Fakulta matematiky, fyziky a informatiky
Univerzita Komenského, Bratislava
Katedra Informatiky

Paralelné kooperujúce systémy gramatík

diplomová práca

autor: Lýdia Hanusková
vedúci dipl. práce: Prof. RNDr. Branislav Rovan PhD.
Bratislava, Apríl 2005

Čestne prehlasujem, že som diplomovú prácu vypracovala samostatne s použitím uvedenej literatúry.

Bratislava, Apríl 2005

Lýdia Hanusková

Ďakujem môjmu diplomovému vedúcemu Prof. RNDr. Branislavovi Rovanovi PhD. za cenné rady a pripomienky pri vypracovávaní tejto diplomovej práce. Chcela by som tiež poďakovať mojej rodine za vrelú podporu počas môjho štúdia.

Obsah

1 Úvod

Model paralelných kooperujúcich systémov gramatík ($PCGS$) sa skúma viac ako 15 rokov. Už v roku 1989 Gh.Paun a L.Sântean v článku *Parallel communicating grammar systems: The regular case* zaviedli tento model. Vzniklo veľa článkov, ktoré skúmali rôzne aspekty tohto modelu. Autori vo svojich výsledkoch často používali rôzne predpoklady na gramatiky v komponentách $PCGS$. Nie je vždy je jasné, či sa zmenou predpokladov mení aj popisná sila $PCGS$ a či sú tie predpoklady podstatné pre platnosť výsledku.

Cieľom predkladanej práce je analyzovať existujúce tvrdenia v oblasti $PCGS$ s regulárnymi komponentami ($PCGSREG$) z hľadiska predpokladov na gramatiky v komponentách. V práci sa zaoberáme vplyvom predpokladu striktnosti na regulárne gramatiky v komponentách $PCGS$. Predkladaná práca pozostáva z kapitol Základné definície a pojmy, $PCGS$ so striktne regulárnymi komponentami a Simulácia $PCGS$ Turingovym strojom.

V kapitole Základné definície a pojmy zavedieme model paralelných kooperujúcich systémov gramatík s regulárnymi komponentami, zadefinujeme regulárnu gramatiku, striktne regulárnu gramatiku a ďalšie pojmy pre potreby predkladanej práce.

Kapitola $PCGS$ so striktne regulárnymi komponentami ($PCGSSREG$) obsahuje tvrdenia popisujúce vlastnosti odvodení $PCGSSREG$, ktoré využívame v tejto práci. Hlavným výsledkom je Veta 3.1, v ktorej tvrdíme, že $PCGSSREG$ stupňa m sú slabšie ako $PCGSREG$ stupňa m pre každé pevné $m \geq 3$. Motiváciou pre vznik tohto tvrdenia bol dôkaz nekonečnosti hierarchie $PCGSREG$ v článkoch [HKK94, SK92]. Konkrétne dôkaz tvrdenia, že jazyk $L = \{a_1^n a_2^n \dots a_{2m}^n \mid n \in N\}$ patrí do triedy jazykov generovaných $PCGSREG$ stupňa $m + 1$, kde $m \geq 1$. Systém skonštruovaný v tomto dôkaze a generujúci jazyk L podstatne využíva reťazové pravidlá. Striktne regulárne gramatiky reťazové pravidlá neobsahujú a to nás priviedlo k dôkazu Vety 3.1, v ktorom použijeme jazyk L.

V kapitole Simulácia $PCGS$ Turingovym strojom sa zaoberáme simuláciou $PCGS$ Turingovym strojom v lineárnom čase. V článku [HKK94] bol dosiahnutý výsledok, že $PCGSREG$ majúce acyklickú komunikačnú štruktúru vieme simulovať Turingovym strojom v lineárnom čase. Ale pre $PCGSREG$ bez obmedzenia komunikčnej štruktúry nie je známy výsledok o simulácii Turingovym strojom v lineárnom čase. To nás inšpirovalo k otázke, či nie je možné simulovať $PCGSSREG$ Turingovym strojom v lineárnom čase. Vo vete 4.1 ukážeme, že $PCGS$ so stále generujúcimi striktne regulárnymi komponentami vieme simulovať Turingovym strojom v lineárnom čase. Dôkaz tohto tvrdenia je konštrukčný. Uvedieme konštrukciu Turingovho stroja simulujúceho $PCGS$ so stále generujúcimi striktne regulárnymi komponentami a technické detaily konštrukcie ilustrujeme na príkladoch, dokážeme že, skonštruovaný Turingov stroj simuluje daný systém, a že ho simuluje v lineárnom čase. Konštrukcia Turingovho stroja v dôkaze Vety 4.1 je technicky náročný a prináša nový pohľad na simuláciu $PCGS$ Turingovym strojom.

Predkladaná práca prináša dva nové výsledky, ktoré prispievajú k porozumeniu významu striktnosti v $PCGSREG$. Našou snahou bolo naplniť cieľ práce a dosiahnuť zaujimavé výsledky, ktorými je možné sa ďalej inšpirovať a tak nadchnúť čitateľa pre ďalšie skúmanie modelu paralelných kooperujúcich systémov gramatík.

2 Základné definície a pojmy

Zavedieme definíciu paralelného kooperujúceho systému gramatík s regulárnymi komponentami (označujeme $PCGSREG$) a súvisiacich pojmov a označení. Tieto definície vychádzajú z [Sla00]. Nech V je abeceda, množinu všetkých slov nad V označujeme V^*, prázdne slovo označujeme ϵ, $|x|$ je dĺžka slova $x, x \in V^*$ a $|x|_U$ je počet výskytov symbolov z abecedy U v slove $x, x \in V^*$.

Definícia 2.1. *Regulárna gramatika* je frázová gramatika, v ktorej $P \subseteq N \times (T^* \cup T^* N)$

Definícia 2.2. Nech $m \geq 1$ je prirodzené číslo. *Paralelným kooperujúcim systémom gramatík s regulárnymi komponentami stupňa m* (označujeme $PCGS_mREG$) nazývame $m+3$-ticu

$$\Gamma \quad = \quad (N, K, T, G_1, \ldots, G_m),$$

kde N je abeceda neterminálov, T je abeceda terminálov, $K = \{Q_1, \ldots, Q_m\}$ je množina komunikačných symbolov (tieto množiny sú disjunktné) a

$$G_i \quad = \quad (N \cup K, T, P_i, \sigma_i), 1 \leq i \leq m,$$

sú regulárne gramatiky, pričom $\forall i, P_i \subseteq N \times (T^* \cup T^*(N \cup K))$ Píšeme $V_\Gamma = N \cup K \cup T$.

Gramatiku $G_i, 1 \leq i \leq m$, nazývame *komponentom systému.* Prvý komponent G_1 nazývame *hlavnou gramatikou. Konfiguráciou systému* nazývame m-ticu $(x_1, \ldots, x_m)$ slov z V_Γ^*. *Komponentom konfigurácie* (alebo vetnou formou) nazývame x_i, pre nejaké $i, 1 \leq i \leq m$ alebo hovoríme len o komponente ak odkaz na konfiguráciu je zrejmý z kontextu.

Definícia 2.3. Nech $\Gamma = (N, K, T, G_1, \ldots, G_m), m \geq 1$ je $PCGS_mREG$, potom *krok odvodenia* systému definujeme ako binárnu reláciu $\Rightarrow_\Gamma$ nad konfiguráciami systému $(x_1, \ldots, x_m)$, $(y_1, \ldots, y_m)$, $x_i, y_i \in V_\Gamma^*, 1 \leq i \leq m$, definujeme takto: platí $(x_1, \ldots, x_m) \Rightarrow_\Gamma (y_1, \ldots, y_m)$ ak nastane jeden z dvoch prípadov:

1. $|x_i|_K = 0, 1 \leq i \leq m$ a $\forall i, 1 \leq i \leq m$ existuje, krok odvodenia v gramatike G_i, $x_i \Rightarrow_{G_i} y_i$ alebo $x_i \in T^*$ potom $y_i = x_i$

2. $\exists i, 1 \leq i \leq m$ také že $|x_i|_K > 0$, potom pre každé také i nech $x_i = zQ_{j_i}$, $1 \leq j_i \leq m, z \in T^*$, ak $|x_{j_i}|_K = 0$, potom $y_i = zx_{j_i}$ a $y_{j_i} = S_{j_i}$. Ak $|x_{j_i}|_K \neq 0$, tak $y_i = x_i, \forall i, 1 \leq i \leq m$, pre všetky vyššie nedefinované y_i, $y_i = x_i$.

V prvom prípade hovoríme o *prepisovacom kroku* a v druhom o *komunikačnom kroku* odvodenia. Prvý krok odvodenia *PCGSREG* je zrejme prepisovací krok. Pri komunikácií komponent x_i danej konfigurácie nahradí komunikačný symbol Q_i. Hovoríme, že Q_i je *uspokojený*. Komunikačný krok má prednosť pred prepisovacím krokom. Ak nejaké komunikačné symboly nie sú uspokojené v danom komunikačnom kroku, budú uspokojené v nasledovných krokoch alebo sa systém zastaví. Prepisovanie nie je možné ak sa v ľubovoľnej vetnej forme gramatík systému vyskytuje aspoň jeden komunikačný symbol.

Definícia 2.4. *Odvodenie PCGSREG* je postupnosť konfigurácií taká, že po sebe nasledujúce konfigurácie sú v relácii $\Rightarrow$ a prvá konfigurácia je $(\sigma_1, \ldots, \sigma_2)$.

Poznámka 1. Reflexívny tranzitívny uzáver relácie $\Rightarrow$, označujeme $\Rightarrow^*$.

Poznámka 2. Odvodenie *PCGSREG* sa môže zastaviť v dvoch prípadoch:

1. v prepisovacom kroku v aktuálnej konfigurácií $(x_1, \ldots, x_m)$, kde komponent x_i nie je terminálny vzhľadom na G_i, ale v G_i neexistuje pravidlo aplikovateľné na x_i

2. v komunikačnom kroku nastane zacyklenie požiadaviek gramatík, vetná forma gramatiky G_{i_1} obsahuje Q_{i_2}, gramatika G_{i_2} obsahuje Q_{i_3}, atď. až vetná forma gramatiky G_{i_k} obsahuje Q_{i_1} $(k \geq 1, 1 \leq i_1, \ldots, i_k \leq m)$. Zrejme žiaden z $Q_{i_j}, 1 \leq j \leq k$ nemôže byť uspokojený.

Poznámka 3. Nech Γ je *PCGSREG*, potom orientovaný graf (V, E), kde $V = \{G \mid G$ je komponent systému $\Gamma\}$, $(G_i, G_j) \in E, G_i, G_j$ sú komponenty systému, a platí že aspoň jedno pravidlo gramatiky G_i obsahuje Q_j, znázorňuje možnú komunikáciu medzi gramatikami, hovoríme o *komunikačnej štruktúre.*

Definícia 2.5. Nech Γ je *PCGSREG*. *Jazyk* generovaný systémom Γ je

$$L(\Gamma) \quad = \quad \{x \in T^* \mid (\sigma_1, \sigma_2, \ldots, \sigma_m) \Rightarrow^*_\Gamma (x, \alpha_2, \ldots, \alpha_m), \alpha_i \in V^*_\Gamma, 2 \leq i \leq m\}$$

Definícia 2.6. Nech Γ je $PCGS_mREG$, kde $m \geq 1$. Gramatiku G_i pre nejaké $1 \leq i \leq m$ systému Γ nazývame *gramatika v hre* pre danú konfiguráciu $(x_1, \ldots, x_m)$ systému Γ ak $|x_i|_N > 0$ alebo $|x_i|_K > 0$. Inak hovoríme o *gramatike mimo hry* v danej konfigurácií.

Poznámka 4. Nech Γ je *PCGSREG*. Hovoríme, že gramatika systému Γ sa *dostane mimo hry* v danom kroku odvodenia systému, $u \Rightarrow v$. Ak pre konfiguráciu u je gramatika v hre a pre konfiguráciu v je mimo hry. Gramatika systému sa *dostane do hry* v danom kroku odvodenia, $u \Rightarrow v$, ak pre konfiguráciu u je gramatika mimo hry a pre konfiguráciu v je v hre.

Definícia 2.7. *Striktne regulárnou gramatikou* nazývame regulárnu gramatiku, pre ktorú platí, že jej pravidlá majú nasledovný tvar:

$$\begin{aligned} A &\rightarrow \epsilon \\ A &\rightarrow a \\ A &\rightarrow aB \qquad \text{kde } A, B \in N \wedge a \in T \end{aligned}$$

Poznámka 5. Paralelné kooperujúce systémy gramatík so striktne regulárnymi komponentami budeme označovať $PCGSSREG$.

Definícia 2.8. Hovoríme, že Γ je $PCGS$ so stále generujúcimi regulárnymi komponentami, ak Γ je $PCGSREG$ a zároveň gramatiky v komponentách systému Γ neobsahujú ϵ-ové pravidlá, pravidlá typu $A \rightarrow \epsilon$, $A \in N$, a žiadna gramatika systému okrem hlavnej gramatiky neobsahuje pravidlá typu $A \rightarrow a, A \in N, a \in T$.

Poznámka 6. Paralelné kooperujúce systémy gramatík so stále generujúcimi regulárnymi komponentami budeme označovať $PCGSgREG$.

Poznámka 7. Pravidlá regulárnej gramatiky typu $A \rightarrow a, A \in N, a \in T^*$ budeme nazývať *deaktivujúcimi pravidlami.*

3 *PCGS* so striktne regulárnymi komponentami

Charakterizujeme odvodenie systému *PCGS* so striktne regulárnymi komponentami a dokážeme, že *PCGS* so striktne regulárnymi komponentami sú slabšie ako *PCGS* so všeobecnými regulárnymi komponentami.

Lema 3.1. *Nech* Γ *je PCGSREG a nech* $L(\Gamma) \neq \emptyset$. *Na začiatku odvodenia* Γ *sú všetky komponenty systému v hre.*

Dôkaz. Jednotlivé komponenty počiatočnej konfigurácie systému sú počiatočné neterminály zodpovedajúcich gramatík systému. Preto všetky gramatiky systému sú na začiatku odvodenia systému v hre. □

Lema 3.2. *Nech* Γ *je PCGSREG a* u, v *sú konfigurácie odvodenia systému, pričom platí* $u \Rightarrow_\Gamma v$. *Nech gramatika systému* G *je v hre pre* u*, ale mimo hry vo* v. *Ak daný krok odvodenia bol prepisovací, gramatika aplikovala deaktivujúce pravidlo. Inak gramatika* G *komunikovala s gramatikou mimo hry v* u.

Dôkaz. Majme Γ, ktoré je *PCGSREG*, a $u \Rightarrow_\Gamma v$ je krok odvodenia Γ, taký že gramatika systému G je pre u v hre a mimo hry vo v. Ak krok $u \Rightarrow v$ je prepisovací krok, zrejme gramatika G použila deaktivujúce pravidlo, keďže G je regulárna. Nech $u \Rightarrow v$ je komunikačný krok. Ak by gramatika G nekomunikovala s inou gramatikou, jej vetná forma by ostala bez zmeny, teda by bola v hre aj pre v. Taktiež, ak by v tomto kroku iná gramatika požiadala o vetnú formu gramatiku G, bola by v hre pre v. Gramatika musela v danom kroku osloviť inú gramatiku, keďže gramatiky sú regulárne, aby gramatika G bola vo v mimo hry, musela byť žiadaná gramatika mimo hry v u. □

Lema 3.3. *Nech* Γ *je PCGSREG. Počet gramatík mimo hry počas odvodenia systému* Γ *je neklesajúci.*

Dôkaz. Máme $\Gamma \in PCGSREG$. Na začiatku odvodenia sú všetky gramatiky systému v hre, vyplýva z Lemy 3.1. Gramatika systému sa môže dostať mimo hry v prepisovacom kroku použitím deaktivujúceho pravidla alebo v komunikačnom kroku odvodenia oslovením gramatiky mimo hry, vyplýva z Lemy 3.2. V prvom prípade počet gramatík mimo hry stúpa a v druhom prípade ostáva rovnaký. Gramatika oslovujúca gramatiku mimo hry je v hre a dostáva sa mimo hry a zároveň oslovená gramatika sa vracia k počiatočnému neterminálu a teda sa dostáva do hry. Komunikujúce gramatiky si vymenia postavenie. Toto je jediný spôsob ako sa môže gramatika mimo hry dostať znovu do hry. Nie je to možné počas prepisovacieho kroku, pretože vetná forma gramatiky mimo hry sa v prepisovacom kroku nemení. Dôsledkom čoho je dokazované tvrdenie. □

Lema 3.4. *Nech* Γ *je PCGSSREG. Platí, že v každom prepisovacom kroku daného systému, každá gramatika v hre v konfigurácii pred uvažovaným krokom predĺži v tomto kroku svoju vetnú formu práve o jeden terminál alebo sa dostane mimo hry.*

Dôkaz. Nech $\Gamma \in PCGSSREG$, $w \in L(\Gamma)$ a nech C_w je odvodenie slova w v Γ. Zoberme ľubovoľný prepisovací krok odvodenia C_w, $u \Rightarrow v$, taký že existuje gramatika systému G, ktorá je v hre pre u, nech x je vetná forma gramatiky G v u. $x = yA, y \in T^*, A \in N$, keďže uvažujeme prepisovací krok. Gramatika G je striktne regulárna, a preto môže použiť iba pravidlá tvaru $A \to aB$, $A \in N$, $a \in T$, $B \in N \cup K$, kedy svoju vetnú formu predĺži, alebo pravidlá tvaru $A \to a$, $A \in N$, $a \in T \cup \{\epsilon\}$, kedy sa dostáva gramatika mimo hry. Ak $a \neq \epsilon$ zároveň predĺži terminálnu časť svojej vetnej formy. □

Lema 3.5. *Nech* $\Gamma \in PCGSSREG$*, potom platí:*
$\forall w \in L(\Gamma)$*;* $w = \epsilon \vee \exists v, u$*;* $w = vu \wedge v \neq \epsilon \wedge v$ *generuje hlavná gramatika.*

Dôkaz. Z liem 3.1, 3.4 vyplýva, že v prvom kroku odvodenia (prepisovací krok) daného systému všetky gramatiky systému predĺžia svoje vetné formy o práve jeden terminál alebo sa dostávajú mimo hry. Ak sa hlavná gramatika dostane mimo hry potom $w = \epsilon \vee w = a, a \in T$, teda tvrdenie platí. Nech použila pravidlo tvaru $A \to aB, a \in T, A \in N, B \in N \cup K$. Ak ďalej v odvodení slova hlavná gramatika nie je požiadaná o vetnú formu, tvrdenie zrejme platí. Ak bola požiadaná, hlavná gramatika sa vráti k počiatočnému neterminálu, potom nastáva rovnaká situácia ako v prvom kroku odvodenia. □

Lema 3.6. *Nech* Γ *je z PCGSSREG. Pre gramatiku systému G v hre v konfigurácií z odvodenia daného systému, platí*

1. *Ak z je počiatočná konfigurácia, vetná forma gramatiky v tejto konfigurácií je jej počiatočný neterminál.*

2. *Ak pre* z *existuje odvodenie systému* $z_0 \Rightarrow^* z_1 \Rightarrow z_2 \Rightarrow^* z$*, také že* $z_1 \Rightarrow z_2$ *je krok odvodenia systému, v ktorom gramatika G v komunikácií odovzdá svoju vetnú formu a medzi konfiguráciami* z_2, z *sú iba komunikačné kroky. Potom vetná forma gramatiky G v z je jej počiatočný neterminál.*

3. *Inak vetná forma gramatiky G v z obsahuje aspoň* $k, k \geq 1$ *terminálov, kde k je počet prepisovacích krokov od počiatočnej konfigurácie, ak gramatika od začiatku odvodenia po konfiguráciu z neodovzdala svoju vetnú formu, alebo od poslednej konfigurácie pred z, v ktorej gramatika odovzdala vetnú formu.*

Dôkaz. Majme Γ, ktoré je z $PCGSSREG$. Nech C je nejaké odvodenie systému Γ. Dokážeme, že pre konfigurácie tohto odvodenia platí dokazované tvrdenie. Prvý a druhý bod tvrdenia vyplýva priamo z definície $PCGSREG$. Dokážeme tretí bod tvrdenia. Uvažujme konfiguráciu z odvodenia C, spĺňajúcu predpoklady tretieho bodu tvrdenia. Nech počet prepisovacých krokov, od začiatku odvodenia po z, ak gramatika G pred z neodovzdala svoju vetnú formu, alebo od poslednej konfigurácie pred z, nasledujúcej po kroku odvodenia, v ktorom gramatika odovzdala vetnú formu, po z, je k. Všetky terminály obsiahnuté v x vygenerované gramatikou G, vznikli v časti odvodenia od posledného odovzdania vetnej formy gramatiky G po z, alebo ak táto situácia pred z nenastala, od počiatočnej konfigurácie po z. Z toho a z Lemy 3.4, z faktu, že G je v hre v z, vyplýva, že x obsahuje aspoň k terminálov (alebo práve k, ak gramatika v uvažovanej časti odvodenia nevykomunikovala vetnú formu obsahujúcu aspoň jeden terminál, teda x obsahuje terminály generované len gramatikou G). Keďže z nie je počiatočná konfigurácia, ani konfigurácia nasledujúca po kroku, v ktorom gramatika G odovzdala vetnú formu, bez toho aby medzi konfiguráciou po tomto kroku a uvažovanou konfiguráciu nebol vykonaný aspoň jeden prepisovací krok, vyplýva, že $k \geq 1$. □

Lema 3.7. *Nech Γ je $PCGS_mSREG, m \geq 1$ a $\forall 1 \leq i \leq m, \sigma_i \to \epsilon \notin P_i$. Gramatika systému, ktorá je v nejakej konfigurácií systému mimo hry, má v nej neprázdnu vetnú formu.*

Dôkaz. Nech Γ je $PCGS_mSREG, m \geq 1$. Nech C je nejaké odvodenie systému Γ. A nech gramatika G systému je mimo hry v konfigurácií z odvodenia C. Nech konfigurácia z_2 odvodenia C je posledná konfigurácia pred z, v ktorej bola gramatika G v hre. A nech $z_2 \Rightarrow z_1$ je krok uvažovaného odvodenia. Vetná forma gramatiky G v z je rovnaká ako v z_1, priamo platí, ak $z = z_1$, inak to vyplýva z faktu, že vetná forma gramatiky mimo hry sa iba kopíruje. Pre platnosť tvrdenia stačí, keď ukážeme, že gramatika G má neprázdnu vetnú formu (označme x_1) v z_1. Z Lemy 3.6 vyplýva, že vetná forma gramatiky G v z_2, označme x_2, je rovná počiatočnému neterminálu gramatiky G alebo obsahuje aspoň jeden terminál. Ak x_2 je počiatočný neterminál gramatiky G, krok $z_2 \Rightarrow z_1$ musel byť prepisovací, aby G mohla byť mimo hry v z_1. Z Lemy 3.2, vyplýva, že G v uvažovanom kroku, použila deaktivujúce pravidlo. Keďže počiatočný neterminál nemôže ísť na prázdne slovo, x_1 je neprázdne. Ak x_2 obsahuje aspoň jeden terminál, x_1 je neprázdne. Keďže z Lemy 3.2, vyplýva, že vetná forma gramatiky, ktorá sa dostane mimo hry v nejakom kroku, ostane rovnaká alebo sa predĺži. Dokázali sme, že x_1 je neprázdne, teda aj platnosť tvrdenia. □

Lema 3.8. *Nech Γ je $PCGS_mSREG$ a $\forall i, 1 \leq i \leq m, \sigma_i \to \epsilon \notin P_i$. V každom* *komunikačnom kroku odvodenia systému* Γ *sa vykomunikuje vetná forma obsahujúca aspoň jeden terminál.*

Dôkaz. Nech Γ je systém spĺňajúci predpoklady tvrdenia a nech $u \Rightarrow v$ je komunikačný krok odvodenia. Nech v tomto kroku je uspokojený komunikačný symbol $Q_i, 1 \leq i \leq m$. Musíme ukázať, že vetná forma gramatiky G_i v konfigurácií u, označme x, obsahuje aspoň jeden terminál. Pretože x je vetná forma vykomunikovaná v uvažovanom kroku. Ak G_i je v hre v konfigurácií u, z Lemy 3.6 vyplýva, že x obsahuje aspoň jeden terminál, keďže u nie je počiatočná konfigurácia (prvý krok odvodenia je prepisovací), ani konfigurácia spĺňajúca predpoklady druhého bodu Lemy. Ak G_i je mimo hry v u z Lemy 3.7 vyplýva, že x je neprázdna a teda aj obsahujúca aspoň jeden terminál. □

Definícia 3.1. Nech Γ je $PCGS_mREG, m \geq 1$ a nech $z_1 = (x_1A_1, \ldots, x_mA_m)$, $z_2 = (y_1B_1, \ldots, y_mB_m)$, $x_i, y_i \in T^*, A_i, B_i \in T \cup K \cup \{\epsilon\}, 1 \leq i \leq m$ sú konfigurácie systému Γ. Hovoríme, že konfigurácie z_1, z_2 sú ekvivalentné (označujeme $z_1 \equiv z_2$) ak, $\forall i, 1 \leq i \leq m, A_i = B_i$.

Poznámka 8. Relácia $\equiv$ je zrejme relácia ekvivalencie. Počet tried tejto ekvivalencie, keď uvažujeme konfigurácie systému stupňa m, pričom N je množina neterminálov systému a K je množina komunikačných symbolov systému, je

$$A = (|N| + |K| + 1)^m$$

Lema 3.9. *Nech* Γ *je* $PCGS_mSREG, m \geq 1$ *a* $z = (x_1A_1, \ldots, x_mA_m)$ *je konfigurácia systému* Γ*, kde* $\forall i, 1 \leq i \leq m, x_i \in T^*, A_i \in T \cup K \cup \{\epsilon\}$*, získateľná z počiatočnej konfigurácie po* n *krokoch odvodenia systému* Γ*. Platí* $\forall i, 1 \leq i \leq m, |x_i| < 2^n$*.*

Dôkaz. Nech Γ je $PCGS_mSREG, m \geq 1$. Nech $z = (x_1A_1, \ldots, x_mA_m)$ je konfigurácia systému Γ, kde $\forall i, 1 \leq i \leq m, x_i \in T^*, A_i \in T \cup K \cup \{\epsilon\}$. Tvrdenie dokážeme matematickou indukciou na počet krokov odvodenia $(\sigma_1, \ldots, \sigma_m) \Rightarrow^* z$, označme i

1. $i = 0$

   Uvažovaná konfigurácia z je počiatočná konfigurácia. Žiadna komponenta počiatočnej konfigurácie neobsahuje terminály, preto tvrdenie platí.

2. Predpokladáme, že pre $\forall j, j < i$ tvrdenie platí. Dokážeme tvrdenie pre i.

   Keďže z nie je počiatočná konfigurácia existuje z_1, taká že $z_1 \Rightarrow z$ je krok odvodenia systému. Nech $z_1 = (y_1B_1, \ldots, y_mB_m)$, $\forall i, 1 \leq i \leq m, y_i \in T^*, B_i \in T \cup K \cup \{\epsilon\}$. Z indukčného predpokladu vyplýva, že $\forall i, 1 \leq i \leq m, |y_i| < 2^{i-1}$ Ak bol krok $z_1 \Rightarrow z$ prepisovací, gramatiky neaktívne v z_1, majú rovnaké vetné formy v konfiguráciach z_1, z a gramatiky, aktívne v z_1, majú v z vetné formy dlhšie maximálne o jeden terminál, Lema 3.4. Ak bol krok $z_1 \Rightarrow z$ komunikačný, nejaká gramatika mohla predĺžiť v tomto kroku svoju vetnú formu iba v prípade ak požiadala o vetnú formu inú gramatiku. Dĺžka vetnej formy takejto gramatiky je v z maximálne $2 \cdot 2^{i-1} - 1$ a to je menšie ako 2^i.

□

Veta 3.1. *$\mathcal{L}(PCGS_mSREG) \subsetneq \mathcal{L}(PCGS_mREG)$, pre každé pevné $m \geq 3$.*

Dôkaz. Zrejme platí $\mathcal{L}(PCGS_mSREG) \subseteq \mathcal{L}(PCGS_mREG)$, pre $m \geq 1$. Majme $m \geq 3$. Nech $L = \{{a_1}^i \dots {a_{2m-2}}^i \mid i \geq 1\}$. Ukážeme, že $L \in \mathcal{L}(PCGS_mREG)$ a $L \notin \mathcal{L}(PCGS_mSREG)$. Jazyk L patrí do $\mathcal{L}(PCGS_mREG)$, dôkaz je uvedený v [HKK94, SK92].

Dokážeme, že $L \notin \mathcal{L}(PCGS_mSREG)$, sporom. Predpokladajme, že existuje systém Γ, taký že $\Gamma = (N, K, T, G_1, \dots, G_m)$ je $PCGS_mSREG$ a $L(\Gamma) = L$. Nech p je počet tried ekvivalencie konfigurácií systému Γ, určený podľa poznámky 8. Uvažujme slovo w z jazyka L, také že $w = a_1^{2^{4p}} \dots a_{2m-2}^{2^{4p}}$ a jeho najkratšie odvodenie C_w v systéme Γ. Keďže uvažujeme najkratšie odvodenie slova w, pre každú konfiguráciu odvodenia C_w platí, že terminálna časť aspoň jedného jej kompomentu je súčasťou slova w.

Tvrdenie 3.1.1. *Nech $xA, x \in T^*, A \in N \cup K\{\epsilon\}$ je vetná forma gramatiky systému Γ v odvodení C_w, pričom x je súčasťou slova w. Potom platí, že $x = a_{j+1}^{i_1} \dots a_{j+k}^{i_k}, j + k \leq 2m - 2 \wedge j, k \geq 0$, kde $\forall l, 2 \leq l \leq k - 1, i_l = 2^{4p}, i_1 \leq 2^{4p}, i_k \leq 2^{4p}$.*

Dôkaz. Tvrdenie vyplýva z tvaru slova w a regulárnosti gramatík systému. □

Z Lemy 3.9 a z tvaru slova w vyplýva, že dĺžka odvodenia C_w je väčšia ako $4p$ a preto v prvých p krokoch odvodenia existujú dve ekvivalentné konfigurácie. Nech

$$\begin{aligned}(\sigma_1, \dots, \sigma_m) &\Rightarrow^* z_1 = (x_1A_1, \dots, x_mA_m)\\ &\Rightarrow^* z_2 = (y_1A_1, \dots, y_mA_m)\\ &\Rightarrow^* (w, \dots),\end{aligned}$$

kde $x_i, y_i \in T^*, A_i \in N \cup K \cup \{\epsilon\}$ a $\forall i, 1 \leq i \leq m, |x_i| < 2^p, |y_i| < 2^p$, pre $1 \leq i \leq m$ je odvodenie C_w.

V časti odvodenia $z_1 \Rightarrow^* z_2$ musí byť vygenerovaný aspoň jeden terminál, ktorý sa stane súčasťou slova w. Inak môžeme tento úsek odvodenia vynechať a dostaneme kratšie odvodenie slova w. Uvažujme vetné formy gramatík, pre ktoré sa príslušné y_i

stane súčasťou slova w. Zrejme musia obsahovať každý z terminálnych symbolov vyskytujúcich sa vo w. Inak vynechaním časti odvodenia za z_1 až po z_2 vrátane dostaneme odvodenie, ktoré bude generovať slovo, ktoré nebude patriť do L. Navyše, pre všetky také $y_i, 1 \leq i \leq m$ platí, že obsahujú maximálne dva rôzne terminály, pretože majú dĺžku menšiu ako 2^{4p} a teda v prípade, že by obsahovali viac rôznych terminálov by nastal spor s Tvrdením 3.1.1.

Z Dirichletovho princípu vyplýva, že v konfigurácii z_2 existuje aspoň $m-2$ komponentov obsahujúcich dva rôzne terminály, pričom terminálne časti týchto komponentov sú súčasťou slova w. Nech l-tý komponent konfigurácie z_2 patrí medzi tieto komponenty, potom $y_l = a_j^{s_l} a_{j+1}^{p_l}, 1 \leq j \leq 2m-3, s_l, p_l \geq 1, s_l + p_l < 2^p$.

Keďže odvodenie C_w je dlhšie ako $4p$, po konfigurácií z_2 sa v ňom nachádzajú ďalšie tri páry ekvivalentných konfigurácií. Nech

$$\begin{aligned}
(\sigma_1, \ldots, \sigma_m) &\Rightarrow^* z_2 = (y_1 A_1, \ldots, y_m A_m) \\
&\Rightarrow^* z_3 = (t_1 B_1, \ldots, t_m B_m) \\
&\Rightarrow^* z_4 = (u_1 B_1, \ldots, u_m B_m) \\
&\Rightarrow^* z_5 = (e_1 C_1, \ldots, e_m C_m) \\
&\Rightarrow^* z_6 = (f_1 C_1, \ldots, f_m C_m) \\
&\Rightarrow^* z_7 = (g_1 D_1, \ldots, g_m D_m) \\
&\Rightarrow^* z_8 = (h_1 D_1, \ldots, h_m D_m) \\
&\Rightarrow^* (w, \ldots),
\end{aligned}$$

kde $t_i, u_i, e_i, f_i, g_i, h_i \in T^*, B_i, C_i, D_i \in N \cup K \cup \{\epsilon\}$ a $\forall i, 1 \leq i \leq m, |t_i| < 2^{2p}, |u_i| < 2^{2p}, |e_i| < 2^{3p}, |f_i| < 2^{3p}, |g_i| < 2^{4p}, |h_i| < 2^{4p}$, pre $1 \leq i \leq m$ je odvodenie C_w.

Z faktu, že všetky terminály sú obsiahnuté v komponentoch konfigurácie z_2, ktoré sú súčasťou slova w a z analogických dôvodov ako sú vyššie uvedené vyplýva, že konfigurácie $z_3, z_4, z_5, z_6, z_7, z_8$ majú rovnaké vlastnosti ako konfigurácia z_2.

Zrejme medzi konfiguráciami z_3, z_4 systém musel vygenerovať, v rámci vetných foriem, ktoré sú súčasťou výsledného slova, všetky terminály rovnaký počet krát, k krát. Keby to tak nebolo a my by sme z odvodenia C_w vyrobili iné, vynechaním časti odvodenia medzi konfiguráciami z_3, z_4, systém by generoval terminálne slovo rôzne od w nepatriace do jazyka L. Keďže C_w je najkratšie odvodenie slova w, medzi konfiguráciami z_3, z_4 sa musel vygenerovať nejaký terminál, ktorý je súčasťou slova w, teda platí, že $k \geq 1$. Analogicky, to isté platí pre konfigurácie z_5, z_6 a konfigurácie z_7, z_8.

V prepisovacích krokoch odvodenia C_w medzi konfiguráciami z_3, z_4, konfiguráciami z_5, z_6 a medzi konfiguráciami z_7, z_8 môžme v gramatike, ktorej vetná forma obsahuje dva rôzne terminály, generovať iba terminál, ktorý je posledný v jej vetnej forme, čo vyplýva z regularity gramatík systému a Tvrdenia 3.1.1.

Tvrdenie 3.1.2. *V konfigurácii z_4 existuje najviac jeden komponent, ktorý neobsahuje dva rôzne terminály.*

Dôkaz. V konfigurácii z_3 sú najviac dva komponenty, ktoré neobsahujú dva rôzne terminály. Ak konfigurácia z_3 obsahuje najviac jeden takýto komponent, potom aj konfigurácia z_4 obsahuje najviac jeden komponent, ktorý neobsahuje dva rôzne terminály. Teda platí dokazované tvrdenie.

Platí, že ak v konfigurácií z_3 existujú dva komponenty s vlastnosťou, že neobsahujú dva rôzne terminály, potom ani jeden z nich neobsahuje terminál, ktorý sa vyskytuje v inom komponente konfigurácie z_3 a zároveň obsahujú aspoň jeden terminál, vyplýva z Dirichletovho princípu a z vlastností konfigurácie z_3. Preto komunikácie realizovanej v komunikačnom kroku v časti odvodenia C_w medzi konfiguráciami z_3, z_4 sa môžu zúčastniť len gramatiky, ktorých vetné formy neobsahujú dva rôzne terminály, aby nenastal spor s Tvrdením 3.1.1. Z uvedeného vyplýva, že ak v časti odvodenia C_w medzi konfiguráciami z_3, z_4 systém vykoná komunikačný krok, v konfigurácii z_4 existuje najviac jeden komponent, ktorý neobsahuje dva rôzne terminály.

Uvedomme si, že pred terminálom a_1, vrámci vetných foriem, ktorých terminálne časti sú súčasťou slova w, sa nemôže vyskytovať iný terminál. Vyplýva to priamo z tvaru slova w. Preto jedna gramatika systému musí byť medzi konfiguráciami z_3, z_4, konfiguráciami z_5, z_6 a medzi konfiguráciami z_7, z_8 schopná generovať vetnú formu, ktorej terminálna časť neobsahuje iný terminál ako terminál a_1. Ak ani jeden z komponentov konfigurácie z_3, neobsahujúcich dva rôzne terminály, neobsahuje terminál a_1, aby mohol systém vygenerovať medzi konfiguráciami z_3, z_4 vetnú formu neobsahujúcu iný terminál ako a_1, musí vrátiť nejakú gramatiku k jej počiatočnému neterminálu, teda systém musí vykonať komunikačný krok. Dôsledkom čoho je dokazované tvrdenie, vyplýva z vyššie uvedeného.

Nech jeden z komponentov konfigurácie z_3, neobsahujúcich dva rôzne terminály, obsahuje terminál a_1, možu nastať dva prípady, systém v časti odvodenia C_w medzi konfiguráciami z_3, z_4, vykoná komunikačný krok alebo nie. Ak systém vykoná v uvažovanej časti odvodenia komunikačný krok, platí dokazované tvrdenie. Predpokladajme, že systém v uvažovanej časti odvodenia nevykoná komunikačný krok. Terminály, pre ktoré neexistuje $i, 1 \leq i \leq m$, také že sú posledným písmenom t_i a zároveň t_i obsahuje dva rôzne terminály, musia medzi konfiguráciami z_3, z_4 generovať gramatiky, ktorých vetné formy v z_3 neobsahujú dva rôzne terminály, aby nenastal spor s Tvrdením 3.1.1. Označme množinu týchto terminálov X_{z_3}. Keďže uvažujeme $m \geq 3$, platí, že $|X_{z_3}| \geq 1$. Z toho a z faktu, že v časti odvodenia C_w medzi konfiguráciami z_3, z_4 systém nevykoná komunikačný krok, vetná forma v konfigurácii z_4 jednej z gramatík, ktorých vetné formy v konfigurácii z_3 neobsahujú dva rôzne terminály, musí obsahovať dva rôzne terminály. Teda platí dokazované tvrdenie. □

Dôsledkom tvrdení 3.1.1, 3.1.2 je, že aj v konfigurácii z_5 existuje najviac jediný komponent, ktorý neobsahuje dva rôzne terminály. Ak v konfigurácii z_5 neexistuje takýto komponent, medzi konfiguráciami z_5, z_6 systém nemôže vykonať komunikačný krok, pretože inak by nastal spor s Tvrdením 3.1.1. Ale potom nemôže medzi týmito konfiguráciami vygenerovať všetky terminály rovnaký nenulový počet krát, čo je spor s vlastnosťami konfigurácií z_5, z_6, ktoré sme ukázali vyššie. Predpokladajme, že v konfigurácii z_5 existuje komponent, ktorý neobsahuje dva rôzne terminály. Nech $e(l)$ neobsahuje dva rôzne terminály, pre nejaké $l, 1 \leq l \leq m$. Gramatiku systému, ktorá prislúcha tomuto komponentu, označíme G. Terminály, pre ktoré neexistuje $i, 1 \leq i \leq m$, také že sú posledným písmenom e_i a zároveň e_i obsahuje dva rôzne terminály (tj. $i \neq l$), musí medzi konfiguráciami z_5, z_6 generovať gramatika G, aby nenastal spor s Tvrdením 3.1.1. Označme množinu týchto terminálov X_{z_5}.

Tvrdenie 3.1.3. *Nech* $Y_{z_5} = \{a \in T \mid \exists i, 1 \leq i \leq m \wedge i \neq l, e(i)$ *obsahuje terminál* $a\}$. *Platí* $Y_{z_5} = T$.

Dôkaz. Sporom, predpokladajme, že $Y_{z_5} \subsetneq T$. Keďže komponenty konfigurácie z_5 musia obsahovať všetky terminály platí, že $\forall a \in T \wedge a \notin Y_{z_5}$ obsahuje $e(l)$, teda $e(l)$ nie je prázdne slovo. Predpokladáme, že $e(l)$ neobsahuje dva rôzne terminály, teda platí $|Y_{z_5}| \geq |T| - 1$. Medzi konfiguráciami z_5, z_6 systém nemôže vykonať komunikačný krok, aby nenastal spor s Tvrdením 3.1.1. Platí $|Y_{z_5}| = 2m - 3$. A z terminálov množiny Y_{z_5} môže najviac $m - 1$ terminálov nepatriť do množiny X_{z_5}. Z uvedeného vyplýva, že $|X_{z_5}| \geq m - 2$. Keďže $m \geq 3$, platí že $|X_{z_5}| \geq 1$. Terminály z množiny X_{z_5} môže generovať iba gramatika G, keďže musí vygenerovať aspoň jeden terminál, jej vetná forma je v konfigurácii z_5 neprázdna, neobsahujúca terminály z množiny X_{z_5}, pretože $X_{z_5} \subseteq Y_{z_5}$, a v odvodení C_w medzi konfiguráciami z_5, z_6 systém nevykoná komunikačný krok, platí že vetná forma gramatiky G v konfigurácii z_6 obsahuje dva rôzne terminály. Všetky komponenty konfigurácie z_6 obsahujú dva rôzne terminály. Teda aj všetky komponenty konfigurácie z_7 obsahujú dva rôzne terminály. Medzi konfiguráciami z_7, z_8 systém nemôže vykonať komunikačný krok, pretože inak by nastal spor s Tvrdením 3.1.1. Ale potom nemôže medzi týmito konfiguráciami vygenerovať všetky terminály rovnaký nenulový počet krát, čo je spor s vlastnosťami konfigurácií z_7, z_8. □

Z Tvrdenia 3.1.3 a z Dirichletovho princípu vyplýva, že platí $|X_{z_5}| = m - 1$. Keďže uvažujeme $m \geq 3$, platí $|X_{z_5}| \geq 2$. Medzi konfiguráciami z_5, z_6 systém musel vygenerovať, v rámci vetných foriem, ktoré sú súčasťou výsledného slova, všetky terminály rovnaký nenulový počet krát, nech tento počet je rovný s.

Tvrdenie 3.1.4. *V časti odvodenia C_w medzi konfiguráciami z_5, z_6 nemôže viac gramatík systému naraz požiadať tú istú gramatiku o jej vetnú formu.*

Dôkaz. Ak by toto tvrdenie neplatilo nastal by spor s Tvrdením 3.1.1. □

Počet prepisovacých krokov v časti odvodenia C_w medzi konfiguráciami z_5, z_6 musí byť aspoň $2s$. Vyplýva z Lemy 3.4, Tvrdenia 3.1.4 a faktu, že v každom prepisovacom kroku uvažovanej časti odvodenia môžu byť terminály z množiny X_{z_5} generované najviac jednou gramatikou, čo je dôsledkom vlastností konfigurácie z_5 a Tvrdenia 3.1.1.

Ak by v uvažovanej časti odvodenia C_w existovala konfigurácia, v ktorej všetky gramatiky, ktorých vetné formy v danej konfigurácii obsahujú dva rôzne terminály, sú v tejto konfigurácii mimo hry, potom by platilo, že v konfigurácii z_6 je aspoň $m - 1$ gramatík mimo hry, vyplýva z Lemy 3.3. Potom už v konfigurácií z_5 muselo $m-1$ gramatík byť mimo hry, keďže konfigurácie z_5, z_6 sú ekvivalentné. Aby systém v časti odvodenia medzi konfiguráciami z_5, z_6 mohol generovať všetky terminály rovnaký nenulový počet krát, musela byť gramatika G v konfigurácií z_5 v hre. Potom všetky komponenty konfigurácie z_6 obsahujú dva rôzne terminály. Teda aj všetky komponenty konfigurácie z_7 obsahujú dva rôzne terminály. Medzi konfiguráciami z_7, z_8 systém nemôže vykonať komunikačný krok, pretože inak by nastal spor s Tvrdením 3.1.1. Ale potom nemôže medzi týmito konfiguráciami vygenerovať všetky terminály rovnaký nenulový počet krát, čo je spor s vlastnosťami konfigurácií z_7, z_8. Preto v časti odvodenia C_w medzi konfiguráciami z_5, z_6, nemôže existovať konfigurácia, v ktorej všetky gramatiky, ktorých vetné formy v danej konfigurácii obsahujú dva rôzne terminály, sú v tejto konfigurácii mimo hry.

Nech množina Z pre nejakú konfiguráciu z (označujeme Z_z) je množina terminálov, ktoré sú poslednými terminálmi vo vetných formách gramatík v hre pre konfiguráciu z, pričom tieto vetné formy obsahujú dva rôzne terminály. Ukážeme, že prienik množín Z pre jednotlivé kofigurácie časti odvodenia C_w, medzi konfiguráciami z_5, z_6 je neprázdny. Nech $u \Rightarrow v$ je krok časti odvodenia C_w medzi konfiguráciami z_5, z_6. Ak to je prepisovací krok platí $Z_v \subseteq Z_u$, vyplýva z Lemy 3.4 a Tvrdenia 3.1.1. Ak krok $u \Rightarrow v$ je komunikačný krok platí $Z_v = Z_u$, čo vyplýva z regulárnosti gramatík a Tvrdenia 3.1.1. Keďže sme vyššie dokázali, že všetky množiny Z pre konfigurácie, v časti odvodenia C_w

medzi konfiguráciami z_5, z_6, sú neprázdne a z vývinu množín Z vzhľadom na odvodenie systému, platí, že ich prienik je neprázdny.

Teda existuje terminál, ktorý v každej konfigurácii pred nejakým prepisovacím krokom v časti odvodenia C_w medzi konfiguráciami z_5, z_6, je posledný terminál nejakého komponentu danej konfigurácie, pričom tento komponent obsahuje dva rôzne terminály a gramatika prislúchajúca tomuto komponentu je v danej konfigurácii v hre. Potom systém generoval tento terminál v každom prepisovacom kroku časti odvodenia C_w medzi konfiguráciami z_5, z_6. Pretože v uvažovanej časti odvodenia C_w gramatiky, v hre v konfigurácii pred nejakým prepisovacím krokom, ktorých vetné formy v tejto konfigurácii obsahujú dva rôzne terminály v tomto kroku môžu generovať len terminál, ktorý je posledný v ich vetnej forme v konfiguráncii pred týmto predchodovým krokom.

Z toho vyplýva, že nejaký terminál systém v časti odvodenia C_w medzi konfiguráciami z_5, z_6 vygeneroval aspoň $2s$ krát. Čo je spor s faktom, že v uvažovanej časti odvodenia C_w systém generuje všetky terminály rovnaký počet krát, keďže $s \geq 1$. Z toho vyplýva, že náš predpoklad, že jazyk L patrí do triedy $\mathcal{L}(PCGS_mSREG)$ je nesprávny. Preto dokazované tvrdenie platí. $\square$

4 Simulácia $PCGS$ Turingovym strojom

V prípade potreby simulácie $PCGSREG$ Turingovym strojom, sa nám ponúka intuitívna konštrukcia, kedy m-páskovým Turingovym strojom simulujeme $PCGS_mREG$ nasledovne. Každá páska zodpovedá jednej gramatike a jej vetnej forme. Neterminály a komunikačné symboly sa pamätajú v stave. V prípade komunikácie sa obsah pásky požiadanej gramatiky prekopíruje na koniec pások simulujúcich gramatiky, ktoré o danú vetnú formu požiadali. V nasledovnom stave sa požiadaná gramatika vráti k počiatočnému neterminálu a príslušná páska sa označí ako prázdna. Táto konštrukcia pre $PCGS$ s acyklickou komunikačnou štruktúrou je bližšie popísana v [HKK94]. V leme 4.1 ukážeme, že ak chceme simulovať $PCGS_mREG$, m-páskovým Turingovym strojom v lineárnom čase, bez ohľadu na komunikačnú štruktúru simulovaného systému, táto konštrukcia nie je vhodná. Ukazuje sa, že pri simulácii kopírovanie vetných foriem v cykloch komunikačnej štruktúry vyžaduje nelineárny čas.

Lema 4.1. *Nech* Γ *je* $PCGS_mREG, m \geq 1$ *a nech m-páskový Turingov stroj A je zostrojený vyššie popísanou konštrukciou k systému* Γ*. Potom* $TIME(A, n) = O(n^2)$*.*

Dôkaz. Majme nasledovné $\Gamma \in PCGS_2SREG$

$$\begin{aligned}
\Gamma &= (N, K, T, G_1, G_2) \\
N &= \{\sigma_1, \sigma_2, A, B\} \\
K &= \{Q_1, Q_2\} \\
T &= \{a\} \\
P_1 &= \{\sigma_1 \to aQ_2, B \to aA, B \to a\} \\
P_2 &= \{\sigma_2 \to aQ_1, \sigma_2 \to aB, A \to aB\}
\end{aligned}$$

$$L(\Gamma) = \{a^{4n-1}, n \geq 1\}$$

Nech $w \in L(\Gamma)$ a C_w je odvodenie slova w v systéme Γ, $|w| = n$. A nech Turingov stroj A je zostrojený podľa vyššie uvedenej konštukcie ku systému Γ.

Tvrdenie 4.0.5. *V i-tom komunikačnom kroku, Turingov stroj A kopíruje $2i - 1$ symbolov.*

Dôkaz. Matematickou indukciou na i.

1. $i = 1$

   Z definície Γ vyplýva, že odvodenie slova, vyzerá nasledovne:
   $(\sigma_1, \sigma_2) \Rightarrow (aQ_2, aB) \Rightarrow \ldots$
   V prvom komunikačnom kroku stroj A kopíruje práve jedno a. Zároveň platí, $2 \cdot 1 - 1 = 1$.

2. Predpokladáme, že dokazované tvrdenie platí pre $\forall j < i$. Ukážeme, že platí aj pre i.

   Podľa definície systému Γ, konfigurácia pred $i-1$ komunikačným krokom má tvar (aQ_2, xB) alebo (xA, aQ_1), kde $x \in a^+$. Uvažujme ako pokračuje odvodenie z tejto konfigurácie až do kofigurácie pred i-tým komunikačným krokom.
   $(aQ_2, xB) \Rightarrow (axB, \sigma_2) \Rightarrow (axaA, aQ_1) \Rightarrow \ldots$
   $(xA, aQ_1) \Rightarrow (\sigma_1, axA) \Rightarrow (aQ_2, axaB) \Rightarrow \ldots$

   V $i-1$ komunikačnom kroku sa kopíruje x, z indukčného predpokladu vieme, že $|x| = 2(i-1) - 1 = 2i - 3$. V konfigurácií pred i-tým komunikačným krokom, má kopírovaná časť tvar axa, vyplýva z tvaru možných odvodení slova. A platí, $2i - 3 + 2 = 2i - 1$.

□

Keďže systém Γ v odvodení slova dĺžky n vykoná $\frac{n-1}{2}$ komunikačných krokov, Turingov stroj A musí pri simulácií odvodenia systému Γ urobiť následovný počet kopírovacích krokov.

$$\sum_{i=1}^{\frac{n-1}{2}} 2i - 1 = 2\sum_{i=1}^{\frac{n-1}{2}} i - \frac{n-1}{2} = 2\frac{\frac{n-1}{2}(\frac{n-1}{2}+1)}{2} - \frac{n-1}{2} = \frac{(n-1)^2}{4} + $$
$$+\frac{n-1}{2} - \frac{n-1}{2} = \frac{1}{4}(n-1)^2 \in O(n^2)$$

Z čoho vyplýva, že $TIME(A, n) = O(n^2)$ □

Dokážeme, že vieme simulovať $PCGS$ stupňa m, so stále generujúcimi striktne regulárnymi komponentami, $2m$-páskovým Turingovym strojom v lineárnom čase, bez ohľadu na komunikačnú štruktúru systému.

Veta 4.1. *$\mathcal{L}(PCGSgSREG) \subseteq NTIME(n)$.*

Dôkaz. Nech $\Gamma = (N, K, T, G_1, \ldots, G_m)$ je $PCGS_m gSREG$ pre nejaké pevné $m \geq 1$, pričom pracuje v čase $O(n)$, kde n je dĺžka slova generovaného systémom. K systému Γ zostrojíme $2m$–páskový Turingov stroj A pracujúci v rovnakom čase.

Idea práce automatu A spočíva v simulácií práce gramatík systému Γ na m-páskach automatu. Každá páska zodpovedá práve jednej gramatike systému. Prepisovacie kroky systému sa zaznamenávajú na tieto pásky. Hlavná otázka je ako sa bude automat správať v komunikačných krokoch. Ak by sme vetné formy gramatík systému kopírovali z pásky na pásku, musíme zabezpečit, aby sa réžia vynaložená na toto kopírovanie v komunikačných cykloch dala vykonať v lineárnom čase. My zvolíme iný prístup, aby sme sa vyhli tomuto problému, nebudeme pri samotnej simulácií systému vôbec kopírovať vetné formy z pásky na pásku. Iba zaznamenáme informácie potrebné na neskoršie zloženie generovaného slova. A to konkrétne, kam mala byť nejaká vetná forma presunutá, a hranice vetných foriem uložených na páskach automatu. Tým sme transformovali jadro problému simulácie systému, na zloženie slova generovaného systémom z informácií uložených na páskach automatu.

Dôležité je uvedomiť si, že nám táto transformácia naozaj pomohla. V skladaní výsledneho slova sa kopírovaniu vetných foriem nevyhneme, ale vyhneme sa opakovanému kopírovaniu vetných foriem v komunikačných cykloch. Pretože pre každú vetnú formu generovanú nejakou gramatikou systému vieme určiť jej konečné umiestnenie, čo vyplýva z faktu, že simulácia systému už bola ukončená.

Po zložení slova generovaného simulovaným systémom stačí už len toto slovo porovnať so vstupným slovom. V prípade rovnosti slov automat akceptuje, inak zastaví a neakceptuje.

Vidíme, že výpočet konštruovaného Turingovho stroja môžeme rozdeliť do troch fáz. V prvej simuluje systém Γ, v druhej zostrojí z informacií získaných počas simulácie systému Γ výsledné slovo a v poslednej ho porovná so vstupným slovom.

Na prvý pohľad vidno, že prvá a tretia fáza automatu nebudú časovo náročne a sú realizovateľné v lineárnom čase. Teraz bližšie popíšeme prácu automatu v jeho druhej fáze.

Práca automatu v druhej fáze sa skladá z m krokov a využívame v nej ďalších m-pások. V jednom kroku prekopírujeme všetky vetné formy jednej z gramatík na umiestenie, ktoré im náleží, podľa aktuálnych informácií na páskach automatu. Potom musíme aktualizovať informácie o cieľovom umiestnení vetných foriem, ktoré mali byť vložené do vetných foriem, ktoré sme premiestňovali. To je prvá fáza jedného kroku. V druhej fáze kroku automat spracuje vetné formy, ktoré podľa aktualizovaných informácií majú cieľové umiestnenie vo vetných formách gramatiky, ktorá ich generovala. Uvedomme si, že priamo po ukončení simulácie systému sa žiadna takáto vetná forma na páskach automatu nenachádza, vyplýva to priamo z definície $PCGSREG$ a prvej fázy automatu. Preto v prvej fáze prvého kroku druhej fázy práce automatu, automat nepremiestňuje takéto vetné formy. A keďže v druhej fáze nejakého kroku sú spracovávané, v prvej fáze nasledujúceho kroku nie sú spracovávané. A teda takéto vetné formy nie sú spracovávané v žiadnej z prvých fáz vykonávanych krokov.

Práve spracovávanie vetných foriem, ktoré majú cieľové umiestnenie vo vetných formách gramatiky, ktorá ich generovala, je klúčové. Takéto vetné formy vznikajú v dôsledku komunikačných cyklov. Nejaká gramatika vygeneruje vetnú formu x, ktorá sa v dôsledku komunikačného cyklu má vložiť do vetnej formy y, ktorú tiež generovala uvažovaná gramatika. Pre vetnú formu y môže nastať podobná situácia, ak nastane ďalšie zopakovanie cyklu. Namiesto postupného premiestňovania vetných foriem, spracujeme všetky takéto formy, pre ktoré nastala takáto situácia v dôsledku nejakého komunikačného cyklu, naraz. Zjednodušená idea je nasledovná, začneme od vonkajšej vetnej formy prekopírujeme pravú stranu formy až po miesto, kde má byť vložená vetná forma generovaná tou istou gramatikou ako spracovávana vetná forma, prekopírujeme jej pravú stranu a takto pokračujeme, až pokým nedospejeme k vetnej forme, do ktorej už nemá byť vložená žiadna vetná forma, túto vetnú formu prekopírujeme celú. Potom začneme kopírovať ľavé časti vetných foriem od najvnútornejšej až po vetnú formu, ktorou sme spracovávanie začali. Vetné formy ležia na jednej páske, pretože ich generovala jedna a tá istá gramatika systému. V poradí od najvnútornejšej po vonkajšiu vetnú formu. Pretože vetná forma gramatiky, do ktorej má byť vložená iná vetná forma, generovaná tou istou gramatikou, musí byť generovaná neskôr ako do nej vkladaná vetná forma. A preto aj na páske zodpovedajúcej danej gramatike leží až po vetnej forme, ktoré má byť do nej vložená. Z toho vyplýva, že spracovanie týchto vetných foriem môže byť realizovateľné na konečný počet čítaní pásky, na ktorej ležia. Nezáleží v ktorom kroku spracovávame vetné formy ktorej gramatiky. My sme zvolili poradie zostupne od gramatiky s najväčším indexom až po hlavnú gramatiku, takéto poradie vyžaduje minimálnu réžiu.

V nasledujúcom texte popíšeme technické detaily konštruovaného automatu. Potom uvedieme príklady, na ktorých ukážeme prácu automatu A. V δ-funkcii A zohľadňuje pravidlá všetkých gramatík a v stavoch si pamätá aktuálne neterminály, komunikačné symboly alebo symbol M pre všetky gramatiky. Predpokladáme, že $M \notin T \cup N \cup K$. Symbol M značí, že daná gramatika je mimo hry. Na páskach $T_1, \ldots, T_m$ sú vetné formy generované jednotlivými gramatikami $G_1, \ldots, G_m$ a pomocné symboly $\{\beta, \Theta_i, \Theta_i^|, \Theta_i^*, [,]_i,]_i^|,]_i^*, a^* \mid 1 \leq i \leq m, a \in T\}$, pričom predpokladáme, že pomocné symboly sú disjunktné so symbolmi, ktoré používa Γ.

Na začiatku prvej fázy si A pamätá pre každú gramatiku systému jej počiatočný neterminál a na každú pásku $T_1, \ldots, T_m$ zapíše symbol [(*začiatok masky*). Jeden krok simulácie pozostáva z jedného kroku odvodenia každej z gramatík. Automat A pre každú gramatiku uhádne, aké pravidlo použila. Nech gramatika $G_i, 1 \leq i \leq m$ použila pravidlo $A \to aB, A \in N, B \in N \cup K \cup \{\epsilon\}, a \in T$. Ak si automat v stave pre gramatiku G_i nepamätá neterminál A, výpočet sa zastaví a automat neakceptuje, pretože buď automat zle uhádol pravidlo, ktoré použila gramatika, alebo sa zastavilo odvodenie systému Γ. Inak v nasledujúcom stave si pre G_i zapamätá B, ak $B \neq \epsilon$, inak si zapamätá symbol M. Na koniec pásky T_i zapíše symbol a. Takto A pokračuje pokým hlavná gramatika nevygeneruje terminálnu vetnú formu, alebo nejaká gramatika systému nevygeneruje komunikačný symbol (tj. existuje gramatika, pre ktorú je v stave zapamätaný komunikačný symbol). Keď hlavná gramatika vygeneruje terminálnu vetnú formu, na koniec pásky T_1 sa zapíše symbol $]_0$ a automat A prejde do svojej druhej fázy.

Nech gramatika systému $G_i, 1 \leq i \leq m$ vygeneruje komunikačný symbol $Q_j, 1 \leq j \leq m$. Ak $i = j$ automat zastaví a neakceptuje, pretože sa zastavilo odvodenie systému Γ. Nech $i \neq j$, ak symbol pamätaný v stave pre gramatiku G_j je komunikačný symbol, automat zastaví a neakceptuje. Pretože v prípade, že viac gramatík naraz vygeneruje komunikačné symboly, automat sa nedeterministicky rozhodne, ktorej požiadavku gramatiky, ktorá požiadala gramatiku, ktorá nevygenerovala komunikačný symbol, začne spracovávať. Inak sa symbol pamätaný v stave pre gramatiku G_j v nasledujúcom stave zapamätá pre gramatiku G_i, na koniec pásky T_i sa zapíše symbol Θ_j a na koniec pásky T_j zapíše symbol $]_i$ (*koniec masky*). Ak neexistuje gramatika G_k, pre ktorú sa v stave pamätá symbol Q_j, na koniec pásky T_j zapíšeme symbol pre začiatok pásky. Naznačuje návrat gramatiky k počiatočnému neterminálu. Pre gramatiku G_j si v nasledujúcom stave pamätáme jej počiatočný neterminál. Pre ostatné gramatiky si v nasledujúcom stave pamätá to isté, čo v aktuálnom stave. Symbol Θ_j znamená, že na miesto neho mala byť na páske prekopírovaná vetná forma gramatiky G_j. Index konca masky, hovoríme tiež index masky, určuje, na ktorú pásku mala byť vetná forma, ktorú zastupuje daná maska, prekopírovaná. Podslovo w ľubovoľnej pásky stroja A, také že $w = [v]_i, 0 \leq i \leq m, v \in (T \cup \{\Theta_1, \ldots, \Theta_m,]_1, \ldots,]_m\})^*$, nazývame *maska*. Ak viac gramatík vygeneruje komunikačné symboly naraz, stroj A spracuje požiadavky na komunikáciu, v poradí ako systém Γ. Ak nastane zacyklenie v Γ, automat A zastaví a neakceptuje.

Celý proces zefektívnime, keď pre každú novú vetnú formu generovanú nejakou gramatikou automat uhádne, či daná vetná forma bude súčasťou výsledného slova alebo nie. Ak áno, zaznamená generovanú vetnú formu podľa vyššie uvedeného. Ak nie, na zodpovedajúcu pásku zapíše pomocný symbol β a ďalej sa správa nezmenene s výnimkou toho, že na danú pásku nezapisuje žiadne symboly. Ak nastane komunikácia medzi dvoma gramatikami systému, najskôr spraví kontrolu rozhodnutí, či vetné formy zúčastnených gramatík sú súčasťou výsledného slova. Ak posledné znaky na páskach zodpovedajúcich daným gramatikám sú oba symboly β alebo oba rôzne od β, výpočet pokračuje ako predtým. Ak jeden zo znakov je symbol β a druhý nie, automat zastaví a neakceptuje, pretože sa rozhodol nesprávne. Keď hlavná gramatika vygeneruje terminálnu vetnú formu, automat kontroluje, či sa pre danú formu rozhodol, že je súčasťou výsledného slova, ak nie, zastaví a neakceptuje. Na konci pások $T_2, \ldots, T_m$ sa musí nachádzať symbol β, keďže uvažujeme systém so stále generujúcimi striktne regulárnymi komponentami. Ak existuje nejaká páska okrem pásky T_1, na ktorej sa na konci nenachádza symbol β, automat zastaví a neakceptuje. Pri kontrole automat symboly β vymaže. Rozhodovanie nastáva na začiatku simulácie pre všetky gramatiky a pre každú gramatiku po návrate k počiatočnému neterminálu (tj. po odovzdaní vetnej formy v nejakej komunikácií). Ak pred návratom gramatika generovala vetnú formu, ktorá nie je súčasťou výsledného slova, po návrate automat zmaže symbol β na konci pásky zodpovedajúcej danej gramatike. Táto myšlienka je prevzatá z [HKK94].

V druhej fáze práce automatu, ako bolo už vyššie spomenuté, automat nahrádza výskyty Θ-symbolov maskami, ktoré patria na ich miesto a spája masky. Nahrádzanie Θ-symbolov prebieha v m krokoch, v i-tom kroku nahradíme všetky výskyty symbolu Θ_{m+1-i}, teda začíname výskytmi Θ-symbolov s najväčším indexom (tj. spracovávame požiadavky na komunikáciu s gramatikou G_m).

Uvažujme $m+1-j$ krok nahrádzania, nahrádzame symboly Θ_j. Jeden krok náhradzania prebieha v dvoch fázach, v prvej nahradíme výskyty symbolu Θ_j na páskach $T_1, \ldots, T_{j-1}$, $T_{j+1}, \ldots, T_m$ a v druhej výskyty symbolu Θ_l na páske T_l, pre všetky $1 \leq l \leq j-1$.

V prvej fáze uvažovaného kroku nahrádzania je práca automatu nasledovná. Pásku T_j automat číta od konca, spracuje masku, ktorej posledný znak číta. Postup spracovávania masky s indexom l, ktorá leží na páske T_j je nasledovný. Posledný znak spracovávanej masky označme z. Automat začne čítať pásku T_l od konca smerom doľava, nech hlava danej pásky číta znak y.

- $y \in T \cup \{[\ ,\ ]_1, \ldots,]_m, \Theta_1, \ldots, \Theta_{j-1}, \Theta_{j+1} \ldots, \Theta_m\}$

  Znak y sa zapíše na pásku P_l a na páske T_l ho označí za zmazaný a pokračuje v čítaní smerom doľava.

- $y = \Theta_j$

  Automat označí znak y na páske T_l ako zmazaný a posunie hlavu pásky doľava.

Ak znak vľavo od znaku z na páske T_j nie je koniec masky, automat pracuje nasledovne. Označí ako zmazaný znak z na páske T_j a prečíta symbol vľavo. Následne automat číta znaky na páske T_j, kopíruje ich na pásku P_l a označuje ich za zmazané na páske T_j. Ak číta symbol $\Theta_s, 1 \leq s \leq m$, navyše na pásku P_j zapíše symbol Z_l^s. Takto pokračuje pokým neprečíta začiatok masky, ten neprekopíruje len označí za zmazaný a posunie hlavu doľava ak sa dá. Vtedy začne spracovávať ďalšiu masku. V prípade že znak vľavo od znaku z je koniec masky, nastala situácia, kedy viac gramatík naraz požiadalo o komunikáciu s gramatikou G_j. Preto nebudeme hneď označovať znaky tvoriace spracovávanu masku za zmazané, ale označíme za zmazaný iba znak z. Presnejšie, automat označí za zmazaný znak z a na páske T_j a číta pásku T_j smerom doľava a kopíruje znaky na pásku P_l, ak číta symbol $\Theta_s, 1 \leq s \leq m$, navyše zapíše na pásku P_j symbol Z_l^s, ale iba v prípade, že pre čítaný znak ešte nebola zapamätaná daná informácia. Takto pokračuje pokým neprečíta začiatok masky, potom vráti hlavu pásky T_j na posledný neoznačený znak ako zmazaný vpravo, čo je koniec masky, ktorý sa nachádza pred znakom z. Automat začne spracovávať túto masku.

Automat tento postup opakuje, pokým neprečíta celú pásku T_j. Potom automat každú z pásiek $T_1, \ldots, T_m$ prečíta až po jej začiatok z aktuálnej pozície hlavy pásky, pričom kopíruje čítané znaky na zodpovedajúcu pomocnú pásku, zároveň ich na čítanej páske označuje za zmazané.

Nakoniec musí automat zaznamenať zmenu informácie, na ktorú pásku majú byť prekopírované masky, ktoré mali byť vložené do masiek ležiacich na páske T_j. Keďže masky ležiace na páske T_j boli vložené do iných nasiek. Automat využije informácie zapamätané na páske P_j. Od konca pásky, k-tý výskyt symbolu Z_l^s, kde s je pevné, znamená, že index k-tej masky s indexom j na páske T_s má byť zmenený na l. Pretože tento symbol vznikol pri nahrádzaní Θ-symbolu na páske T_l maskou, ktorá obsahovala výskyt Θ-symbolu s indexom s, pričom tento výskyt bol k-tý na páske T_j odzadu. Hlavy pásiek $P_1, \ldots, P_m$ sú na ich konci. Automat číta pásku P_j zprava doľava až pokým neprečíta celú pásku. Nech aktuálne číta znak $Z_l^s, 1 \leq s, l \leq m$. Začne čítať pásku P_s, kopíruje čítané znaky na pásku T_s a označuje ich za zmazané na páske P_s, pokým neprečíta symbol $]_j$. Označí ho za zmazaný a posunie hlavu pásky P_s vľavo. Na pásku T_s zapíše symbol $]_l$. Tým spracoval aktuálne čítaný symbol na páske P_j, ten označí za zmazaný a hlavu posunie doľava, pokračuje v spracovávaní. Po prečítaní celej pásky P_j, prekopíruje zostávajúci obsah pomocných pásiek, späť na pásky $T_1, \ldots, T_m$. Pomocné pásky sú vyprázdnené. Kedže sme kopírovali na pomocné pásky masky odzadu a z pomocných pásiek kopírujeme masky znovu odzadu, budú znaky na páskach $T_1, \ldots, T_m$

v pôvodnom poradí.

V druhej fáze nahrádzania uvažovaného kroku nahrádzame výskyty symbolu Θ_l na páske T_l, pre všetky $l, 1 \leq l \leq j-1$. Nech $s \in \{1, \ldots, j-1\}$, popíšeme prácu automatu pri nahrádzaní výskytu symbolu Θ_s na páske T_s, analogicky automat pracuje pri spracovávaní ostatných pások.

Automat použije pomocnú pásku P_s. Hlavy pásiek $T_1, \ldots, T_m$ sa nachádzajú na ich konci. Teda pásku T_s začne čítať od konca smerom doľava. Automat číta na páske T_s koniec masky s indexom rôznym od s. Automat spracuje masku, ktorej číta posledný symbol nasledujúcim spôsobom. Zapíše koniec spracovávanej masky na pásku P_s a označí ho horným indexom | na páske T_s. V stave si zapamätá, že hľadá symbol Θ_s. Potom číta masku smerom doľava podľa aktuálne čítaneho znaku z a stavu, v ktorom sa automat nachádza, sa správa nasledovne.

- Automat hľadá symbol Θ_s.
  - $z \in \{\Theta_1, \ldots, \Theta_{s-1}, \Theta_{s+1}, \ldots, \Theta_m\} \cup \{]_1, \ldots,]_m\} \cup T$

    Symbol z sa iba kopíruje, automat na pásku P_s zapíše symbol z, na páske T_s posunie hlavu doľava.
  - $z = \Theta_s$

    Automat nedeterministicky uhádne, či čítaný výskyt symbolu Θ_s je vrámci spracovávanej masky posledný v smere od konca pásky. Ak sa rozhodne, že je posledný, znak z na páske T_s označí, zapíše $\Theta_s^|$ a posunie hlavu doľava. Automat si v stave zapamätá, že hľadá symbol $]_s$. Inak si v stave zapamätá, že duplikuje spracovávanú masku, posunie hlavu pásky T_s doľava a na pásku P_j zapíše symbol Θ_s.
  - $z = [$

    Pásku T_s začne čítať smerom doprava. Číta pokým, neprečíta symbol $\Theta_s^|$ alebo symbol $]_l^|, 1 \leq l \leq m$. Nech prečíta symbol $\Theta_s^|$, ten odznačí a začne čítať pásku T_s smerom doľava, pričom kopíruje čítané znaky na pásku P_s, až pokým neprečíta začiatok masky alebo neprečíta symbol Θ_s. Ak prečíta symbol Θ_s, automat zastaví a neakceptuje, pretože automat sa nesprávne rozhodol, ktorý výskyt symbolu Θ_s je posledný v rámci masky, ktorej znaky automat aktuálne číta. Ak automat prečíta začiatok masky, celý proces opakuje. Ak automat prečíta symbol $]_l^|, 0 \leq l \leq m$, označí ho ako zmazaný, posunie hlavu pásky T_s doľava, na pásku P_s zapíše začiatok masky a v stave si zapamätá, že hľadá koniec masky s indexom rôznym od s.

- Automat duplikuje spracovávanú masku.
  - $z \in \{\Theta_1, \ldots, \Theta_{s-1}, \Theta_{s+1}, \ldots, \Theta_m\} \cup T$

    Symbol z sa prekopíruje na pásku P_j a na páske T_s posunie hlavu doľava.
  - $z = \Theta_s$

    Automat nedeterministicky uhádne, či čítaný výskyt symbolu Θ_s je vrámci spracovávanej masky posledný v smere od konca pásky. Ak sa rozhodne, že je posledný, znak z na páske T_s označí, zapíše $\Theta_s^{|}$ a posunie hlavu doľava, hlavu pásky P_s vráti na začiatok pásky. Automat si v stave zapamätá, že spracováva pásku P_j. Inak kopíruje znak z na pásku P_j a posunie hlavu pásky T_s doľava.
  - $z = [$

    Automat zastaví a neakceptuje, pretože sa zle rozhodol, ktorý výskyt symbolu Θ_s je v rámci spracovávanej masky posledný.
- Automat spracováva pásku P_j

  Automat číta znaky na páske P_j a kopíruje ich na pásku P_s a označuje ako zmazané, pokým neprečíta symbol Θ_s alebo pokým neprečíta celú pásku. Keď prečíta symbol Θ_s označí ho ako zmazaný a na páske T_s, číta znaky smerom doľava pokým neprečíta symbol $]_s$, potom pokračuje v čítaní, pokým neprečíta terminál, ten zapíše na pásku P_s pokračuje v čítaní pričom čítané symboly kopíruje na pásku P_s, pokým neprečíta začiatok masky. Vtedy celý proces opakuje. Ak automat prečítal celú pásku P_j, hlavu pásky P_j vráti na začiatok a v stave si zapamätá, že hľadá symbol $]_s$.
- Automat hľadá symbol $]_s$.
  - $z \in \{\Theta_1, \ldots, \Theta_m\} \cup \{[\} \cup \{\ ]_1, \ldots,]_{s-1},]_{s+1}, \ldots,]_m\} \cup T$

    Automat posunie hlavu pásky T_s doľava.
  - $z =]_s$

    V stave si zapamätá, že hľadá symbol Θ_s.
- Automat hľadá koniec masky s indexom rôznym od s.
  - $z \in \{\Theta_1, \ldots, \Theta_m\} \cup \{\ ]_s\} \cup T$

    Čítaný symbol automat označí za zmazaný a posunie hlavu pásky T_s doľava ak sa dá.

– $z \in \{\ ]_1, \ldots,]_{s-1},]_s, \ldots,]_m\}$
Automat označí znak z, na pásku T_s zapíše $z^|$ a posunie hlavu pásky doľava a v stave si zapamätá, že hľadá symbol Θ_s.

Po prečítaní celej pásky T_s automat prečíta celú pásku P_s zľava doprava, pričom kopíruje čítané znaky na pásku T_s a označuje ich za zmazané na páske P_s. Tým automat ukončí nahrádzanie výskytov symbolou Θ_s na páske T_s, opakuje postup pre všetky ostané pásky, ktoré prichádzajú do úvahy. Čím automat ukončí druhú fázu jedného kroku nahrádzania Θ-symbolov. A teda aj uvažovaný krok nahrádzania Θ-symbolov.

Keď automat nahradí všetky Θ-symboly nastáva tretia fáza práce automatu. Po druhej fáze práce automatu sa na páske T_1 nachádza jediná vetná forma tvaru $[w]_0, w \in T^+$. Pričom hlava danej pásky sa nachádza na jej konci. Automat presunie hlavu tejto pásky na prvý znak po symbole začiatku masky na páske a začne porovnávanie. Potom číta pásky smerom doprava, pričom kontroluje, či čítané znaky na páskach sa rovnajú, ak nie, automat zastaví a neakceptuje. Automat zastaví a neakceptuje aj v prípade, keď dočítame vstupnú pásku skôr ako hlava pásky T_1, číta znak vľavo od konca masky. V prípade, že automat prečíta na páske T_1 symbol konca masky akceptuje vstupné slovo.

Prácu automatu zostrojeného podľa vyššie uvedenej konštrukcie ukážeme na príkladoch, po jednotlivých fázach práce automatu. Pre prvú fázu automatu, uvažujeme časť nejakého odvodenia systému spĺňajúceho predpoklady tvrdenia. K nemu vytvoríme zodpovedajúce tvary pások automatu, po jednotlivých krokoch odvodenia systému.

- časť odvodenia systému[1]

G_1	σ_1	aA	σ_1	aA	aA	aaA	aaA	$aaaA$
G_2	σ_2	bB	bB	bbB	σ_2	bQ_3	$bccbbcC$	$bccbbcbB$
G_3	σ_3	cC	cC	ccQ_2	$ccbbB$	$ccbbcC$	σ_3	cQ_2
G_4	σ_4	dQ_1	daA	$dadD$	$dadD$	$daddD$	$daddD$	$dadddD$

$aaaA$	$aaaaQ_3$	$aaaacbccbbcbdC$	$aaaacbccbbcbda$
σ_2	bB	bB	bbB
$cbccbbcbB$	$cbccbbcbdC$	σ_3	cC
$dadddD$	$daddddD$	$daddddD$	$dadddddD$

- zodpovedajúce tvary pások a neterminály pamätané v stave automatu

[1] plnými čiarami sú znázornené prepisovacie kroky a prerušovanými čiarami komunikačné kroky

$$
\begin{array}{c||ll|ll|ll|ll|ll|ll|ll|ll|}
T_1 & \beta & \sigma_1 & \beta & A & [& \sigma_1 & [a & A & [a & A & [aa & A & [aa & A \\
T_2 & [& \sigma_2 & [b & B & [b & B & [bb & B & [bb]_3[& \sigma_2 & [bb]_3[b & Q_3 & [bb]_3[b\Theta_3 & C \\
T_3 & [& \sigma_3 & [c & C & [c & C & [cc & Q_2 & [cc\Theta_2 & B & [cc\Theta_2 c & C & [cc\Theta_2 c]_2[& \sigma_3 \\
T_4 & \beta & \sigma_4 & \beta & Q_1 & \beta & A & \beta & D & \beta & D & \beta & D & \beta & D
\end{array}
$$

$$
\begin{array}{|ll|ll|ll|ll|}
[aaa & A & [aaa & A & [aaaa & Q_3 & [aaaa\Theta_3 & C \\
[bb]_3[b\Theta_3 b & B & [bb]_3[b\Theta_3 b]_3\beta & \sigma_3 & [bb]_3[b\Theta_3 b]_3\beta & B & [bb]_3[b\Theta_3 b]_3\beta & B \\
[cc\Theta_2 c]_2[c & Q_2 & [cc\Theta_2 c]_2[c\Theta_2 & B & [cc\Theta_2 c]_2[c\Theta_2 d & C & [cc\Theta_2 c]_2[c\Theta_2 d]_1\beta & \sigma_3 \\
\beta & D & \beta & D & \beta & D & \beta & D
\end{array}
$$

$$
\begin{array}{|ll}
[aaaa\Theta_3 a]_0 & M \\
[bb]_3[b\Theta_3 b]_3\beta & B \\
[cc\Theta_2 c]_2[c\Theta_2 d]_1\beta & C \\
\beta & D
\end{array}
$$

Ako ukážku práce automatu v druhej fáze uvedieme jeden krok nahrádzania Θ-symbolov. Najskôr uvedieme príklad práce atomatu v prvej fáze nahrádzania a potom v druhej. Príklad práce automatu v prvej fáze nahrádzania:

- pásky pred zahájením prvej fázy nahrádzania symbolov Θ_3 [2]

$$
\begin{array}{c||ll||l}
T_1 & [aaaa\Theta_3 a]_0 \Lsh & P_1 & \Lsh \\
T_2 & [bb]_3[b\Theta_3 b]_3 \Lsh & P_2 & \Lsh \\
T_3 & [cc\Theta_2 c]_2[c\Theta_2 d]_1 \Lsh & P_3 & \Lsh \\
T_4 & \Lsh & P_4 & \Lsh
\end{array}
$$

- spracovanie prvej masky na páske T_3 [3]

$$
\begin{array}{c||ll||l}
T_1 & [aaaa \Lsh \Theta_3^* a^*]_0^* & P_1 &]_0 ad\Theta_2 c \Lsh \\
T_2 & [bb]_3[b\Theta_3 b]_3 \Lsh & P_2 & \Lsh \\
T_3 & [cc\Theta_2 c]_2 \Lsh [^* c^* \Theta_2^* d^*]_1^* & P_3 & Z_1^2 \Lsh \\
T_4 & \Lsh & P_4 & \Lsh
\end{array}
$$

- po prečitaní celej pásky T_3 a po dočítaní pások T_1, T_2

$$
\begin{array}{c||ll||l}
T_1 & \Lsh [^* a^* a^* a^* a^* \Theta_3^* a^*]_0^* & P_1 &]_0 ad\Theta_2 caaaa[\Lsh \\
T_2 & \Lsh [^* b^* b^*]_3^* [^* b^* \Theta_3^* b^*]_3^* & P_2 &]_3 bc\Theta_2 ccb[\ ]_3 bb[\Lsh \\
T_3 & \Lsh [^* c^* c^* \Theta_2^* c^*]_2^* [^* c^* \Theta_2^* d^*]_1^* & P_3 & Z_1^2 Z_2^2 \Lsh \\
T_4 & \Lsh & P_4 & \Lsh
\end{array}
$$

[2] znak $\Lsh$ znamená, že hlava pásky číta znak pred ním, označuje pozíciu hlavy pásky, nie je súčasťou pásky

[3] index $*$ znaku znamená, že znak bol označený ako zmazaný

- po dokončení prvej fázy nahrádzania symbolu Θ_3

$$\begin{array}{l||l l||l} T_1 & [aaaac\Theta_2da]_0 \Lsh & P_1 & \Lsh \\ T_2 & [bb]_2[bcc\Theta_2cb]_1 \Lsh & P_2 & \Lsh \\ T_3 & \Lsh & P_3 & \Lsh \\ T_4 & \Lsh & P_4 & \Lsh \end{array}$$

Príklady práce automatu v druhej fáze nahrádzania. Automat nahrádza výskyty symbolu Θ_2. Znázornené sú určité časti nahrádzania.

- Nahrádzanie výskytu symbolu Θ_2 na páske T_2 z predchádzajúceho príkladu.
  - počiatočný stav pások T_2, P_2

    $$\begin{array}{l||l l||l} T_2 & [bb]_2[bcc\Theta_2cb]_1 \Lsh & P_2 & \Lsh \end{array}$$
  - automat hľadá symbol Θ_2

    $$\begin{array}{l||l l||l} T_2 & [bb]_2[bcc\Theta_2cb \Lsh]^{|}_1 & P_2 &]_1 \Lsh \end{array}$$
  - automat hľadá symbol $]_2$

    $$\begin{array}{l||l l||l} T_2 & [bb]_2[bcc \Lsh \Theta^{|}_2cb]^{|}_1 & P_2 &]_1bc \Lsh \end{array}$$
  - automat hľadá symbol Θ_2

    $$\begin{array}{l||l l||l} T_2 & [bb \Lsh]_2[bcc\Theta^{|}_2cb]^{|}_1 & P_2 &]_1bc \Lsh \\ T_2 & [\Lsh bb]_2[bcc\Theta^{|}_2cb]^{|}_1 & P_2 &]_1bcbb \Lsh \\ T_2 & [bb]_2[bcc \Lsh \Theta_2cb]^{|}_1 & P_2 &]_1bcbb \Lsh \\ T_2 & [bb]_2[\Lsh bcc\Theta_2cb]^{|}_1 & P_2 &]_1bcbbccb \Lsh \\ T_2 & [bb]_2[bcc\Theta_2cb]^{|}_1 \Lsh & P_2 &]_1bcbbccb \Lsh \end{array}$$
  - automat hľadá symbol konca masky nemajúci index 2

    $$\begin{array}{l||l l||l} T_2 & [bb]_2[bcc\Theta_2cb \Lsh]^{*}_1 & P_2 &]_1bcbbccb[\Lsh \\ T_2 & \Lsh [^*b^*b^*]^*_2[^*b^*c^*c^*\Theta^*_2c^*b^*]^*_1 & P_2 &]_1bcbbccb[\Lsh \end{array}$$
  - automat prekopíroval obsah pásky P_2 na pásku T_2

    $$\begin{array}{l||l l||l} T_2 & [bccbbcb]_1 \Lsh & P_2 & \Lsh \end{array}$$
- Príklad skladania slova, v ktorého odvodení dve rôzne gramatiky naraz požiadali o komunikáciu tú istú gramatiku.

T_2	$[b]_2]_1[b\Theta_1 bc\Theta_2 ccb]_1 \Lsh$	P_2	$\Lsh$
T_2	$[b]_2]_1[b\Theta_1 bc \Lsh \Theta_2^{\mid} ccb]_1^{\mid}$	P_2	$]_1 bcc \Lsh$
T_2	$[\Lsh b]_2]_1[b\Theta_1 bc\Theta_2^{\mid} ccb]_1^{\mid}$	P_2	$]_1 bccb \Lsh$
T_2	$[b]_2]_1[b\Theta_1 bc \Lsh \Theta_2 ccb]_1^{\mid}$	P_2	$]_1 bccb \Lsh$
T_2	$[b]_2]_1[\Lsh b\Theta_1 bc\Theta_2 ccb]_1^{\mid}$	P_2	$]_1 bccbcb\Theta_1 b \Lsh$
T_2	$[b]_2]_1 \Lsh [^*b^*\Theta_1^* b^* c^* \Theta_2^* c^* c^* b^*]_1^*$	P_2	$]_1 bccbcb\Theta_1 b \Lsh$
T_2	$[\Lsh b]_2]_1^{\mid}[^*b^*\Theta_1^* b^* c^* \Theta_2^* c^* c^* b^*]_1^*$	P_2	$]_1 bccbcb\Theta_1 b[]_1 b \Lsh$
T_2	$\Lsh [^*b^*]_2^*]_1^*[^*b^*\Theta_1^* b^* c^* \Theta_2^* c^* c^* b^*]_1^*$	P_2	$]_1 bccbcb\Theta_1 b[]_1 b[\Lsh$
T_2	$[b]_1[b\Theta_1 bcbccb]_1 \Lsh$	P_2	$\Lsh$

- Príklad skladania slova, kde maska obsahuje viac výskytov nahrádzaného symbolu.
  - na páske T_1 nahrádzame výskyty symbolu Θ_1, počiatočný tvar pásky T_1

    $$T_1 \;\Big\|\; [aba]_1[a]_1[ab\Theta_1 b\Theta_1 ba]_1[acccccca]_1[a]_1[ab\Theta_1 bb\Theta_1 b\Theta_1 ba]_0 \Lsh$$
  - automat hľadá symbol Θ_1

    $$T_1 \;\Big\|\; [aba]_1[a]_1[ab\Theta_1 b\Theta_1 ba]_1[acccccca]_1[a]_1[ab\Theta_1 bb\Theta_1 b\Theta_1 ba \Lsh]_0^{\mid}$$
  - automat duplikuje spracovávanú masku

    $$T_1 \;\Big\|\; [aba]_1[a]_1[ab\Theta_1 b\Theta_1 ba]_1[acccccca]_1[a]_1[ab\Theta_1 bb\Theta_1 b\Theta_1 \Lsh ba]_0^{\mid}$$
  - automat spracováva pásku P_2

    $$T_1 \;\Big\|\; [aba]_1[a]_1[ab\Theta_1 b\Theta_1 ba]_1[acccccca]_1[a]_1[ab \Lsh \Theta_1^{\mid} bb\Theta_1 b\Theta_1 ba]_0^{\mid}$$
  - automat hľadá symbol $]_1$

    $$T_1 \;\Big\|\; [aba]_1[a]_1[ab\Theta_1 b\Theta_1 ba]_1 \Lsh [acccccca]_1[a]_1[ab\Theta_1^{\mid} bb\Theta_1 b\Theta_1 ba]_0^{\mid}$$
  - automat hľadá symbol Θ_1

    $$T_1 \;\Big\|\; [aba]_1[a]_1[ab\Theta_1 b\Theta_1 ba]_1 \Lsh [acccccca]_1[a]_1[ab\Theta_1^{\mid} bb\Theta_1 b\Theta_1 ba]_0^{\mid}$$
  - automat duplikuje spracovávanú masku

    $$T_1 \;\Big\|\; [aba]_1[a]_1[ab\Theta_1 b\Theta_1 \Lsh ba]_1[acccccca]_1[a]_1[ab\Theta_1^{\mid} bb\Theta_1 b\Theta_1 ba]_0^{\mid}$$
  - automat spracováva pásku P_2

    $$T_1 \;\Big\|\; [aba]_1[a]_1[ab \Lsh \Theta_1^{\mid} b\Theta_1 ba]_1[acccccca]_1[a]_1[ab\Theta_1^{\mid} bb\Theta_1 b\Theta_1 ba]_0^{\mid}$$
  - automat hľadá symbol $]_1$, následne hľadá symbol Θ_1

T_1	$[aba]_1 \Lsh [a]_1[ab\Theta_1^{\mid} b\Theta_1 ba]_1[acccccca]_1[a]_1[ab\Theta_1^{\mid} bb\Theta_1 b\Theta_1 ba]_0^{\mid}$
T_1	$[\Lsh aba]_1[a]_1[ab\Theta_1^{\mid} b\Theta_1 ba]_1[acccccca]_1[a]_1[ab\Theta_1^{\mid} bb\Theta_1 b\Theta_1 ba]_0^{\mid}$
T_1	$[aba]_1[a]_1[ab \Lsh \Theta_1 b\Theta_1 ba]_1[acccccca]_1[a]_1[ab\Theta_1^{\mid} bb\Theta_1 b\Theta_1 ba]_0^{\mid}$
T_1	$[aba]_1[a]_1[ab\Theta_1 b\Theta_1 ba]_1[acccccca]_1[a]_1[ab \Lsh \Theta_1 bb\Theta_1 b\Theta_1 ba]_0^{\mid}$
T_1	$[aba]_1[a]_1[ab\Theta_1 b\Theta_1 ba]_1[acccccca]_1[a]_1[ab\Theta_1 bb\Theta_1 b\Theta_1 ba]_0^{\mid} \Lsh$

– automat hľadá koniec masky s indexom rôznym od 0

$$T_1 \;\Big\|\; [aba]_1[a]_1[ab\Theta_1 b\Theta_1 ba]_1[acccccca]_1[a]_1[ab\Theta_1 bb\Theta_1 b\Theta_1 ba \Lsh]_0^*$$

– pomocné pásky P_1, P_2 zobrazené v rovnakých fázach ako páska T_1

P_1	$\Lsh$	P_2	$\Lsh$
P_1	$]_0 \Lsh$	P_2	$\Lsh$
P_1	$]_0ab \Lsh$	P_2	$\Lsh$
P_1	$]_0ab \Lsh$	P_2	$\Lsh \Theta_1 b\Theta_1 bb$
P_1	$]_0ababaccccccabb \Lsh$	P_2	$\Lsh \Theta_1^* b^* \Theta_1^* b^* b^*$
P_1	$]_0ababaccccccabb \Lsh$	P_2	$\Lsh$
P_1	$]_0ababaccccccabbab \Lsh$	P_2	$\Lsh$
P_1	$]_0ababaccccccabbab \Lsh$	P_2	$\Lsh \Theta_1 b$
P_1	$]_0ababaccccccabbabab \Lsh$	P_2	$\Lsh \Theta_1^* b^*$
P_1	$]_0ababaccccccabbabababa \Lsh$	P_2	$\Lsh$
P_1	$]_0ababaccccccabbababababa \Lsh$	P_2	$\Lsh$
P_1	$]_0ababaccccccabbabababababa \Lsh$	P_2	$\Lsh$
P_1	$]_0ababaccccccabbabababababa[\Lsh$	P_2	$\Lsh$

– pásky T_1, P_1, P_2 po skončení nahrádzania symbolu Θ_1 na páske T_1

$$T_2 \;\Big\|\; [ababababababbaccccccababa]_0 \Lsh \qquad P_1 \;\Big\|\; \Lsh \qquad P_2 \;\Big\|\; \Lsh$$

Popísali sme ako zostrojíme automat A k danému systému Γ, v nasledujúcom texte ukážeme, že platí $L(\Gamma) = L(A)$, a že automat A pracuje v lineárnom čase.

Tvrdenie 4.1.1. *Pásky $T_1, \ldots, T_m$ automatu A po prvej fáze akceptačného výpočtu automatu obsahujú masky alebo sú prázdne.*

Dôkaz. Na začiatku prvej fázy, pásky obsahujú symbol začiatok masky alebo symbol β. Ak nejaká páska automatu z pásiek $T_1, \ldots, T_m$ neobsahuje symbol β, z popisu prvej fázy práce automatu a z faktu, že uvažujeme akceptačný výpočet vyplýva, že automat na danú pásku zapíše symbol pre koniec masky a následne symbol β alebo začiatok masky. Ak sa na nejakej páske nachádza symbol β, počas simulácie odvodenia systému Γ, automat na danú pásku nezapisuje žiadne symboly, až pokým sa nerozhodne tento symbol zmazať, následne na pásku zapíše začiatok masky. Na konci prvej fázy práce automat kontroluje výskyty β symbolov, vtedy na pásky nezapisuje žiadne symboly iba maže tieto symboly. Z uvedeného vyplýva platnosť dokazovaného tvrdenia. □

Tvrdenie 4.1.2. *Páska T_1 automatu A po prvej fáze akceptačného výpočtu obsahuje aspoň jednu masku a posledná maska na nej ležiaca má index 0.*

Dôkaz. Z Tvrdenia 4.1.1 vyplýva, že páska T_1 po prvej fáze práce automatu A obsahuje masky alebo je prázdna. Ak by mala byť páska T_1 po prvej fáze práce automatu prázdna, v prvom kroku výpočtu automatu, automat musel na danú pásku zapísať symbol β inak by obsahovala po prvej fáze aspoň symbol pre začiatok masky, keďže automat počas svojej prvej fázy maže iba symboly β. Počas simulácie odvodenia systému Γ sa automat nemohol rozhodnúť zmazať symbol β, ktorý obsahuje páska T_1, pretože páska by obsahovala aspoň začiatok masky. Ale ak páska T_1 počas kontroly výskytov symbolu β obsahuje symbol β automat zastaví a neakceptuje. Z uvedeného vyplýva, že po prvej fáze akceptačného výpočtu automatu A páska T_1 obsahuje aspoň jednu masku. Posledný symbol, ktorý automat zapíše na pásku T_1 počas uvažovanej časti výpočtu je symbol $]_0$. Z predchádzajúceho vyplýva dokazované tvrdenie. □

Tvrdenie 4.1.3. *Počas druhej fázy výpočtu automatu A platí, že maskou s indexom $i, 1 \leq i \leq m$ má byť nahradený symbol Θ ležiaci na páske T_i.*

Dôkaz. Dokážeme, že po každom kroku druhej fázy výpočtu automatu A platí, že maskou s indexom $i, 1 \leq i \leq m$ má byť nahradený symbol Θ ležiaci na páske T_i. Ukážeme, matematickou indukciou na počet vykonaných krokov druhej fázy výpočtu automatu, tento počet označíme k.

1. $k = 0$

   Priamo po prvej fáze práce automatu A platí dokazované tvrdenie, vyplýva priamo z konštrukcie automatu.

2. predpokladáme, že tvrdenie platí po $l, l < k$ krokoch druhej fázy výpočtu automatu, dokážeme, že platí aj po k krokoch

   Z indukčného predpokladu vieme, že na začiatku k-teho kroku platí, že maskou s indexom $i, 1 \leq i \leq m$ má byť nahradený symbol Θ ležiaci na páske T_i. Iba v prvej fáze každého kroku druhej fázy výpočtu automatu, symboly Θ menia svoju pozíciu z nejakej pásky na inú pásku. Ak sú súčasťou masky, ktorá je vkladaná do inej masky v uvažovanej fáze výpočtu. Počas tohto vkladania, automat zaznamená pre symboly Θ obsiahnuté vo vkladanej maske, zmenu ich pozície a následne na konci prvej fázy kroku druhej fázy výpočtu automatu aktualizuje indexy masiek, ktorými majú byť nahradené tieto symboly Θ, tak aby platilo dokazované tvrdenie.

Z dokázaného, vyplýva platnosť tvrdenia. □

Tvrdenie 4.1.4. *Po prvej fáze výpočtu automatu A platí nasledovné. Nech znak z ležiaci na páske $T_i, 1 \leq i \leq m$ automatu A, je symbol $\Theta_j, j \neq i \wedge 1 \leq j \leq m$ a je to l-tý* *výskyt tohto symbolu na páske T_i od jej konca. Potom znak z má byť nahradený maskou s indexom i ležiacou na páske T_j automatu A, ktorá je l-tá v poradí od konca pásky T_j spomedzi masiek s indexom i.*

Dôkaz. Nech znak z ležiaci na páske $T_i, 1 \leq i \leq m$ automatu A, je symbol $\Theta_j, j \neq i \wedge 1 \leq j \leq m$ a je to l-tý výskyt tohto symbolu na páske T_i od jej konca. Index j symbolu Θ znamená, že na miesto daného symbolu, patrí vetná forma generovaná gramatikou G_j. Po prvej fáze výpočtu automatu A všetky vetné formy reprezentované maskami, ktoré ešte neboli spracované (tj. existuje symbol Θ, ktorý má byť nahradený danou maskou), generované gramatikami systému Γ sa nachádzajú na páskach zodpovedajúcim jednotlivým gramatikám. Teda hľadaná maska sa nachádza na páske T_j. Z masiek nachádzajúcich sa na páske T_j prichádzajú do úvahy iba masky s indexom i, ktorý značí, že reprezentujú vetnú formu, ktorá mala byť vložená do vetnej formy generovanej gramatikou G_i, Tvrdenie 4.1.3. Pre každý symbol Θ_j na páskach $T_1, \ldots, T_m$ automatu existuje na páske T_j maska, ktorou má byť nahradený, zároveň platí, že na páske T_j sa nenachádza viac masiek ako symbolov Θ_j na páskach $T_1, \ldots, T_m$ automatu A. V okamihu keď automat A na pásku T_i zapísal symbol Θ_j, existujú všetky masky potrebné k zloženiu vetnej formy, ktorá má byť prekopírovaná na jeho miesto. Z uvedeného vyplýva, dokazované tvrdenie. □

Tvrdenie 4.1.5. *Nech $1 \leq i \leq m-1$. Po prvej fáze i-teho kroku druhej fázy výpočtu automatu A sú pásky $T_{m-i+1}, \ldots, T_m$ prázdne a ostanú prázdne až do konca výpočtu automatu.*

Dôkaz. Nech $1 \leq i \leq m-1$. Automat A v prvej fáze i-teho kroku druhej fázy svojho výpočtu vyprázdni pásku T_{m+i-1} a potom už viac naňu nezapíše žiaden symbol, vyplýva priamo z konštrukcie automatu A. A teda platí dokazované tvrdenie. □

Tvrdenie 4.1.6. *Po prvej fáze výpočtu automatu A platí nasledovné. Nech znak z ležiaci na páske $T_i, 1 \leq i \leq m$ automatu A, je symbol Θ_i a je to l-tý výskyt tohto symbolu v maske x od jej konca, kde $l \geq 1$. Potom znak z má byť nahradený maskou s indexom i ležiacou na páske T_i automatu A pred masku x, pričom je l-tá v poradí od masky x spomedzi masiek s indexom i.*

Dôkaz. Nech znak z, ktorý je súčasťou masky x ležiacej na páske $T_i, 1 \leq i \leq m$ automatu A, je l-tý výskyt symbolu Θ_i vrámci masky x od jej konca, kde $l \geq 1$. Index i symbolu Θ znamená, že na miesto daného symbolu, patrí vetná forma generovaná gramatikou G_i. Po prvej fáze výpočtu automatu A všetky vetné formy reprezentované maskami, ktoré ešte neboli spracované (tj. existuje symbol Θ, ktorý má byť nahradený danou maskou), generované gramatikami systému Γ sa nachádzajú na páskach zodpovedajúcim jednotlivým gramatikám. Teda hľadaná maska sa nachádza na páske T_i. Z masiek nachádzajúcich sa na páske T_i prichádzajú do úvahy iba masky s indexom i, ktorý značí, že reprezentujú vetnú formu, ktorá mala byť vložená do vetnej formy gramatiky G_i, Tvrdenie 4.1.3. Maska, ktorá má nahradiť znak z sa musí nachádzať na páske T_i pred maskou x, keďže v okamihu keď automat A na pásku T_i zapísal symbol Θ_i, existujú všetky masky potrebné k zloženiu vetnej formy, ktorá má byť prekopírovaná na jeho miesto. A z toho istého dôvodu vyplýva, že je l-tá v poradí od masky x spomedzi masiek s indexom i ležiacich na páske T_i. $\square$

Tvrdenie 4.1.7. *Počas druhej fázy výpočtu automatu A platí nasledovné. Nech x je maska ležiaca na páske $T_i, 1 \leq i \leq m$, pričom obsahuje viac symbolov $\Theta_j, i \neq j \wedge 1 \leq j \leq m$ (tj. $|x|_{\{\Theta_j\}} \geq 2$). Potom iba maska, ktorou má byť nahradený prvý výskyt (zľava) symbolu Θ_j v maske x, môže obsahovať symbol Θ_i.*

Dôkaz. Nech počas druhej fázy výpočtu automatu A, sa na páske $T_i, 1 \leq i \leq m$ automatu nachádza maska x obsahujúca viac symbolov $\Theta_j, i \neq j \wedge 1 \leq j \leq m$. Potom čo bol zapísaný na pásku T_i prvý výskyt symbolu Θ_j v maske x, gramatika G_j sa vrátila k počiatočnému neterminálu, ak v ďalšom simulovanom odvodení gramatika G_j požiada o vetnú formu gramatiku G_i, automat ukončí masku x a preto masky, ktorými majú byť nahradené výskyty symbolu Θ_j, okrem prvého výskytu, v maske x, nemôžu obsahovať symbol Θ_i, pretože automat A ich začiatky zapísal na pásku T_j neskôr ako začiatok masky x. $\square$

Tvrdenie 4.1.8. *Pre každé $i, 1 \leq i \leq m$ platí, že páska T_i automatu A po prvej fáze akceptačného výpočtu automatu neobsahuje symbol Θ_i.*

Dôkaz. V uvažovanej časti výpočtu automat symbol $\Theta_i, 1 \leq i \leq m$ zapíše na pásku $T_j, 1 \leq j \leq m$, iba v prípade keď gramatika G_j systému Γ v simulovanom odvodení vygenerovala komunikačný symbol Q_i. Ale ak $i = j$ automat zastaví a neakceptuje. Preto platí dokazované tvrdenie. $\square$

Tvrdenie 4.1.9. *Nech $1 \leq i \leq m$. Nech na začiatku i-teho kroku druhej fázy výpočtu automatu A, pre každé $j, 1 \leq j \leq m$ platí, že symbol Θ_j sa nenachádza na páske T_j. Potom aj po skončení uvažovaného kroku druhej fázy výpočtu, platí že pre každé $l, 1 \leq l \leq m$ platí, že páska T_l automatu A neobsahuje symbol Θ_l.*

Dôkaz. Nech $1 \leq i \leq m$. Predpokladajme, že na začiatku i-teho kroku druhej fázy výpočtu automatu A, pre každé $j, 1 \leq j \leq m$ platí, že na páske T_j automatu A sa nenachádza symbol Θ_j. Automat v prvej fáze uvažovaného kroku druhej fázy výpočtu automatu, nahrádza symboly Θ_{m-i+1} na páskach $T_1, \ldots, T_{m-i}$ automatu A, pretože pásky $T_{m-i+2}, \ldots, T_m$ sú prázdne, vyplýva z Tvrdenia 4.1.5 a predpokládame, že na páske T_{m-i+1} sa nenachádza symbol Θ_{m-i+1}. Po prvej fáze uvažovaného kroku druhej fázy výpočtu automatu, môže existovať $l, 1 \leq l \leq m-i$ také, že na páske T_l sa nachádza symbol Θ_l, len v prípade ak v tejto časti výpočtu automatu, automat nahradil na páske T_l výskyt symbolu Θ_{m-i+1} obsiahnutý v nejakej maske x, maskou y obsahujúcou symbol Θ_l. Z Tvrdenia 4.1.7 vyplýva, že maskou y v maske x je nahrádzaný prvý výskyt symbolu Θ_{m-i+1} a ešte, že iba maska, ktorou má byť nahradený prvý výskyt symbolu Θ_l v maske y, môže obsahovať symbol Θ_{m-i+1} a teda po jeho nahradení, symbol Θ_l. Z uvedeného a konštrukcie automatu A vyplýva, že po ukončení uvažovaného kroku druhej fázy výpočtu, platí pre každé $j, 1 \leq j \leq m$ páska T_j neobsahuje symbol Θ_j. □

Tvrdenie 4.1.10. *Po skončení druhej fázy každého kroku druhej fázy akceptačného výpočtu automatu A platí, že na páske $T_i, 1 \leq i \leq m$ sa nenachádza symbol Θ_i.*

Dôkaz. Po prvej fáze akceptačného výpočtu automatu A platí, že na páske $T_i, 1 \leq i \leq m$ sa nenachádza symbol Θ_i, vyplýva z Tvrdenia 4.1.8. Z toho a Tvrdenia 4.1.9 vyplýva, že po druhej fáze prvého kroku druhej fázy výpočtu, pre každé $j, 1 \leq j \leq m$ platí, že páska T_i neobsahuje symbol Θ_i. Analogicky potom to isté platí pre všetky ostatné kroky druhej fázy výpočtu automatu. Z toho vyplýva platnosť dokazovaného tvrdenia. □

Tvrdenie 4.1.11. *Pre každé $i, 1 \leq i \leq m$ platí, že v druhej fáze i-teho kroku druhej fázy akceptačného výpočtu automatu A automat pre každé $j, 1 \leq j \leq m$, nahradí všetky výskyty symbolu Θ_j na páske T_j príslušnymi maskami.*

Dôkaz. Nech $1 \leq i \leq m$. Uvažujme i-tý krok druhej fázy výpočtu automatu A. Z Tvrdenia 4.1.10 vyplýva, že v druhej fáze uvažovaného kroku druhej fázy výpočtu, automat pre každé $j, 1 \leq j \leq m$ nahradí všetky výskyty symbolu Θ_j na páske T_j. Fakt, že ich nahradí príslušnými maskami vyplýva z Tvrdenia 4.1.6 a konštrukcie automatu A. □

Tvrdenie 4.1.12. *Nech $1 \leq i \leq m$. Na začiatku i-teho kroku druhej fázy výpočtu automatu A sa na páskach $T_1, \ldots, T_m$ nachádzajú len masky s indexom z množiny $\{0, 1, \ldots, m-i+1\}$.*

Dôkaz. Pre prvý krok druhej fázy výpočtu automatu A tvrdenie platí, vyplýva to priamo z konštrukcie automatu. Nech $2 \leq i \leq m$. Z Tvrdenia 4.1.5 vyplýva, že na začiatku i-teho kroku druhej fázy výpočtu automatu A sú pásky $T_{m-i+2}, \ldots, T_m$ prázdne. Z Tvrdenia 4.1.3 a z faktu, že symbol Θ sa nenáchadza na páskach $T_{m-i+2}, \ldots, T_m$, vyplýva, že masky ležiace ma páskach $T_1, \ldots, T_m$ majú indexy z množiny $\{0, 1, \ldots, m-i+1\}$. $\square$

Tvrdenie 4.1.13. *Pre každé $i, 1 \leq i \leq m-1$ platí, že v prvej fáze i-teho kroku druhej fázy akceptačného výpočtu automatu A automat nahradí všetky výskyty symbolu Θ_{m-i+1} na páskach $T_1, \ldots, T_m$ príslušnymi maskami.*

Dôkaz. Nech $1 \leq i \leq m-1$. Na začiatku prvej fázy i-teho kroku druhej fázy výpočtu automatu A sa všetky výskyty symbolu Θ_{m-i+1} nachádzajú na páskach $T_1, \ldots, T_{m-i}$ automatu, vyplýva z Tvrdenia 4.1.10 a Tvrdenia 4.1.5. Všetky masky, ktorými majú byť nahradené výskyty symbolu Θ_{m-i+1} na páskach $T_1, \ldots, T_{m-i}$ automatu, sa nachádzajú na páske T_{m-i+1}, vyplýva z Tvrdenia 4.1.4. Na páske T_{m-i+1} ležia len masky s indexami z množiny $\{1, \ldots, m-i\}$, vyplýva z tvrdení 4.1.10, 4.1.12 a z faktu, že maska s indexom 0 sa nachádza len na páske T_1. V i-tom kroku druhej fázy akceptačného výpočtu automatu A automat prechádza od konca pásku T_{m-i+1} a každú masku prekopíruje na nejakú pásku z pásiek $T_1, \ldots, T_{m-i}$ na miesto určené symbolom Θ_{m-i+1}. Uvažovaný krok končí, keď automat spracuje všetky masky ležiace na páske T_{m-i+1}. Z toho vyplýva, že automat v i-tom kroku druhej fázy výpočtu nahradí všetky výskyty symbolu Θ_{m-i+1}. Automat masky ležiace na páske T_{m-i+1} spracováva nasledovne, nech spracováva masku, ktorá je prvá, vrámci masiek s indexom $l, 1 \leq l \leq m-i$ na páske T_{m-i+1}. Hlava pásky T_l sa nachádza na jej konci. To znamená, že keď automat číta túto pásku smerom doľava, nájde prvý výskyt symbolu Θ_{m-i+1}, ktorý zodpovedá spracovávanej maske, Tvrdenie 4.1.4. Nech spracovávaná maska je s-tá v poradí, v rámci masiek s indexom $l, 1 \leq l \leq m-i$ na páske T_{m-i+1}, to znamená, že sme spracovali $s-1$ masiek s indexom l, pretože hlava pásky T_l sa nachádza, pred $s-1$ výskytom symbolu Θ_{m-i+1} na páske T_l a teda čítaním tejto pásky smerom doľava automat nájde s-tý výskyt daného symbolu na páske T_l, ktorý zodpovedá spracovávanej maske, Tvrdenie 4.1.4. Z uvedeného vyplýva dokazované tvrdenie. $\square$

Tvrdenie 4.1.14. *Pre každé $i, 1 \leq i \leq m-1$ platí, že na páskach $T_1, \ldots, T_m$ automatu A po i-tom kroku druhej fázy akceptačného výpočtu automatu sa nenachádzajú symboly $\Theta_j, m-i+1 \leq j \leq m$.*

Dôkaz. Dôkaz matematickou indukciou na i.

1. $i = 1$

   V prvom kroku druhej fázy automatu nahradíme všetky výskyty symbolu Θ_m, vyplýva z Tvrdenia 4.1.13. Z toho a faktu, že automat počas druhej fázy svojho výpočtu na páskach $T_1, \ldots, T_m$ symboly Θ iba premiestňuje, nezapisuje nové, vyplýva dokazované tvrdenie.

2. predpokladáme, že tvrdenie platí, pre všetky $j < i$, dokážeme tvrdenie pre i

   Z indukčného kroku vyplýva, že na páskach $T_1, \ldots, T_m$ automatu pred i-tým krokom druhej fázy výpočtu automatu A sa nenachádzajú symboly $\Theta_j, m-i+2 \leq j \leq m$. V i-tom kroku druhej fázy výpočtu nahradíme všetky výskyty symbolu Θ_{m-i+1}, Tvrdenie 4.1.13. Z toho a faktu, že automat počas druhej fázy svojho výpočtu na páskach $T_1, \ldots, T_m$ symboly Θ iba premiestňuje, nezapisuje nové, vyplýva dokazované tvrdenie.

□

Tvrdenie 4.1.15. *Páska T_1 automatu A po druhej fáze práce automatu obsahuje masku $[w]_0$, kde w je slovo patriace do jazyka $L(\Gamma)$.*

Dôkaz. Z Tvrdenia 4.1.5 vyplýva, že po druhej fáze výpočtu automatu A sú pásky $T_2, \ldots, T_m$ prázdne. Z Tvrdenia 4.1.10 a Tvrdenia 4.1.14 vyplýva, že na páske T_1 automatu A sa nenachádza symbol Θ. To znamená, že sme nahradili všetky výskyty symbolu Θ. Z toho a tvrdení 4.1.1, 4.1.2 vyplýva, že na páske T_1 leží práve jedna maska s indexom 0. Nech páska T_1 obsahuje masku $[w]_0, w \in T^*$. Slovo $w \in L(\Gamma)$, vyplýva z tvrdení 4.1.11, 4.1.13. □

Tvrdenie 4.1.16. $L(A) = L(\Gamma)$.

Dôkaz. Z práce automatu A v jeho tretej fáze a Tvrdenia 4.1.15 vyplýva dokazované tvrdenie. □

Ukázali sme, že platí $L(A) = L(\Gamma)$. Teraz ukážeme, že automat A pracuje v lineárnom čase.

Tvrdenie 4.1.17. *Automat A počas svojej prvej fázy pracuje v čase $O(n)$.*

Dôkaz. Na začiatku prvej fázy práce automatu, automat urobí jeden krok, v ktorom rozhodne, pre ktoré gramatiky systému budú zapamätávane vetné formy. Každý prepisovací krok systému Γ automat simuluje v jednom kroku. Komunikačný krok systému automat simuluje v konštantom počte krokov, vyplýva priamo z konštrukcie automatu.

Na konci prvej fázy automat ešte zkontroluje výskyt symbolov β a zmaže tieto symboly na konci pások automatu, načo protrebuje konštantný počet krokov. Z toho, že systém Γ pracuje v čase $O(n)$ a z vyššie uvedeného vyplýva, že práca automatu A v jeho prvej fáze je v čase $O(n)$. □

Tvrdenie 4.1.18. *Pásky $T_1, \ldots, T_m$ automatu A po prvej fáze akceptačného výpočtu automatu obsahujú najviac $(m+1)n$ symbolov.*

Dôkaz. Počet symbolov, ktoré sú terminály systému Γ, na páskach automatu $T_1, \ldots, T_m$ po danej časti výpočtu je rovný n, keďže počas simulácie systému automat A na svoje pásky zaznamenáva len terminály generované gramatikami systému, ktoré sú súčasťou slova, ktoré je generované v simulovanom odvodení systému Γ a predpokladáme, že dĺžka tohto slova je n. Ukážeme, že pomocných symbolov na páskach $T_1, \ldots, T_m$ po danej časti výpočtu automatu nie je viac ako mn. Pomocné symboly, ktoré sa nachadzajú na páskach automatu A po danej časti výpočtu automatu sú symboly začiatok masky, koniec masky a Θ-symboly. Na jeden začiatok masky môže pripadnúť najviac $m-1$ symbolov pre koniec masky, pretože v odvodení nejakého slova systémom Γ môže najviac $m-1$ gramatík naraz požiadať o tú istú vetnú formu. Po každom začiatku masky na páskach $T_1, \ldots, T_m$ nasleduje aspoň jeden terminál, výplýva to zo striktnej regularity komponent systému Γ. Pred a po každom výskyte symbolu Θ na páskach $T_1, \ldots, T_m$ sa nachádza terminál, čo vyplýva z faktu, že automat A simuluje systém so stále generujúcimi striktne regulárnymi komponentami. Z uvedého vyplýva, že na každý terminál na páskach $T_1, \ldots, T_m$ pripadne najviac m pomocných symbolov. Z čoho vyplýva platnosť dokazovaného tvrdenia. □

Tvrdenie 4.1.19. *Automat A jeden krok svojej druhej fázy realizuje v čase $O(n)$.*

Dôkaz. Jeden krok druhej fázy výpočtu automatu A sa skladá z dvoch fáz. Prvá fáza kroku trvá $O(2(m+1)n)$, keďže na páskach $T_1, \ldots, T_m$ sa nachádza najviac $(m+1)n$ znakov, vyplýva z Tvrdenia 4.1.18 a faktu, že automat počas druhej fázy výpočtu nezapisuje na pásky $T_1, \ldots, T_m$ symboly, ktoré sa na týchto páskach nenachádzali už po prvej fáze výpočtu automatu. Nech na nejakej páske na začiatku druhej fázy kroku druhej fázy výpočtu automatu A je k znakov. Duplikovanie masiek trvá maximálne $O(4k)$ času, prípad kedy sa duplikujú všetky znaky pásky. Spracovávanie masiek môžeme rozdeliť na dve časti, a to keď hlava pásky sa hýbe smerom doľava a keď sa hýbe smerom doprava. Keď sa hýbe doprava tak automat vykoná $O(k)$ krokov, keď sa hýbe doľava vykoná $O(3k)$ krokov. Záverečné prekopírovanie pomocnej pásky zaberie $O(k)$ krokov. Celkovo druhá fáza kroku pre jednú pásku trvá $O(9k)$ času a teda druhá fáza kroku trvá $O(9(m+1)n)$. Teda jeden krok zaberie $O(11(m+1)n)$ času. Keďže m je konštanta, platí dokazované tvrdenie. □

Tvrdenie 4.1.20. *Automat A vo svojej tretej fáze pracuje v čase $O(n)$.*

Dôkaz. Automat A na začiatku svojej tretej fázy, presunie hlavu pásky na začiatok pásky, čo vyžaduje $O(n)$ krokov, vyplýva z Tvrdenia 4.1.15. Samotné porovnávanie vstupného slova a slova generovaného systémom vyžaduje tiež $O(n)$ krokov výpočtu automatu. Teda dokazované tvrdenie platí. □

Tvrdenie 4.1.21. *Automat A pracuje v čase $O(n)$.*

Dôkaz. Predpokladáme, že simulovaný systém pracuje v čase $O(n)$, kde n je dĺžka slova generovaného systémom. V prvej fáze automat pracuje v čase $O(n)$, vyplýva z Tvrdenia 4.1.17. Z Tvrdenia 4.1.19 a konštrukcie automatu A vyplýva, že vo svojej druhej fáze automat pracuje v čase $O(mn)$. A tretia fáza výpočtu automatu trvá $O(n)$ času, Tvrdenie 4.1.20. Z toho vyplýva, že automat pracuje v čase $O((m+2)n)$, z toho vyplýva dokazované tvrdenie. □

Z tvrdení 4.1.16, 4.1.21 vyplýva $\mathcal{L}(PCGSgSREG) \subseteq NTIME(n)$. □

5 Záver

Predkladaná práca splnila svoj cieľ. Ukázali sme, že predpoklad striktnosti na regulárne gramatiky v komponentách $PCGS$ je dôležitý. Veta 3.1 hovorí, že $PCGSSREG$ stupňa m sú slabšie ako $PCGSREG$ stupňa m, pre $m \geq 3$. A Veta 4.1 hovorí, že $PCGS$ so stále generujúcimi striktne regulárnymi komponentami je možné simulovať Turingovým strojom v lineárnom čase. Možnými rozšíreniami tejto práce je dokázanie Vety 3.1 pre $m = 2$ a dôkaz Vety 4.1 pre $PCGSSREG$. Model paralelných kooperujúcich systémov gramatík je zaujímavý a už pri práci s jeho najjednoduchšou variantou $PCGSSREG$ sa ukazuje sila tohto modelu a elegencia jeho vlastností.

Literatúra

[HKK94] Juraj Hromkovič, Jarkko Kari, and Lila Kari. Some hierarchies for the communication complexity measures of cooperating grammar systems. *Theoretical Computer Science*, 127:123–147, 1994.

[PS89] Gh. Paun and Lila Santean. Parallel communicating grammar systems: The regular case. *Annals of the University of Bucharest, Mathematics-Informatics Series*, 38(2):55–63, 1989.

[Rov00] Branislav Rovan. Teória paralelných výpočtov. Elektronické materiály spísané poslucháčmi fakulty FMFI UK, Fakulta matematiky, fyzika a informatiky, Univerzita Komenského, Bratislava, 2000. Poznámky k prednáške Teória paralelných výpočtov, prednášajúci Prof. RNDr. Branislav Rovan Phd.

[RS01] Branislav Rovan and Marián Slašťan. Eliminating communication by parallel rewriting. *LNCS 2295 Springer*, pages 369–378, 2001.

[SK92] Lila Santean and Jarkko Kari. The impact of the number of cooperating grammars on the generative power. *Theoretical Computer Science*, 98:249–262, 1992.

[Sla00] Marián Slašťan. Parallel communicating grammar systems. Master thesis, Faculty of Mathematics and Physics, Comenius University, Bratislava, Department of Computer Sience, 2000.