Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Formálne výpočtové modely s bezdrôtovou komunikáciou

(diplomová práca)

Pavol Mravec

Vedúci: RNDr. Dana Pardubská, PhD.

Bratislava, 2007

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím citovaných zdrojov.

. . . . . .

Abstrakt

V diplomovej práci preskúmame výpočtový model so špeciálnym typom komunikácie, ktorý je istou analógiou bezdrôtovej komunikácie. Paralelne pracujúcim procesom pridáme tzv. kanálovú pásku, ktorej obsah bude určovať kanál, na ktorom budú môcť jednotlivé procesy nachádzajúce sa vo výpočte vysielať určité informácie ostatným procesom.

Charakterizujeme priestorovo ohraničené deterministické ako aj nedeterministické siete strojov. Taktiež ukážeme ako schopnosť komunikácie zmení silu konečných automatov. Porovnáme taktiež náš model s iným druhom paralelizmu a to s istým typom synchronizovaného alternovania.

Kľúčové slová: Turingove stroje, paralelizmus, komunikácia formou komunikačných kanálov

Obsah

Kapitola 1

Úvod

Teória výpočtových modelov ekvivalentných súčasným počítačom sa rozvinula hlavne v druhej polovici 20. storočia. Najprv sa skúmali modely, ktoré si za svoj cieľ kládli zistiť, ktoré problémy sú a ktoré nie sú algoritmicky riešiteľné. Všetky sekvenčné modely algoritmickej vypočítateľnosti napr. Turingove stroje, Minského registrové stroje, rekurzívne funkcie alebo λ-kalkul sa ukázali byť navzájom ekvivalentné. Takže sa istým spôsobom charakterizovala sila súčasných počítačov.

Neskôr sa začala skúmať aj veľkosť prostriedkov – napr. času a pamäte, ktoré sú potrebné na vyriešenie daného problému. Sekvenčné stroje sa podarilo usporiadať do jednotlivých hierarchií v závislosti na tom koľko času a pamäte potrebujú na svoj výpočet. Niektoré problémy sa ukázali byť ťažko na sekvenčných strojoch riešiteľné, pretože ich časové resp. pamäťové nároky boli priveľké.

Aj preto sa začali skúmať rôzne paralelné modely, v ktorých by sa dalo využiť viac strojov bežiacich paralelne hlavne na zníženie časových nárokov. Vzniklo mnoho modelov paralelizmu. Spomeňme napríklad model, ktorý v roku 1981 predstavili Chandra, Kozen a Stockmeyer v [CKS81] – alternujúci Turingov stroj (ATM). ATM patrí medzi nedeterministické modely a spojil nedeterminizmus s paralelizmom. ATM na rozdiel od klasického nedeterministického Turingovho stroja má dva druhy stavov – existenčné stavy a univerzálne stavy. Z obidvoch druhov stavov môže existovať viacero ciest, ktorými sa výpočet môže uberať. V existenčnom stave sa stroj nedeterministicky rozhodne, ktorou cestou pôjde. Stačí, aby existovala jedna cesta z

existenčného stavu taká, ktorá skončí v akceptačnom stave. V univerzálnych stavoch sa stroj rozvetví na viacero svojich kópií – každá z nich bude pokračovať paralelne inou časťou výpočtu a každá z nich musí akceptovať vstup na to, aby akceptoval aj ATM. ATM môže počas svojho výpočtu prechádzať z jedného druhu stavu do druhého t.j. alternovať.

Ukázalo sa, že ATM nevie rozpoznať iné jazyky ako Turingov stroj, avšak alternovanie posunulo ATM o jedno miesto v časo-priestorovej hierarchii strojov oproti Turingovym strojom samozrejme v prospech ATM.

Čisté alternovanie nedovoľuje svojim procesom spolu komunikovať po tom, čo sa rozdelia. Vzniklo niekoľko rôznych možností, akým spôsobom umožniť komunikáciu paralelne bežiacim procesom. Synchronizovanie paralelných procesov takým spôsobom, že paralelné procesy si za určitých okolností nedeterministicky zvolia rovnaký krok z nejakého existenčného stavu, bolo navrhnuté Wiedermannom v [Wie89].

Iný druh synchronizácie bol preskúmaný Slobodovou v [Slo88] a [Slo92]. V tomto type synchronizácie si ľubovoľný proces vie niekoľko krát počas svojho výpočtu nedeterministicky určiť tzv. synchronizačný prvok. V okamihu, keď si nejaký synchronizačný prvok zvolí, zastane a počká, kým si všetky ostatné paralelne bežiace procesy nejaký synchronizačný prvok nezvolia taktiež. Výpočet môže úspešne pokračovať iba v prípade, že si všetky procesy zvolili ten istý prvok alebo ak svoj vstup akceptovali.

V [WP05a] Pardubská a Wiedermann predstavili nový model paralelnej komunikácie Turingovych strojov – (bezdrôtové) siete strojov. V tomto modeli sú jednotlivé procesy vybavené okrem pracovných pások aj tzv. kanálovou páskou. Obsah kanálovej pásky určuje číslo kanála, na ktorom môže daný proces vysielať resp. prijímať správy od ostatných procesov naladených na ten istý kanál. Obsah tejto pásky môže proces kedykoľvek zmeniť a preladiť sa tak na iný kanál a komunikovať tak s inými procesmi. Tento model bol čisto deterministický bez existenčných stavov.

V [WP06] Pardubská a Wiedermann ukázali, že tento nový model vie simulovať alternujúce Turingove stroje v asymptoticky rovnakom čase a priestore a naopak, že ATM vedia siete strojov simulovať tiež v tej istej časovej aj priestorovej zložitosti. Takto sa podarilo získať rovnako silný deterministický model, akým sú ATM.

V tejto diplomovej práci podrobnejšie preskúmame tento nový model sietí strojov. V prvej kapitole formálne definujeme túto sieť strojov a zavedieme

skúmané miery zložitostí – čas, pracovný priestor, kanálový priestor a paralelizmus.

V ďalšej kapitole preskúmame rôzne vlastnosti sietí strojov. Uvedieme ako ovplyvňuje akceptačné kritérium silu strojov. Potom sa zameriame na konečné automaty, ktorým pridáme schopnosť komunikácie. Neskôr charakterizujeme priestorovo ohraničené siete strojov. Následne ukážeme vzťah nášho modelu s tzv. globálne deterministickými synchronizovanými alternujúcimi strojmi a nakoniec sa pokúsime zistiť ako ovplyvní kanálová páska silu sietí strojov.

V poslednej kapitole pridáme do nášho modelu nedeterminizmus a ukážeme, že nedeterministický model má mnohé vlastnosti podobné ako pôvodný deterministický model.

Kapitola 2

Definície modelu a pojmov

2.1 Sieť strojov

Ako definíciu modelu zoberieme definíciu z [WP05b] a z [WP06], ktorú rozšírime o väčší počet hláv na vstupnej páske.

Definícia 2.1. *Sieť k-hlavových t-páskových paralelných strojov (WPM) s jednou vstupnou a jednou kanálovou páskou je systém*

$$M = (k, t, Q, R, \Sigma, \Gamma, \Delta, q_0, r_0, q_{accept}, q_{reject})$$

kde

- *k je počet čítacích hláv na vstupnej páske*
- *t je počet pracovných pások*
- *$Q \times R$ je konečná množina stavov*
  - *Q je množina pracovných stavov, $q_0 \in Q$ je počiatočný stav*
  - *R je množina komunikačných stavov, ktorá obsahuje tri špeciálne stavy: počiatočný komunikačný stav r_0, akceptujúci stav q_{accept} a zamietajúci stav q_{reject}*
- *Σ je konečná vstupná abeceda ($¢ \notin \Sigma$ je ľavá zarážka, $\$ \notin \Sigma$ je pravá zarážka)*

- Γ *je konečná pracovná abeceda (*$\# \in \Gamma$ *je prázdny symbol,* $\$, ¢ \notin \Gamma$*)*
- $\Delta \subseteq Q \times R \times (\Sigma \cup \{\$, ¢\})^k \times (\Gamma \cup \{\$, ¢\})^{t+1} \times Q \times R \times (\Gamma - \{\#\})^{t+1} \times \{-1, 0, 1\}^{k+t+1}$ *je prechodová relácia*

Pokiaľ nevznikne možnosť viacznačnosti s inými typmi strojov, budeme sieť strojov volať skrátene *stroj*.

Stroj má jednu iba čítaciu vstupnú pásku, na ktorej má k hláv. Táto páska obsahuje postupne zľava ľavú zarážku ¢, vstupné slovo a nakoniec pravú zarážku \$. Stroj M má ďalej t pracovných pások a jednu tzv. kanálovú pásku, na ktoré vie aj zapisovať. Tieto pásky sú jednosmerne nekonečné smerom doprava. Na začiatku sú okrem najľavejšej pozície vyplnené prázdnym symbolom #, najľavejšia pozícia je vyplnená ľavou zarážkou ¢. Pozície jednotlivých hláv budeme zľava číslovať od nuly.

Prvkom relácie Δ hovoríme prechody stroja M. Nech

$$\delta = (q, r, x_1, \ldots, x_k, a_1, \ldots, a_t, k, q', r', a'_1, \ldots, a'_t, k', v_1, \ldots, v_k, p_1, \ldots, p_t, c)$$

je prechod stroja M. Tento prechod určuje, že stroj M, ktorý sa nachádza v pracovnom stave q, komunikačnom stave r, jeho k vstupných hláv číta zo vstupnej pásky symboly postupne $x_1, \ldots, x_k$, jeho t pracovných hláv číta postupne symboly $a_1, \ldots, a_t$ z pracovných pások $1, \ldots, t$ a jeho kanálová hlava číta k z kanálovej pásky, prejde po jednom kroku do nového pracovného stavu q', nového komunikačného stavu r' (obidva sa môžu zhodovať s pôvodným stavom), na pracovné pásky $1, \ldots, t$ zapíše postupne symboly $a'_1, \ldots, a'_t$ a na kanálovú pásku zapíše k' a pohne vstupnými hlavami $1, \ldots, k$ o $v_1, \ldots, v_k$, pracovnými hlavami $1, \ldots, t$ o $p_1, \ldots, p_t$ a kanálovou hlavou o c.

Keďže $(v_1, \ldots, v_k, p_1, \ldots, p_t, c) \in \{-1, 0, 1\}^{t+k+1}$, stroj M vie v jednom kroku každou svojou hlavou na ľubovoľnej páske pohnúť o jedno políčko doľava, o jedno políčko doprava alebo zostať stáť na pôvodnom políčku.

Dohodou zakážeme prechody doľava, keď sa hlava nachádza na ¢ a zakážeme prechody doprava, keď sa hlava nachádza na \$.

Teraz si definujeme niekoľko pojmov, ktoré využijeme neskôr pri definovaní výpočtu stroja a spôsobu jeho akceptovania vstupu.

Definícia 2.2. *Konfigurácia stroja M je prvok z*

$$Q \times R \times \Sigma^* \times ((\Gamma - \{\#\})^*)^{t+1} \times \mathbb{N}^{k+t+1}$$

Konfigurácia stroja M popisuje jeho pracovný a komunikačný stav, vstup, neprázdne obsahy pracovných pások a kanálovej pásky a pozíciu jeho všetkých $k + t + 1$ hláv - vstupných, pracovných a kanálovej.

Definícia 2.3. *Hlavová konfigurácia stroja M je prvok z*

$$Q \times R \times (\Sigma \cup \{\$, ¢\})^k \times (\Gamma \cup \{\$\})^{t+1}$$

Hlavová konfigurácia stroja M popisuje jeho pracovný a komunikačný stav a obsahy políčok snímaných jednotlivými hlavami.

Hovoríme, že prechod s novým komunikačným stavom r' *vysiela* stav $r' \in R$. Na prechody kladieme jedno syntaktické obmedzenie, tzv. *pravidlo súhlasného vysielania*: Všetky prechody odpovedajúce rovnakej hlavovej konfigurácii vysielajú rovnaký komunikačný stav. Líšiť sa však môžu vo všetkých ostatných položkách – v novom pracovnom stave, v nových symboloch zapísaných na pásky ako aj v posunoch hláv.

Ďalej hovoríme, že konfigurácia je naladená na kanál $c \in \Gamma^*$, ak obsahom kanálovej pásky je naľavo od aktuálnej pozície kanálovej hlavy je c. Ak c je neprázdny reťazec, tak c hovoríme *číslo kanála/kanálové číslo*. Ľavú zarážku nepovažujeme za súčasť c. Všimnime si, že práve snímané políčko *nie je* súčasťou c. V prípade, že $c \neq \varepsilon$ a aplikujeme na konfiguráciu prechod s novým komunikačným stavom r', hovoríme, že konfigurácia *vysiela* stav r' na kanáli c.

Definícia 2.4. *Nevysielajúca konfigurácia resp. tichá konfigurácia je konfigurácia stroja vysielajúca ε.*

Tiché konfigurácie sú výhodné hlavne v prípade, keď konfigurácia nechce nič vysielať napríklad pri preladzovaní kanála alebo pri očakávaní informácie od ostatných paralelne bežiacich konfigurácií.

Definícia 2.5. *Konfigurácia β je δ-nasledovník konfigurácie α vzhľadom na prechod $\delta \in \Delta$, ak vznikla aplikovaním prechodu δ na konfiguráciu α. Píšeme $\alpha \vdash^\delta \beta$. Prechodu $\alpha \vdash^\delta \beta$ hovoríme jednoduchý krok.*

Definícia 2.6. *Terminálna konfigurácia je taká konfigurácia, ktorá nemá žiadneho nasledovníka.*

Všimnime si, že ľubovoľná konfigurácia môže mať viacero nasledovníkov. Práve táto vlastnosť umožňuje sieti strojov vykonávať paralelné výpočty.

Definujme si niekoľko pomocných projekcií. Symbolom $\perp$ označíme nedefinovanú hodnotu:

Definícia 2.7. *Funkcia* $Tuned$*:* $Q \times R \times \Sigma^* \times ((\Gamma - \{\#\})^*)^{t+1} \times \mathbb{N}^{k+t+1} \to (\Gamma - \{\#\})$ *priraďuje konfigurácii číslo jej kanála. Rozšírením na neprázdnu množinu konfigurácií* $L \neq \emptyset$ *dostaneme:*

$$Tuned(L) := \begin{cases} c & \text{ak/práve vtedy, keď } Tuned(\alpha) = c \text{ pre všetky } \alpha \in L \\ \perp & \text{v ostatných prípadoch} \end{cases}$$

Podobne definujeme funkciu $Broadcast$:

Definícia 2.8. *Funkcia* $Broadcast$*:* $Q \times R \times \Sigma^* \times ((\Gamma - \{\#\})^*)^{t+1} \times \mathbb{N}^{k+t+1} \to R$ *priraďuje konfigurácii jej vysielaný komunikačný stav po aplikovaní nejakého aplikovateľného prechodu. Vďaka pravidlu súhlasného vysielania je táto funkcia definovaná jednoznačne. Rozšírením na neprázdnu množinu konfigurácií* $L \neq \emptyset$ *dostaneme:*

$$Broadcast(L) := \begin{cases} b & \text{ak pre všetky } \alpha, \beta \in L\ Tuned(\alpha) = Tuned(\beta) \\ & \text{a zároveň } Broadcast(\alpha) = b \\ \perp & \text{v ostatných prípadoch} \end{cases}$$

Poslednou z uvedených troch projekcií je $Comm$:

Definícia 2.9. *Funkcia* $Comm$*:* $Q \times R \times \Sigma^* \times ((\Gamma - \{\#\})^*)^{t+1} \times \mathbb{N}^{k+t+1} \to R$*, ktorá priraďuje konfigurácii jej komunikačný stav.*

Nech $u, v \in R$ sú komunikačné stavy a α je konfigurácia v komunikačnom stave u. Potom symbolom $\alpha|_{u:=v}$ označíme takú konfiguráciu, ktorá vznikne nahradením komunikačného stavu u v pôvodnej konfigurácii α za komunikačný stav v.

Definícia 2.10. *Nech* L *je nejaká množina konfigurácií. Označme* $L_e \subseteq L$ *podmnožinu všetkých takých konfigurácií* L*, ktoré sú naladené na kanál* e*. Nech* $\alpha, \beta \in L$ *sú konfigurácie, pričom konfigurácia* β *je naladená na kanál* e *a nech* $\alpha \vdash^{\delta} \beta$ *je jednoduchý krok. Potom konfigurácia* γ *– tzv.* δ_{L_e}*-nasledovník konfigurácie* α *vzhľadom na prechod* δ *modifikovaný vysielaním z* L *(označujeme* $\alpha \vdash^{\delta_{L_e}} \gamma$*), je definovaný nasledovne:*

$$\gamma := \begin{cases} \beta & ak\ \forall \varphi \in L_e : Broadcast(\varphi) = \varepsilon \\ \beta|_{Comm(\beta):=b} & ak\ \forall \varphi \in L_e : Broadcast(\varphi) = b\ a\ zároveň \\ & \forall \varphi \in L_e :\ Broadcast(\varphi) \in \{b, \varepsilon\} \\ \bot & v\ ostatných\ prípadoch \end{cases}$$

Keď to zoberieme z neformálneho hľadiska, každá konfigurácia pri aplikovaní nejakého aplikovateľného prechodu z Δ vysiela na svojom naladenom kanáli svoj nový komunikačný stav. Na úspešné prijatie vysielaného stavu musí konfigurácia byť naladená na rovnaký kanál po vykonaní jednoduchého kroku ako vysielajúca konfigurácia pred vykonaním jednoduchého kroku. Ďalej požadujeme, aby jedna konfigurácia mohla na jednom kanáli počas jedného kroku vysielať iba jeden stav alebo ε. Vysielanie ε zodpovedá nevysielaniu resp. iba počúvaniu na danom kanáli. Ak nikto na danom kanáli nevysiela, nový komunikačný kanál sa nezmení – to zodpovedá prvému prípadu. Ak niekto na danom kanáli vysiela nejaký komunikačný stav, tento komunikačný stav nahradí náš nový komunikačný stav – to zodpovedá druhému prípadu. Ak sa niektoré konfigurácie pokúsia vysielať na danom kanáli rôzne komunikačné stavy, výsledok nedefinujeme – to zodpovedá tretiemu prípadu.

Všimnime si, že dovoľujeme na jednom kanáli vysielať viacerým konfiguráciám naraz, avšak tieto všetky musia vysielať rovnaký komunikačný stav.

Ďalej si všimnime, že jedna konfigurácia môže mať viacero nasledovníkov. V tomto prípade hovoríme, že konfigurácia α *vytvorí* všetky konfigurácie γ_i, pre ktoré existuje $\delta_i \in \Delta$ také, že $\alpha \vdash^{(\delta_i)_L} \gamma_i$ pre nejaké r a všetky $1 \leq i \leq r$. Toto vytváranie korešponduje s univerzálnym vetvením alternujúcich Turingovych strojov (ATM).

Definícia 2.11. *Pre ľubovoľný vstup w stroja M výpočtový graf $G(w)$ stroja M na danom vstupe je zakorenený, orientovaný, potencionálne nekonečný acyklický multigraf, ktorého vrcholy zodpovedajú konfiguráciám a hrany zodpovedajú prechodom a komunikačným linkám. Hĺbka vrchola v $G(w)$ je jeho hranová vzdialenosť od koreňa. Tento graf $G(w)$ je definovaný rekurzívne:*

1. *Počiatočná konfigurácia*

$$c_{init} = (q, e, w, \underbrace{v, \ldots, v}_{t+1}, \underbrace{0, \ldots, 0}_{k+t+1})$$

kde v je prázdny reťazec, je koreň grafu $G(w)$ v hĺbke $d = 0$.

2. *Nech C_d je množina konfigurácií grafu $G(w)$ v hĺbke $d \geq 0$. Potom pre všetky neterminálne konfigurácie $\alpha \in C_d$, množina C_{d+1} obsahuje všetkých Δ_{C_d}-nasledovníkov konfigurácie α, t.j. všetky konfigurácie vytvorené konfiguráciou α. V prípade, že niektorí Δ_{C_d}-nasledovníci nie sú definovaní, potom aj celý graf $G(w)$ je nedefinovaný.*

3. *V grafe $G(w)$ existujú dva druhy hrán medzi vrcholmi:*

   - *prechodové hrany, ktoré vedú z každej konfigurácie $c \in C_d$ do každého z jej Δ_{C_d} nasledovníkov $\gamma \in C_{d+1}$.*
   - *vysielacie hrany, ktoré vedú z každej vysielajúcej konfigurácie $\alpha \in C_d$ do každej konfigurácie $\beta \in C_{d+1}$, pre ktorú platí $Tuned(\alpha) = Tuned(\beta)$.*

Definícia 2.12. *Ľubovoľnú konfiguráciu $c \in C_d$ nazveme dosiahnuteľnou konfiguráciou v hĺbke d.*

Definícia 2.13. *Konfiguráciu $c \in C_d$ nazveme vetviacou konfiguráciou ak z nej vychádzadzajúci počet prechodových hrán (t.j. počet jej Δ_{C_d} nasledovníkov) je väčší ako jedna.*

Definícia 2.14. *Vrcholy grafu $G(w)$ stroja M bez nasledovníkov nazveme listy grafu $G(w)$.*

Všimnime si, že výpočtový graf $G(w)$ stroja M je vybudovaný úplne deterministicky – každá hĺbka d grafu $G(w)$ stroja M jednoznačne určuje všetky konfigurácie a hrany v hĺbke $d + 1$.

Ľubovoľnú konfiguráciu zodpovedajúcu nejakému vrcholu výpočtového grafu $G(w)$ stroja M budeme nazývať taktiež *proces*. Pojmy konfigurácia a proces budeme považovať za rovnaké v prípade, že sa vyskytuje v nejakom výpočtovom grafe $G(w)$ stroja M.

Definícia 2.15. *Výpočtový graf $G(w)$ stroja M nazveme výpočtový graf $G(w)$ akceptujúci (resp. zamietajúci) vstup w práve vtedy, keď spĺňa nasledovné podmienky:*

1. *Konečnosť: $G(w)$ je konečný graf.*

2. *Akceptačné kritérium (striktná verzia): všetky listy v grafe $G(w)$ sú terminálne v rovnakej hĺbke, v komunikačnom stave q_{accept} (q_{reject}) a naladené na rovnaký kanál.*

Z praktického hľadiska akceptujúce kritérium hovorí, že všetky procesy zdieľajú informáciu o tom, či všetky ostatné procesy akceptujú alebo nie. Ak sa nejaký proces rozhodne akceptovať, má istotu o tom, že akceptujú aj všetky ostatné procesy.

Definícia 2.16. *Stroj M akceptuje vstup w práve vtedy, keď výpočtový graf $G(w)$ akceptuje vstup w.*

Definícia 2.17. *Jazyk, ktorý akceptuje stroj M – označujeme ho $L(M)$, je množina všetkých reťazcov, ktoré stroj M akceptuje.*

2.2 Skúmané triedy zložitosti

V nasledujúcom odseku definujeme 4 triedy zložitosti, ktoré budeme skúmať – čas, priestor, počet využitých kanálov a paralelizmus. Všetky budeme definovať tak, ako by sme to od nich intuitívne očakávali. To nám umožní porovnávať nami definované triedy zložitosti siete strojov so všeobecne známymi triedami zložitosti iných sekvenčných ako aj paralelných tried.

Časová zložitosť bude závisieť od výšky výpočtového grafu. Paralelizmus bude závisieť od počtu vetviacich konfigurácií vo výpočte. Budeme rozlišovať priestor použitý na kanálovej páske a priestor použitý na pracovných páskach. To nám umožní porovnať akým spôsobom ovplyvňuje komunikácia silu a ostatné zložitosti nášho modelu. Pod (pracovným) priestorom budeme rozumieť maximálny počet políčok na nejakej pracovnej páske zatiaľ čo kanálová zložitosť bude definovaná ako počet políčok použitých na kanálovej páske.

Definícia 2.18. *Časová zložitosť $T(G(w))$ akceptujúceho (resp. zamietajúceho) výpočtu siete strojov na vstupe w s výpočtovým grafom $G(w)$ je definovaná ako maximálna vzdialenosť z koreňa c_{init} (počiatočnej konfigurácie) do ľubovoľnej inej konfigurácie v tomto výpočte.*

$$T(G(w)) = \max\{d(c_{init}, c) \mid c \in G(w)\}$$

Definícia 2.19 (Časová zložitosť)**.** *Časová zložitosť $T(n)$ stroja M je definovaná ako funkcia $\mathbb{N}_0 \to \mathbb{N}_0$, ktorá priradí každej dĺžke n časovú zložitosť najdlhšieho akceptujúceho výpočtu stroja M na slovách dĺžky n.*

$$T(n) = \max\{T(G(w)) \mid w \in (L(M) \cap \Sigma^n)\}$$

Definícia 2.20. *Priestorová (kanálová) zložitosť $S(c)$ $(C(c))$ konfigurácie c stroja M je definovaná ako maximum z dĺžok neprázdnych obsahov všetkých pracovných pások (dĺžka neprázdneho obsahu komunikačnej pásky) stroja M v konfigurácii c.*

Definícia 2.21. *Priestorová (kanálová) zložitosť $S(G(w))$ $(C(G(w)))$ akceptujúceho (resp. zamietajúceho) výpočtu stroja M na vstupe w s výpočtovým grafom $G(w)$ je maximum z priestorových (kanálových) zložitostí $S(c)$ $(C(c))$ zo všetkých konfigurácií v grafe $G(w)$.*

$$S(G(w)) = \max\{S(c) \mid c \in G(w)\}$$
$$C(G(w)) = \max\{C(c) \mid c \in G(w)\}$$

Definícia 2.22 (Priestor (veľkosť) kanálu)**.** *Priestorová (kanálová) zložitosť $S(n)$ $(C(n))$ stroja M je definovaná ako funkcia $\mathbb{N}_0 \to \mathbb{N}_0$, ktorá priradí každej dĺžke n maximum z priestorových (kanálových) zložitostí akceptujúcich výpočtov stroja M na slovách dĺžky n.*

$$S(n) = \max\{S(G(w)) \mid w \in (L(M) \cap \Sigma^n)\}$$
$$C(n) = \max\{C(G(w)) \mid w \in (L(M) \cap \Sigma^n)\}$$

Do priestorovej zložitosti sme nezarátali veľkosť kanálovej pásky. Na zjednodušenie označenia definujeme pojem *Celková priestorová zložitosť* ako maximum priestorovej a kanálovej zložitosti.

Definícia 2.23 (Celková priestorová zložitosť)**.** *Celková priestorová zložitosť $S'(p)$ konfigurácie, výpočtu resp. stroja je definovaná ako*

$$S'(p) = \max\{S(p), C(p)\}$$

kde p je buď konfigurácia, výpočet alebo stroj.

Na odlíšenie od celkovej priestorovej zložitosti budeme zložitostnú triedu *priestor* volať aj pracovný priestor. Poslednou z tu definovaných tried zložitostí je *paralelizmus*.

Definícia 2.24. *Zložitosť paralelizmus $P(G(w))$ akceptujúceho (resp. zamietajúceho) výpočtu $G(w)$ stroja M na vstupe w je počet vetviacich konfigurácií v grafe $G(w)$.*

Definícia 2.25 (Paralelizmus)**.** *Zložitosť paralelizmus $P(n)$ stroja M je definovaná ako funkcia $\mathbb{N}_0 \to \mathbb{N}_0$, ktorá funkcia priradí každej dĺžke n maximum z tried zložitosti paralelizmus akceptujúcich výpočtov stroja M na slovách dĺžky n.*

$$P(n) = \max\{P(G(w)) \mid w \in (L(M) \cap \Sigma^n)\}$$

Nami definovaný model je možné rozšíriť o ďalšie vlastnosti, ktoré by sa mohli taktiež skúmať. Jedná sa napríklad o pridanie niekoľkých komunikačných pások resp. nedeterminizmu do modelu.

2.3 Označenia

2.3.1 Modely

Sieť strojov s k vstupnými hlavami a t páskami definovanú v definícii 2.1 označíme $wpm(k, t)$.

$wpm(1, t)$ nazveme sieťou Turingovych strojov a označíme $wptm(t)$.

$wpm(t)$, ktorý so svojou pracovnou aj kanálovou páskou narába ako s počítadlom, nazveme sieťou počítadlových strojov a označíme $wpcm(t)$.

$wpcm(t)$, ktorý pracuje s počítadlami, pričom nepozná ich obsah – t.j. nevie priamo testovať, či sú prázdne, nazveme sieťou strojov s t slepými počítadlami a označíme $wpbm(t)$.

$wpm(k, 0)$, ktorého kanálová páska obsahuje iba jedno políčko, nazveme sieťou k-hlavových konečných automatov a označíme $wpfa(k)$.

V práci budeme skúmať aj vplyv obmedzenia na pohyb hláv. V prípade, že každá vstupná hlava sa musí v každom kroku pohnúť o jedno políčko

doprava, dostaneme tzv. *striktne jednosmernú* verziu zariadenia, budeme ju označovať prefixom $1'$ – napr. $1'wpm(k,t)$ je striktne jednosmerná sieť k-hlavových, t-páskových strojov.

V prípade, že zakážeme vstupným hlavám pohyb doľava – t.j. v každom kroku môžu zostať na mieste alebo sa pohnúť doprava, dostaneme *jednosmernú* verziu zariadenia, budeme ju označovať prefixom 1.

Obojsmerné zariadenia nebudeme označovať žiadnym prefixom alebo im na zdôraznenie obojsmernosti pridáme prefix 2.

Triedu jazykov rozpoznávaných nejakým typom zariadenia budeme označovať rovnakými avšak veľkými písmenami – napr. $2WPFA(k)$ je trieda jazykov rozpoznávaných sieťou obojsmerných k-hlavových konečných automatov.

2.3.2 Zložitostné triedy

Triedu jazykov rozpoznávaných zariadením s ohraničenou zložitosťou označíme tak, že jej pridáme sufix danej zložitosti. Sufixy jednotlivých zložitostí:

- Čas – $TIME$
- Priestor – $WSPACE$
- Celkový priestor – $SPACE$
- Kanálová zložitosť – $CHANNEL$

Napríklad $WPTM(t)SPACE(S(n))$ je trieda jazykov akceptovaných sieťou Turingovych strojov s t-páskami s celkovou priestorovou zložitosťou $S(n)$.

Kapitola 3

Vlastnosti modelu

V tejto kapitole sa budeme venovať viacerým vlastnostiam modelu. Bližšie preskúmame voľnejšie akceptačné kritérium ako sme v 2.15 a ukážeme, že takto zvoľnené akceptačné kritérium neobmedzí silu zariadenia. Preskúmame taktiež niektoré striktne jednosmerné zariadenia. Ukážeme, že konečným automatom stačí pridať jedno slepé počítadlo, aby vedelo akceptovať ľubovoľný rekurzívne vyčísliteľný jazyk. Ďalej sa pokúsime obmedziť paralelnú, priestorovú ako aj kanálovú zložitosť zariadení a zistiť akým spôsobom toto obmedzenie zmení výpočtovú silu zariadenia ako aj ostatné zložitostné triedy. Porovnáme tiež náš model s iným typom paralelných zariadení, konkrétne s globálne deterministickými alternujúcimi strojmi.

3.1 Vplyv akceptačného kritéria na silu modelu

Akceptačné kritérium definované v 2.15 prikazuje procesom, aby do akceptačného/zamietajúceho stavu prešli naraz a aby nemohli pokračovať vo výpočte. Dôvodom, prečo sme ho takto definovali je, že každý proces vie, že všetky ostatné procesy vstup akceptovali a nemusí ďalej pokračovať vo výpočte. Pri konštruovaní siete strojov sa nám však môže stať, že nejaký proces už bude chcieť akceptovať skôr ako ostatné. Tento bude musieť počkať na ostatné procesy a čo je dôležitejšie, musí sa s nimi synchronizovať, aby do akceptačného stavu prešiel naraz spolu s nimi. Preto sa nám môže hodiť iná definícia akceptačného kritéria, ktorou nahradíme definíciu v 2.15:

Definícia 3.1 (Voľné akceptačné kritérium)**.** *Všetky listy v grafe $G(w)$ sú v komunikačnom stave q_{accept}. Zároveň zakážeme v Δ funkcii zariadenia prechody zo stavu q_{accept} do ľubovoľného iného stavu.*

Vynechali sme podmienky o rovnakom kanáli a rovnakej hĺbke – t.j. každý proces vo výpočte môže v ľubovoľnom kroku akceptovať prechodom do akceptačného stavu avšak v tomto akceptačnom stave musí ostať do konca výpočtu – nesmie zmeniť názor na akceptovanie, ak sa už raz rozhodol. Podobne môžeme definovať aj *voľné zamietajúce kritérium*.

Veta 3.1 (O voľnom akceptačnom kritériu)**.** *Pre každú sieť strojov M akceptujúcu voľným akceptačným kritériom v čase $T(N)$, priestore $S(N)$ a kanálovej zložitosti $C(N)$ existuje ekvivalentná sieť strojov M' akceptujúca jazyk $L(M)$ striktným akceptačným kritériom v čase $O(T(N))$, priestore $S(N)$ a kanálovej zložitosti $C(N)$.*

Dôkaz. Základná myšlienka dôkazu je nasledovná: M' vytvorí na každom vo výpočte použitom kanáli nový proces tzv. „observer“. Observer bude počúvať na svojom kanáli, či všetky „pracovné“ procesy, ktoré sú naladené na daný kanál chcú skončiť (akceptovať resp. zamietnuť) alebo chcú pokračovať. Pracovné procesy budú korešpondovať s procesmi siete M a každý z nich bude simulovať výpočet jedného procesu siete M. Na začiatku výpočtu sa vytvorí ešte tzv. „centrálny proces“, ktorý bude komunikovať s observermi a bude mať prehľad o tom, či ešte nejaký pracovný proces počíta alebo už môže začať so synchronizáciou všetkých procesov.

Označenie: Ak observer o je naladený na kanáli $c = wa$, kde $w \in (\Gamma - \#)^*$ a $a \in (\Gamma - \#)$, potom observera p naladeného na kanál w nazveme *rodičom* observera o. V prípade, že $w = \varepsilon$, rodičom observera o bude centrálny proces. Observerov naladených na ľubovoľný kanál wb, kde $b \in (\Gamma - \#)$ budeme volať *súrodencami* observera o. Observera o spolu so všetkými jeho súrodencami nazveme *deťmi* observera p.

Všimnime si, že observeri (a tým zároveň kanály) sú logicky usporiadaní do stromovej štruktúry, ktorej koreňom je centrálny proces. Z technických príčin sa môže na jednom kanáli vyskytnúť viacero observerov, títo však budú tesne po svojom vytvorení navzájom zameniteľní a budú vykonávať rovnakú činnosť.

Najprv predpokladajme, že už na každom kanáli, ktorý je použitý vo výpočte, počúva nejaký observer. Zároveň predpokladajme, že každý observer

si v stave pamätá, či má nejaké „deti“ a paritu „vzdialenosti“ od koreňa (t.j. paritu dĺžky svojho kanála). M' bude pracovať rovnako ako M s rovnakými stavmi, abecedami a páskami. V určitom intervale pracovné procesy siete M' odsimulujú jeden krok výpočtu M. Následne budú pracovné procesy M' komunikovať s observermi.

Pracovné procesy budú pracovať v cykle. V prvom kroku tohoto cyklu každý pracovný proces, ktorý ešte chce vo výpočte pokračovať, oznámi vyslaním príslušného komunikačného stavu svojmu observeru, že chce pokračovať. Následne procesy, ktoré chcú posunúť svoju kanálovú hlavu doprava oznámia observeru, že tak chcú urobiť. V treťom a zároveň poslednom kroku cyklu odsimulujú jeden krok svojho prislúchajúceho procesu siete M.

Keď observer o prijme aspoň jednu správu, vie, že ešte výpočet nemôže skončiť. V prípade, že tomu tak nie je, observer o vie, že všetky pracovné procesy naladené na tom istom kanáli v danom kroku už chcú skončiť. Taktiež vie, či niektorý z jeho pracovných procesov vytvoril proces na kanáli patriacom nejakému observeru p, ktorý je v hierarchii observerov dieťaťom observera o. Čiže observer o si môže obnoviť informáciu pamätanú v svojom stave o tom či je alebo nie je listom v hierarchii observerov.

Centrálny proces bude počúvať na špeciálnom kanáli, pričom nebude hýbať svojimi hlavami a kanálovú hlavu bude mať počas celého výpočtu na druhom políčku – t.j. dĺžka jeho kanála bude stále 1.

Observeri budú podobne ako pracovné procesy pracovať v cykle. Okrem komunikácie so svojimi pracovnými procesmi budú komunikovať aj so svojim rodičom. V prvom kroku cyklu sa každý observer o preladí na kanál svojho rodiča p (observer o to vie urobiť v jednom kroku posunom kanálovej hlavy doľava). V prípade, že nejaký pracovný proces na jeho sledovanom kanáli chce ešte pokračovať vo výpočte, zakričí o túto informáciu rodičovi (a zároveň všetkým svojim súrodencom naladeným v tom okamihu na kanál svojho rodiča). V prípade, že nikto nekričal (komunikačný stav observera p ostane na ε), zapamätá si observer p túto informáciu v svojom stave. Následne sa o preladí späť na svoj kanál.

V druhom kroku svojho cyklu bude každý observer o počúvať na svojom pridelenom kanáli informácie vysielané od svojich pracovných procesov. Tento druhý krok pracovného cyklu observerov bude samozrejme synchronizovaný s prvým krokom pracovného cyklu pracovných procesov.

Z technických príčin prvý krok rozdelíme na dva podkroky. V prvom pod-

kroku si všetci observeri s nepárnou dĺžkou svojho kanála vymenia informácie so svojimi rodičmi a v druhom podkroku si observeri s párnou dĺžkou kanála vymenia informácie so svojimi rodičmi. Takto zabezpečíme, aby observeri boli naladení na svoj kanál, keď im chcú ich deti poslať informácie a zároveň, aby každý observer poslal svoje informácie svojmu rodiču.

Všetky observeri budú teda pracovať nasledovne v cykle:

1. (a) Všetky observeri s nepárnou dĺžkou svojho kanála si vymenia informácie so svojim rodičom
   (b) Všetky observeri s párnou dĺžkou svojho kanála si vymenia informácie so svojim rodičom
2. Všetky observeri počúvajú na svojom kanáli, či ešte nejaký pracovný proces chce pokračovať vo výpočte

Centrálny proces zároveň bude posielať „ping“ správu svojim potomkom. Každý observer prepošle „ping“ svojim potomkom iba v prípade, že všetky jeho sledované pracovné procesy už chcú skončiť. V prípade, že „ping“ dorazí do nejakého listu, tento list odpovie späť správou „echo“ svojmu rodičovi, ktorý túto správu bude preposielať svojim predkom. Každý observer prepošle „echo“ iba v prípade, že stále všetky pracovné procesy, ktoré sleduje chcú skončiť. Z technických dôvodov observer svojmu rodičovi vždy vyšle správu „not-echo“ ak nechce poslať „echo“, inak nezakričí nič. Takže rodič vie, či aspoň jeden z jeho synov vyslal „not-echo“, alebo všetky chcú správu „echo“ preposlať. Podobne budú observeri prenášať aj ostatné správy svojim rodičom.

Keď centrálny proces obdrží „echo“ správu od všetkých synov, pošle podobným spôsobom svojim potomkom správu „init-collapse“. Keď túto správu obdržia listy, pošlú svojim pracovným procesom správu „do-collapse“ a spolu so svojimi pracovnými procesmi sa preladia na kanál svojho rodiča a pošlú správu „do-collapse“ všetkým procesom naladeným na tento kanál (z technických príčin procesy, ktoré správu do-collapse nechcú poslať vyšlú správu „dont-collapse“, aby vedeli zistiť, či všetky procesy chcú pokračovať).

Takto sa po určitom čase všetky procesy naraz naladia na kanál k centrálneho procesu. V tomto okamihu všetky procesy, ktoré chcú zamietnuť vstup oznámia na tomto kanáli k správu „chcem zamietnuť“. Ak nikto nekričal, prejdú všetky procesy do q_{accept}, v opačnom prípade prejdú do stavu q_{reject}.

Ešte popíšeme ako sa vytvoria observeri na každom kanáli. Na začiatku počiatočný proces vytvorí centrálny proces a zapamätá si, že je listom. V priebehu výpočtu každý list, keď vytvorí proces naladený na dlhší kanál, odovzdá tomuto procesu informáciu, že od tohoto kroku je listom on. Zároveň vytvorí na tomto dlhšom kanáli nového observera. Ešte pred vytvorením nových procesov naladených na dlhší kanál oznámi skutočnosť, že nejaké procesy chce vytvárať svojmu observeru, takže aj jemu prislúchajúci observer už vie, že od tohoto kroku nie je listom.

Dôkaz správnosti konštrukcie je zjavný z vyššie uvedeného popisu. Jediná problematická vec je ukázať, že keď centrálny proces prijme „echo“ správu od všetkých svojich synov, tak už neexistuje žiadny proces, ktorý by chcel pokračovať vo výpočte. Táto vlastnosť sa dá ukázať indukciou pre každý observer vzhľadom na hĺbku podstromu, ktorý je pod ním zavesený.

Dôležité je si uvedomiť, že všetky observeri *naraz* zistia stav o svojich pracovných procesoch – t.j. pracovné procesy vždy čakajú so simuláciou, kým si observeri odkomunikujú informácie so svojimi rodičmi a synmi.

Všimnime si, že ak nejaký podstrom stíchol, tak tento podstrom sa môže oživiť už iba cez svoj koreň. To platí z toho dôvodu, že ostatné procesy, keď sa chcú preladiť na nejaký kanál v tomto podstrome, musia sa preladiť najprv na kanál koreňa tohoto podstromu, ktorý stopne správu „echo“, keď cez neho prejde nejaká komunikácia do jeho podstromu.

Jeden krok výpočtu siete M vie sieť M' odsimulovať na konštantný počet krokov – pracovné procesy odsimulujú jeden krok, oznámia výsledok svojmu observeru, ten odkomunikuje správy so svojim rodičom a svojimi synmi. Posledná synchronizačná fáza prebehne zjavne v čase $O(C(N)) = O(T(N))$. Procesy siete M' pracujú v rovnakej kanálovej aj pracovnej priestorovej zložitosti ako procesy siete M. □

Všimnime si, že sme dokázali o niečo silnejšiu vec ako sme pôvodne požadovali. Nemusíme požadovať aby každý proces siete M v stave q_{accept} už nemohol zmeniť svoj stav. Stačí požadovať, aby tento proces počúval na svojom kanáli, či nejaký iný proces na kanáli nevysiela a zmeniť svoje akceptačné rozhodnutie môže iba v prípade, že nejaký proces na jeho kanáli vysiela (t.j. zobudí ho).

3.2 Konečné automaty s komunikáciou

V nasledujúcich častiach bližšie preskúmame konečné automaty, ktorým pridáme schopnosť komunikovať pomocou kanálov. Preskúmame striktne jednosmerné konečné automaty, konečné automaty so slepým počítadlom ako kanálovou páskou a obojsmerné konečné automaty s obmedzením na počet vetviacich konfigurácií vo výpočte.

3.2.1 Striktná jednosmernosť

V časti 2.3.1 sme definovali obmedzenie sietí strojov, ktoré sme nazvali striktná jednosmernosť. Striktná jednosmernosť obmedzuje pohyb všetkých vstupných hláv siete strojov v každom kroku iba na pohyb doprava. V tejto časti preskúmame niektoré striktne jednosmerné modely.

Ako ukážeme nižšie striktne jednosmerným k-hlavovým konečným automatom paralelizmus nepomôže. Keďže kanálová páska siete konečných automatov obsahuje iba jedno políčko, považujme toto políčko za ďalšiu zložku vnútorného stavu siete konečných automatov.

Viachlavové striktne jednosmerné konečné automaty nemá význam uvažovať, pretože ich hlavy budú v ľubovoľnom okamihu čítať jedno a to isté políčko, ktoré môže ekvivalentný jednohlavový konečný automat čítať svojou jednou hlavou.

Veta 3.2. *$1'WPFA(1) = R$, kde R označuje triedu regulárnych jazykov.*

Dôkaz.

- Inklúzia „$\supseteq$“ platí triviálne, pretože trieda naľavo je iba zovšeobecnením triedy vpravo. Takže simulácia prebehne tak, že všeobecnejší automat nevyužije svoje vlastnosti, ktoré má navyše.
- $1'WPFA(1) \subseteq R$: Nech $M \in 1'wpfa(1)$, zostrojme konečný automat A taký, že $L(A) = L(M)$. Počet rôznych konfigurácií v ľubovoľnom výpočtovom kroku siete M na vstup w je maximálne $C = |Q \times R \times \Gamma|$. C je konečné číslo, takže konečný automat A si bude v stave pamätať všetky rôzne existujúce konfigurácie siete M. Keďže každá hlava číta rovnaké písmenko, automat A bude mať pre každé písmenko $p \in \Sigma$ vo

svojej prechodovej funkcii „zadrôtované“ ako sa zmenia konfigurácie po prečítaní p. Akceptačné stavy automatu A budú tie, v ktorých všetky konfigurácie sú akceptačné. Dôkaz, že $L(A) = L(M)$ je triviálny a vyplýva z konštrukcie. □

Vidíme, že komunikácia nepomohla striktne jednosmerným jednohlavovým konečným automatom.

Skúsme sa však pozrieť na triedu striktne jednosmerných sietí automatov s jedným slepým počítadlom – $1'WPBM(1)$, použitým namiesto kanálovej pásky.

Lema 3.3. $\{a^n b^n c^n \mid n \geq 0\} \in 1'WPBM(1)$

Dôkaz. Zostrojme striktne jednosmernú sieť konečných automatov s jedným slepým počítadlom A použitým namiesto kanálovej pásky takú, že $L := \{a^n b^n c^n \mid n \geq 0\} = L(A)$. Sieť A sa v prvom kroku rozdelí na 4 procesy – u, v, w a x. Procesy u a v budú zvyšovať svoje počítadlo vždy, keď prečítajú písmeno a. Keď sa procesy dostanú na vstupe na prvé písmeno b, začne proces v svoje počítadlo znižovať, proces w svoje počítadlo zvyšovať a proces u nerobí s počítadlom nič. Keď sa procesy dostanú na písmenko c (označme túto pozíciu vo výpočte ako P_1), začnú procesy u a w svoje počítadlá znižovať, pričom v nerobí nič so svojim počítadlom. Keď všetky procesy prídu na pravú zarážku u zakričí na svojom kanáli q_{accept}. Ostatné procesy dostanú tento stav iba vtedy, keď sú všetky tri naladené na jednom kanáli. Proces x počas výpočtu kontroluje, či má vstup tvar $a^n b^m c^k$ pre $n, m, k \geq 0$.

Dôkaz správnosti konštrukcie: Rozoberieme niekoľko prípadov. Nech vstupné slovo w má tvar $a^n b^m c^k$ pre $n, m, k \geq 0$.

V prípade, že $n < m$, proces v sa pokúsi znížiť svoje počítadlo pod nulu, čo má za následok zamietnutie slova.

V prípade, že $n > m$, proces v od pozície P_1 bude mať svoje počítadlo nenulové. Taktiež na pozícii P_1 budú mať procesy u a w svoje počítadlá nastavené na inej hodnote (proces u ho bude mať na n a proces w na m). Pre ľubovoľný počet písmen c sa teda nemôže stať, aby w a u dostali počítadlá na rovnakú hodnotu, teda na konci nemôže w akceptovať slovo.

Takže $n = m$. V tomto prípade proces v na pozícii P_1 má svoje počítadlo nulové a procesy u a w ho majú na stavené na n. Je zrejmé, že počet písmen c musí byť n na to, aby si procesy u a w vynulovali svoje počítadlá a mohli akceptovať vstup. □

Striktne jednosmerným automatom so slepým počítadlom pomohla komunikácia až natoľko, že vedia akceptovať dokonca jazyk, ktorý nie je bezkontextový.

3.2.2 Konečné automaty so slepými počítadlami

V tejto časti sa budeme zaoberať sieťami automatov so slepými počítadlami. Ukážeme, že už len jedno slepé počítadlo stačí na to, aby sme mohli odsimulovať ľubovoľný rekurzívne vyčísliteľný jazyk.

Lema 3.4. *Pre každý deterministický Turingov stroj M pracujúci v čase $T(N)$ a pamäti $S(N)$ existuje sieť automatov A s jedným slepým počítadlom použitým ako číslo kanála pracujúca v čase $O(T(N))$ bez pracovných pások s maximálnou hodnotou počítadla vo výpočte $S(N)$.*

Dôkaz. Sieť automatov A bude fungovať tak, že pre každé políčko a_i pracovnej pásky stroja M vytvoríme jeden proces p_i, ktorý bude naladený na kanál i – t.j. vo svojom počítadle si bude pamätať hodnotu i.

Všimnime si, že proces p_i vie v jednom kroku komunikovať s procesmi p_{i-1} a p_{i+1} (v prípade, že existujú). Keď proces chce p_i poslať správu procesu p_{i-1}, dekrementuje si hodnotu svojho počítadla – t.j. preladí svoj kanál na kanál procesu p_{i-1}, pošle na tomto kanáli správu a následne sa preladí späť na kanál i. Analogicky vie proces p_i komunikovať s procesom p_{i+1}. Jedinou podmienkou úspešného prenosu je, že procesy p_{i-1} a p_{i+1} musia túto správu očakávať – t.j. nemôžu oni sami komunikovať s inými procesmi, aby sa náhodou nepreladili na iný kanál a správu by nedostali.

Počas celého výpočtu sieť A bude mať 3 špeciálne procesy: p_0 – ľavý kraj, p_t – proces vlastniaci „token“ a p_j – pravý kraj (t.j. proces naladený na najväčší kanál j z pomedzi ostatných v tom okamihu existujúcich procesov). Na začiatku výpočtu bude proces $p_0 = p_t = p_j = p_i$ inicializačný proces.

Sieť A bude fungovať tak, že bude postupne simulovať kroky stroja M. Na začiatku simulácie j-tého kroku platí nasledovný invariant: Každý proces p_i si bude pamätať v stave obsah i-tého políčka pásky stroja M tesne pred vykonaním j-tého kroku a bude mať svoju hlavu na i-tom políčku vstupu w resp. na pravej zarážke keď $i > |w|$. Nech stroj M má svoju hlavu na pozícii t. Potom vlastníkom tokenu bude proces p_t – t.j. bude si pamätať v svojom stave stav stroja M.

Simulácia j-tého jednoduchého kroku stroja M sieťou A vyzerá nasledovne:

Proces p_t najprv vyhodnotí ako by sa správal stroj M – zistí ako sa zmení stav stroja M a ktorým smerom pohne svojou hlavou. V prípade, že hlava stroja M sa v j-tom kroku nepohne, simulácia môže pokračovať ďalším krokom. V prípade, že sa hlava pohne smerom doľava, proces p_t pošle nový stav stroja M procesu p_{t-1}. Novým procesom vlastniacim token bude odteraz p_{t-1}. Analogicky sa odsimuluje aj pohyb hlavy doprava.

Pri pohybe doprava môže nastať situácia, že proces p_{t+1} neexistuje. To je vtedy keď $p_j = p_t$ teda $j = t$. Túto skutočnosť teda proces p_t pozná – vie, že je najpravejší a teda vytvorí proces p_{t+1}, ktorému pošle nový stav stroja M, informáciu o tom, že je odteraz najpravejší p_{t+1} a hlavu mu nastaví na pozíciu $t+1$ prípadne na pravú zarážku ak $t \geq |w|$. Takže proces p_{t+1} pozná obsah $t+1$-vého políčka pásky stroja M.

V tomto okamihu zjavne platí invariant pre krok $j+1$.

Takto bude simulácia pokračovať až dovtedy, pokiaľ p_t nezistí, že má akceptovať vstup. Vtedy pošle správu „init-accept“ svojmu pravému susedovi, ktorý ju prepošle ďalej, až sa dostane najpravejšiemu procesu. Tento pošle správu „akceptuj“ svojmu ľavému susedovi q a preladí sa na jeho kanál. Potom obidva procesy pošlú naraz správu ľavému susedovi procesu q a preladia sa na jeho kanál, až kým správu nedostane proces p_0, ktorý ostatným procesom oznámi, že je najľavejší a všetky procesy naraz akceptujú.

Dôkaz správnosti vyššie uvedenej konštrukcie zjavne vyplýva z jej popisu. Nami zostrojená sieť potrebuje na svoj výpočet čas $O(T(N))$ a $S(N)$ rôznych kanálov bez potreby ďalšieho pracovného priestoru. □

Simuláciu opačným smerom ukážeme neskôr vo vete 3.9.

Dôsledok 3.5. *Jednosmerné jednohlavové siete strojov bez pracovných pások s jedným slepým počítadlom použitým ako číslo kanála vedia akceptovať ľubovoľný rekurzívne vyčísliteľný jazyk.*

Dôkaz. Uvedený dôsledok priamo vyplýva z dôkazu lemy 3.4, pretože sieť strojov v nej zostrojená nepotrebuje poznať dopredu žiadne ohraničenia – ani na čas ani na priestor. □

Paralelná komunikácia zdvihla silu jednosmerných automatov so slepým počítadlom až na silu Turingovych strojov, teda vedia akceptovať ľubovoľný rekurzívne vyčísliteľný jazyk.

Podobný dôsledok vieme dostať aj nasledovne: Využijeme známy fakt, že Turingov stroj vieme simulovať obojsmerným automatom s dvomi počítadlami. Tieto dve počítadlá by sme vedeli odsimulovať dvomi procesmi – každý z nich by obsluhoval jedno počítadlo a medzi sebou by komunikovali na jednom kanáli. Takýto priamočiary dôkaz však vyrobí sieť strojov s vyššou celkovou priestorovou ako aj časovou zložitosťou.

3.2.3 Jednotková paralelná zložitosť

Pod pojmom jednotková paralelná zložitosť nejakej siete M budeme rozumieť obmedzenie miery zložitosti paralelizmus také, že v ľubovoľnom výpočte siete M na ľubovoľnom vstupnom slove w sa bude môcť nachádzať maximálne jedna taká konfigurácia, ktorá sa môže rozdeliť na viacero procesov.

Lema 3.6. *Pre každú sieť M s jednotkovou paralelnou zložitosťou existuje ekvivalentná sieť M' s jednotkou paralelnou zložitosťou, v ktorej jediná vetviaca konfigurácia je počiatočná, pracujúca v rovnakej časovej, priestorovej ako aj kanálovej zložitosti.*

Dôkaz. Vetviaca konfigurácia vo výpočte sieti M na ľubovoľnom vstupe sa môže rozdeliť maximálne na konštantný počet k konfigurácií daný popisom siete M. M' bude pracovať tak, že sa hneď rozdelí na k procesov p_i. Každý proces p_i si bude v stave pamätať svoje číslo i. Na začiatku budú procesy simulovať sekvenčný výpočet siete M. Keď prídu na krok výpočtu, kde by sa mal pôvodný výpočet vetviť, proces p_i bude simulovať i-ty proces vo vetvení M. V prípade, že vetvení je menej ako k, zvyšné procesy budú pracovať rovnako ako proces p_1. M' pracuje zjavne v rovnakej časovej, priestorovej aj kanálovej zložitosti ako M.

Dôkaz uvedenej konštrukcie je zrejmý z jej popisu. □

Poznámka 3.7. Dôkaz lemy 3.6 sa dá jednoduchým spôsobom zovšeobecniť na situáciu, keď vo výpočte povolíme ľubovoľný konštantný počet vetvení k, z ktorých každé sa môže rozvetviť naraz na maximálne p vetiev. Sieť M' sa v tomto prípade rozdelí najprv na p^k procesov, z ktorých si každý pamätá svoje

číslo v svojom stave. Keď príde k deleniu, tak sa týchto p^k procesov rozdelí na p skupín, každá po p^{k-1} procesov, ktoré budú simulovať svoju vetvu. Počet vetvení je maximálne k, takže vždy bude existovať dostatok procesov, ktoré budú môcť pokračovať vo výpočte.

V nasledujúcej vete charakterizujeme siete s jednotkovou paralelnou a konštantnou celkovou priestorovou zložitosťou:

Veta 3.8. *Trieda jazykov akceptovaných sieťami s jednou vstupnou hlavou, s jednotkovou paralelnou a konštantnou celkovou priestorovou zložitosťou je* $DLOGSPACE$.

Dôkaz. Treba ukázať dve inklúzie. Najprv ukážeme inklúziu „$\subseteq$“:

Majme teda sieť M pracujúcu s jednou vstupnou hlavou a s jednotkovou paralelnou a konštantnou celkovou priestorovou zložitosťou. Táto sieť sa bez ujmy na všeobecnosti na každom vstupe w rozvetví hneď na začiatku na k procesov. Náš Turingov stroj M', ktorý bude akceptovať jazyk $L(M)$ má dostatok priestoru na svojej pracovnej páske na zapamätanie si pozície vstupnej hlavy každého z týchto k procesov - potrebuje na to pamäť $O(\log N)$.

Stroj M' bude pracovať tak, že bude postupne simulovať kroky siete M. Na začiatku každého kroku si prečíta z pracovnej pásky polohy vstupných hláv jednotlivých procesov, pozrie si aké písmenko čítajú zo vstupu, ostatné informácie si pamätá v svojom stave. Následne zistí, ako sa pohnú jednotlivé hlavy siete M, uloží si to na svoju pracovnú pásku a do svojho stavu si zapamätá stavy simulovaných procesov modifikované prípadnou komunikáciou. Následne môže pristúpiť k simulovaniu nasledujúceho kroku siete M.

Stroj M' akceptuje svoj vstup práve vtedy, keď aj všetky procesy siete M akceptujú vstup.

Správnosť vyššie uvedenej konštrukcie triviálne vyplýva z jej popisu.

Inklúzia „$\supseteq$“: Majme Turingov stroj M pracujúci v pamäti $k \log N$ pre vhodne zvolenú konštantu k. Bez ujmy na všeobecnosti tento stroj pracuje s binárnou pracovnou abecedou. Naša sieť M' bude pozostávať z niekoľkých procesov. Každý z nich si bude pozíciou svojej hlavy na vstupe pamätať nejaké prirodzené číslo. Na pamätanie si ľubovoľného čísla $0 \leq c \leq N$ využijeme jeden proces p, ktorý bude mať svoju (vstupnú) hlavu na pozícii c.

Pri simulovaní stroja M budeme potrebovať s takto pamätanými číslami robiť jednoduché operácie, ktoré si tu popíšeme. Pri každej predpokladáme,

že výsledok je menší ako veľkosť vstupného slova N. Prípadu, že tomu tak nie je, sa budeme venovať neskôr.

- operácia $+$: Nech procesy p a q si pamätajú čísla a a b. S pomocou pomocných procesov r, s nastavíme hlavu procesu t na hodnotu $a+b$. Na začiatku procesy r, s a t majú hlavu pozícii 0. V prvej fáze procesy r a t budú posúvať v každom kroku svoju hlavu doprava a proces p svoju hlavu doľava. Keď proces p dosiahne ľavú zarážku, zakričí to ostatným procesom. Vtedy procesy r a t budú mať svoju hlavu na pozícii a. V druhej fáze budú rovnako postupovať procesy s, t a q. V tomto okamihu si procesy r a s vymenili pozície hláv s procesmi p a q a proces t má svoju hlavu na pozícii $a + b$.

- operácia $\times$: Nech procesy p a q si pamätajú čísla a a b. Chceme nastaviť hlavu procesu t na hodnotu ab. Pomocnému procesu p nastavíme hlavu na pozíciu a (podobným spôsobom aký je popísaný v operácii $+$). Následne a-krát vykonáme operáciu $+$. Po každom pričítaní hodnoty b k pozícii hlavy procesu t si proces p posunie svoju vstupnú hlavu doľava. Keď dosiahne ľavú zarážku, zakričí ostatným procesom, že už môžu skončiť.

- operácia 2^s: Nech proces p má svoju hlavu na pozícii s. Proces t bude mať svoju hlavu na pozícii 1. Proces p bude posúvať svoju hlavu doľava. Pokiaľ nebude mať hlavu na ľavej zarážke, proces t si zdvojnásobí pozíciu svojej hlavy na vstupe vyššie uvedeným spôsobom. Keď proces p dosiahne ľavú zarážku, proces t bude mať svoju hlavu na hodnote 2^s. Zároveň pomocný proces r si bude svoju hlavu posúvať doprava (začne na ľavej zarážke), takže na konci bude mať svoju hlavu na pozícii s – tým nestratíme pôvodnú hodnotu s.

- operácia $\log_2 s$: Nech proces p má svoju hlavu na pozícii s. Chceme dostať hlavu procesu t na pozíciu $\log_2 s$. Proces t začne na ľavej zarážke a bude postupne posúvať svoju hlavu doľava. Po každom posunutí hlavy na nejakú pozíciu k si vypočíta hodnotu 2^k a porovná ju s pozíciou hlavy procesu p. Keď 2^k presiahne hodnotu s, proces p má svoju hlavu na hodnote $\lceil \log_2 s \rceil$.

- operácia $\frac{s}{2}$: Majme proces p, ktorý si pamätá hodnotu s. Pustíme dva procesy a a b, ktoré začnú na hodnote s a budú si dekrementovať svoju

pamätanú hodnotu – t.j. budú svoju hlavu posúvať smerom doľava. Proces b si svoju hlavu posunie raz za každé dva posuny procesu a, takže keď sa proces a dostane na ľavú zarážku, proces b bude presne v polovici čísla s.

- polovica intervalu $\langle a, b\rangle$: Majme procesy p a q, ktoré si pamätajú postupne hodnoty a a b. Najprv si vytvoríme kópie procesov p a q. Tieto kópie si budú svoju pamätanú hodnotu dekrementovať, až kým kópia procesu p nepríde na ľavú zarážku. V tomto okamihu si kópia procesu pamätá hodnotu $b-a$. Zistíme polovicu tejto hodnoty vyššie uvedeným spôsobom a následne pripočítame k tejto hodnote späť číslo a.

- k-ty bit binárneho zápisu čísla c: Majme proces p, ktorý si pamätá hodnotu c svojimi hlavami. Nájdeme si najbližšiu väčšiu mocninu čísla 2 od c – označme ju d. To urobíme napríklad tak, že budeme postupne rátať hodnoty 2^b pre b od 0 až po c, ktorú potom porovnáme s hodnotou c. Zistíme polovicu p intervalu $\langle 0, d\rangle$. Porovnáme, či je hodnota c väčšia alebo menšia p. V prípade, že $c < p$, tak prvý bit zápisu čísla c je nula a pokračujeme ďalej vo výpočte s intervalom $\langle 0, p\rangle$, inak prvý bit zápisu čísla c je jedna a ďalej pokračujeme vo výpočte s intervalom $\langle p, d\rangle$. Výpočet i-teho bitu vyzerá rovnako ako výpočet prvého bitu, avšak budeme pracovať s i-tym získaným intervalom. Pokračujeme, až kým nezistíme k-ty bit.

  Môže sa stať, že hodnota d je väčšia ako dĺžka vstupu n, avšak d môže nadobudnúť hodnotu maximálne $2n$. Hodnotu d si preto nejaký proces bude pamätať tak, že polohou hlavy si bude pamätať číslo c od 0 po n a v stave, či táto c určuje číslo z intervalu $\langle 0, n\rangle$ alebo $\langle n, 2n\rangle$.

Všimnime si, že pomocou k procesov si môžeme pamätať číslo veľkosti $c \leq n^k$, kde n je veľkosť vstupu a to tak, že si číslo c napíšeme v n-árnej sústave a každý z týchto k procesov si bude pamätať jednu cifru čísla c v n-árnej sústave. Vyššie uvedené operácie sa jednoduchým spôsobom dajú modifikovať, aby sme vedeli pracovať s číslami do veľkosti n^k.

Vráťme sa späť k simulácii stroja M. Počet všetkých jeho konfigurácií je maximálne $2^{k\lceil \log_2 n\rceil} \leq n^{k+1}$ pre dostatočne veľké n. Pre malé n si vieme celú vstupnú aj pracovnú pásku zapamätať v stave. Takže si teda sieť M' vie pamätať obsah pracovnej pásky (chápanú ako číslo c v dvojkovej sústave) s

pomocou $k+1$ procesov. Ďalej si bude v jednom procese pamätať polohu hlavy na tejto pracovnej páske a v ďalšom procese polohu vstupnej hlavy.

Sieť M' bude postupne simulovať kroky výpočtu stroja M takto: extrahuje hodnotu políčka snímaného pracovnou hlavou stroja M, vyhodnotí čo urobí stroj M a následne zapíše nejaký symbol na zapamätanú pracovnú pásku.

Extrahovanie hodnoty k-teho políčka pracovnej pásky stroja M je identické extrahovaniu k-teho najľavejšieho bitu hodnoty c, v ktorej si pamätáme celý obsah pásky stroja M.

Na prepísanie nuly na jednotku potrebujeme pripočítať k obsahu pracovnej pásky c hodnotu $2^{(a-b)}$ a na prepísanie jednotky na nulu potrebujeme od neho odpočítať hodnotu $2^{(a-b)}$, kde a je veľkosť pracovnej pásky a b je pozícia hlavy stroja M na pracovnej páske, kam chceme zapisovať. Hodnota a je $\log_2 n$, ktorú si vieme vypočítať vyššie uvedeným spôsobom ako aj všetky ďalšie potrebné operácie.

Keď sieť M' zistí, že stroj M chce akceptovať, akceptuje taktiež.

Správnosť uvedenej konštrukcie vyplýva z jej popisu. Zostrojená sieť si počas svojho výpočtu pamätá konštantný počet hodnôt – na to potrebuje konštantný počet procesov. S týmito hodnotami potrebuje robiť operácie, na ktoré potrebuje konštantný počet pomocných procesov. Všetky potrebné procesy si teda vie vytvoriť jediným vetvením už na začiatku výpočtu a teda pracuje v jednotkovej paralelnej zložitosti. □

3.3 Priestorovo obmedzené siete strojov

V nasledujúcich častiach sa pokúsime zistiť, aký vplyv majú rôzne obmedzenia priestorovej zložitosti na výpočtovú silu sietí strojov. V tejto časti charakterizujeme výpočtovú silu strojov s konštantnou priestorovou zložitosťou:

Veta 3.9. *Pre ľubovoľnú funkciu $S(n)$ platí:*

$$WPM(k,0)CHANNEL(S(n)) = \bigcup_{c} DSPACE\left(n^k c^{S(n)}\right)$$

Uvedená veta hovorí, že výpočtová sila sietí strojov bez pracovných pások a s kanálovou zložitosťou $S(n)$ je rovnaká ako výpočtová sila priestorovo obmedzených deterministických Turingovych strojov.

Dôkaz. Myšlienka dôkazu tejto vety spočíva v tom, že priestor $c^{S(n)}$ si vieme zapamätať v čísle kanála a priestor n^k si vieme zapamätať pozíciami vstupných hláv. V tejto vete treba ukázať dve inklúzie. Začneme inklúziou „$\supseteq$“ t.j. budeme sa snažiť odsimulovať prácu priestorovo ohraničeného deterministického Turingovho stroja pomocou siete strojov bez pracovných pások:

Majme teda Turingov stroj $M \in DSPACE(n^k c^{S(n)})$ pre nejakú konštantu c. Zostrojme sieť M' ekvivalentnú k stroju M. Sieť M' bude fungovať tak, že bude postupne simulovať kroky stroja M. V jednom okamihu si sieť bude nižšie uvedeným spôsobom pamätať celú konfiguráciu stroja M.

Popíšme invariant I, ktorý bude platiť v sieti M' po každom odsimulovaní jedného kroku stroja M:

Pre každé políčko i pracovnej pásky stroja M si sieť M' vytvorí jeden proces – označíme ho p_i, ktorý si bude v stave pamätať obsah svojho políčka pracovnej pásky stroja M. Pásku veľkosti $n^k c^{S(n)}$ rozdelíme na $c^{S(n)}$ častí, každá o veľkosti n^k. Procesy v časti i budú naladené na kanál i zapísaný na kanálovej páske v c-árnej sústave. Nech stroj M má svoju pracovnú hlavu nad políčkom $0 \leq i < n^k$ v časti j. Popíšme ako si budú jednotlivé procesy pamätať, ktoré políčko im prislúcha:

- Procesy v časti j si budú svojimi hlavami $h_1, \ldots, h_k$ pamätať absolútnu hodnotu d vzdialenosti im prislúchajúceho políčka a políčka i. Číslo d si budú pamätať v n-árnej sústave tak, že pozíciu hlavy h_m určuje m-tá najnevýznamnejšia cifra zápisu d v n-árnej sústave. V stave si budú pamätať, či sú naľavo alebo napravo od i.

- Procesy v častiach $0, \ldots, j-1$ si budú pamätať vzdialenosť od najpravejšieho políčka v danej časti svojimi vstupnými hlavami tak, ako je popísané vyššie. V stave si budú pamätať, že ich časť (kanál) sa nachádza naľavo od aktívnej časti (aktívneho kanála).

- Procesy v častiach $j+1, \ldots, c^{S(n)}$ si budú pamätať vzdialenosť od najľavejšieho políčka v danej časti svojimi vstupnými hlavami tak, ako je popísané vyššie. V stave si budú pamätať, že ich časť (kanál) sa nachádza napravo od aktívnej časti (aktívneho kanála).

Proces, ktorý prislúcha políčku práve snímaným hlavou stroja M nazveme aktívny proces. Aktívny proces si bude okrem obsahu jemu prislúchajúceho políčka navyše pamätať v svojom stave aktuálny stav stroja M. Časť, v ktorej sa nachádza aktívny proces nazveme aktívna časť.

Výpočet siete M' sa bude skoro po celú dobu výpočtu diať iba v aktívnej časti. Aktívna časť sa však môže zmeniť, keď stroj M presunie svoju hlavu do inej (zjavne susednej) časti.

Jeden krok výpočtu stroja M odsimulujeme nasledovne:

Aktívny proces má k dispozícii všetky potrebné informácie k určeniu toho, čo spraví stroj M. Do svojho stavu si zapamätá novú hodnotu políčka, ktoré mu prislúcha, ktorú by tam v danom kroku zapísal stroj M. Na kanáli oznámi ostatným procesom v aktívnej časti nový pracovný stav stroja M a taktiež pohyb hlavy stroja M.

V prípade, že stroj M by pohol hlavou doprava, všetkým procesom prislúchajúcim políčkam napravo od pozície hlavy sa zníži o jedna ich vzdialenosť od pozície hlavy. Preto všetky procesy v aktívnej časti (naladené na rovnaký kanál ako aktívny proces), ktoré sa nachádzajú napravo od pozície hlavy si znížia o jedna pamätanú hodnotu svojimi hlavami. Zároveň sa procesom, ktoré sa nachádzajú naľavo od pozície hlavy, zvýši ich vzdialenosť od pozície hlavy a preto si zvýšia pamätanú hodnotu svojimi hlavami. V prípade pohybu hlavou doľava sa procesy v aktívnej časti budú správať analogicky.

Popíšeme ako inkrementovať hodnotu pamätanú hlavami:

Každý proces má svoje hlavy očíslované a vie pre každú hlavu koľkú cifru zápisu pamätaného čísla c v n-árnej sústave si daná hlava svojou pozíciou pamätá. Všetky najnevýznamnejšie cifry, ktoré si pamätajú hodnotu $n-1$ treba vynulovať a prvú cifru, ktorá nie je rovná $n-1$ treba inkrementovať. Tieto operácie urobíme nasledovne:

Proces posunie doprava každú svoju hlavu a pozrie si, ktoré hlavy sú na pravej zarážke. Všetky najnevýznamnejšie hlavy na pravej zarážke pošle doľava až na ľavú zarážku. Prvú najnevýznamnejšiu hlavu, ktorá nie je na pravej zarážke nechá na mieste a ostatné hlavy (posunuté na začiatku doprava) posunie smerom doľava späť na svoje miesto.

Dekrementovanie pozície funguje analogicky: Všetky svoje najnevýznamnejšie hlavy na ľavej zarážke pošleme doprava na pravú zarážku a prvú najnevýznamnejšiu hlavu, ktorá sa nenachádza na ľavej zarážke dekrementuje.

Všimnime si, že tento proces trvá rôznym procesom rôzne časy. Preto treba uvedené procesy späť zosynchronizovať. Každý proces bude v párnych krokoch pracovať podľa algoritmu uvedeného vyššie a v nepárnych krokoch každý proces, ktorý ešte uvedený algoritmus neskončil oznámi túto informáciu ostatným procesom v aktívnej časti, ktoré sú naladené na rovnaký kanál.

V prípade, že všetky procesy už skončili a sú pripravené pokračovať ďalším krokom – t.j. na kanáli prislúchajúcom aktívnej časti nevyšle v nepárnom kroku žiadny proces žiadnu správu, nový aktívny proces (ktorý o sebe vie, že je aktívny, pretože má všetky svoje hlavy na ľavej zarážke) môže pokračovať so simuláciou ďalšieho kroku výpočtu.

V prípade, že stroj M sa svojou hlavou dostal do inej (t.j. susednej) časti prislúchajúcej susednému kanálu, aktívny proces sa preladí na tento susedný kanál – t.j. inkrementuje resp. dekrementuje hodnotu pamätanú v svojom kanáli. Následne oznámi všetkým procesom na tomto susednom kanáli, že odteraz bude aktívny tento susedný kanál. Aktívnym procesom sa na tomto susednom kanáli stane proces pamätajúci si hodnotu nula t.j. má všetky svoje hlavy na ľavej zarážke.

Ešte musíme ukázať, akým spôsobom sa dostať do invariantu platného pre počiatočnú konfiguráciu. Všimnime si, že sieti M' stačí vytvárať kanály „online“ – t.j. sieť M' bude mať len tie kanály ktoré naozaj potrebuje vo výpočte a ostatné si vytvorí, keď ich bude potrebovať.

Takže sieti M' stačí vytvoriť procesy na začiatku iba na prvom kanáli. To urobí nasledovne: Na začiatku inicializačný proces bude posúvať svojou najnevýznamnejšou hlavou h_1 doprava a vyrábať svoje kópie. Keď svojou hlavou h_1 príde nakoniec, pohne svojou druhou najnevýznamnejšou hlavou h_2 o jedno políčko doprava a svoju hlavu h_0 pošle späť na ľavú zarážku. Takto bude pokračovať až dovtedy, kým nebudú všetky hlavy na pravej zarážke.

Prvý rad kópií (s hlavami $h_2,\ldots,h_k$ na ľavej zarážke) si bude v svojom stave pamätať hodnotu vstupu, zatiaľ čo ostatné procesy si budú v svojom stave pamätať prázdny symbol ako hodnotu pracovnej pásky stroja M.

Počas výpočtu si procesy naladené na kanál s najvyšším číslom budú pamätať, že ich kanál je najvyšší a v prípade, že sa nejaký z týchto procesov preladí na ešte vyšší kanál, vytvorí podobnú štruktúru ako je popísané v predchádzajúcom odstavci na tomto vyššom kanáli a odovzdá procesom na tomto kanáli informáciu o tom, že ich kanál je odteraz najvyšší.

V prípade, že stroj M akceptuje svoj vstup, môžeme využiť techniku z vety 3.1 a akceptovať taktiež. Sieť M' používa k vstupných hláv a $c^{S(N)}$ rôznych kanálov, ktoré si vie pamätať pomocou $S(N)$ políčok kanálovej pásky.

Ukázali sme jednu inklúziu, ešte ostáva inklúzia „$\subseteq$“ – t.j. našou úlohou je odsimulovať prácu siete strojov bez pracovných pások na deterministickom Turingovom stroji.

Majme teda sieť M s k vstupnými hlavami a bez pracovných pások pracujúcu v kanálovej zložitosti $S(n)$. Zostrojme k nej ekvivalentný Turingov stroj M'. Stroj M' si bude pamätať konfigurácie siete M na svojej páske nasledovne:

Páska stroja M bude mať tvar $¢c_1☺c_2☺c_3☺\ldots\$$, kde zarážkou ☺ sú oddelené popisy konfigurácií c_i. c_i je popis konfigurácií, ktoré majú na svojej kanálovej páske i-ty lexikograficky najmenší obsah braný bez prázdnych symbolov.

c_i má tvar $☻k_1☻k_2\ldots☻$, kde k_j je popis tých konfigurácií, ktoré majú kanálovú hlavu na pozícii j a obsah kanálovej pásky je i-ty najmenší.

k_j má tvar $♥g_0♥g_1\ldots♥$, kde g_m je popis takej konfigurácie, ktorej pozície vstupných hláv tvoria číslo m, keď ich zoberieme ako cifry v n-árnej sústave, kde n je veľkosť vstupného slova.

V g_m si pamätáme pracovné aj komunikačné stavy všetkých konfigurácii, ktoré sa práve vo výpočte nachádzajú.

Stroj M' bude mať dve poschodia na svojej páske, na jednom si pamätá stav výpočtu pred vykonaním kroku a na druhé poschodie bude postupne zapisovať stav po vykonaní jedného kroku. Následne vymení sémantiku jednotlivých poschodí.

Stroj M' vie na takto si pamätaných poschodiach jednoducho simulovať jednotlivé kroky výpočtu siete M:

Najprv zistí všetky nasledujúce konfigurácie a zapíše ich nad druhé poschodie. Následne zistí aké stavy vysielajú jednotlivé konfigurácie, dopíše ich novovzniknutým konfiguráciám a skontroluje, či na jednom kanáli sa vysiela iba jeden komunikačný stav. Potom zistí, či všetky konfigurácie chcú akceptovať a ak áno, akceptuje aj on.

Stroj M' si môže budovať svoju pásku „online“ – vždy keď bude chcieť prísť za pravú zarážku, vybuduje si tam štruktúru pre konkrétny kanál.

Všetkých kanálov je maximálne $g^{S(n)}$, kde g je veľkosť pracovnej abecedy siete M. Všetkých konfigurácií, ktoré sú naladené na jeden konkrétny kanál je abn^k, kde a resp. b je veľkosť množiny pracovných resp. komunikačných stavov.

Takže stroj M' zjavne potrebuje maximálne $n^k c^{S(n)}$ políčok pre vhodnú konštantu c a pracuje v čase $n^k c^{S(n)} T(n)$, kde $T(n)$ je časová zložitosť siete M. □

Veta 3.9 je zovšeobecnením výsledku v [WP05b] tým, že do úvahy sme zobrali viachlavé stroje a zbavili sme sa predpokladu pre funkciu $S(n) \geq \log N$.

Veta 3.9 má taktiež niekoľko zaujímavých dôsledkov:

Dôsledok 3.10. *$WPFA(k) = DSPACE(n^k)$. Špeciálne trieda sietí jednohlavých automatov je práve trieda deterministických kontextových jazykov – $WPFA(1) = \mathcal{L}_{DCS}$.*

Dôsledok 3.11. *Pre ľubovoľnú funkciu $S(n) \in o(\log N)$ platí:*

$$WPM(k-1,0)CHANNEL(S(n)) \subsetneq WPFA(k,0)CHANNEL(S(n))$$

špeciálne

$$WPFA(k-1) \subsetneq WPFA(k)$$

.

Dôsledok 3.12. *Pre ľubovoľnú funkciu $S(n) \in \Omega(\log N)$ platí:*

$$WPM(k-1,0)CHANNEL(S(n)) = WPM(k,0)CHANNEL(S(n))$$

Dôkaz. Uvedené dva dôsledky zjavne vyplývajú zo známych výsledkov o pamäťovej hierarchii deterministických Turingovych strojov a z faktov, že pre $S(n) = o(\log N)$ platí:

$$n^{k-1} c^{S(n)} = o\left(n^k c^{S(n)}\right)$$

a zároveň pre $S(n) = \Omega(\log N)$ platí:

$$n^k c^{S(n)} = O\left(n^{k-1} c^{S(n)}\right)$$

□

Dôsledok 3.13.

$$\bigcup_{k\in\mathbb{N}} WPFA(k) = PSPACE$$

Poznámka 3.14. V dôkaze vety 3.9 sme si pri simulácii Turingovho stroja pomocou siete strojov v n-árnej sústave pamätali vstupnými hlavami siete M' pozíciu hlavy h na páske nejakého deterministického stroja M. Pri posune hlavy h si určité procesy pamätanú hodnotu inkrementovali resp. dekrementovali. Tieto operácie môžu spomaliť danú simuláciu. V prípade, že si danú pozíciu budú hlavy siete M pamätať v n-árnom Grayovom kóde [Gua98], budú operácie inkrementovania a dekrementovania prebiehať v konštantnom čase. Taktiež si nemusí sieť M' vždy, keď príde na nový kanál vytvoriť všetky procesy na tomto kanáli, ale si ich bude vytvárať postupne ako ich bude potrebovať – t.j. proces s aktuálne najväčším číslom pamätaným vstupnými hlavami si bude v stave pamätať, že je posledný a keď nastane situácia, že novým aktuálnym procesom by mal byť jeho pravý sused, tak ho „online“ vytvorí. Týmto spôsobom vieme simulovať v asymptoticky rovnakom čase Turingove stroje pracujúce v priestorovej zložitosti $n^k c^{S(n)}$ pomocou k-hlavových sietí strojov bez pracovných pások s kanálovou zložitosťou $S(n)$.

3.4 Vzťah sietí strojov a GDSA

V tejto časti porovnáme náš model s modelom publikovaným v [Slo92] a [Slo88]. Najprv v stručnosti definujeme globálne deterministické synchronizované stroje ($GDSA$) a neskôr ukážeme ako súvisia s naším modelom.

Stroj $GDSA$ sa správa podobne ako nami definovaný model, ale s iným typom komunikácie. Nepoužíva kanálovú pásku, ale v istým okamihoch každý proces, ktorý sa nachádza vo výpočte na nejakom vstupe w si môže nedeterministicky uhádnuť nejaký „synchronizačný prvok“ – ekvivalent komunikačného stavu v našom modeli. V tomto okamihu proces vo výpočte čaká, kým aj ostatné procesy vo výpočte sa nedostanú do tohoto špeciálneho synchronizačného stavu. Keď sú všetky procesy v synchronizačnom stave, všetky prejdú do jedného a toho istého synchronizačného prvku, ktorý si na začiatku nedeterministicky uhádli. Podmienka globálneho determinizmu však zaručuje, že aspoň jeden proces si musí synchronizačný stav vybrať deterministicky a teda istým spôsobom prenesie/oznámi svoj synchronizačný prvok všetkým ostatným procesom.

Pre formálnu definíciu $GDSA$ pozri napríklad [Slo92].

Lema 3.15. *Pre ľubovoľné páskovo konštruovateľné funkcie $S(n)$ a $C(n)$ platí:*

$$WPM(k,t)WSPACE(S(n))CHANNEL(C(n))TIME(T(n)) \subseteq \bigcup_h GDSA(k,t)SPACE(S(n)+C(n))TIME\left(O\left(T(n)(S(n)+C(n))h^{S(n)+C(n)}\right)\right)$$

Uvedená lema hovorí, že triedu sietí strojov pracujúcich v priestore $S(n)$, čase $T(n)$ a kanálovej zložitosti $C(n)$) vieme simulovať globálne deterministickými strojmi pracujúcimi v priestore $S(n)+C(n)$ a v čase

$$O\left(T(n)(S(n)+C(n))h^{S(n)+C(n)}\right)$$

pre vhodnú konštantu h.

Dôkaz. Majme teda sieť strojov M patriacu do uvedenej triedy. Nájdeme k nej ekvivalentný $GDSA$ stroj M', ktorý bude spĺňať vyššie uvedené podmienky. Všimnime si, že $GDSA$ stroje majú k dispozícii vlastne jeden kanál na ktorom si vedia prenášať informácie. Presne ako v našom modeli, môže na danom kanáli vysielať viacero procesov avšak každý z nich musí vyslať rovnakú hodnotu. Musíme ešte vyriešiť nasledovný problém: V našom modeli vie proces detekovať, či na kanáli niekto vysiela alebo nie, zatiaľ čo v modeli $GDSA$ musí vždy niekto vyslať nejakú informáciu – vždy musí existovať stroj, ktorý deterministicky prejde do svojho synchronizovaného stavu.

Našou úlohou je na tomto jednom kanáli odsimulovať komunikáciu na $h^{C(n)}$ kanáloch, kde h je veľkosť pracovnej abecedy, pretože až toľko kanálov môže sieť M použiť vo svojom výpočte.

Urobíme to takto: Na začiatku si stroj M' vyrobí procesy p_c zodpovedajúce všetkým možným konfiguráciám c siete M, ktoré sa môžu vyskytnúť pri výpočte siete M. Tu potrebujeme páskovú konštruovateľnosť funkcií $S(n)$ a $C(n)$. Tieto procesy p_c budú mať rovnaký počet pások ako sieť M (kanálová páska procesu p_c bude v tomto prípade obyčajná pracovná páska). Na svojich páskach si bude proces p_c pamätať obsah prislúchajúcich pások konfigurácie c siete M, zároveň bude mať označené to políčko každej pásky, na ktorom sa nachádza hlava príslušnej pásky siete M.

Pri simulácii i-tého kroku výpočtu v_i siete M si bude proces p_c pamätať, či sa jemu prislúchajúca konfigurácia c nachádza vo výpočtu pred i-tym krokom alebo nie.

Simulácia jedného kroku bude vyzerať nasledovne: Všetky konfigurácie c siete M si vieme zoradiť do postupnosti $(c_1, c_2, \ldots)$. V simulácii každého kroku výpočtu siete M procesy p_c v tomto poradí oznámia ostatným procesom, či sa im prislúchajúca konfigurácia nachádza v tomto kroku vo výpočte a ak áno, tak aj čo by urobila (nový pracovný a komunikačný stav a pohyb jednotlivých hláv). Toto oznámenie sa urobí technicky samozrejme tak, že ostatné procesy si nedeterministicky uhádnu obsah správy, zatiaľ čo aktívny proces im ho deterministicky oznámi. Takto vie proces p_c každej konfigurácie c siete M, či bude aktívny aj v ďalšom kroku. Zároveň vedia všetky procesy aj skontrolovať, aký komunikačný stav sa na akom kanáli vysiela, prípadne či sa na danom kanáli nevysiela alebo či sa snažia dva procesy vysielať rôznu hodnotu.

Všimnime si, že týmto spôsobom sme zariadili, že sa vysiela aj na nevyužitých kanáloch – konkrétne konfigurácie naladené na takého kanály vysielajú informáciu, že vlastne nič vysielať nechcú.

V prípade, že všetky nové aktívne konfigurácie sú akceptačné, prejdú všetky procesy p_c stroja M' do akceptačného stavu. V prípade, že sa vyskytne problém pri vysielaní alebo nejaká konfigurácia siete M neakceptuje vstup, prejde jej prislúchajúci proces do zamietajúceho stavu.

Zjavne na simuláciu potrebujeme priestor $S(n) + C(n)$. Počas simulácie jedného kroku siete M musia všetky procesy vyslať správu sekvenčne všetkým ostatným procesom. Všetkých konfigurácií je $h^{O(S(n)+C(n))}$. Jedno vysielanie spolu s réžiou okolo trvá $O(S(n) + C(n))$, takže celkový čas výpočtu stroja M' je $O(T(n)(S(n) + C(n))h^{O(S(n)+C(n))})$. □

Lema 3.16.

$$GDSA(k,t)SPACE(S(n))TIME(T(n)) \subseteq$$
$$WPM(k,t)WSPACE(S(n))CHANNEL(1)TIME(O(T(n)))$$

Uvedená lema hovorí o tom, že každý globálne deterministický synchronizovaný stroj vieme odsimulovať sieťou strojov s jedným kanálom v rovnakom priestore a v asymptoticky rovnakom čase.

Dôkaz. Majme teda $GDSA$ stroj M. Ekvivalentnú sieť strojov M' k nemu zostrojíme takto:

Sieť M' bude pracovať rovnako ako stroj M. Vždy keď nejaký proces stroja M deterministicky prejde do synchronizačného stavu s, k nemu prislúchajúci proces siete M' na kanáli oznámi komunikačný stav s. Keď nejaký proces stroja M chce nedeterministicky prejsť do nejakého synchronizačného stavu s z množiny S, k nemu prislúchajúci proces siete M' príjme na kanáli nejaký komunikačný stav a skontroluje, či je tento stav prvkom množiny S.

Procesom stroja M však môže trvať rôzny čas, kým prejdú do synchronizačného stavu, a preto každý proces, ktorý ešte pracuje, bude v nepárnych krokoch vysielať na kanáli správu „still-working“. a v párnych krokoch bude simulovať prácu k nemu prislúchajúceho procesu stroja M.

Keď žiadny proces už nevyšle správu „still-working“, môže na kanáli prebehnúť synchronizácia. Na konci pomocou techniky z vety 3.1 sieť M' akceptuje vstup, alebo ho zamietne. Sieť M' zjavne potrebuje na svoju prácu jeden kanál, pracovný priestor $S(n)$ a čas $O(T(n))$.

Všimnime si, že ak bol jednosmerný pôvodný stroj M, je jednosmerná aj výsledná sieť M'. □

Dôsledok 3.17. *Pre ľubovoľnú páskovo konštruovateľnú funkciu platí:*

$$WPM(k,t)SPACE(S(n)) = GDSA(k,t)SPACE(S(n))$$

Dôsledok 3.18. *Pre ľubovoľnú páskovo konštruovateľnú funkciu platí:*

$$1WPM(1,t)SPACE(S(n)) = 2WPM(1,t)SPACE(S(n))$$

Dôkaz. Inklúzia „$\subseteq$“ je zrejmá z definície. K dvojsmernej sieti strojov pracujúcej v celkovej priestorovej zložitosti $S(n)$ existuje podľa lemy 3.15 ekvivalentný $GDSA$ v rovnakej priestorovej zložitosti. K nemu existuje z [Slo92] ekvivalentný jednosmerný $GDSA$ a k nemu existuje podľa lemy 3.16 ekvivalentná jednosmerná sieť strojov v celkovej priestorovej zložitosti $S(n)$. □

3.5 Vplyv kanálovej pásky na silu modelu

V časti 3.4 sme zistili, že na odsimulovanie celej kanálovej pásky nám stačí jeden kanál, avšak s veľkým časovým postihom. Skúsme tento postih o niečo znížiť:

Veta 3.19.

$$WPM(k,t)WSPACE(S(n))CHANNEL(C(n))TIME(T(n)) \subseteq$$
$$WPM(k,t+1)WSPACE(S(n)+C(n))CHANNEL(1)TIME\left(T(n)c^{C(n)}\right)$$

Dôkaz. Dá sa využiť zjednodušená verzia dôkazu lemy 3.15. Sieť M', ktorá má k dispozícii iba jeden kanál, bude simulovať sieť M s kanálovou zložitosťou $C(n)$ nasledovne:

Výpočet siete M' bude prebiehať rovnako ako výpočet siete M - s rovnakými konfiguráciami, ale obsah kanálovej pásky si sieť M' bude pamätať v normálnej pracovnej páske. Sieť M' bude postupne krok po kroku simulovať výpočtové kroky siete M.

Jediný problém je odsimulovať komunikáciu na viacerých kanáloch. V simulácii každého kroku procesy siete M' budú na druhom poschodí svojej „kanálovej“ pásky počítať od nuly až po číslo najväčšieho kanála nachádzajúceho sa vo výpočte. Vždy keď toto počítadlo bude rovnaké ako číslo kanála, na ktorom je daný proces siete M' naladený, tento proces pošle na kanáli tú správu, ktorú by poslal jemu prislúchajúci proces siete M.

Takto vieme každý krok siete M odsimulovať na $c^{C(N)}$ krokov siete M' a v rovnakej celkovej priestorovej zložitosti. □

Dôsledok 3.20.

$$WPM(k,1)WSPACE(S(n)) = \bigcup_c DSPACE\left(n^k c^{S(n)}\right)$$

Dôkaz. Uvedený dôsledok vyplýva z viet 3.19 a 3.9. □

Dôkaz vety 3.19 sa dá zovšeobecniť na prípad, keď namiesto kanálovej pásky dĺžky $C(n)$ budeme mať k dispozícii kanálovú pásku dĺžky $D(n) \in O(C(n))$, kde $C(n)$ aj $D(n)$ sú páskovo konštruovateľné funkcie. V takomto prípade sa časová zložitosť zmení z $T(n)$ na $T(n)c^{\frac{C(n)}{D(n)}}$ a priestorová zložitosť z $S(n)$ na $S(n) + \frac{C(n)}{D(n)}$.

Kapitola 4

Model rozšírený o nedeterminizmus

V tejto kapitole rozšírime náš model o možnosť nedeterministických prechodov a pokúsime sa zistiť akým spôsobom ovplyvní nedeterminizmus silu modelu a jednotlivé zložitostné triedy.

4.1 Definícia

Stroj budeme definovať rovnako ako v definícii 2.1. Množinu pracovných stavov Q však rozdelíme na dve disjunktné podmnožiny. Jednu podmnožinu $Q_U \subseteq Q$ nazveme množinou *univerzálnych pracovných stavov* a druhú podmnožinu $Q_E \subseteq Q$ nazveme množinou *existenčných pracovných stavov*. Dohodou určíme akceptačný a zamietajúci stav za univerzálny.

Konfiguráciu, hlavovú konfiguráciu, počiatočnú aj terminálnu konfiguráciu, jednoduchý krok ako aj konfiguráciu modifikovanú vysielaním definujeme rovnako ako pre deterministický model. Pravidlo súhlasného vysielania taktiež zachováme v platnosti aj pre nedeterministický model.

Prechody (konfigurácie) sa nám rozdelia na dve skupiny. *Univerzálne prechody (konfigurácie)* sú prechody z univerzálnych pracovných stavov (konfigurácie v univerzálnych stavoch) a *existenčné prechody (konfigurácie)* sú prechody z existenčných stavov (konfigurácie v existenčných stavoch).

V deterministickom modeli sme mali jediný výpočtový graf. Po pridaní existenčných prechodov nám vznikne viacero výpočtových grafov. Neformálnym jazykom povedané každá kombinácia existenčných prechodov nám vytvorí jeden deterministický výpočtový graf spĺňajúci definíciu 2.11.

Formálne definujeme výpočtové grafy $G(w)$ pre vstup w nedeterministickej siete strojov M rekurzívne pre každú dĺžku výpočtu d. Množinu výpočtových grafov pre hĺbku d označme $G_d(w)$. Prvkami množiny $G_d(w)$ budú všetky deterministické výpočtové grafy hĺbky d, ktoré mohli vzniknúť z počiatočnej konfigurácie – každý z nich je určený konkrétnymi nedeterministickými rozhodnutiami siete M v existenčných konfiguráciách.

1. Množina výpočtových grafov $G_0(w)$ pre hĺbku 0 obsahuje jediný výpočtový graf, ktorý obsahuje jediný vrchol – počiatočnú konfiguráciu výpočtu.

2. Nech $G_d(w)$ je množina výpočtových grafov pre hĺbku d. Potom množinu $G_{d+1}(w)$ výpočtových grafov pre hĺbku $d+1$ získame nasledovne:

   Z každého grafu $G \in G_d(w)$ nám vznikne niekoľko grafov (prípadne jeden alebo žiadny) v množine $G_{d+1}(w)$. Množinu existenčných listov grafu $G \in G_d(w)$ označme G_E. Z každej konfigurácie $c \in G_E$ existuje niekoľko existenčných prechodov, ktorých množinu si označme c_P. Množinu všetkých množín c_P pre ľubovoľné $c \in G_E$ si označme P_E.

   Každý prvok p množiny P_E spolu so všetkými univerzálnymi prechodmi z G určuje jeden výpočtový graf $G' \in G_{d+1}(w)$ pre hĺbku $d+1$ nasledovne: G' obsahuje graf G ako svoj podgraf, navyše obsahuje všetkých Δ_{C_d}-nasledovníkov univerzálnych listov grafu G a zároveň jedného Δ_{C_d}-nasledovníka pre každý existenčný list grafu G určeného prvkom $p \in P_E$.

   Graf G' obsahuje všetky hrany grafu G a zároveň prechodové a komunikačné hrany z listov G do nových listov v hĺbke $d+1$ rovnako ako pri deterministickom modely podľa definície 2.11.

Výpočtový graf $G \in G_d(w)$ na vstupe w nazveme akceptačným ak všetky jeho listy sa nachádzajú v akceptačnom stave. Nepožadujeme aby boli v rovnakej hĺbke alebo naladené na rovnaký kanál. Táto podmienka pri nedeterminizme stráca svoj význam.

Hovoríme, že sieť akceptuje svoj vstup, ak existuje pre nejakú hĺbku d nejaký graf $G \in G_d(w)$, ktorý je akceptačný.

Jednotlivé zložitosti (časová, priestorová, kanálová, paralelizmus) výpočtu nedeterministickej siete na vstupe w definujeme ako zložitosti takého akceptačného výpočtu $G_d(w)$, ktorý má danú zložitosť najmenšiu.

Zložitosť nedeterministickej siete a jednotlivé zložitostné triedy definujeme analogicky ako pri deterministických strojoch.

Pri deterministických strojoch sme používali skratku WP pri označení strojov a ich zložitostných tried. Pre nedeterministické stroje použijeme skratku WN – napr. $WNFA(k)$ je trieda jazykov rozpoznávaných sieťami nedeterministických konečných automatov s k hlavami.

4.2 Vlastnosti nedeterministických zariadení

Ukazuje sa, že mnohé z uvedených vlastností deterministického modelu sa pretransformujú na ekvivalentné vlastnosti nedeterministického modelu.

Veta 4.1. *Pre ľubovoľnú funkciu $S(n)$ platí:*

$$WNM(k,0)CHANNEL(S(n)) = \bigcup_c NSPACE\left(n^k c^{S(n)}\right)$$

Uvedená veta je ekvivalentom vety 3.9 platnej pre deterministické zariadenia. Hovorí, že výpočtová sila nedeterministických sietí strojov bez pracovných pások a s kanálovou zložitosťou $S(n)$ je rovnaká ako výpočtová sila priestorovo obmedzených nedeterministických Turingovych strojov.

Dôkaz. Myšlienka dôkazu tejto vety je rovnaká ako vo vete 3.9 – priestor $c^{S(n)}$ si vieme zapamätať v čísle kanála a priestor n^k si vieme zapamätať pozíciami vstupných hláv.

Keďže dôkaz obidvoch inklúzii je skoro totožný dôkazu vety 3.9 zameriame sa iba na odlišnosti.

Začneme inklúziou „$\supseteq$“: Majme teda nedeterministický Turingov stroj M, pre ktorý zostrojíme ekvivalentnú nedeterministickú sieť strojov M'. Sieť M' bude pracovať rovnako sieť zostrojená v dôkaze vety 3.9. Sieť M; bude

simulovať postupne každý krok stroja M. Platiť bude pred každým krokom siete M' aj rovnaký invariant.

Jediný rozdiel je v tom, že v tomto prípade je stroj M nedeterministický, čiže si môže nedeterministicky uhádnuť ďalší krok výpočtu. Avšak nedeterministická je taktiež sieť M'. Preto aktívny proces siete M' (pozri dôkaz vety 3.9) nedeterministicky určí ako by sa rozhodol stroj M a bude pokračovať v simulácii tohoto určeného kroku.

Opačná inklúzia „$\subseteq$“: V tomto prípade chceme simulovať nedeterministickú sieť strojov M nedeterministickým Turingovym strojom M'. Stroj M' si bude pamätať všetky použité konfigurácie siete M rovnako ako v dôkaze vety 3.9 a bude rovnako simulovať postupne všetky kroky siete M.

Jediný rozdiel je aj tu v tom, že jednotlivé procesy siete M si môžu svoju ďalšiu konfiguráciu nedeterministicky uhádnuť. V takomto prípade stroj M' nedeterministicky uhádne všetky existenčné prechody siete M, v prípade, že sa vo výpočte nejaké vyskytnú.

Pri obidvoch inklúziách zostrojené ekvivalentné zariadenia pracujú v požadovaných zložitostiach z rovnakých príčin ako je ukázané v dôkaze vety 3.9. □

Dôsledok 4.2. *$WNFA(k) = NSPACE(n^k)$. Špeciálne trieda nedeterministických sietí jednohlavých automatov je práve trieda kontextových jazykov – $WNFA(1) = \mathcal{L}_{CS}$.*

Dôsledok 4.3. *Pre ľubovoľnú funkciu $S(n) \in o(\log N)$ platí:*

$$WNM(k-1,0)CHANNEL(S(n)) \subsetneq WNFA(k,0)CHANNEL(S(n))$$

špeciálne

$$WNFA(k-1) \subsetneq WNFA(k)$$

.

Dôsledok 4.4. *Pre ľubovoľnú funkciu $S(n) \in \Omega(\log N)$ platí:*

$$WNM(k-1,0)CHANNEL(S(n)) = WNM(k,0)CHANNEL(S(n))$$

Dôsledok 4.5.

$$\bigcup_{k\in\mathbb{N}} WNFA(k) = NSPACE$$

Dôsledok 4.6.

$$WNFA(k) = WPFA(2k)$$

Dôsledok 4.7. *V prípade, že pre ľubovoľnú konštantu* $a \geq 1$ *platí:*

$$WNFA(k) = WPFA(ak)$$

potom $DSPACE(n^{ak}) = NSPACE(n^k)$.

Uvedený dôsledok je zaujímavý hlavne pre konštanty $a < 2$.

Dôsledok 4.8. *Pre ľubovoľnú funkciu* $S(n) \in \Omega(\log N)$ *platí:*

$$WPM(k,0)CHANNEL(S(n)) = WNM(k,0)CHANNEL(S(n))$$

Dôkaz. Uvedený dôsledok priamo vyplýva z viet 4.1, 3.9 a Savitchovej vety [For04]. □

Ak by vyššie uvedený dôsledok platil aj pre niektoré funkcie $S(n) \in o(\log N)$, znamenalo by to vylepšenie odhadu zo Savitchovej vety na simuláciu nedeterministického priestoru pomocou deterministického pre konkrétnu triedu strojov. Napríklad ak by tento dôsledok platil pre $S(n) = O(1)$, znamenalo by to, že $DSPACE(n^k) = NSPACE(n^k)$.

Podarilo sa nám ukázať ekvivalent vety 3.9. V nasledujúcej časti ukážeme ekvivalent vety 3.8.

Veta 4.9. *Trieda jazykov akceptovaných nedeterministickými sieťami s jednou vstupnou hlavou, s jednotkovou paralelnou a konštantnou celkovou priestorovou zložitosťou je* $NLOGSPACE$.

Dôkaz. Dôkaz tejto vety je podobný dôkazu vety 3.8 pre deterministické siete strojov.

Najprv sa zaoberajme inklúziou „$\subseteq$“ – simulovanie nedeterministických sietí Turingovými strojmi. Majme teda nedeterministickú sieť strojov M pracujúcu s jednou vstupnou hlavou a s jednotkovou paralelnou a konštantnou celkovou priestorovou zložitosťou. Nech táto sieť pracuje s k procesmi. K nej ekvivalentný Turingov stroj si bude na svojej páske pamätať všetky konfigurácie nedeterministickej siete M. Na to mu stačí priestor $O(\log N)$.

Následne bude postupne simulovať kroky siete M. Jeden nedeterministickej siete M odsimuluje tak, že pre každý proces zapamätaný na páske nedeterministicky uhádne jeho nasledovníka. Potom pre odsimuluje krok každého procesu – pohyb hlavy, zmenu stavu a vysielanie. Nakoniec skontroluje, či dva procesy nevysielajú na tom istom kanáli rôzne symboly.

Ak všetky procesy nedeterministickej siete M akceptujú vstup, stroj M' akceptuje taktiež.

Ostáva ukázať opačnú inklúziu – simulovanie triedy $NLOGSPACE$ pomocou nedeterministických sietí s jednotkovou paralelnou zložitosťou a konštantnou priestorovou zložitosťou.

Majme teda Turingov stroj $M \in NLOGSPACE$. Nedeterministickú sieť M' k nemu zostrojíme rovnako ako v dôkaze vety 3.8. Sieť M' najprv nedeterministicky uhádne krok stroja M, ktorý následne odsimuluje rovnakým spôsobom ako zostrojená sieť vo vete 3.8. Ak stroj M akceptoval svoj vstup, sieť M' akceptuje taktiež. □

Dôsledok 4.10. *Ak trieda jazykov akceptovaných deterministickými sieťami je rovná triede jazykov akceptovaných nedeterministickými sieťami, potom* $NLOGSPACE = DLOGSPACE$.

Kapitola 5

Záver

V diplomovej práci sme sa pokúsili preskúmať formálny výpočtový model, ktorý využíva analógiu bezdrôtovej komunikácie pri svojom výpočte. Najprv sme sa zaoberali definíciou akceptačného kritéria a ukázali sme ako sa vedia jednotlivé procesy zosynchronizovať tak, aby všetky naraz akceptovali resp. zamietli výpočet.

V ďalšej kapitole sme sa venovali konečným automatov, ktorým sme pridali schopnosť komunikácie. Podarilo sa nám charakterizovať siete konečných automatov s jednotkovou paralelnou zložitosťou. Taktiež sme ukázali, že konečným automatom stačí iba jedno slepé počítadlo k tomu, aby vedeli akceptovať ľubovoľný rekurzívne vyčísliteľný jazyk.

V diplomovej práci sa podarilo charakterizovať priestorovo ohraničené siete strojov. Napríklad jedným z dôsledkov tejto charakterizácie je, že siete konečných automatov akceptujú práve deterministické kontextové jazyky. Ďalej sa nám podarilo ukázať, že zvýšenie počtu hláv zvýši silu sietí strojov, ktorých priestorová zložitosť je $o(\log N)$.

V poslednej kapitole sme sa venovali nedeterminizmu a ukázali sme, že pre nedeterministické siete strojov platia ekvivalentné vety ako pre deterministické siete strojov. Napríklad nedeterministické siete konečných automatov akceptujú práve kontextové jazyky.

Ostáva však ešte niekoľko otvorených otázok o danom modeli. Zaujímavé by bolo zistiť, ako presne ovplyvní veľkosť kanála silu strojov – resp. či existuje rýchlejšia simulácia sietí strojov s viacerými kanálmi pomocou sietí strojov s jedným kanálom než aká je uvedená v tejto diplomovej práci.

Podarilo sa charakterizovať priestorovo obmedzené nedeterministické siete strojov, avšak nepodarilo sa charakterizovať časovo obmedzené nedeterministické siete strojov. Je možné simulovať takéto siete strojov deterministickými sieťami strojov rýchlejšie ako vyskúšaním všetkých možností podobne ako sú simulované napríklad alternujúce stroje v [WP06]?

Literatúra

[CKS81] A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28:114–133, 1981.

[For04] Michal Forišek. Formálne jazyky a automaty. Skriptá, FMFI UK, Feb 2004.

[Gua98] Dah-Jyh Guan. *Generalized Gray Codes with Applications*, volume 22, pages 841–848. Proceedings of the National Science Council, Republic of China, 1998.

[Slo88] Anna Slobodová. Alternujúce výpočtové modely s komunikáciou. Master's thesis, MFF UK, Bratislava, 1988.

[Slo92] Anna Slobodová. *Synchronizované alternujúce výpočty.* PhD thesis, MFF UK, Bratislava, 1992.

[Wie89] Wiedermann. On the power of synchronization. *EIK: Journal of Information Processing and Cybernetics EIK (formerly Elektronische Informationsverarbeitung und Kybernetik)*, 25, 1989.

[WP05a] Jiří Wiedermann and Dana Pardubská. Alternatívna charakterizácia synchronizovaného alternovania. In ITAT, pages 133–144, 2005. ISBN 80-7097-609-8.

[WP05b] Jiří Wiedermann and Dana Pardubská. On the power of broadcasting in mobile computations. Technical Report V-944, ICS AS CR, Prague, 2005.

[WP06] Jiří Wiedermann and Dana Pardubská. Computing by broadcasting: Alternation in disguise (extended version). Technical Report V-975, ICS AS CR, Prague, 2006.