{\Rightarrow}, 1, \bar{b}) = \{(q^{copy}_{\Rightarrow}, \bar{b}, 0, 1)\} \mid b \in \{0, 1\}$
(k) $\delta(q^{copy}_{\Rightarrow}, 1, \#_2) = \{(q^{copy}_{\rightarrow}, \#_2, 0, 1)\}$
(l) $\delta(q^{unmark}_{\leftarrow}, 1, \bar{b}) = \{(q^{unmark}_{\leftarrow}, \bar{b}, 0, -1)\} \mid b \in \{0, 1\}$
(m) $\delta(q^{unmark}_{\leftarrow}, 1, \#_1) = \{(q^{unmark}_{\rightarrow}, \#_1, 0, 1)\}$
(n) $\delta(q^{unmark}_{\rightarrow}, 1, \bar{b}) = \{(q^{unmark}_{\rightarrow}, b, 0, 1)\} \mid b \in \{0, 1\}$
(o) $\delta(q^{unmark}_{\rightarrow}, 1, \#_2) = \{(q^{unmark}_{\rightarrow}, \#_2, 0, 1)\}$
(p) $\delta(q^{unmark}_{\rightarrow}, 1, B) = \{(q_{rewind}, B, 0, -1)\}$

7. Posledný prechod - riešený separátne

(a) $\delta(q^{comp}_{read}, 1, \#_1) = \{(\bar{q}^{comp}_{\rightarrow}, \#_1, 0, 1)\}$
(b) $\delta(\bar{q}^{comp}_{\rightarrow}, 1, c) = \{(\bar{q}^{comp}_{\rightarrow}, c, 0, 1)\} \mid c \in \{0, 1\}$
(c) $\delta(\bar{q}^{comp}_{\rightarrow}, 1, \#_2) = \{(\bar{q}^{dec}, \#_2, 0, -1)\}$
(d) $\delta(\bar{q}^{dec}, 1, 0) = \{(\bar{q}^{dec}, 1, 0, -1)\}$
(e) $\delta(\bar{q}^{dec}, 1, 1) = \{(\bar{q}^{dec}_{return}, 0, 0, -1)\}$
(f) $\delta(\bar{q}^{dec}, B, 1) = \{(q^{last}_{one}, 1, 0, -1)\}$
(g) $\delta(\bar{q}^{dec}_{return}, 1, c) = \{(\bar{q}^{dec}_{return}, c, 0, 1)\} \mid c \in \{0, 1\}$
(h) $\delta(\bar{q}^{dec}_{return}, 1, \#_2) = \{(\bar{q}^{dec}, \#_2, 1, -1)\}$

(i) $\delta(q_{one}^{last}, B, 0) = \{(q_{one}^{last}, 0, 0, -1)\}$

(j) $\delta(q_{one}^{last}, B, \#_1) = \{(q_F, \#_1, 0, 0)\}$

Začneme tým, že na pracovnú pásku (1(a) a 1(b)) napíšeme k symbolov $\bar{1}$ (aspoň dva), taktiež overíme, či sa na začiatku vstupnej pásky nachádzajú dva symboly 0. Prejdeme do kontrolného cyklu, ktorý sa začína uhádnutím úseku na vstupnej páske, ktorý budeme kontrolovať (1(c)). Presunieme sa hlavou na pracovnej páske na začiatočnú pozíciu (1(d) a 1(e)). Porovnáme dĺžky úseku na vstupnej páske a aktuálny obsah pracovnej pásky (1(f) a 1(g)). Pomocou 1(h) buď opakujeme cyklus, alebo ho ukončíme.

Presunieme hlavu pracovnej pásky na začiatok (2(a) a 2(b)) a začneme kontrolný cyklus (2(c) - 2(h)), ktorý overí, či máme na pracovnej páske správny počet symbolov ($3k - 1$). Robíme to nasledovne: v jednom prechode cyklom prepíšeme jeden symbol $\bar{1}$ na $\tilde{1}$ a dva symboly 1 na $[1]$. Teda za každý symbol $\bar{1}$ prepíšeme dva symboly 1. Posledný prechod je riešený separátne, lebo v ňom treba prepísať len jeden symbol 1 (2(i)), a potom treba ukončiť cyklus. Po ukončení cyklu skontrolujeme (2(i) - 2(k)), či neostali nejaké nespracované symboly. Ak nejaké nájdeme, tak sme zle hádali počet symbolov na pracovnej páske a dôjde k zaseknutiu. Pri tejto kontrole nastavíme hlavu pracovnej pásky na jej začiatok. Špeciálne opatrenie obsahuje 2(i). Tento prípad nielen ošetruje posledný prechod, ale naviac po načítaní $BLANK$-u, napíše ešte jeden symbol $[1]$ naviac. Z technických príčin chceme mať pri kontrole vypchávky nie $3k - 1$ symbolov na pracovnej páske, ale $3k$.

Overíme, či počet núl v permutačnej časti je správny. Nastavíme čítaciu hlavu na začiatok (3(a)). Preskočíme prvé dve nuly (3(b) a 3(c)), lebo tie slúžia ako špeciálny oddeľovač. Nás zaujímajú iba nuly, ktoré sú medzi jednotlivými úsekmi. Za každé dve nuly, ktoré postupne nájdeme na vstupe, prečítame jeden symbol $\tilde{1}$ z pracovnej pásky (3(d), 3(e) ,3(f) ,3(h) a 3(i)). Keď prečítame všetky symboly $\tilde{1}$ z pracovnej pásky, ukončíme čítanie núl (3(g)); hlavu vstupnej pásky nastavíme na prvý symbol vypchávky, t.j. prvú jednotku, ktorá je hneď napravo od posledne skontrolovanej nuly. Znamená to, že sme skontrolovali celú permutačnú časť slova. Ďalej nemusíme kontrolovať, či sú nuly vo vypchávke, lebo to overíme pri kontrole vypchávky. Celkovo sme overili nasledovné vlastnosti permutačnej časti:

- Pre každý počet jednotiek, ktorý bol na pracovnej páske (od k až po $3k - 1$), existuje nejaký úsek na vstupe, ktorý má zhodný počet jednotiek, naviac je ukončený symbolom 0
- Počet úsekov v permutačnej časti slova je práve $2k$
- Počet núl v permutačnej časti slova (nerátajúc prvé dve) je práve $2k$

Z týchto vlastností nutne vyplýva tvar permutačnej časti slova podľa definície jazyka.

Preskúmame vypchávku. Treba si všimnúť, že pri celej kontrole vypchávky, nedovoľujeme čítať nuly na vstupe. Presunieme hlavu pracovnej pásky na jej koniec (4(a)). Vytvoríme prvú adresu (4(b) a 4(c)), ktorá začína symbolom $\#_1$ a potom nasleduje $log(n)$ núl a jedna jednotka. Hneď za ňou vytvoríme druhú adresu (4(d) a 4(e)) obdobným spôsobom, len bude začínať symbolom $\#_2$. Dĺžky oboch adries uhádneme rovnako dlhé (toto neskôr overíme) a naviac ich uhádneme dosť dlhé na to aby sa výpočet neskôr nezasekol.

Naštartujeme cyklus, ktorý skontroluje počet jednotiek vo vypchávke. V jednom prechode cyklu sa vykoná nasledovné: presunieme hlavu pracovnej pásky na prvý (najľavejší) nespracovaný symbol (4(f) - 4(h)), t.j. $\tilde{1}$ alebo $[1]$. Spracujeme ho (prepíšeme na (1)) (4(i)) a presunieme sa na koniec prvej adresy (na pracovnej páske) (4(j) a 4(k)). Spustíme vnútorný cyklus, ktorý bude posúvať hlavou na vstupe. V jednom prechode znížime hodnotu adresy o jeden (5(a) a 5(b)) a posunieme hlavu na vstupe o jedno políčko doprava. Toto robíme (5(d) a 5(e)), kým neznížime prvú adresu na nulovú (5(c)). Presunieme sa na koniec druhej adresy (5(f) a 5(g)) a zdvojnásobíme jej hodnotu vykonaním operácie SHIFT LEFT (5(h) a 5(i)). Po ukončení operácie SHIFT (5(j)), skopírujeme obsah druhej adresy do prvej.

Kopírovanie pracuje v cykle. Presunieme hlavu pracovnej pásky na najpravejší nespracovaný symbol (bez pruhu) (6(a) - 6(c)) druhej adresy. Spracujeme ho a uložíme do stavu (6(d)). Presunieme sa na prvý (najpravejší) nespracovaný symbol prvej adresy (6(e) - 6(h)) a prepíšeme ho spracovaným znakom, ktorý máme uložený v stave (6(i)). Presunieme sa do druhej adresy (6(j) a 6(k)). Cyklus opakujeme, kým nie sú všetky znaky spracované, resp. skopírované. Po skončení kopírovania odstránime pruhy (6(l) - 6(p)) a začneme nový prechod vonkajšieho cyklu. Treba si všimnúť, že pri práci s adresami vyžadujeme ich rovnakú dĺžku. Ak je dĺžka rôzna, dôjde k zaseknutiu. Taktiež, ak adresa nie je dosť dlhá, dôjde k zaseknutiu. Vonkajší cyklus prebehne presne $3k$-krát. Jednotiek má byť vo vypchávke $2^{(3k-1)+1} - 1$. Preto posledný prechod cyklom musíme ošetriť špeciálne.

Posledný prechod identifikujeme tak, že už máme všetky symboly na pracovnej páske (v časti počítadla) spracované (7(a)). Prejdeme do pruhovanej kópie vnútorného cyklu. 7(b) a 7(c) presunie hlavu pracovnej pásky na koniec prvej adresy. Vykonáme zníženie adresy (7(d) a 7(e)), presunutie hlavy na pracovnej páske na koniec adresy a posunutie hlavy na vstupe o jedno políčko doprava. Rozdiel je, že v poslednom prechode nerobíme SHIFT druhej adresy, lebo tú adresu už nepotrebujeme. Taktiež, ak je slovo správneho tvaru, tak sa nikdy nestane, že by sme išli znížiť minimálnu adresu. Správne slovo v poslednom prechode na vstupe číta $BLANK$. Toto detekujeme v 7(f) a overíme, či je čítacia hlava na správnej pozícii, t.j. či prvá adresa je tvaru $0^{O(log(n))}1$ (7(i) a 7(j)).

Keďže $k \leq log(n)$ priestor, ktorý potrebujeme na výpočet je $O(k) + O(log(n)) + O(log(n))$, teda celkovo $O(log(n))$.

Našli sme aspoň jeden jazyk, ktorý vylučuje možnosť tesnej simulácie. Teraz sa pokúsime rozšíriť tento jazyk na triedu jazykov.

Využijeme lemu o bariére, pomocou ktorej ukážeme, že G-systém potrebuje aspoň $O(n)$ krokov na akceptovanie jazyka $L = \{ww \mid w \in \Sigma^+\}$.

Ukázali sme, že $L \in ASPACE(log(n))$ (dokonca $L \in NSPACE(log(n))$). Teraz sporom ukážeme, že $L \notin GTIME(log(n))$. Predpokladajme, že existuje G-systém, ktorý generuje L v logaritmickom čase. Vyberme si ľubovoľné slovo z jazyka L. Nech je to $w = w_1 \# w_2$. Podľa lemy o bariére je možné preniesť najviac $O(log(n))$ symbolov z ľavej časti vetnej formy do pravej. Keďže dĺžka podslov w_1 a w_2 je $O(n)$ podslovo w_2 sa môže líšiť od w_1 v $n - log(n)$ znakoch. Formálnejšie povedané, vieme, že k danému odvodeniu podslov w_1, w_2, existuje slovo $\bar{w}_2$, ktoré sa líši od w_2 v najviac $\frac{n}{2} - log(n)$ znakoch. Keďže je komunikácia obmedzená na $O(log(n))$ infomácií, G-systém nevie rozlíšiť slová w_2 a $\bar{w}_2$. Vygeneruje aj slovo $w_1 \# \bar{w}_2$, ktoré ale nepatrí do jazyka, čo je spor.

Vieme, že vyššie spomenutý jazyk $L \in ASPACE(log(n))$ a súčasne $L \notin GTIME(log(n))$. Naviac $L \notin GTIME(f(n))$, pre $f(n) = o(n)$. Najbližšia trieda, do ktorej L patrí, je $GTIME(n)$ podľa lemy o bariére. Tento jazyk vylučuje možnosť tesnej simulácie.

Našli sme jazyk, ktorý odhaľuje slabosť G-systému - pomalá vnútorná komunikácia. Aby sme ukázali, že sa nedá spraviť tesná simulácia, potrebujeme viac ako tento jeden jazyk. Fakt, že $\exists L \in ASPACE(log(n))$ a súčasne $L \notin GTIME(log(n))$, nevylučuje možnosť, že existuje tesná simulácia pre iné funkcie ako $log(n)$. Aby sme ukázali, že toto nie je možné, urobíme rozšírenie jazyka L. Uvažujme jazyk

$$L_{f(n),L} = \{ww \mid w \in L, L \in ASPACE(f(n)), L \notin ASPACE(g(n)), g(n) = o(f(n))\}$$

pre $f(n) = \Omega(log(n)) \wedge f(n) = o(n)$. Akceptácia takéhoto jazyka pre ATS vyzerá nasledovne. Keďže $w \in L$ a $L \in ASPACE(f(n))$, existuje ATS, ktorý akceptuje w v priestore $f(n)$. Spustíme tento ATS na prvé podslovo. Náročnosť sa asymptoticky nemení, lebo vstup sa zmenšil iba o polovicu, čo je prenásobenie konštantou. Potom spustíme ATS na porovnanie oboch podslov (popísaný vyššie). Je zrejmé, že celková simulácia potrebuje $O(f(n))$ priestoru.

Táto konštrukcia nie je použiteľná pre funkcie $f(n) = o(log(n))$, lebo nech $f(n)$ je akokoľvek malé, tak celková priestorová náročnosť bude $log(n)$ (lebo toľko priestoru potrebujeme na porovnanie).

Z [6] vieme, že existuje neregulárny jazyk $L_f \in ASPACE(f(n))$, pričom $L_f \notin NSPACE(f(n))$, pre $f(n) = \Omega(loglog(n)) \wedge f(n) = o(log(n))$. Keďže G-systém vie v každom kroku zväčšiť vetnú formu, len o k násobok (k je maximum cez všetky dĺžky prepisovacích pravidiel 1-a prekladača), vieme ním vygenerovať počas p krokov vetnú formu dĺžky najviac k^p. Inými slovami, pre $f(n) = o(log(n))$ vie vygenerovať iba konečné jazyky. Môžeme teda povedať, že $L_f \in ASPACE(f(n))$ a $L_f \notin GTIME(f(n))$, pre $f(n) = \Omega(loglog(n)) \wedge f(n) = o(log(n))$.

Veta 2.2.3: $ATIME(f(n)) \subseteq \bigcup_c GTIME(c^{f(n)})$, pre $f(n) = o(n)$.

Dôkaz: Zostrojíme pre jazyk $L = \{ww \mid w \in \Sigma_L\}$ ATS s rýchlym prístupom na pásku, ktorý akceptuje L v čase $O(log(n))$. ATS má jednu vstupnú pásku, z ktorej môže len čítať, jednu pracovnú pásku a register. Podrobný komentár ku δ - funkcii sa nachádza hneď za konštrukciou. $A = (K, \Sigma, \Gamma, \delta, q_0, F)$, kde $\Sigma = \Sigma_L$, $\Gamma = \Sigma_L$,$K_\exists = \{q_0, q_{init}, q_{init}^{\leftarrow}, q^{\leftarrow}, q_{move}, q^{\Leftarrow}, q_F\} \cup \{q_{save_a} \mid a \in \Sigma\}$, $K_\forall = \{q_{split}\}$, $F = \{q_F\}$ a δ je v tvare

δ(state, input, work, register) = {(state, work, register, input dir, work dir, register dir)}.

- $\delta(q_0, a, B, 0) = \{(q_{init}, \bar{0}, 0, 0, 1, 1)\} \mid a \in \Sigma$
- $\delta(q_{init}, a, B, 0) = \{(q_{init}, 0, 0, 0, 1, 1)\} \mid a \in \Sigma$
- $\delta(q_{init}, a, B, \$) = \{(q_{init}^{\leftarrow}, B, \$, 0, -1, -1)\} \mid a \in \Sigma$
- $\delta(q_{init}^{\leftarrow}, a, 0, 0) = \{(q_{init}^{\leftarrow}, 0, 0, 0, -1, -1)\} \mid a \in \Sigma$
- $\delta(q_{init}^{\leftarrow}, a, \bar{0}, 0) = \{(q_{split}, \bar{0}, 0, 0, 1, 0)\} \mid a \in \Sigma$

- $\delta(q_{split}, a, 0, 0) = \{(q_{split}, 0, 0, 0, 1, 0), (q_{split}, 1, 0, 0, 1, 0)\} \mid a \in \Sigma$
- $\delta(q_{split}, a, B, 0) = \{(q^{\leftarrow}, B, 0, 0, -1, 0)\} \mid a \in \Sigma$
- $\delta(q^{\leftarrow}, a, \triangle, 0) = \{(q^{\leftarrow}, \triangle, 0, 0, -1, 0)\} \mid \triangle \in \{0, 1\}, a \in \Sigma$
- $\delta(q^{\leftarrow}, a, \bar{0}, 0) = \{(q_{move}, \bar{0}, 0, 0, 1, 1)\} \mid a \in \Sigma$
- $\delta(q_{move}, a, \triangle, 0) = \{(q_{move}, \triangle, \triangle, 0, 1, 1)\} \mid \triangle \in \{0, 1\}, a \in \Sigma$
- $\delta(q_{move}, a, B, \$) = \{(q^{\Leftarrow}, B, \$, J, -1, 0)\} \mid a \in \Sigma$
- $\delta(q^{\Leftarrow}, a, \triangle, 0) = \{(q^{\Leftarrow}, \triangle, 0, 0, -1, 0)\} \mid \triangle \in \{0, 1\}, a \in \Sigma$
- $\delta(q^{\Leftarrow}, a, \bar{0}, 0) = \{(q_{save_a}, \bar{1}, 1, J, 0, 0)\} \mid a \in \Sigma$
- $\delta(q_{save_a}, a, \bar{1}, 1) = \{(q_F, \bar{1}, 1, 0, 0, 0)\} \mid a \in \Sigma$

Najprv inicializujeme pracovnú pásku. Skopírujeme obsah registra ($0^{log(n)}$) na pracovnú pásku a prvý symbol 0 označíme pruhom (má špeciálny význam) - $\bar{0}$. Po skončení kopírovania presunieme hlavu registra aj pracovnej pásky na začiatok. Posunieme hlavu pracovnej pásky tesne za symbol $\bar{0}$. Teraz budeme odvetvovať nové procesy tak, že prvý proces prepíše symbol 0 na 0 a druhý proces ho prepíše na 1. V oboch prípadoch sa posunieme o jeden symbol doprava. Vetvenie pokračuje, až kým neprídeme na prvý $BLANK$, t.j. prejdeme $log(n) - 1$ symbolov. Znova presunieme hlavu pracovnej pásky na jej začiatok (symbol $\bar{0}$). Presunieme hlavu za tento symbol a taktiež posunieme hlavu registra o jeden symbol doprava. Skopírujeme zvyšok pracovnej pásky (až po $BLANK$) do registra. Po prekopírovaní použijeme inštrukciu JUMP a skočíme s hlavou na vstupe na políčko pásky s adresou, ktorá je v registri. Opäť presunieme hlavu registra a pracovnej pásky na začiatok. Uložíme symbol, ktorý číta hlava zo vstupu do stavu. Prepíšeme najľavejší bit registra z 0 na 1. Teraz každý proces skočí na takto zmodifikovanú adresu a porovná čítané políčko so vstupom. V prípade zhody akceptuje.

Časové nároky sú nasledovné: inicializácia trvá $O(log(n))$ krokov. Presun pracovnej a registrovej hlavy na začiatok trvá $O(log(n))$ krokov. Vetvenie trvá $O(log(n))$ krokov. Presun pracovnej a registrovej hlavy na začiatok trvá $O(log(n))$ krokov. Kopírovanie adresy z pracovnej pásky do registra trvá $O(log(n))$ krokov. Inštrukcia JUMP je vykonaná v konštantnom čase. Presun pracovnej a registrovej hlavy na začiatok trvá $O(log(n))$ krokov. Modifikácia najľavejšieho bitu adresy je v konštantnom čase. Inštrukcia JUMP je vykonaná v konštantnom čase. Porovnanie symbolov je vykonané v konštantnom čase. Celkový čas potrebný na akceptáciu je $O(log(n))$.

Vieme, že vyššie spomenutý jazyk $L \in ATIME(log(n))$ a súčasne $L \notin GTIME(log(n))$. Naviac $L \notin GTIME(f(n))$, pre $f(n) = o(n)$. Najbližšia trieda, do ktorej L patrí je $GTIME(n)$ podľa lemy o bariére. Tento jazyk nám vylučuje možnosť tesnej simulácie.

Dolný odhad urobíme obdobne ako v predchádzajúcom dôkaze - rozšírením jazyka L.

$$L_{f(n),L} = \{ww \mid w \in L, L \in ATIME(f(n)), L \notin ATIME(g(n)), g(n) = o(f(n))\}$$

pre $f(n) = \Omega(log(n)) \wedge f(n) = o(n)$.

Veta 2.2.4: $GTIME(f(n)) \subseteq ATIME(f^2(n))$, pre $f(n) = \Omega(log(n)) \wedge f(n) = o(n)$.

Dôkaz: Vieme z [1], že $GTIME(f(n)) \subseteq NSPACE(f(n))$ a
$NSPACE(f(n)) \subseteq ATIME(f^2(n))$, pre $f(n) = \Omega(log(n))$. Z týchto tvrdení vyplýva naše tvrdenie.

V ďalšom popíšeme kandidáta na kontrapríkladový jazyk pre naše tvrdenie. Využijeme vlastnosti rýchlogenerovateľných jazykov. Treba pripomenúť, že sa zaoberáme jednopáskovým modelom ATS, pokiaľ nie je povedané inak.

Uvažujme jazyk $L = (ab)^{2^m}$. Tento jazyk vieme ľahko vygenerovať G-systémom v čase $log(n)$. Tento jazyk je natoľko jednoduchý, že ho vieme akceptovať ATS tiež v čase $log(n)$. Popíšeme konštrukciu ATS kvôli tomu, aby sme videli, v čom spočíva jednoduchosť tohoto rýchlo generovateľného jazyka. ATS najprv na pracovnú pásku skopíruje počiatočný obsah registra (samé nuly). Potom sa začne binárne vetviť tak, že na pracovnej páske prepisuje adresu zľava, vždy jeden proces prepíše jeden bit na 0 a druhý proces na 1. Vetvenie ukončíme tesne pred posledným bitom adresy (najpravejším). Toto opatrenie zabezpečí, že každý proces bude mať na pracovnej páske adresu políčka, ktorým začína nejaké podslovo. Teraz už len každý proces nakŕmi adresu z pracovnej pásky do registra, skočí na príslušné políčko, a ak prečíta podslovo ab, tak akceptuje. Vetvenie trvá $O(log(n))$ krokov a rovnako ako naplnenie registra. Celkový čas potrebný na akceptáciu je $O(log(n))$.

Skúsme odhaliť vlastnosť, ktorá robí tento rýchlo generovateľný jazyk tak ľahko akceptovateľným. Zdá sa, že to spočíva v rovnakej dĺžke podslov a vetvení binárneho stromu nad vstupnou páskou. Skúsime zmeniť dĺžku podslova na viac ako dva symboly.

Uvažujme jazyk $L = (abc)^{3^m}$. V tomto prípade môžeme skúsiť postupovať rovnako. G-systém nemá problém tento jazyk vygenerovať v čase $O(log(n))$. ATS začne mať problémy, lebo dĺžka slova a vetvenie sa nezhodujú. Toto môžeme riešiť viacerými spôsobmi.

Prvý spôsob je písať na pracovnú pásku adresy v k-árnej sústave. Potom je potrebné spraviť preklad z k-árnej sústavy do binárnej v čase $O(log(n))$. Uvažujme vstupné slovo dĺžky n v k-árnej sústave. Urobiť preklad z k-árnej sústavy do binárnej sa zrejme nedá urobiť v lineárnom čase, lebo zvyšok po delení, ktorý si musíme pamätať, sa nedá ohraničiť konštantou. Ľahko sa nahliadne, že jeho veľkosť môže byť až $O(n)$. Stačí zobrať vstupné slovo, ktoré má vo všetkých cifrách symbol $k-1$, teda ide o najväčšie číslo, ktoré sa dá zapisať v tejto sústave vzhľadom na túto dĺžku. V každom kroku zvyšok, ktorý si pamätáme, rastie, až kým nedočítame vstupné slovo. Až potom začne zvyšok klesať. Keďže zvyšok môže byť až veľkosti $O(n)$, nemôžeme si ho uložiť v stave, ale niekde na páske. Musíme sa teda pri preklade vždy k nemu vracať a to vedie k dĺžke výpočtu $\omega(n)$ krokov. Ďalší intuitívny dôvod, prečo sa nedá spraviť redukcia z k-árneho stromu na binárny bez nejakej časovej straty, je fakt, že počet pások TS tiež nevieme zredukovať na jednu bez časovej straty. Preklad adresy nie je vhodný prístup.

Iný prístup je prispôsobiť početnosť vetvenia stromu nad vstupnou páskou. Teda v tomto prípade by stupeň vetvenia nebol dva, ale tri. Na pracovnú pásku by sme písali čísla v trojkovej sústave a zvyšok postupu by bol rovnaký. Otázka je, či model ATS s rýchlym prístupom na vstup s binárnym stromom je ekvivalentný s ternárnym stromom (všeobecnejšie k-árnym stromom).

Treba si všimnúť, že zvýšenie stupňa vetvenia o konštantný počet, asymptoticky nemení hĺbku stromu a ani dĺžku adresy (zmena základu logaritmu je prenásobenie konštantou). Definujeme rozšírenie pásky s rýchlym prístupom na vstup, ktorá nebude mať nad páskou zavesený binárny, ale k-árny strom. Intuitívne ide o podobné rozšírenie ako je k-páskový TS, vzhľadom ku jednopáskovému. Adresa registra sa, očakávateľne, tiež mení. Namiesto abecedy $\{0, 1\}$, bude mať abecedu $\{0, 1, \ldots, k-1\}$. S takýmto rozšírením vieme akceptovať jazyky

$$L = \{(w)^{k^m} \mid w \in \Sigma, \mid w \mid = k, m \geq 0\}$$

pomocou ATS s rýchlym prístupom na vstup s k-árnym vetvením. Teda tieto jazyky nie sú pre naše účely zaujímavé.

Zdá sa, že tá vlastnosť, ktorú hľadáme, je synchronizácia dĺžky jednotlivých podslov. Pevná dĺžka podslov implikuje pevné pravidlá na výpočet adresy. Preskúmame, čo sa stane, ak povolíme mať rôzne dĺžky podslov. Nech $L_K = \{v_1, v_2, \ldots, v_k \mid k > 0\}$ je ľubovoľný neprázdny konečný jazyk, neobsahujúci ϵ. Nech $L = \{w_1 w_2 \ldots w_{2^m} \mid w_i \in L_K, m \geq 0\}$. Tento rýchlo generovateľný jazyk je G-systémom vygenerovateľný v čase $O(log(n))$. Exponenciálne množenie počiatočných neterminálov trvá $O(log(n))$ krokov. Potom v konštantom čase vieme z každého neterminálu vygenerovať slovo z konečného jazyka. Na druhej strane ATS začne mať veľké problémy, lebo výpočet adresy neprichádza do úvahy, nakoľko dĺžky slov a teda aj adresy ich začiatkov sú úplne ľubovoľné. Bez alternovania nemá ATS šancu na úspech, nakoľko nestihne prečítať celý vstup. Pri alternovaní dochádza ku deleniu vstupu medzi procesy, ktoré medzi sebou nekomunikujú.

Uvažujme konkrétny jazyk:

$$L = \{w_1 w_2 \ldots w_{2^m} \mid w_i \in L_{ab}, m \geq 0\}, \text{ kde } L_{ab} = \{ab, ab^2, \ldots, ab^k \mid k > 0\}$$

Vstup medzi procesy môžeme rozdeliť na časti rovnakej alebo rôznej dĺžky. V prípade častí rovnakej dĺžky dochádza ku problému. Môže sa stať, že jedno podslovo bude rozdelené na dve časti, pričom každú časť dostane na vstup iný proces. Keďže procesy nekomunikujú, nastáva problém pri akceptácii. Pri častiach rôznej dĺžky musíme uhádnuť rozdelenie slov pre vstupy jednotlivých procesov tak, aby nedošlo ku rozbitiu niektorého podslova na dve časti, lebo by sme skončili s rovnakým problémom ako v predchádzajúcom prípade. Zamyslíme sa, ako uhádnuť adresy začiatkov podslov. Zrejme neexistuje deterministický spôsob, ako vypočítať jednotlivé adresy, ako to bolo v prípade pevnej dĺžky podslov. Skúsme využiť nedeterminizmus. Nech každý proces uhádne adresu, a potom to overíme. Správna adresa je vtedy, ak sa proces po skoku ocitne na políčku obsahujúce a. Potom každý proces skontroluje, či príslušné podslovo patrí do jazyka L_{ab}. Druhá podmienka je, že celý vstup musí byť prečítaný. Ale keďže procesy nekomunikujú, tento fakt nemáme ako overiť.

Ďalší spôsob je uhádnuť cieľové adresy pre všetky procesy (okrem prvého, lebo ten má adresu jasnú). Potom overíme, či sme hádali správne. Budeme to robiť takto: keď si zoberieme dva susedné procesy (t.j. procesy, ktoré kontrolujú dva úseky vstupu, ktoré sú vedľa seba), pred odvetvením sa dohodnú na cieľovej adrese pravého procesu. Pravý proces skočí na túto adresu a ľavý skočí na svoju adresu. Ľavý proces potom bude čítať vstup a zvyšovať svoju adresu až kým nepríde na prvé a. Toto zaberie konštantne veľa času, lebo podslovo je konštantnej dĺžky. Potom

porovná svoju adresu a adresu, na ktorej sa dohodol s pravým procesom. Ak sú zhodné, tak akceptuje. Teda všeobecnejšie i-ty proces, kontroluje svoju časť vstupu a zároveň kontroluje, či $i+1$-vý uhádol svoju cieľovú adresu správne (prvý proces je výnimka, toho nekontroluje nikto, lebo jeho adresa je od začiatku známa). Táto konštrukcia má jeden problém, a to je fakt, že procesy sa pred odvetvením musia dohodnúť na spoločnej adrese. Postupné odvetvovanie (t.j. strom hrebeň - má málo uzlov a veľkú hĺbku, je nevyvážený) je príliš pomalé. Aby sa to dalo stihnúť v čase $O(log(n))$, musí byť akceptačný výpočet vyváženejší. Toto je ale sporná požiadavka so spoločným dohodnutím adresy. Totiž ak má byť strom vyvážený, musia sa čo najskôr procesy odvetviť a ešte sa nestihnú dohodnúť na spoločnej adrese. Problém je najlepšie vidieť, ak zoberieme najpravejší listový proces ľavého podstromu (z pohľadu koreňa stromu) a najľavejší listový proces pravého podstromu. Tieto dva procesy sú susedné, ale nemohli sa stihnúť dohodnúť na spoločnej adrese.

Lepší spôsob sa zdá byť rekurzívny prístup. Hlavný proces (koreň) má overiť celý vstup, t.j. interval od adresy $0^{log(n)}$ až po adresu $1^{log(n)}$. Overí to takto: uhádne stredový deliaci bod. Tento bod nie je nutne v strede vstupu, ale je niekde v strede, ak berieme do úvahy len políčka, ktoré obsahujú symbol a. Uhádneme ho tým, že uhádneme adresu políčka, ktoré obsahuje symbol a a je stredovým deliacim bodom. Uhádnutie adresy trvá $O(log(n))$ krokov, nakoľko jej dĺžka je $O(log(n))$. Rozvetvíme sa na dva procesy, jeden bude kontrolovať interval od adresy $0^{log(n)}$ až po uhádnutú adresu a druhý skontroluje interval od uhádnutej adresy až po $1^{log(n)}$. Takto sa vetvíme, až kým sa nedopracujeme k listovým procesom, t.j. takým, že majú skontrolovať iba úsek, ktorý už neobsahuje viac ako jedno a. V tomto prípade proces skočí na začiatočnú adresu intervalu, skontroluje či je tam a, dočíta zvyšok podslova (overí či patrí do L_{ab}), pričom si zvyšuje počiatočnú adresu. Toto robí, kým nepríde na adresu, ktorú ma zaznamenanú ako koniec intervalu. V tomto okamihu musí platiť, že cestou neprečítal ani jedno ďalšie a a zároveň najbližšie políčko napravo obsahuje a. V tomto prípade proces akceptuje. Strom vetvenia má hĺbku $O(log(n))$. Teda celkový čas je $O(log(n)).O(log(n))$, t.j. $O(log^2(n))$. Tento prístup vedie k príliš pomalému riešeniu.

Modifikácia jazyka $L = \{w_1\#w_2\#\ldots\#w_{2^m} \mid w_i \in L_K, m \geq 0\}$ trochu uľahčuje problém, lebo procesy vedia, kde začína a končí podslovo. Hlavný problém to však nerieši. Naviac, ak sa ukáže, že to nejde pre jazyk L_{ab}, tak to nepôjde ani pre verziu s mrežami. Ak dovolíme, aby $\epsilon \in L_K$ tak to na podstate problému nič nemení, len dovolíme procesom akceptovať ϵ.

Hypotéza: $L \notin ATIME(log(n))$, kde $L = \{w_1w_2\ldots w_{2^m} \mid w_i \in L_{ab}, m \geq 0\}$, pričom $L_{ab} = \{ab, ab^2, \ldots, ab^k \mid k > 0\}$.

2.3 Porovnanie tried zložitosti pre $f(n) = O(c)$

Veta 2.3.1: $GTIME(f(n)) \subseteq ASPACE(f(n))$, pre $f(n) = O(c)$.

Dôkaz: G-systém vie vygenerovať v konštantnom čase iba všetky konečné jazyky, teda $GTIME(c) = \mathcal{F}$. Naproti tomu ATS v konštantnom priestore vie akceptovať všetky regulárne jazyky (alternujúce konečné automaty, ktorých sila je presne $\mathcal{R}$). Teda $ASPACE(c) = \mathcal{R}$. Opačná inklúzia evidentne neplatí, a teda vieme dokonca povedať, že $GTIME(f(n)) \subset ASPACE(f(n))$, pre $f(n) = O(c)$.

Veta 2.3.2: $GTIME(f(n)) \subseteq ATIME(f(n))$, pre $f(n) = O(c)$.

Dôkaz: Z predchádzajúcej vety vieme, že $GTIME(c) = \mathcal{F}$. Naproti tomu ATS v konštantnom čase vie akceptovať všetky konečné jazyky, ale ešte aj niečo naviac. Napríklad jazyk $\{w \mid w \in \Sigma^*\}$ vie akceptovať v konštantnom čase, zatiaľ čo G-systém ho v konštatnom čase nevie vygenerovať. Opačná inklúzia evidentne neplatí, a teda vieme, že $GTIME(f(n)) \subset ATIME(f(n))$, pre $f(n) = O(c)$. Toto tvrdenie samozrejme platí len za predpokladu, že ATS nemusí prečítať celý vstup, a taktiež nemusí kontrolovať, či symboly na páske patria do abecedy. Bez týchto predpokladov vie ATS akceptovať len konečné jazyky.

ZÁVER

V práci sme prezentovali prehľad už známych vzťahov tried zložitosti oboch modelov. Úspešne sme analyzovali oba modely a identifikovali ich slabé a silné miesta. Na základe týchto novonadobudnutých poznatkov sa nám podarilo, za určitých podmienok, skonštruovať tesnú simuláciu ATS pomocou G-systému a následne sme dokázali jej správnosť. Ďalej sme našli kontrapríkladový jazyk, ktorý, za určitých podmienok, dokáže ATS akceptovať, ale G-systém ho nevie generovať. Oba výsledky čiastočne objasňujú vzťahy modelov.

Sformulovaním dvoch nedokázaných tvrdení sme pripravili podklady pre ďalšiu prácu. Našli sme jazyky, ktoré sú dobrými kandidátmi na kontrapríkladové jazyky pri oboch tvrdeniach (dolných odhadoch).

V oblasti ešte ostáva viacero nepreskúmaných vzťahov - otázka vzťahu oboch modelov je stále otvorená a aktuálna, preto ďalšia práca v tejto oblasti je výzvou a zaujímavým problémom. Budúcim prácam v tejto oblasti želáme veľa úspechov.

LITERATÚRA

[1] BRANISLAV ROVAN: Teória paralerných výpočtov (prednáška), Katedra informatiky, FMFI UK

[2] BRANISLAV ROVAN: Formálne Jazyky a Automaty (prednáška, spísal Michal Forišek), Katedra informatiky, FMFI UK

[3] MARTIN KRÁLIK: Deterministic generative systems, diplomová práca, Katedra informatiky, FMFI UK, apríl 2002

[4] DUŠAN KRCHO: Non-determinism in Generative systems, diplomová práca, Katedra informatiky, FMFI UK, december 2001

[5] MARIÁN SLAŠŤAN: Paralell communicating grammar systems, diplomová práca, Katedra informatiky, FMFI UK, apríl 2000

[6] VILIAM GEFFERT, CARLO MEREGHETTI, GIOVANNI PIGHIZZINI: Sublogaritmic bounds on space and reversals, SIAM Journal on Computing, Vol. 28, No. 1, pp. 325 - 340, február 1999