UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY FYZIKY A INFORMATIKY

SKÚMANIE VÝPOČTOVEJ SILY WPTM PRI OHRANIČENIACH

2014 Bc. Milan Mikula

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY FYZIKY A INFORMATIKY

SKÚMANIE VÝPOČTOVEJ SILY WPTM PRI OHRANIČENIACH

Diplomová práca

Študijný program: Informatika

Študijný odbor: 2508 Informatika

Školiace pracovisko: Katedra informatiky FMFI

Vedúci: doc. RNDr. Dana Pardubská, CSc.

Evidenčné číslo:

Bratislava 2014

Bc. Milan Mikula

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Milan Mikula
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Skúmanie výpočtovej sily WPTM pri ohraničeniach

Cieľ: Skúmanie výpočtovej sily WPTM pri ohraničeniach (počet paralelných procesov, počet komunikačných krokov, modifikácia komunikácie, slabší model ako TS)

Vedúci: doc. RNDr. Dana Pardubská, CSc.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 16.11.2011

Dátum schválenia: 23.11.2011 prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Abstrakt

Jedným z formálnych modelov skúmajúcich paralelné výpočty je bezdrôtový paralelný Turingov stroj. Tento deterministický model dovoľuje počas výpočtu vytvárať nové procesy, ktoré spolu vedia komunikovať. Na komunikáciu procesy používajú špeciálnu kanálovú pásku, ktorá udáva, na aký kanál je proces naladený. Procesy naladené na rovnakom kanáli môžu spolu komunikovať.

V práci sme ukázali, že použitie jednosmerných vstupných hláv nezmenší silu priestorovo ohraničeného modelu. Ďalej sme ukázali, aký vplyv má ohraničenie počtu procesov na skúmaný model.

Kľúčové slová: Turingov stroj, paralelné výpočty, komunikácia

Abstrakt

One of the formal models used to study parallel computations is wireless parallel Turing machine (WPTM). During the computation of this deterministic model it is possible to create new processes, which can communicate together. Each process uses a special channel tape for communication and the contend of the tape specifies the channel the process is tuned on. Processes tuned on the same channel can exchange messages.

In the thesis, we show that the use of one-way input heads does not decrease the power of space-bounded WPTM. We also show how the use of a limited number of processes in a computation would impact the power of our model.

Key words: Turing machine, parallel computation, communication

Čestné vyhlásenie

Čestne vyhlasujem, že som túto prácu vypracoval samostatne, s použitím uvedených zdrojov.

Poďakovanie

Chcel by som sa poďakovať mojej vedúcej, doc. RNDr. Dane Pardubskej, CSc., za rady, pomoc, trpezlivosť a čas, ktorý mi venovala. Ďalej mojej rodine a priateľom za povzbudenie a podporu.

Obsah

Zoznam obrázkov

Úvod

Pri riešení praktických problémov je kľúčové poznať, koľko prostriedkov musíme vynaložiť na vyriešenie daného problému. Preto skúmame zložitostné triedy na rôznych modeloch, ktoré sú motivované určitým výpočtovým strojom. V prípade paralelných výpočtov sa ako formálny model často používa alternujúci Turingov stroj (ATM) [1]. ATM má dva druhy stavov, univerzálne stavy a existenčné stavy. V prípade existenčného stavu sa ATM podľa δ-funkcie nedeterministicky rozhodne, do ktorej konfigurácie sa dostane. V univerzálnom stave sa výpočet vydá všetkými cestami naraz a dôjde k rozvetveniu na viac súbežne bežiacich procesov. Vstupné slovo akceptujeme, ak všetky paralelne bežiace procesy akceptujú. ATM je pomerne dobre preskúmaný model a sú známe aj vzťahy medzi ním a inými paralelnými modelmi, ako napríklad uniformné booleovské obvody [6].

Počas výpočtu ATM sú jednotlivé paralelne bežiace procesy oddelené. Prirodzeným krokom je pridať procesom možnosť komunikácie. Skúmali sa rôzne možnosti, ako umožniť komunikáciu medzi jednotlivými procesmi.

V prácach [5] a [4] bol Slobodovou skúmaný model synchronizovaného alternovania. Synchronizovaný alternujúci Turingov stroj (SATM) je model ATM rozšírený o možnosť použitia synchronizačného prvku. Proces si môže počas behu zvoliť nejaký synchronizačný prvok a tento proces potom čaká na ostatné procesy, kým si aj oni nezvolia synchronizačný prvok. Výpočet sa zasekne a vstupné slovo neakceptuje, ak sa zvolený synchronizačný prvok nezhoduje. Použitie synchronizačného prvku umožňuje posielanie správ medzi procesmi. Odosielateľ si vyberá synchronizačné prvky podľa obsahu správy a prijímateľ nedeterministicky háda obsah správy a synchronizáciou svoje hádanie overuje.

Deterministické paralelné výpočty s komunikáciou boli skúmané v práci [3], kde bol navrhnutý model SATM bez existenčných stavov, teda nedeterminizmu. Uvedený model však nedovoľuje procesom priamo posielať správy.

Na skúmanie sily deterministických paralelných výpočtov s komunikáciou navrhli Pardubská a Wiedermann model nazvaný bezdrôtový paralelný Turingov stroj (WPTM) [7]. Tento deterministický model dovoľuje procesom rozvetvovať sa rovnakým spôsobom, ako použitím univerzálnych stavov ATM. Procesy majú navyše špeciálnu kanálovú pásku, ktorej obsah určuje, na akom kanáli sú jednotlivé procesy naladené. Počas výpočtu si môžu procesy naladené na zhodný kanál posielať správy.

V tejto práci sa budeme zaoberať prevažne tým, ako sa zmení sila WPTM pri ohraničení prostriedkov. V prvej kapitole zadefinujeme WPTM a niektoré jeho zložitostné triedy. Taktiež uvedieme používané označenia.

V ďalšej kapitole budeme skúmať prevažne bezdrôtové paralelné konečné automaty (WPFA), model, ktorý nemá pracovné pásky a vie sa naladiť iba na konštantný počet rôznych kanálov. V tejto kapitole preskúmame vplyv jednosmernosti vstupných hláv na výpočtovú silu WPFA. Neskôr preskúmame vplyv jednosmernosti aj na model s kanálovou páskou.

V poslednej kapitole sa zaoberáme vplyvom počtu procesov využitých počas výpočtu stroja WPTM na jeho silu. Budeme skúmať, za akých podmienok pridanie ďalších procesov zväčší silu priestorovo ohraničeným WPTM.

Kapitola 1

Bezdrôtový paralelný Turingov stroj

Model WPTM vychádza z alternujúceho Turingovho stroja. Každý proces môže rovnako ako pri ATM vytvoriť konečný počet procesov, ktoré pokračujú vo výpočte z rovnakej konfigurácie ako ich rodič. Procesy môžu spolu komunikovať, a to tak, že sa naladia na rovnaký *kanál*, čo sa docieli zápisom kanálového čísla na *kanálovú pásku*. Potom všetky procesy naladené na rovnaký kanál môžu navzájom komunikovať a vymieňať si správy. Správy modelujeme ako prvok *komunikačného stavu*. Správa poslaná v čase t sa doručí v čase $t+1$ všetkým počúvajúcim procesom, ktoré sú naladené na daný kanál v čase $t+1$. Požadujeme, aby všetky správy vysielané v jednom kanáli v konkrétnom čase boli rovnaké. V opačnom prípade sa výpočet zasekne. Keďže náš model je deterministický, jeho akceptácia je vyriešená inak ako pri ATM, kde výpočet automaticky akceptuje, ak vo výpočtovom strome existuje akceptačný podstrom. Na akceptáciu sa využije komunikácia. WPTM akceptuje, ak sa všetky procesy navzájom dohodnú, že chcú akceptovať.

1.1 Definícia modelu

Neformálne definujeme bezdrôtový paralelný Turingov Stroj. Formálna definícia sa nachádza v práci [7].

Definícia 1.1 *k-páskový bezdrôtový paralelný Turingov stroj so vstupnou páskou (na ktorú sa nedá zapisovať) a kanálovou páskou je dvanástica*
$M = (k, B, L, Q, R, \Sigma, \Gamma, \Delta, g_0, \varepsilon, q_{accept}, g_{reject})$, *kde*

- *k je počet pracovných pások,*

- *B, L, Q a R sú navzájom disjunktné konečné množiny vysielacích, načúvacích, lokálnych a komunikačných stavov.*
  Množinu $K = B \cup L \cup Q$ nazveme množinou pracovných stavov.
  $R = \{q_{accapt}, g_{reject}, \varepsilon\}$, kde ε je prázdny komunikačný stav, ktorý sa používa, keď práve nekomunikujeme.

- *Σ, Γ sú konečné neprázdne vstupné a pracovné abecedy. $\$ \notin \Sigma$ je zarážka, $\sharp \in \Gamma \wedge \sharp \notin \Sigma$ je blank.*

- *$\Delta \subseteq K \times R \times (\Sigma \cup \{\$\}) \times \Gamma^{k+1} \times K \times R \times (\Gamma \setminus \{\sharp\})^{k+1} \times \{-1, 0, 1\}^{k+2}$ je prechodová relácia, prvky Δ nazývame prechody. Prechod s novým komunikačným stavom r' nazývame vysielací prechod. Hovoríme tiež, že prechod vysiela r'.*

Definícia 1.2 *Konfigurácia WPTM je prvok $K \times R \times \Sigma^* \times ((\Gamma \setminus \{\sharp\})^*)^{k+1}$. Konfigurácia môže byť vysielacia, načúvacia alebo lokálna, podľa druhu pracovného stavu.*

Hlavová konfigurácia je prvok $K \times R \times (\Sigma \cup \{\$\}) \times \Gamma^{k+1}$, a teda zodpovedá súčasnému stavu stroja a obsahu práve čítaných znakov. Zavedieme ešte jedno obmedzenie na Δ: všetky vysielacie prechody z rovnakej hlavovej konfigurácie musia vysielať ten istý stav.[1]

Konfigurácia je naladená na kanál c, ak reťazec c je napísaný naľavo od pozície hlavy na kanálovej páske. Vysielacia konfigurácia vysiela komunikačný stav r' na kanáli c,$r' \neq \varepsilon$, ak z tejto konfigurácie prechádzame prechodom, ktorý má r' ako nový komunikačný stav.

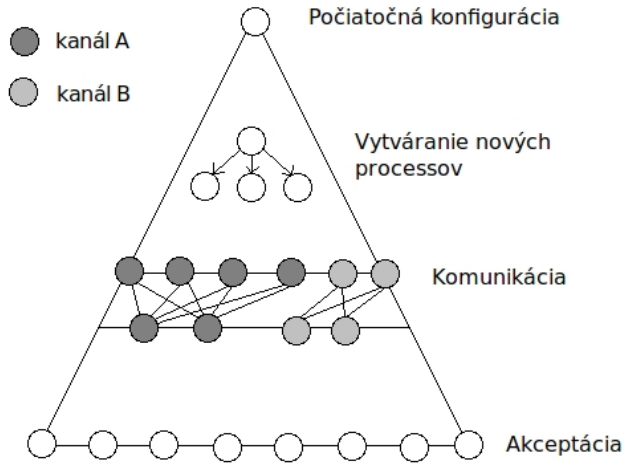
Definícia 1.3 *Výpočtový krok má dve fázy. V prvej fáze sa na každú konfiguráciu aplikujú všetky možné prechody podľa prechodovej relácie. Ak je viacero možných prechodov pre jednu konfiguráciu, vytvorí sa viacero nových konfigurácií. Táto fáza zodpovedá univerzálnemu vetveniu ATM. V druhej fáze sa uskutočňuje komunikácia. Komunikačné stavy načúvacích konfigurácií sa prepíšu komunikačným stavom vysielaným na rovnakom kanáli.*

Definícia 1.4 *Pre každý vstup w stroja M, výpočtový graf $G(w)$ stroja M je zakorenený orientovaný potenciálne nekonečný acyklický multigraf, ktorého vrcholmi sú konfigurácie stroja M a hrany zodpovedajú prechodom a komunikačným hranám. Hĺbka vrcholu (konfigurácie) v $G(w)$ je vzdialenosť od koreňa. Pre $d \geq 0$ je komunikačná hrana medzi vrcholom*

[1] Toto obmedzenie je čisto syntaktické.

α v hĺbke d a vrcholom β v hĺbke $d+1$, práve vtedy, keď α zodpovedá vysielacej konfigurácii, β zodpovedá načúvacej konfigurácii a obidve sú naladené na tom istom kanáli.

Poznámka. Pojmom proces stroja M budeme označovať cestu od koreňa k listu v príslušnom výpočtovom grafe $G(w)$. Procesom v čase t rozumieme konfiguráciu na tejto ceste v hĺbke t.



Obr. 1.1: Na obrázku je výpočtový graf $G(w)$. Zvýraznený je komunikačný krok na dvoch rôznych kanáloch a tiež vytváranie nových procesov spôsobom zodpovedajúcim univerzálnemu vetveniu ATM. Upravené podľa [7].

Definícia 1.5 *Stroj M akceptuje svoj vstup w, ak výpočtový graf $G(w)$ je konečný a všetky jeho listy sú terminálne konfigurácie, teda konfigurácie v komunikačnom stave q_{accept}, a sú naladené na rovnakom kanáli.*

1.2 Zložitostné triedy

Pre WPTM zadefinujeme triedy zložitosti, ktoré sú analógiou k triedam zložitosti ATM, ale aj nové triedy zložitosti, ktoré nemajú priamu analógiu.

Definícia 1.6 *WPTM M má pracovnú priestorovú zložitosť $S(n)$, ak pre $\forall w \in L(M)$ dĺžky n je priestor použitý na všetkých pracovných páskach vo všetkých konfiguráciách vo*

výpočtovom grafe $G(w)$ menší ako $S(n)$. $L \in WSPACE(S(n))$ práve vtedy, ak existuje WTPM M, ktorý má pracovnú priestorovú zložitosť $S(n)$ a akceptuje jazyk L.

Definícia 1.7 *WPTM M má kanálovú priestorovú zložitosť $C(n)$, ak pre $\forall w \in L(M)$ dĺžky n je priestor použitý na kanálovej páske vo všetkých konfiguráciách vo výpočtovom grafe $G(w)$ menší ako $C(n)$.*

Definícia 1.8 *WPTM M má celkovú priestorovú zložitosť $S(n)$, ak má pracovnú priestorovú zložitosť $S(n)$ a súčasne aj kanálovú priestorovú zložitosť $S(n)$.*

Definícia 1.9 *WPTM M má kanálovú zložitosť $CC(n)$, ak pre $\forall w \in L(M)$ dĺžky n použije na výpočet najviac $CC(n)$ rôznych kanálov.*

Definícia 1.10 *WPTM M pracuje v čase $T(n)$,ak pre $\forall w \in L(M)$ dĺžky n je príslušný výpočtový strom $G(w)$ hlboký najviac $T(n)$. Prirodzene $L \in WTIME(S(n))$ práve vtedy, ak existuje WTPM M, ktorý pracuje v priestore $S(n)$ a akceptuje jazyk L.*

Keďže budeme chcieť pracovať aj so sublineárnou časovou zložitosťou, budeme používať model s rýchlym prístupom na pásku. Na novú indexovú pásku môžeme binárne zapísať index i, potom ak stroj prejde do nového stavu q_{index}, tak sa v jednom kroku sa čítacia hlava presunie na pozíciu i.

Definícia 1.11 *WPTM M má procesorovú zložitosť $P(n)$, ak pre $\forall w \in L(M)$ dĺžky n je maximálny počet konfigurácií jednej hĺbky v príslušnom výpočtovom strome $G(w)$ najviac $P(n)$.*

1.3 Použité označenia

Niektoré použité označenia sme prebrali z práce [2].

Definícia 1.12 *Zložitostná trieda $WTP(k,t)$ je trieda jazykov, ktorú akceptuje bezdrôtový paralelný Turingov stroj s k vstupnými hlavami a t pracovnými páskami.*

Triedu $WTP(k,t)$ môžeme ďalej zúžiť ohraničením niektorej zložitostnej miery použitého stroja. Pridanie ohraničenia na danú triedu značíme sufixom.

Použité ohraničenia sú:

1. $SPACE(S(n))$ – celková priestorová zložitosť
2. $WSPACE(S(n))$ – pracovná priestorová zložitosť
3. $CHANNEL(C(n))$ – kanálová priestorová zložitosť
4. $PROCESS(P(n))$ – procesorová zložitosť
5. $ALPH(c)$ – ohraničenie veľkosti pracovnej abecedy

Navyše, použitím prefixu 1 označíme triedu strojov s jednosmernými vstupnými hlavami. V prípade potreby implicitne zdôrazníme triedu strojov s obojsmernými vstupnými hlavami prefixom 2.

Napríklad trieda $1WTP(k,t)WSPACE(S(n))CHANNEL(C(n))$ značí triedu jazykov, ktorú akceptuje bezdrôtový paralelný Turingov stroj s k jednosmernými vstupnými hlavami, t pracovnými páskami veľkosti najviac $S(n)$ a kanálovou páskou veľkosti najviac $C(n)$.

Triedu $WTP(k,0)CHANNEL(1)$ nazveme triedou k-hlavových konečných bezdrôtových automatov a označíme $WPFA(k)$.

Kapitola 2

Jednosmerný bezdrôtový paralelný konečný automat

V tejto kapitole preskúmame vplyv jednosmernosti na silu jedno- a viachlavých konečných automatov. Najprv ukážeme simuláciu dvojsmerného WPFA jednosmerným s konštantnou veľkosťou kanála, potom si budeme všímať vplyv obmedzenia pohybu vstupnej hlavy siete automatov, ktoré majú nekonštantnú veľkosť kanála.

2.1 Vzťah 1WPFA a 2WPFA

Vieme, že striktne jednosmerný bezdrôtový paralelný konečný automat rozpoznáva iba regulárne jazyky [2]. Pre obojsmerné automaty platí $2WPFA(k) = DSPACE(n^k)$ [2].

Pripomeňme, že k-hlavový jednosmerný WPFA je taký WPFA, že každá z jeho k hláv môže v jednom kroku stáť alebo sa pohnúť doprava. Prvé a posledné políčko vstupnej pásky obsahuje zarážku.

Veta 2.1 *Pre ľubovoľné $k \in \mathbb{N}^+$ platí:*

$$1WPFA(k) = DSPACE(n^k).$$

Dôkaz. Inklúzia „$\subseteq$“ triviálne platí, kedže podľa [2] platí

$$2WPFA(k) = DSPACE(n^k)$$

a zrejme

$$1WPFA(k) \subseteq 2WPFA(k)).$$

Dôkaz inklúzie „$\supseteq$“ spravíme simulovaním TS M pomocou siete k-hlavých jednosmerných WPFA W. Predpokladajme, že veľkosť vstupu je n'. Predpokladajme, že TS M v každom kroku najskôr prečíta a prepíše aktuálne políčko pásky a až potom sa pohne. Pre jednoduchosť tiež predpokladajme, že M má iba jednu pásku s presne $(n'+2)^k$ políčkami, teda aj vstup dostane na pracovnú pásku. Toto zjednodušenie si môžeme dovoliť, keďže $(n'+2)^k > n$ je konštruovateľná a tiež $DSPACE(n^k) = DSPACE((n+2)^k)$. Políčka pások všetkých strojov budeme indexovať od nuly. Každý proces v sieti W má $n'+2$ políčok na páske (na dvoch políčkach sú zarážky). Preto v ďalšom texte budeme namiesto n' používať $n = n'+2$. Sieť W bude mať niekoľko procesov, pomocou ktorých si bude pamätať konfiguráciu TS M.

1. Stav stroja M si bude pamätať proces R (riadiaci proces) vo svojom stave.

2. Očíslujeme si políčka pásky M číslami od 0 do $n^k - 1$. Nech aktuálna pozícia hlavy i zapísaná v sústave so základom n je $i = (i_0, i_1, \ldots, i_{k-1})_n$. Potom číslo i si uchová proces R tak, že jeho j-tá hlava bude na políčku i_j.

3. Majme množinu procesov $A = \{A_0, \ldots, A_{n^k-1}\}$, pričom proces A_i má pomocou hláv uchované číslo i rovnakým spôsobom ako proces R. Proces A_i si pamätá obsah i-tého políčka vo svojom stave.

Teraz popíšeme, ako bude prebiehať simulácia jedného kroku TS M. Počas simulácie budú všetky procesy komunikovať na rovnakom kanáli. Prípravu siete W do počiatočného stavu, v ktorom si pamätá počiatočnú konfiguráciu TS M, popíšeme neskôr. Simulácia jedného kroku TS M bude prebiehať v dvoch fázach:

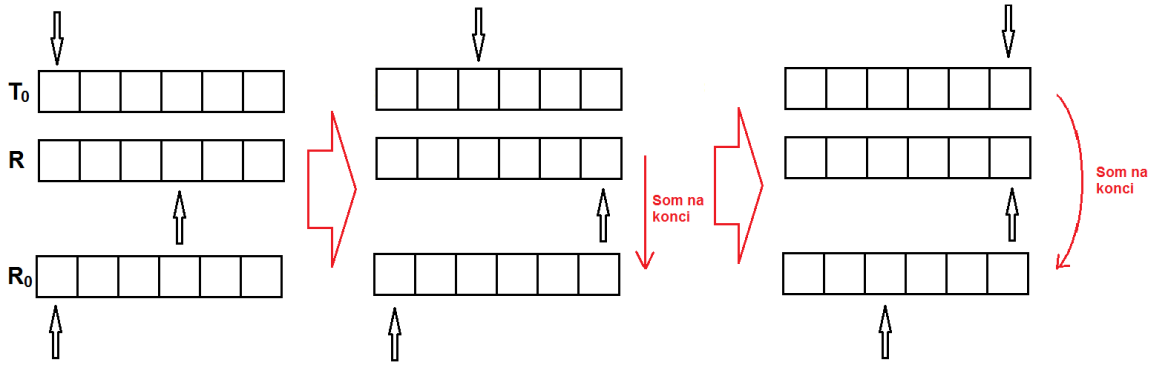
I [identifikácia symbolu pod hlavou] Proces R si pomocou pozícií svojich hláv pamätá číslo aktuálnej pozície hlavy stroja M – číslo i. V prvej fáze nájdeme taký proces $A_i \in A$, ktorý si pamätá obsah aktuálneho políčka. Nájdený proces sa označí zmenou stavu.

II [aplikácia prechodovej funkcie] V druhej fáze proces A_i pošle zapamätané políčko pásky procesu R, ktorý na základe toho z δ-funkcie stroja M vypočíta nový stav, smer pohybu hlavy a nový symbol. Nový symbol pošle naspäť procesu A_i, ktorý si ho zapamätá v stave. Potom proces R pohne svojimi hlavami v súlade s vypočítaným smerom.

Najskôr popíšeme druhú fázu, ktorá je jednoduchšia a použité techniky využijeme v prvej fáze v zložitejšej forme. Kľúčovým problémom v tejto fáze je posun hláv procesu R z pozícií reprezentujúcich číslo i na pozície reprezentujúce číslo $j = i+c$, kde $c \in \{-1, 0, 1\}$ je smer pohybu hlavy stroja M.

Najskôr ukážeme techniku, s akou si vie ľubovoľný proces, v našom prípade proces R, pohnúť jednu zvolenú hlavu h doľava. Ako prvý vytvoríme proces T, ktorý bude pomocou komunikačných symbolov odpočívavať n kôl. Proces S, ktorý má počas celého výpočtu všetky hlavy na prvom políčku, sa rozvetví a získame nový proces T. Proces T sa tiež rozvetví, čím získame proces T_0, ktorý bude slúžiť ako časovač odpočítavajúci n kôl. Proces T_0 na začiatku odvysiela symbol 0, signalizujúci nulté kolo a potom v každom kole pohne hlavou 0. Toto proces T_0 opakuje, až kým sa nedostane hlavou 0 na pravú zarážku, čo nastane za $n-1$ kôl. V tomto kole T_0 odvysiela symbol „$n-1$“ a prejde do akceptačného stavu, kde zostane do konca simulácie, čo budeme nazývať, že proces *zomrie*. Na signál $n-1$ zareaguje proces T a vytvorí nový proces T_0. To znamená, že procesy T_0 vysielajú signály $0, \varepsilon, \varepsilon, \ldots, \varepsilon, n-1, 0, \varepsilon, \ldots, n-1, \ldots$, pričom od signálu 0 do $n-1$ prejde presne $n-1$ kôl. Tento odpočet použijeme na posun hlavy h procesu R doľava. Aby sme neblokovali komunikáciu iným procesom, použijeme časové multixplexovanie. Časové multiplexovanie, ktoré tu využijeme, je technika, ktorá umožňuje k procesom naraz komunikovať na jednom kanáli. Celý výpočet sa spomalí k-krát a každý proces komunikuje iba raz za k kôl.

Teraz už môžeme posunúť hlavu h doľava. Proces S vytvorí proces R_0, ktorý má všetky hlavy na začiatku. Všetky hlavy procesu R_0, postupne po jednej hlave, premiestnime na pozíciu, akú má táto hlava v procese R. Nakoniec bude mať proces R_0 hlavy na rovnakých pozíciách, ako mal pôvodne proces R. Nech teda hlava j procesu R je na políčku l. Proces R počká na signál 0 a potom v každom kole pohne hlavou j doprava. Keď sa dostane na pravú zarážku, dá signál procesu R_0, ktorý začne hýbať svojou hlavou j doprava. Keď



Obr. 2.1: Príklad posunu hlavy doľava. Proces R_0 si začne posúvať hlavu, keď dostane signál od procesu R a prestane, keď prijme signál od procesu T_0. Aby sme realizovali posun doľava, tak jeden krok vynecháme.

R_0 prijme signál $n-1$, zastane. Hlava j dokopy vykonala cyklický posun o n políčok, a teda je na tom istom políčku, ako bola v procese R. Postup zopakujeme pre všetky hlavy s výnimkou hlavy h, ktorú sme chceli posunúť doľava. V tomto prípade vynecháme prvý pohyb hlavou, čo znamená, že vykonáme cyklický posun o $n-1$ políčok. Po skončení má proces R_0 všetky hlavy na rovnakých pozíciach, ako mal R, okrem hlavy h, ktorá je posunutá o jedno políčko doľava. Nakoniec R pošle procesu R_0 svoj stav, do ktorého proces R_0 prejde. Proces R zomrie a R_0 sa stane novým procesom R. Obrázok 2.1 ilustruje posun hlavy doľava.

Ukázali sme postup, ako posúvať hlavu jedného konkrétneho procesu doľava. Prečo nemôžeme tento postup priamo použiť na simuláciu ľubovoľnej obojsmernej siete WPFA jednosmernou? Uvedomme si, že na tento posun sme potrebovali vytvorenie nového procesu, ktorého zviazanie s pôvodným procesom je realizované číslom kanála. Po presunutí hláv na nový proces musíme cez komunikačný kanál poslať informáciu o stave. Kanálov je však iba konečne veľa, čo dovoľuje pohnúť hlavou doľava iba konečnému počtu procesov.

Teraz sme pripravení vyriešiť kľúčový problém druhej fázy: posunúť hlavy procesu R na pozície reprezentujúce číslo $j = i + c$ v sústave so základom n, kde c je smer pohybu hlavy simulovaného stroja M a $i = (i_0, i_1, \ldots, i_{k-1})_n$ je číslo políčka pásky, na ktorom mal stroj M hlavu. Ak $c = 0$, hlavy už sú na správnej pozícii. V prípade, že $c = -1$ a $i_{k-1} \neq 0$, t.j. hlava $k-1$ nie je na ľavej zarážke, tak použijeme predchádzajúci postup na posun

hlavy $k-1$ doľava. V druhom prípade, $i_{k-1}=0$, nech $i_l \neq 0$ a $i_{l+1}=\cdots=i_{k-1}=0$, hlavu l posunieme doľava a hlavy $l+1,\ldots,k-1$ budeme posúvať doprava, až kým sa nedostanú na pravú zarážku. Prípad $c=1$ je symetrický. Či $i_l=n-1$ zistíme tak, že sa pozrieme, či je hlava l na pravej zarážke.

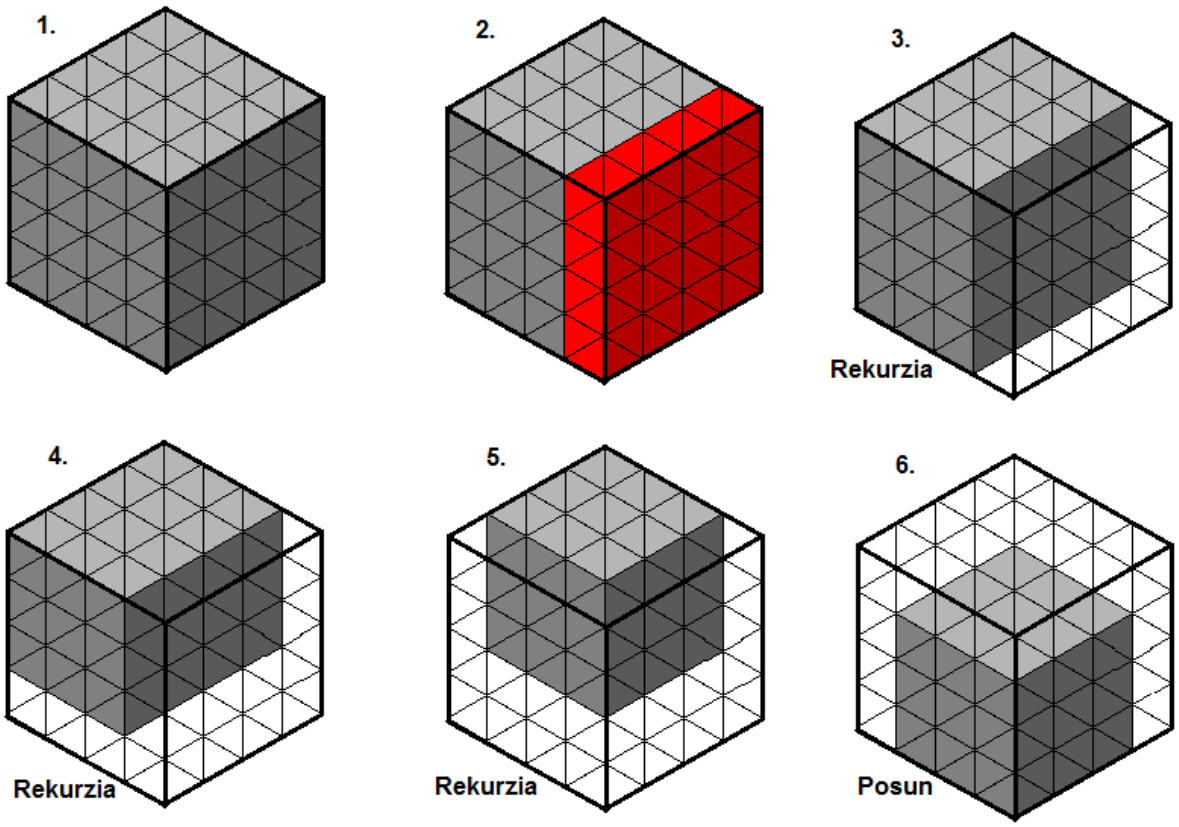
Teraz sa vrátime a popíšeme prvú fázu. Kľúčovým problémom tejto fázy je zistenie aktuálneho písmena pod hlavou stroja M. Uvedomme si, čo treba spraviť, aby sme našli proces $A_i \in A$, ktorý si pamätá aktuálne políčko pásky stroja M. A_i je proces, ktorý má umiestnenie svojich hláv rovnaké ako riadiaci proces R. Predstavme si, že procesy sú usporiadané do k-rozmernej hyperkocky, v ktorej umiestnenie procesu v i-tej dimenzii zodpovedá pozícii i-tej hlavy. V tejto kocke sú jedinými procesmi, s ktorými je možná komunikácia, procesy v rohoch kocky, pretože tie sa vedia jednoznačne identifikovať, keďže majú všetky svoje hlavy na zarážkach. V našom algoritme sa budeme snažiť dostať všetky procesy A_i v určitom deterministickom poradí do pozície, keď majú všetky hlavy na pravej zarážke. Túto pozíciu nazývame *koncová*. V koncovej pozícii proces odovzdá zapamätané písmeno a zomrie. Vytvoríme jeho nástupcu, ktorý preberie toto písmeno a ktorému musíme vrátiť jeho hlavy do pôvodnej pozície. Procesy nástupcov sú v stave *vybavený* do konca algoritmu. Tento algoritmus bude mať navyše vlastnosť, že ak niektoré z procesov A_i chýbajú, tak ostatné procesy týmto nie sú ovplyvnené. Preto ak spustíme súbežne algoritmus na sieť procesov A a aj na riadiaci proces R, tak proces R sa dostane do pozície, keď má všetky hlavy na pravej zarážke práve vtedy, keď aj nejaký proces A_j. A_j bude ten proces, ktorý mal na začiatku rovnaké pozície hláv – náš hľadaný proces. Proces R odvysiela signál, že má všetky hlavy na pravej zarážke a proces A_j prijme signál a označí sa. Týmto vyriešime kľúčový problém tejto fázy. Po skončení algoritmu budeme vedieť zistiť hľadané písmeno, ktoré si pamätá označený proces A_j a prípadne ho zmeniť.

Zaveďme označenia. Bez ujmy na všeobecnosti nech $K=\{0,\ldots,k-1\}$ je množina hláv. Nech $I \subseteq K$ a nech $p: I \to \{0,\ldots,n-1\}$ je funkcia, ktorá priradí hlavám z I pozíciu na páske. Potom podkockou $K_{I,p}$ označíme množinu procesov

$$K_{I,p}=\{A_i \in A \mid \text{pre hlavy } h \in I \text{ je ich pozícia } p(h)\},$$

teda hlavy v I sú fixované a ostatné voľné.

$F(m,I,p)$, kde $m \in \{0,\ldots,n-1\}$ je rekurzívny program, ktorý pre každý proces



Obr. 2.2: Príklad priebehu vybavovania kocky volaním rekurzívneho programu F. Každé „odrezanie“ stany sa vykoná rekurzívnym volaním. Následne sa zbytok kocky znova posunie na steny. Tento postup budeme opakovať, až kým nevybavíme celú kocku.

A_i z $K_{I,p}$ posunie všetky jeho hlavy z $K - I$ maximálne o m políčok a pričom sa každý proces dostane do koncovej pozície, zomrie a vytvorí sa k nemu nástupca. Tento nástupca bude mať po skončení programu svoje hlavy na tých istých pozíciách, ako pôvodný proces A_i na začiatku celej fázy. Opísanú funkcionalitu programu $F(m, I, p)$ nazveme *vybavením* podkocky $K_{I,p}$.

Priebeh vybavovania podkocky volaním rekurzívneho programu F ilustruje obrázok 2.2. Každé volanie vybaví jednu stenu podkocky. Keď je všetkých k stien vybavených, tak zostávajúce procesy si posunú voľné hlavy doprava, čím sa nové procesy dostanú na steny kocky. Následne znova vybavíme všetky steny, tento postup opakujeme, až kým nevybavíme celú podkocku.

Náš algoritmus pozostáva z volania $F(n-1, \emptyset, \emptyset)$. Parametre volania programu sa

odovzdajú pomocou dvoch procesov. Proces M si pamätá číslo m tak, že má 0-tú hlavu na pozícii $n - m - 1$. Proces P si pamätá I v stave a p v pozíciách svojich hláv. Volania podprogramov budú rekurzívne a hĺbka rekurzie bude ohraničená počtom hláv – k. Každý proces P a M si pamätá aktuálnu hĺbku rekurzie d. Rekurzívne volania budeme vykonávať sekvenčne, preto v každom okamihu existuje maximálne jeden proces P a M s konkrétnou hĺbkou rekurzie. Po skončení podprogramu s hĺbkou d proces P odvysiela symbol $d-1$, čo je signál pre proces s hĺbkou $d-1$, ktorý podprogram zavolal, aby pokračoval vo svojom výpočte. Potom procesy P a M zomrú. Počas celej rekurzie proces P s hĺbkou rekurzie d bude vydávať príkazy a komunikovať s procesom M a s procesmi z $K_{I,p}$ vo forme správ s prefixom d, aby nedošlo k miešaniu komunikácie. Hĺbku rekurzie d si nemusíme posielať ako ďalší parameter, lebo $d = |K - I|$.

Vstupné predpoklady programu F:

1) Nech sú vstupné parametre zapamätané pomocou procesov P, M.

2) Nech platí, že práve všetky nevybavené procesy z $K_{I,p}$ majú svoje hlavy z I na pravej zarážke.

3) Nech tiež majú všetky nevybavené procesy z $K_{I,p}$ všetky svoje hlavy z $K-I$ posunuté o $n-m$, oproti ich pôvodnej pozícii.

Pseudokód programu F:

1: **procedure** F(m, I, p)
2: Označíme všetky nevybavené procesy z $K_{I,p}$ ako d-aktívne,
kde $d \leftarrow |K-I|$ je hĺbka rekurzie. Využijeme predpoklad 2).
Ďalší kód sa bude vykonávať iba na d-aktívnych procesoch.
3: **if** I=K **then** ▷ *báza rekurzie*
4: Aktívny bude maximálne jeden proces Q.
5: Proces P sa stane nástupcom procesu Q, ktorý mu odovzdá zapamätané písmeno a zomrie.
6: **else** ▷ *telo rekurzie*
7: **for** $i \leftarrow m$ **downto** 0 **do**
8: **for** $\forall h \in K-I$ **do**
9: Vytvoríme kópiu procesu M – proces M_0.
10: Vytvoríme kópiu procesu P – proces P_0.
11: Procesu P_0 posunieme hlavu h na pozíciu i.
12: Pomocou procesov M_0, P_0 zavoláme F(i, $I \cup \{h\}$, $p \cup \{(h,i)\}$).
13: **end for**
14: Proces M posunie svoju 0-tú hlavu o jedno políčko doprava. Zodpovedá to dekrementácii i o jedna.
15: Všetky d-aktívne procesy pohnú hlavami z $K-I$ o jedno políčko doprava.
16: **end for**
17: **end if**
18: **end procedure**

Formálne dokážeme správnosť algoritmu. Matematickou indukciou vzhľadom na $|K-I|$ ukážeme platnosť nasledovnej implikácie:

Ak sú splnené predpoklady volania funkcie $F(m, I, p)$, tak dané volanie je korektné, a teda vybaví podkocku $K_{I,p}$.

Pripomenieme, že vybavenie podkocky $K_{I,p}$ znamená, že každý proces A_i z $K_{I,p}$ sa posúvaním hláv z $K - I$ dostane do koncovej pozície, zomrie a vytvorí sa nástupca, ktorý po skončení programu bude mať svoje hlavy na tých istých pozíciách, ako mal proces A_i na začiatku celej fázy.

1° Báza. $K = I$ ide teda o bázu rekurzie. Podľa predpokladov si proces P pamätá podkocku $K_{K,p}$ – jediný proces Q. P má teda rovnaké umiestnenie hláv, ako mal proces Q na začiatku fázy. Navyše proces Q má všetky svoje hlavy na pravej zarážke. Teda proces Q sa vie jednoznačne identifikovať. Potom, ako Q odovzdá zapamätané písmeno procesu P, Q zomrie a proces P sa stane nástupcom a následne prejde do stavu vybavený, kde zostáva až do konca fázy.

2° $I \neq K$, a teda nenastáva báza rekurzie. Z toho vyplýva, že volanie pozostáva z dvojitého for cyklu, v ktorom voláme $F(i, I \cup \{h\}, p \cup \{(h, i)\})$. Najskôr ukážeme, že tieto volania budú mať splnené predpoklady. Vyberme si ľubovoľné volanie $F(q, I \cup \{l\}, p \cup \{(l, q)\})$ a predpokladajme, že všetky predchádzajúce volania v rámci dvojitého for cyklu boli korektné. Z toho podľa IP vieme, že ovplyvňovali iba procesy v ich podkockách. Vieme, že celý vonkajší for cyklus sa už vykonal $(m-q)$-krát. Teda aj proces M si posunul svoju 0-tú hlavu $(m-q)$-krát (riadok 19). Hlava je na $(n-m-1)+(m-q) = (n-q-1)$-tom políčku čo znamená, že si naozaj pamätá číslo q. Podobne všetky zostávajúce procesy z podkocky $K_{I,p}$ si svoje hlavy z $K - I$ posunuli $(m - q)$-krát. Čiže ich hlavy z $K - I$ sa od začiatku fázy posunuli $(n - m - 1) + (m - q) = (n - q - 1)$-krát. Pozrime sa teraz, ako sa vykonal vnútorný for cyklus pre hlavu l z volania funkcie F. Proces P_0 si s využitím M_1 (ďalšej kópie procesu M) postupne posunie hlavu l na pozíciu q tak, že v každom kole si M_1 posunie hlavu 0 a P_0 hlavu l, až kým M_1 nepríde na pravú zarážku, t.j. q-krát.

Teraz môžeme overiť, či sú splnené predpoklady volania $F(q, I \cup \{l\}, p \cup \{(l, q)\})$.

1) Proces M_0 má 0-tú hlavu na políčku $(n-q-1)$ t.j. pamätá si číslo q. Proces P_0 si pamätá v stave $I \cup \{l\}$ a zároveň hlavy z I má na správnych pozíciach a hlavu l má na políčku q. Teda P_0 si pamätá správnu podkocku $K_{I\cup\{l\},p\cup\{(l,q)\}}$. T.j. vstupné parametre sú zapamätané správne.

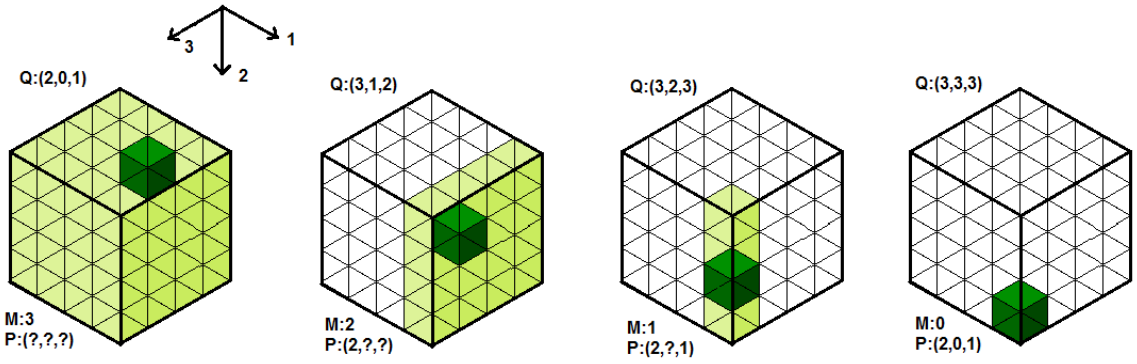
2) Podľa vstupných predpokladov majú všetky nevybavené procesy z $K_{I\cup\{l\},p\cup\{(l,q)\}}$ svoje hlavy z I na pravej zarážke. Zostáva teda iba ukázať, že majú aj hlavu l na pravej zarážke. Keďže procesy patria do podkocky $K_{I\cup\{l\},p\cup\{(l,q)\}}$, tak na začiatku fázy mali svoju hlavu l na políčku q. Hlava $l \in K - I$ sa teda od začiatku fázy pohla $(n-q-1)$-krát, a teda je na pravej zarážke. Zostáva ešte overiť, že každý nevybavený proces so všetkými hlavami z $I \cup \{l\}$ na pravej zarážke patrí do podkocky $K_{I\cup\{l\},p\cup\{(l,q)\}}$. Podľa predpokladu vieme, že tento proces patrí do podkocky $K_{I,p}$. Ak by jeho hlava l bola na začiatku fázy na pozícii väčšej ako q, tak by ho podľa predpokladov niektoré z predchádzajúcich volaní vybavilo. Ak by bola na pozícii menšej ako q, tak by po $(n-q-1)$ posunutiach nebola na pravej zarážke. To znamená, že hlava l musela byť na políčku q, a teda proces patrí do podkocky $K_{I\cup\{l\},p\cup\{(l,q)\}}$.

3) Vieme, že procesy z podkocky $K_{I,p}$ si svoje hlavy z $K-I$ posunuli $(n-q-1)$-krát od začatia fázy. Z toho vyplýva, že procesy z podkocky $K_{I\cup\{l\},p\cup\{(l,q)\}}$ majú svoje hlavy z $K-(I\cup\{l\})$ posunuté $(n-q-1)$-krát.

Ukázali sme, že všetky volania funkcie $F(i, I\cup\{h\}, p\cup\{(h,i)\})$ majú splnené počiatočné podmienky. Zostáva nám ukázať, že volanie $F(m, I, p)$ vybaví celú podkocku $K_{I,p}$. Nech Q je ľubovoľný proces z $K_{I,p}$ v stave pred začatím celej fázy. Nech q je číslo políčka, ktoré je najviac vpravo a je obsadené aspoň jednou hlavou z $K-I$. Nech je to hlava l. Potom $Q \in K_{I\cup\{l\},p\cup\{(l,q)\}}$, ale z IP vyplýva, že volanie funkcie $F(q, I\cup\{l\}, p\cup\{(l,q)\})$ ho vybaví. Tým sme dokázali správnosť procedúry F.

Potrebujeme ešte overiť, že prvé volanie $F(n-1, \emptyset, \emptyset)$ spĺňa počiatočné podmienky. Vytvoríme procesy P, M so všetkými hlavami na ľavej zarážke. Platí, že všetky procesy z $A = K_{\emptyset,\emptyset}$ majú hlavy z K posunuté o 0 políčok a tiež, že všetky hlavy z $I = \emptyset$ sú na pravej zarážke.

Priebeh rekurzie z pohľadu ľubovoľného procesu Q ilustruje obrázok 2.3. Kľúčovým bodom rekurzívneho programu F je odovzdávanie informácie medzi procesmi M a P.



Obr. 2.3: Príklad priebehu rekurzie z pohľadu ľubovoľného procesu Q. Každá kocka zodpovedá jednej hĺbke rekurzie, ktorá vybavuje proces Q. Všimnime si, ako sa pri každom volaní rekurzie skopíruje informácia z procesu M do procesu P, čím postupne odhaľujeme pôvodnú pozíciu procesu Q.

Proces M, tým, že si posúva hlavu spolu so všetkými voľnými hlavami procesu Q, si pamätá, koľko krokov už všetky voľné hlavy vykonali (druhý predpoklad). Keď sa niektorá vybraná voľná hlava dostane na pravú zarážku, tak proces M vie, kde sa pôvodne nachádzala. Táto hlava sa stane fixovanou a informáciu o jej pôvodnom umiestnení odovzdá proces M procesu P, ktorý je zodpovedný za uchovávanie informácie o fixných hlavách (riadok 11).

Každý krok DTS M sme vyššie uvedeným postupom simulovali. Počas simulácie akceptujeme, ak sa R dostane do stavu, v ktorom si pamätá akceptačný stav.

Zostáva vyriešiť, ako pripravíme sieť do počiatočnej konfigurácie stroja M. Použijeme upravenú konštrukciu z prvej fázy. Konštrukcia sa bude týkať iba procesov P, M a v mieste, kde nanovo vytvárame procesy A_i, ich iba vytvoríme. Táto konštrukcia však vytvorí viacero duplikátov jedného procesu. Preto na ne ešte raz spustíme pôvodnú konštrukciu z prvej fázy. Keďže nástupca sa vždy vytvorí iba jeden, zbaví nás to duplikátov. Potrebujeme ešte zabezpečiť, aby si procesy $A_0, \ldots, A_{n-2}$ pamätali vstupné slovo. Najskôr si procesy $A_1, \ldots, A_{n-1}$, t.j. tie, ktoré majú 0-tú hlavu mimo zarážiek a ostatné hlavy na ľavej zarážke, zapamätajú písmeno pod 0-tou hlavou. Vstupné slovo máme však zapamätané tak, že je posunuté o jedno políčko doprava, preto do δ-funkcie stroja M pridáme podprogram, ktorý na začiatku posunie obsah pásky doľava. □

Dôsledok 2.2 *Pre ľubovoľné $k \in \mathbb{N}^+$ platí:*

$$1WPFA(k) = 2WPFA(k).$$

2.2 Jednosmernosť WPTM

Ukážeme, že aj po pridaní kanálovej pásky jednosmernosť vstupných hláv neprekáža a jednosmerný model je ekvivalentný obojsmernému.

Veta 2.3 *Pre ľubovoľnú funkciu $S(n) : \mathbb{N} \to \mathbb{N}$, $S(n) \geq 1$ a ľubovoľné $k \in \mathbb{N}^+$ platí:*

$$1WPM(k,0)CHANNEL(S(n)) = \bigcup_{c \geq 0} DSPACE(n^k c^{S(n)}).$$

Dôkaz. Tvrdenie je rozšírením vety 2.1. Keďže podľa [2] platí:

$$2WPM(k,0)CHANNEL(S(n)) = \bigcup_{c \geq 0} DSPACE(n^k c^{S(n)}),$$

tak stačí ukázať iba inklúziu „$\supseteq$“. Majme DTS M pracujúci v priestore $n^k c^{S(n)}$ pre konkrétne $k, c, S(n)$. Stroj M budeme simulovať sieťou WPTM W s kanálovou zložitosťou $S(n)$ a s k jednosmernými hlavami na vstupnej páske.

Myšlienka dôkazu je podobná ako v predchádzajúcom dôkaze a konštrukcie z neho budeme aj priamo využívať. Ukážme, ako sa bude pamätať konfigurácia TS M. Políčka pracovnej pásky stroja M budeme číslovať tak, že nulté políčko, políčko na ktorom mal M hlavu na začiatku celého výpočtu, bude mať index 0. Indexom ostatných políčok bude vzdialenosť od nultého políčka s kladným znamienkom, ak sú napravo od neho, a so záporným, ak sú naľavo.

1. Stav stroja M si bude pamätať proces R vo svojom stave.

2. Pracovnú pásku stroja M si rozdelíme na potenciálne $2c^{S(n)}$ úsekov dĺžky n^k. Nech je hlava v s-tom úseku, $s \in \{-c^{S(n)} + 1, \ldots, c^{S(n)} - 1\}$. Číslo úseku s si proces R zapamätá na kanálovej páske v sústave so základom c, na čo mu stačí veľkosť jeho pásky. Znamienko čísla s si tiež pamätá na kanálovej páske. Pozíciu v rámci s-tého

úseku si pamätáme rovnako ako v predchádzajúcom dôkaze. Pozíciu – číslo i – si v sústave so základom n zapíšeme ako $i = (i_0, i_1, \ldots, i_{k-1})_n$. Potom R si pamätá i v pozíciach hláv na vstupnej páske.

3. Na zapamätanie obsahu i-teho políčka s-tého úseku pracovnej pásky použijeme proces A_i^s. Proces A_i^s si v stave pamätá obsah políčka. Čísla i a s si pamätá rovnakým spôsobom ako proces R.

Všetky procesy A_i^s rozdeľme na skupiny procesov A^s, veľkosti n^k, s procesmi s rovnakým obsahom kanálovej pásky. Skupina A^s si pamätá celý s-tý úsek. Keďže ale funkcia $S(n)$ nemusí byť konštruovateľná, tak budeme skupiny A^s vytvárať dynamicky. Ak si potrebujeme zapamätať políčko z nového úseku, tak pre tento úsek vytvoríme novú skupinu procesov A^s.

Simulácia jedného kroku TS M bude opäť prebiehať v dvoch fázach:

I [identifikácia symbolu pod hlavou] Proces R si pomocou pozícií svojich hláv na vstupnej páske a obsahu kanálovej pásky pamätá aktuálnu pozíciu hlavy stroja M. V prvej fáze nájdeme taký proces A_i^s, ktorý si pamätá obsah aktuálneho políčka. Teda taký proces, ktorý má rovnaký obsah kanálovej pásky a rovnaké pozície hláv. Nájdený proces sa označí zmenou stavu.

II [aplikácia prechodovej funkcie] V druhej fáze proces A_i^s pošle zapamätané políčko pásky procesu R, ktorý na základe toho z δ-funkcie stroja M vypočíta nový stav, smer pohybu hlavy a nový symbol. Nový symbol pošle naspäť procesu A_i^s, ktorý si ho zapamätá v stave. Potom proces R pohne svojimi hlavami a zmení obsah svojej kanálovej pásky v súlade s vypočítaným smerom.

V prvej fáze použijeme algoritmus z vety 2.1. Komunikácia medzi procesom R a procesmi A_i^s však bude prebiehať len na kanáli, na ktorom je naladený proces R. Preto sa algoritmus vykoná iba na procesoch tej skupiny A^s, v ktorej majú všetky procesy rovnaký obsah kanálovej pásky ako proces R. Po skončení rekurzívneho algoritmu sa označí proces, ktorý má rovnaké pozície hláv na vstupnej páske a tiež aj rovnaký obsah kanálovej pásky – náš hľadaný proces.

V druhej fáze, po odkomunikovaní písmena, potrebujeme zmeniť zapamätanú pozíciu hlavy stroja M, ktorú má uloženú proces R v pozíciach hláv a obsahu kanálovej pásky. V prípade, že nedošlo k pohybu medzi rôznymi úsekmi pracovnej pásky stroja M, zmeníme pozície hláv procesu R rovnakým spôsobom ako v predošlom dôkaze. V prípade, že došlo k prechodu do iného úseku pracovnej pásky a hlava sa pohla doprava, resp. doľava, tak zapamätanú pozíciu v rámci úseku nastavíme pozíciami hláv na 0, resp. $n^k - 1$ tak, že presunieme všetky hlavy na ľavú, resp. pravú zarážku. Následne inkrementujeme, resp. dekrementujeme obsah kanálovej pásky. Potrebujeme ešte zistiť, či sme neprešli na nový úsek, ktorému sme ešte nevytvorili skupinu procesov A^s, ktorá by si ho pamätala. Proces R vyšle signál *ping* na svojom aktuálnom kanáli, na ktorý procesy A_i^s odpovedajú. Ak nedostaneme odpoveď, musíme vytvoriť novú skupinu procesov s obsahom kanálovej pásky rovnakej, ako má proces R. Vytvoríme kópiu procesu R, ktorému posunieme hlavy na vstupnej páske na ľavú zarážku. Z neho za pomoci konštrukcie inicializácie procesov z predchádzajúceho dôkazu vytvoríme novú skupinu procesov.

Dôkaz správnosti konštrukcie je zrejmý z jej popisu. □

Dôsledok 2.4 *Pre ľubovoľnú funkciu $S(n) : \mathbb{N} \to \mathbb{N}$, $S(n) \geq 1$ a ľubovoľné $k \in \mathbb{N}^+$ platí:*

$$1WPM(k,0)CHANNEL(S(n)) = 2WPM(k,0)CHANNEL(S(n)).$$

Poznámka. Veta 2.2 a dôsledok 2.4 sa dajú zovšeobecniť na model s pracovnými páskami. Upravíme simuláciu siete WPTM Turingovým strojom v práci [2], tak, aby okrem kanálovej pásky simuloval aj pracovnú pásku.

Predchádzajúca simulácia využíva veľké množstvo procesov. Pri každom kroku simulovaného stroja sa znehodnotí $O(n^k)$ procesov. Za celý výpočet sa použije $O(T_M(n)n^k) = O(c^{c^{S(n)n^k}})$ procesov. Ak by však $S(n) = \Omega(\log n)$, tak môžeme využívať procesy efektívnejšie.

Veta 2.5 *Pre ľubovoľnú funkciu $S(n) : \mathbb{N} \to \mathbb{N}$, $S(n) = \Omega(\log n)$ a ľubovoľnú konštantu $k \in \mathbb{N}^+$ platí:*

$$\bigcup_{c>0} 1WPM(1,0)CHANNEL(S(n))PROCESS(c^{S(n)}) = \bigcup_{c>0} DSPACE(n^k c^{S(n)}).$$

Dôkaz. Inklúzia „$\subseteq$“ triviálne vyplýva z predchádzajúcej vety. Ak $S(n) = \Omega(\log n)$, tak $\bigcup_{c>0} DSPACE(n^k c^{S(n)}) = \bigcup_{c>0} DSPACE(c^{S(n)})$. Opačnú inklúziu dokážeme podobnou simuláciou, ako vo vete . Uvedieme iba rozdiely. Na zapamätanie $c^{S(n)}$ políčok nám stačí $c^{S(n)}$ procesov s rôznymi obsahmi kanálovej pásky. Nepotrebujeme využívať pozície vstupných hláv. Preto pri identifikácii procesu, ktorý si pamätá obsah políčka pod hlavou simulovaného stroja, nehýbeme hlavami, čo znamená, že tieto procesy sa dajú znova použiť. Vstupnú hlavu siete využijeme iba na začiatku simulácie, keď skopírujeme obsah vstupu do prvých n procesov. Celkový počet procesov, ktoré použijeme, je $c^{S(n)} + d$, kde d je počet pomocných procesov. Aby sme zachovali procesorovú zložitosť $c^{S(n)}$, na zapamätanie pásky použijeme iba $c^{S(n)} - d$ procesov, pričom prvý z nich si bude pamätať v stave d políčok namiesto jedného. □

Všimnime si, že pri simulácii v inklúzii „$\supseteq$“ sme na zapamätanie $c^{S(n)}$ políčok použili presne $c^{S(n)}$ procesov. Pri opačnej inklúzii nám $c^{S(n)}$ políčok vo všeobecnosti nestačí na simulovanie $c^{S(n)}$ procesov.

2.3 Bezzarážkový WPFA

Pri vyššie uvedených tvrdeniach sme uvažovali, že vstupné slovo je z oboch strán ohraničené zarážkami. Tento fakt sme aj využívali pri zistení, či daný proces má hlavu na konci. Ako zaujímavosť ukážeme, že jednosmerný WPFA aj bez zarážok vie rozpoznať jazyky silnejšie ako regulárne jazyky.

Definícia 2.6 *Jednohlavý jednosmerný bezzarážkový WPFA je upravený WPFA, ktorého vstupná páska nie je ohraničená zarážkami. Po prečítaní posledného políčka pásky proces prejde do stavu, v ktorom zostáva do konca výpočtu. Ak proces vysielal stav, tak sa bude tento stav vysielať do konca výpočtu. Triedu jazykov akceptovanú týmto modelom nazveme 1bWPFA(1).*

Táto definícia sa podobá na definíciu konečných automatov alebo deterministických zásobníkových automatov. Stroj sa musí po prečítaní vstupu „okamžite“ rozhodnúť o akceptácii. Definícia tiež pripomína prehľadávanie grafov s „čiernymi dierami“, pretože proces, ktorý prečíta posledné písmeno vstupu sa „stratí“.

Veta 2.7

$$1bWPFA(1) \supseteq \bigcup_k 2FA(k),$$

kde $2FA(k)$ *je trieda akceptovaná* k*-hlavovým obojsmerným konečným automatom.*

Dôkaz. Simulovať budeme k-hlavý obojsmerný konečný automat M pomocou jednohlavej bezzarážkovej siete WPFA W. Nech H je množina hláv stroja M. Nech Σ je vstupná abeceda.

Keďže po prečítaní posledného písmena vstupu sa proces musí rozhodnúť o akceptácii a prípadne odvysielať akceptačný symbol, tak celú simuláciu spustíme pre všetky možné posledné písmená vstupu $a \in \Sigma$. Výsledky akceptácie pre každé písmeno si zapamätá proces R. Keď simulácia skončí pre všetky možné posledné písmená vstupu, tak všetky procesy okrem R akceptujú a proces R prečíta skutočné posledné písmeno a rozhodne sa, či akceptovať. Ukážeme, ako si sieť W pamätá konfiguráciu stroja M.

1. Stav stroja M si bude pamätať proces R (riadiaci proces) vo svojom stave.

2. Pozície hláv si pamätá množina procesov $A = \{A_h | h \in H\}$. Pozícia h-tej hlavy stroja M je rovnaká ako pozícia hlavy procesu A_h. Výnimkou je, ak h-ta hlava stroja M je na:

   a) ľavej zarážke – hlava procesu A_h je na prvom políčku vstupnej pásky a informáciu, že je na zarážke, si pamätá v stave.

   b) na poslednom písmene vstupu – hlava procesu A_h je na predposlednom políčku vstupnej pásky a v stave si pamätá, že je na poslednom písmene.

   c) na pravej zarážke – hlava je na predposlednom políčku a v stave si pamätá, že je na zarážke.

3. Aké je posledné písmeno vstupu, ktoré práve simulujeme, si pamätajú všetky procesy v stave.

Najskôr ukážeme, ako vytvoriť časovač odpočítavajúci $n-1$ kôl. Majme procesy T a T_0 s hlavou na prvom políčku. Proces T_0 si bude hlavu posúvať doprava, pričom vždy, keď prečíta nové písmeno, prejde do akceptačného stavu a vyšle akceptačný symbol. V ďalšom

kole si T_0 posunie hlavu doprava a vyšle symbol *ok*. V prípade, že proces T_0 sa dostal na posledné písmeno svojho vstupu, zostane v akceptačnom stave a symbol *ok* sa neodvysiela. Keď proces T zistí, že symbol *ok* nebol odvysielaný, tak sám odvysiela symbol *stop*. Od začiatku behu procesu T_0 do signálu *stop* sa signál *ok* odvysiela presne $(n-1)$-krát. Preto všetky procesy, ktoré sa chcú riadiť týmto časovačom, spravia krok na každý *ok* signál, až pokiaľ nezachytia signál *stop*.

Pri simulácii jedného kroku stroja M musíme pohnúť hlavami podľa smeru z δ funkcie. Postupne posunieme hlavu všetkým procesom A_h. Najskôr ukážeme, ako vykonať posun hlavy doľava. Hlavu premiestnime na nástupcu – proces A'_h. Nech je hlava pôvodne na p-tom políčku. Spustíme časovač odpočítavajúci $n-2$ kôl. Hlavu budeme posúvať doprava, vždy na symbol *ok* od časovača. Okrem toho budeme vysielať symbol *h-ok* rovnakým spôsobom ako časovač, teda čítať budeme v akceptačnom stave a hýbať a komunikovať až potom. Symbol *h-ok* dostane riadiaci proces $(n-p-1)$-krát. V kole, v ktorom nedostal tento symbol sa vytvorí nástupný proces A'_h s hlavou na prvom políčku. Časovač už odpočítal $(n-p)$ kôl (počas jedného kola sa vytvoril proces A'_h). Proces A'_h si bude posúvať hlavu doprava na každý symbol *ok* od časovača. To sa vykoná $(p-1)$-krát, pokiaľ časovač nevyšle symbol *stop*. Preto hlava procesu A'_h bude na políčku $p-1$, teda posunutá doľava. Výnimkou bude prípad, keď hlava bola na prvom políčku, vtedy dôjde iba k zmene stavu.

Ak chceme pohnúť hlavu doprava, tak ju najskôr pohneme doľava hore uvedeným postupom a potom ešte o dve políčka doprava. Navyše ešte použijeme ďalší časovač, ktorý zaistí, že sa hlava pohne doprava maximálne $(n-2)$-krát. Hlava sa teda nedostane na posledné políčko. V prípade, že máme ešte urobiť pohyb doprava viac ako $(n-2)$-krát, iba zmeníme stav. Tým si zapamätáme, že sa nachádzame na poslednom políčku, resp. pravej zarážke.

Pre simuláciu jedného kroku automatu M si riadiaci proces R najskôr postupne vypýta písmená pod hlavami procesov A_h. Potom im R pošle z δ funkcie vypočítaný smer pohybu. Každý proces A_h si podľa toho posunie hlavu.

Počas simulácie všetky procesy, ktoré už prečítali celý vstup, prešli do akceptačného stavu a vysielajú akceptačný symbol. Ak sa počas simulácie R dostane do situácie, keď si pamätá akceptačný stav, vyšle signál ostatným procesom, aby akceptovali. Všetky procesy

sú v akceptačnom stave a vysielajú akceptačný symbol čo znamená, že sieť W akceptovala. V opačnom prípade bude proces R brániť akceptácii. □

Dôsledok 2.8

$$DLOGSPACE \subseteq 1bWPFA(1) \subseteq DSPACE(n)$$

Dôkaz. Prvá inklúzia platí, pretože:

$$DLOGSPACE = \bigcup_k 2FA(k).$$

Platnosť druhej inklúzie je zrejmá, keďže:

$$1bWPFA(1) \subseteq 1WPFA(1) = DSPACE(n).$$

□

Kapitola 3

Vplyv počtu procesov

V tejto kapitole preskúmame vplyv počtu procesov na silu WPTM. Neobmedzený WPTM s jedným procesom je ekvivalentný Turingovému stroju. Preto budeme skúmať vplyv počtu procesov na priestorovo ohraničené WPTM. Vo všetkých výsledkoch charakterizujeme silu skúmanej triedy WPTM pomocou vhodne priestorovo ohraničeného Turingového stroja. Ako prvé charakterizujeme silu k-hlavových WPTM s celkovou priestorovou zložitosťou $S(n)$ a počtom procesov najviac $O(n^{k-1}c^{S(n)})$.

Lemma 3.1 *Pre ľubovoľné $k \in \mathbb{N}^+$, funkcie $S(n) : \mathbb{N} \to \mathbb{N}$, $S(n) \geq 1$ a $P(n) : \mathbb{N} \to \mathbb{N}$, $P(n) \geq 1$ platí:*

$$WPM(k,1)SPACE(S(n))PROCESS(P(n)) \subseteq DSPACE(P(n)(S(n) + k \log n)).$$

Dôkaz. Sieť WPTM W s maximálnym počtom procesov $P(n)$, kanálovou a pracovnou priestorovou zložitosťou $S(n)$ a k hlavami budeme simulovať TS M s priestorovou zložitosťou $P(n)(S(n)+k \log n)$. Keďže nás nezaujíma čas simulácie, bez ujmy na všeobecnosti môžeme predpokladať, že TS M je dvojpáskový. Stroj M si na prvej páske pamätá konfigurácie procesov siete WPTM W, teda jednu úroveň výpočtového grafu $G(w)$. Na zapamätanie konfigurácie jedného procesu potrebujeme $O(S(n)+k \log n)$ pásky, kde využijeme $k \log n$ pásky na uloženie informácie o pozíciách hláv. Na zapamätanie konfigurácie celej siete W stačí $O(P(n)(S(n) + k \log n))$ pásky. Na druhej páske si pamätáme maximálny kanál siete W. Realizovanie kroku výpočtu siete W je priamočiare. V prvej fáze aplikujeme δ-funkciu na všetky zapamätané procesy. V prípade, že došlo k univerzálnemu vetveniu, t.j. vytvoreniu nových procesov, konfiguráciu nových procesov pripíšeme na koniec pásky.

V druhej fáze vyriešime komunikáciu. Na druhú pásku si zapíšeme maximálny použitý kanál. Na každom kanáli, menšom alebo rovnom ako zapamätaný maximálny kanál, skontrolujeme, či vysielacie procesy vysielajú rovnaký symbol. Tento symbol zapíšeme všetkým načúvacím procesom.

Správnosť konštrukcie je zrejmá z popisu. □

V nasledujúcich lemách ukážeme opačnú inklúziu. Budeme simulovať Turingov stroj sieťou WPTM s daným počtom procesov. Pre väčšiu prehľadnosť rozdelíme dôkaz do dvoch lem. V prvej leme zafixujeme veľkosť pracovnej abecedy simulovaného stroja. V druhej leme ukážeme, že nárast veľkosti pracovnej abecedy spôsobí konštantné zväčšenie počtu potrebných procesov.

Lemma 3.2 *Pre ľubovoľné $c, k \in \mathbb{N}^+$, konštruovateľnú funkciu $S(n) \geq 1$ a funkciu $P(n) : \mathbb{N} \to \mathbb{N}$, ktorá spĺňa $P(n) \in O(n^{k-1}c^{S(n)})$, $P(n) \geq 1$ platí:*

$$WPM(k,1)SPACE(S(n))PROCESS(P(n)) \supseteq$$
$$DSPACE(P(n)(S(n) + \log_{|\Sigma|} n))ALPH(|\Sigma|).$$

Dôkaz. TS M s pracovnou abecedou Σ a priestorovou zložitosťou $P(n)(S(n) + \log_{|\Sigma|} n)$ budeme simulovať sieťou WPTM W s k-hlavami, kanálovou a pracovnou páskou veľkosti $S(n)$ a s maximálne $P(n)$ procesmi. Bez ujmy na všeobecnosti predpokladajme, že páska stroja M je jednostranne nekonečná. Pásku stroja M si rozdelíme na $P(n)$ úsekov veľkosti $S(n) + \log_{|\Sigma|} n$. Konfiguráciu stroja M si sieť W pamätá tak, že každý proces A_i si pamätá i-ty úsek pásky stroja M. Keďže $S(n)$ je konštruovateľné, proces A_i vie presne ohraničiť svoj úsek pásky stroja M. Využijeme dve rôzne techniky, pri každej využijeme časť prostriedkov, ktoré máme k dispozícii:

Identifikovanie susedov Procesy $A = (A_0, \ldots, A_{P(n)-1})$ zoradíme do postupnosti A. Každý proces bude vedieť identifikovať svojho ľavého a pravého suseda v postupnosti A. Keďže $P(n) \in O(n^{k-1}c^{S(n)})$, každý proces jednoznačne identifikujeme pozíciami prvých $k-1$ hláv a obsahom kanálovej pásky veľkosti $S(n)$. Veľkosť kanálovej abecedy Σ_c zvolíme tak, aby $n^{k-1}|\Sigma_c|^{S(n)} \geq P(n)$.

Uchovanie úseku pásky Každý proces si bude pamätať $S(n) + \log_{|\Sigma|} n$ políčok pásky stroja M. Prvých $S(n)$ políčok si pamätá na pracovnej páske. Zostávajúcich $\log_{|\Sigma|} n$ políčok si pamätá pozíciou poslednej hlavy tak, že hlava bude na políčku j, ak obsah $\log_{|\Sigma|} n$ políčok je číslo j v číselnej sústave zo základom $|\Sigma|$.

Technika identifikovania susedov bola použitá v práci Pavla Mravca [2]. Stručne popíšeme konštrukciu. Nech proces A_i chce identifikovať svojho suseda $A_j, j = i + c$. Proces A_i si pamätá číslo i obsahom kanálovej pásky a v pozíciach $k - 1$ hláv. Ako prvé, proces A_i zmení pozície hláv, prípadne obsah kanálovej pásky, aby reprezentovali číslo j. Následne sa snaží nájsť proces A_j, ktorý má rovnaký obsah kanálovej pásky a rovnaké pozície $k-1$ hláv. Komunikácia bude prebiehať na kanáli určeného kanálovou páskou, takže zostáva určiť, ktorý proces má aj pozície daných hláv rovnaké. Postupne porovnáme pozície hláv. Na porovnanie pozície hlavy h použijeme proces K – krokovač, ktorý má hlavy na ľavej zarážke. Všetky procesy z A také, ktoré sú naladené na rovnaký kanál ako A_i, si spolu s procesom K naraz posúvajú hlavu h doprava, až kým nenarazia na pravú zarážku, kde zmenia smer posunu. Posúvanie končí, keď sa hlava h procesu A_i dostane na pravú zarážku. Označíme procesy, ktoré majú rovnakú pozíciu hlavy h. Následne použijeme proces K, aby sme hlavu h vo všetkých procesoch vrátili na pôvodné miesto. Procesy vykonávajú opačný pohyb, až kým sa hlava h procesu K nedostane opäť na na ľavú zarážku, čím sa hlava h vo všetkých procesoch vráti na pôvodnú pozíciu. Týmto spôsobom porovnáme pozície všetkých hláv a identifikujeme hľadaný proces A_j. Následne si proces A_i vráti hlavy a kanálovú pásku na pôvodnú pozíciu, čím si pamätá číslo i.

Keďže funkcia $P(n)$ nemusí byť konštruovateľná, procesy z postupnosti A budeme vytvárať dynamicky. Začínať budeme s jedným procesom A_0. V prípade, že pri identifikácii susedov sme nenašli suseda, znamená to, že ho musíme vytvoriť. Proces A_i si počas hľadania pozíciami svojich $k - 1$ hláv a kanálovou páskou pamätá číslo $j = i + c$. Proces A_i vytvorí proces A_j ako svoju kópiu. Následne si proces A_j nastaví obsah pracovnej pásky a poslednú hlavu tak, aby si pamätal úsek pásky stroja M, vyplnený iba blankmi.

Popíšme uchovanie úseku pásky stroja M jedným procesom. Každý proces A_i si pamätá úsek pásky stroja M veľkosti $S(n) + \log_{|\Sigma|} n$. Prvých $S(n)$ políčok si pamätá na pracovnej páske. Zostávajúcich $\log_{|\Sigma|} n$ políčok si pamätá pozíciou poslednej hlavy. Okrem toho si

potrebujeme pamätať aj pozíciu hlavy stroja M na pracovnej páske. Ak si proces A_i pamätá úsek na ktorom sa nachádza hlava, označí sa zmenou stavu. Ak sa hlava nachádza v časti dĺžky $S(n)$ ktorú si A_i pamätá na svojej pracovnej páske, tak pozíciu hlavy stroja M si A_i zapamätá pozíciou pracovnej hlavy. Ak sa hlava stroja nachádza v časti dĺžky $\log_{|\Sigma|} n$ ktorú si A_i pamätá pozíciou hlavy, potom si jej pozíciu bude pamätať špeciálny proces H. Pozícia prvej hlavy procesu H bude rovnaká ako pozícia hlavy stroja M v rámci časti dĺžky $\log_{|\Sigma|} n$, ktorú si A_i pamätá pozíciou hlavy. Ak chceme zistiť alebo zmeniť obsah zapamätaného úseku pod aktuálnou pozíciou hlavy, tak v prípade, že aktuálne políčko je v časti zapamätanej na pracovnej páske procesu A_i, jednoducho prečítame a zmeníme príslušné políčko pracovnej pásky. V prípade, že ide o políčko v časti zapamätanej pozíciou hlavy, využijeme nasledovné aritmetické operácie.

Posledná hlava procesu je na políčku i, čím si pamätá časť pásky dĺžky $\log_{|\Sigma|} n$. Túto časť budeme indexovať odzadu: páska má obsah $i_{\log_{|\Sigma|} n-1}, \ldots, i_1, i_0$, ak číslo i v číselnej sústave so základom$|\Sigma|$ je práve $i = (i_{\log_{|\Sigma|} n-1}, \ldots, i_1, i_0)_{|\Sigma|}$. Aby sme zistili alebo zmenili obsah k-tého políčka v tejto časti, potrebujeme zistiť k-tu cifru čísla i, respektíve pripočítať alebo odpočítať k číslu i číslo $|\Sigma|^k$. Ukážeme, ako uskutočniť jednoduché aritmetické operácie na číslach zapamätaných pozíciou hlavy.

$\mathbf{a \pm 1}$ Majme proces, ktorý má hlavu na pozícii i, čím si pamätá číslo i, potom inkrementáciu, resp. dekrementáciu, uskutočníme posunom hlavy doprava, resp. doľava.

$\mathbf{a = 0}$ Test na nulu. Pozrieme sa, či je hlava na ľavej zarážke.

$\mathbf{a \pm b}$ Majme dva procesy A a B, ktoré si pamätajú čísla a a b. Proces A si každé kolo inkrementuje, resp. dekrementuje, zapamätané číslo. Zároveň si proces B dekrementuje zapamätané číslo. Postup opakujeme, kým $a \neq 0$. Na konci si proces A pamätá číslo $a \pm b$.

$\mathbf{a * |\Sigma|}$ Majme dva procesy A a B, ktoré si pamätajú čísla a a $b = 0$. Vykonáme $b \leftarrow b + a$ $|\Sigma|$-krát.

$\mathbf{|\Sigma|^a}$ Majme dva procesy A a B, ktoré si pamätajú čísla a a $b = 1$. a-krát vykonáme $b \leftarrow b * |\Sigma|$.

a div $|\Sigma|$ Majme proces A, ktorý si pamätá číslo a a proces B, do ktorého uložíme výsledok. Každé kolo vykonáme $a \leftarrow a - |\Sigma|$ a inkrementujeme číslo zapamätané v procese B, skončíme, ak $a < |\Sigma|$. Proces B si pamätá a div $|\Sigma|$.

a mod $|\Sigma|$ Vykonáme $a - |\Sigma|(a \text{ div } |\Sigma|)$.

(k-ta cifra čísla a) Poslednú cifru čísla i zistíme ako a mod $|\Sigma|$. Následne vykonáme „bitový posun“ čísla i doprava, pomocou operácie a div $|\Sigma|$. Operácie a mod $|\Sigma|$ a $a \leftarrow a$ div $|\Sigma|$ zopakujeme k-krát, čím zistíme k-tu cifru čísla a.

S pomocou týchto operácií vieme zistiť, aké je zapamätané políčko pod pracovnou hlavou, ktorej pozíciu si pamätá proces H. Na zistenie hodnoty políčka i_k vykonáme operáciu k-ta cifra čísla i. Na zmenenie zapamätanej hodnoty z jednotky na nulu, resp. nuly na jednotku, vykonáme $i \leftarrow i - |\Sigma|^k$, resp. $i \leftarrow i + |\Sigma|^k$. Pri žiadnej operácii neprekročí veľkosť medzivýsledku číslo n.

Všimnime si, že nemá zmysel použiť viac hláv na uchovávanie úseku pásky. Jednou hlavou si vieme pamätať úsek veľkosti $\log_{|\Sigma|} n$ a k hlavami iba $k \log_{|\Sigma|} n$.

Ukážeme, ako simulovať jeden krok stroja M s využitím techniky identifikácie susedov a uchovania úseku pásky. Procesy A si pamätajú obsah celej pásky stroja M. Aktuálnu pozíciu pracovnej hlavy si pamätáme označením procesu A_i, ktorý si pamätá daný úsek pásky a pozíciu v rámci úseku si pamätá pracovná hlava, resp. proces H. Ďalší proces V číta vstup rovnako ako hlava na vstupnej páske stroja M. Proces V si tiež pamätá stav stroja M. Proces V je naladený na rovnaký kanál ako proces A_i. Proces V si od procesu A_i vypýta zapamätané písmeno pod aktuálnou pozíciou hlavy. Podľa δ-funkcie stroja M zistí nové písmeno a pohyb vstupnej a pracovnej hlavy. Proces V si posunie svoju vstupnú hlavu požadovaným smerom a pošle nové písmeno a pohyb hlavy procesu A_i. Proces A_i si zapamätá nové písmeno a pohne svojou pracovnou hlavou, resp. pošle signál procesu H, aby pohol hlavou. Ak nastane pohyb hlavy mimo úsek zapamätaný procesom A_i, tak technikou identifikácie susedov nájde, prípadne vytvorí proces A_j, ktorý sa označí a bude si pamätať pozíciu hlavy. V prípade, že proces A_j má iný obsah kanálovej pásky ako majú procesy A_i a V, tak proces V zmení obsah svojej kanálovej pásky na rovnaký ako má proces A_j.

Sieť W akceptuje, ak sa proces V dostane do stavu, v ktorom si pamätá akceptačný stav. □

Lemma 3.3 *Pre ľubovoľné $c, k \in \mathbb{N}^+$, konštruovateľnú funkciu $S(n) \geq 1$ a funkciu $P(n) : \mathbb{N} \to \mathbb{N}$, ktorá spĺňa $P(n) \in O(n^{k-1}c^{S(n)})$, $P(n) \geq 1$ platí:*

$$\bigcup_{d>0} WPM(k,1)SPACE(S(n))PROCESS(dP(n)) \supseteq DSPACE(P(n)(S(n) + \log n)).$$

Dôkaz. Podľa lemy 3.2 pre každé $s \geq 2$ platí:

$$\begin{aligned} WPM(k,1)SPACE(S(n))PROCESS(P'(n)) \supseteq \\ DSPACE(P'(n)(S(n) + \log_s n))ALPH(s). \end{aligned} \tag{3.1}$$

Keďže platí:

$$P'(n)(S(n) + \log_s n) = P'(n)(S(n) + \frac{\log_2 n}{\log_2 s}) \geq \frac{P'(n)}{\log_2 s}(S(n) + \log_2 n), \tag{3.2}$$

potom

$$\begin{aligned} WPM(k,1)SPACE(S(n))PROCESS(P'(n)) \supseteq \\ DSPACE(\frac{P'(n)}{\log_2 s}(S(n) + \log_2 n))ALPH(s). \end{aligned} \tag{3.3}$$

Nech $P(n) = \frac{P'(n)}{\log_2 s}$, potom

$$\begin{aligned} WPM(k,1)SPACE(S(n))PROCESS(P(n)\log_2 s) \supseteq \\ DSPACE(P(n)(S(n) + \log_2 n))ALPH(s). \end{aligned} \tag{3.4}$$

Urobme zjednotenie cez všetky $s \geq 2$:

$$\begin{aligned} \bigcup_{s\geq 2} WPM(k,1)SPACE(S(n))PROCESS(P(n)\log_2 s) \supseteq \\ \bigcup_{s\geq 2} DSPACE(P(n)(S(n) + \log_2 n))ALPH(s) = \\ DSPACE(P(n)(S(n) + \log_2 n)). \end{aligned} \tag{3.5}$$

Keďže funkcie $S(n)$ a $P(n)$ spĺňajú predpoklady, dosadením $d = \log_2 s$ dostávame dokazované tvrdenie. □

Veta 3.4 *Pre ľubovoľné $c, k \in \mathbb{N}^+$, konštruovateľnú funkciu $S(n) \geq 1$ a funkciu $P(n) : \mathbb{N} \to \mathbb{N}$, ktorá spĺňa $P(n) \in O(n^{k-1}c^{S(n)})$, $P(n) \geq 1$ platí:*

$$\bigcup_{d>0} WPM(k,1)SPACE(S(n))PROCESS(dP(n)) = DSPACE(P(n)(S(n) + \log n)).$$

Dôkaz. Tvrdenie priamo vyplýva z lem 3.3 a 3.1 a z platnosti

$$DSPACE(P(n)(S(n) + \log n)) = DSPACE(P(n)(S(n) + k \log n)).$$

□

Predchádzajúca veta charakterizuje silu priestorovo ohraničených WPTM v prípade, že môžu použiť najviac $O(n^{k-1}c^{S(n)})$ procesov. Pre nízky počet procesov môžeme využívať jednu hlavu na pamätanie si úseku pásky. V ďalších vetách charakterizujeme silu priestorovo ohraničených WPTM pre väčší počet procesov. Pre tento prípad dokázané ohraničenia sily WPTM nebudú tesné.

Veta 3.5 *Pre ľubovoľné $c, k \in \mathbb{N}^+$, konštruovateľnú funkciu $S(n) \geq 1$ a funkciu $P(n) : \mathbb{N} \to \mathbb{N}$, kde $P(n) \in \omega(n^{k-1}c^{S(n)})$ a zároveň $P(n) \in O(n^k c^{S(n)})$ platí:*

$$WPM(k,1)SPACE(S(n))PROCESS(P(n)) \supseteq DSPACE(P(n)S(n)).$$

Dôkaz. Štruktúra dôkazu bude rovnaká ako leme 3.2. Budeme simulovať výpočet Turingovho stroja M s priestorovou zložitosťou $P(n)S(n)$ sieťou WPTM W s celkovou priestorovou zložitostou $S(n)$, $P(n)$ procesmi a s k hlavami. Obsah pásky stroja M si bude pamätať $P(n) = O(n^k c^{S(n)})$ procesov, pričom každý z nich si bude pamätať úsek dĺžky $S(n)$.

Na rozdiel od dôkazu lemy 3.2 musíme na identifikovanie procesov využiť všetkých k hláv. Každý proces A_i je unikátny obsahom kanálovej pásky a pozíciami všetkých svojich vstupných hláv. Takto vieme vyrobiť $n^k c^{S(n)}$ unikátnych procesov. Keďže sme použili všetky vstupné hlavy, tak na pamätanie si úseku pásky stoja M procesom A_i môžeme použiť iba pracovnú pásku veľkosti $S(n)$. Simulácia jedného kroku stroja M a akceptácia je rovnaká ako v dôkaze lemy 3.2. □

Lemma 3.6 *Pre ľubovoľné $d \in R^+$, $k \in \mathbb{N}^+$, funkciu $S(n) : \mathbb{N} \to \mathbb{N}$, kde $S(n) \geq 1$ platí:*

$$WPM(k,1)SPACE(S(n)) \subseteq \bigcup_{c>0} DSPACE(n^k c^{S(n)}) \quad (1)$$

$$\bigcup_{c>0} WPM(k,1)SPACE(S(n))PROCESS(dn^k c^{S(n)}) \supseteq \bigcup_{c>0} DSPACE(n^k c^{S(n)}) \quad (2)$$

Dôkaz. Na dokázanie tvrdení upravíme dôkaz tvrdenia

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

v práci Mravca [2]. Stručne zhrnieme použitý dôkaz, na ktorý sa v práci niekoľkokrát odvolávame. Pre dokázanie inklúzie „$\supseteq$“ sa simuluje TS M s priestorovou zložitosťou $n^k c^{S(n)}$ sieťou W s $n^k c^{S(n)}$ procesmi. Každý proces si pamätá jedno políčko pásky stroja M. Dorozumievanie medzi procesmi sme už popísali pri uvedení techniky nájdenia susedov v dôkaze lemy 3.2. Pri dôkaze opačnej inklúzie sa simuluje k-hlavová sieť W s priestorovou zložitosťou $S(n)$ pomocou TS M s priestorovou zložitosťou $n^k d^{S(n)}$, kde d je veľkosť pracovnej abecedy siete W. Sieť W má iba $O(n^k d^{S(n)})$ rôznych procesov. TS M si na každom svojom políčku pamätá, či je daný proces v sieti prítomný. Simulácia jedného kroku siete W je priamočiara.

Popísaný dôkaz Mravca môže byť triviálne rozšírený na dokázanie tvrdenia

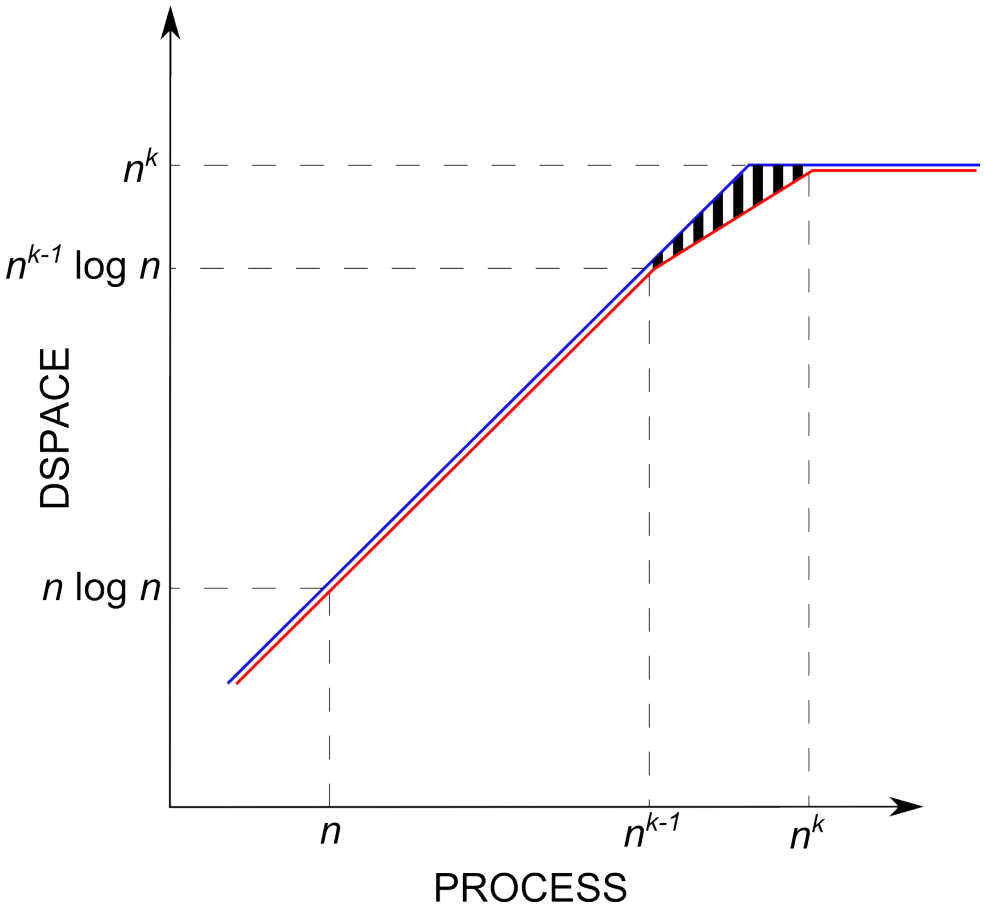
$$WPM(k,1)SPACE(S(n)) = \bigcup_{c>0} DSPACE(n^k c^{S(n)}).$$

Pri dôkaze inklúzie „$\subseteq$“ použijeme $O(n^k (d^2)^{S(n)})$ políčok stroja M na zapamätanie všetkých $O(n^k (d^2)^{S(n)})$ možných procesov siete W, s dvomi páskami veľkosti $S(n)$. Pri dôkaze inklúzie „$\supseteq$“ sa použilo $n^k c^{S(n)}$ procesov, pričom každý proces si pamätá jedno políčko simulovaného Turingovho stroja. Ak potrebujeme použiť iba $dn^k c^{S(n)}$ procesov, kde $d < 1$, potom simuláciu upravíme tak, aby si jeden proces pamätal $\lceil 1/d \rceil$ políčok. □

Nasledujúca veta sumarizuje horné hranice sily WPTM s daným počtom procesov.

Veta 3.7 *Pre ľubovoľné $c, k \in \mathbb{N}^+$, funkcie $S(n) : \mathbb{N} \to \mathbb{N}$ a $P(n) : \mathbb{N} \to \mathbb{N}$, kde $S(n) \geq 1$ a $P(n) \geq 1$ platí:*

$$WPM(k,1)SPACE(S(n))PROCESS(P(n)) \subseteq DSPACE(P(n)(S(n) + \log n)) \quad (1)$$



Obr. 3.1: k-hlavový WPTM pri ohraničení počtu procesov a $S(n) = 1$. Na osi x sa nachádza rádový počet procesov, na osi y sila modelu. Červená krivka vyjadruje dolnú hranicu sily, teda koľko procesov treba na simuláciu príslušného TS. Naopak, modrá krivka vyjadruje hornú hranicu sily WPTM, teda aký TS sa dá simulovať s využitím daného počtu procesov. Vyšrafovaná oblasť zodpovedá nie tesnému určeniu sily WPTM vo vetách 3.5 a 3.7

$$WPM(k,1)SPACE(S(n))PROCESS(P(n)) \subseteq \bigcup_{d>0} DSPACE(n^k d^{S(n)}) \qquad (2)$$

Dôkaz. Platnosť prvého tvrdenia vyplýva z lemy 3.1. Platnosť druhého tvrdenia z lemy 3.6. □

Vetami 3.4, 3.5 a 3.7 sme charakterizovali silu priestorovo ohraničených WPTM pri obmedzení počtu procesov. Výsledky pre jednoduchý prípad $S(n) = 1$ prehľadne znázorňuje obrázok 3.1. Pre väčšie $S(n)$ je situácia komplikovanejšia.

Z predchádzajúcich tvrdení a z vety o pamäťovej hierarchii Turingových stojov vieme bližšie popísať hierarchiu priestorovo ohraničených WPFA vzhľadom na procesorovú zložitosť, pre niektoré konkrétne počty procesov.

Dôsledok 3.8 *Pre ľubovoľné* $c, k \in \mathbb{N}^+$*, konštruovateľné funkcie* $S(n) \geq 1$*,* $P_1(n) \geq 1$ *a* $P_1(n) \geq 1$*, kde* $P_1(n) \in o(P_2(n))$ *a* $P_1(n) \in o(n^{k-1}c^{S(n)})$*, potom*

$$WPM(k,1)SPACE(S(n))PROCESS(P_1(n)) \subsetneq$$
$$\subsetneq WPM(k,1)SPACE(S(n))PROCESS(P_2(n)).$$

Dôkaz. Pre dôkaz tvrdenia rozoberieme dva prípady. V prvom prípade predpokladajme, že funkcia $P_2(n) \in O(n^{k-1}c^{S(n)})$. Tvrdenie dokážeme sporom. Predpokladajme, že tvrdenie neplatí, a teda platí:

$$WPM(k,1)SPACE(S(n))PROCESS(P_1(n)) =$$
$$= WPM(k,1)SPACE(S(n))PROCESS(P_2(n)).$$

Nech $P_1'(n) = dP_1(n)$ a $P_2'(n) = dP_2(n)$, pre ľubovoľné $d \geq 0$. Potom,

$$WPM(k,1)SPACE(S(n))PROCESS(dP_1'(n)) =$$
$$= WPM(k,1)SPACE(S(n))PROCESS(dP_2'(n)).$$

Potom platí:

$$\bigcup_{d>0} WPM(k,1)SPACE(S(n))PROCESS(dP_1'(n)) =$$
$$= \bigcup_{d>0} WPM(k,1)SPACE(S(n))PROCESS(dP_2'(n)).$$

Z vety 3.4, vyplýva:

$$DSPACE(P_1'(n)(S(n) + \log n)) = DSPACE(P_2'(n)(S(n) + \log n)).$$

To je spor s vetou o pamäťovej hierarchii Turingových strojov, keďže

$$P_1'(n)(S(n) + \log n) \in o(P_2'(n)(S(n) + \log n)).$$

Platnosť tvrdenia v prípade, ak $P_2(n) \in \omega(n^{k-1}c^{S(n)})$, dokážeme spojením dvoch inklúzii. Z prvého prípadu pre ľubovoľné d platí:

$$WPM(k,1)SPACE(S(n))PROCESS(P_1(n)) \subsetneq$$
$$\subsetneq WPM(k,1)SPACE(S(n))PROCESS(dn^{k-1}c^{S(n)}).$$

A zrejme

$$WPM(k,1)SPACE(S(n))PROCESS(dn^{k-1}c^{S(n)}) \subseteq$$
$$\subseteq WPM(k,1)SPACE(S(n))PROCESS(P_2(n)).$$

□

Poznámka. V hierarchii vzhľadom na počet procesov pre $P_1(n) \in \omega(n^{k-1}c^{S(n)})$ z našich výsledkov nevieme, kedy nastáva rovnosť alebo ostrá nerovnosť príslušných tried. Napriek tomu v niektorých prípadoch vieme, že nastáva nerovnosť. Napríklad nerovnosť nastáva pre $S(n) = 1$, $P_1 = n^{k-1}\log n$ a $P_2 = n^k$.

Nasledujúca lema nám umožní zosilniť všetky predchádzajúce tvrdenia tak, že použitie pracovnej pásky modelu možno nahradiť použitím výlučne jeho kanálovej pásky.

Lemma 3.9 *Pre ľubovoľné $k \in \mathbb{N}^+$, konštruovateľnú funkciu $S(n) \geq 1$ a $P(n) : \mathbb{N} \to \mathbb{N}$, $P(n) \geq 1$ platí:*

$$WPM(k,1)SPACE(S(n))PROCESS(P(n)) =$$
$$WPM(k,0)CHANNEL(S(n))PROCESS(P(n)).$$

Dôkaz. Iklúzia „$\supseteq$“ vyplýva z definície. Na dôkaz obrátenej inklúzie využijeme, že podľa definície je proces naladený na kanál c, ak je obsah kanálovej pásky *naľavo* od kanálovej hlavy práve c. Priestor napravo od hlavy môžeme využiť inak. Sieť W s pracovnou a kanálovou páskou veľkosti $S(n)$ budeme simulovať sieťou W' s kanálovou páskou veľkosti $2S(n)$. Bez ujmy na všeobecnosti predpokladajme, že každý proces siete W komunikuje iba v párnych kolách, v ktorých nemení obsah svojich pások ani pozície hláv. Každý proces siete W bude simulovať jeden proces siete W', pričom kanálovú pásku tohto procesu rozdelíme na dve časti veľkosti $S(n)$ oddelené symbolom ♠. V ľavej časti simulujeme kanálovú pásku a pohyb kanálovej hlavy. V pravej časti si pamätáme obsah pracovnej pásky, pričom pozíciu pracovných hláv si pamätáme tak, že označíme obsah políčka pod pracovnou hlavou. Simulácia jedného kroku procesu A siete W procesom A' siete W' bude prebiehať nasledovne:

a) V prípade, že simulujeme párny, teda komunikačný krok, simulujeme krok procesu A priamo vykonaním δ-funkcie na procese A'.

b) V prípade, že simulujeme nepárny, „pracovný“ krok, potrebujeme zistiť znaky pod všetkými hlavami. Pozíciu kanálovej hlavy si označíme a následne hľadáme všetky označené políčka, obsahujúce písmená pod pracovnými hlavami procesu A. Následne podľa δ-funkcie siete W prepíšeme označené políčka a posunieme značky, čo zodpovedá zmene obsahu pások pod hlavami procesu A a posunutiu hláv. Hlavy na vstupnej páske si proces A' posunie rovnako ako proces A. Na záver nájdeme značku označujúcu pozíciu kanálovej pásky, odstránime ju a na tomto políčku zostaneme. Proces A bude naladený na rovnaký kanál ako A'. V prípade, že podľa δ-funkcie siete W došlo k vetveniu procesov, sa proces A' rovnako rozvetví.

Simulácia jedného kroku jedného procesu trvá maximálne $4S(n)$ krokov. Špeciálny proces R bude riadiť simuláciu všetkých procesov. Procesy naladené na rovnaký kanál budeme simulovať naraz, aby medzi sebou mohli komunikovať. Proces R prechádza postupne všetky kanály c, ktoré sa dajú naladiť na kanálovej páske veľkosti $S(n)$. Pre každý kanál c, dá proces R signál na simulovanie všetkých procesov na tomto kanáli. Proces R čaká $4S(n)$ kôl. Všetky odsimulované procesy sa označia, aby sa v prípade, že sa preladia na iný kanál, nesimulovali znova. Po $4S(n)$ kolách sú všetky procesy naladené na daný kanál odsimulované a proces R dá pokyn na simuláciu na ďalšom kanáli. Po prejdení všetkých možných kanálov sme odsimulovali jeden krok siete W. Z popisu je zrejmé, že siete W a W' sú ekvivalentné.

Sieť W' využíva $P(n)+1$ procesov. Funkciu procesu R môže vykonávať aj prvý simulujúci proces A'_1, ktorý simuluje proces A_1 siete W. Proces A'_1 bude mať kanálovú pásku veľkosti $4S(n)$, pričom obsah pásky procesu R bude naľavo od pôvodného obsahu pásky procesu A'_1. Pozíciu pôvodnej kanálovej pásky si označíme očiarkovaním príslušného písmena. Proces A'_1 si vždy predtým, ako vyšle pokyn na simulovanie procesov na kanáli c zistí, či nejde o kanál, na ktorom sa má aj on simulovať. Ak je obsah časti pásky so zapamätaným kanálom procesu R rovnaký ako zapamätaný kanál pôvodného procesu A'_1, potom si proces A'_1 najskôr pozrie obsah políčok pod simulovanými hlavami. Proces A'_1 podľa δ-funkcie stroja W vie, ako simulovať proces A_1. Následne sa proces A'_1 opäť naladí

na svoj kanál a dá pokyn na simuláciu, pričom aj on simuluje svoj proces A_1. Po dokončení simulácie procesu A_1, proces A'_1 čaká $4S(n)$ kôl, kým skončí simulácia všetkých procesov na tomto kanáli.

Po tejto úprave sieť W' využíva $P(n)$ procesov a $4S(n)$ políčok kanálovej pásky. Sieť W' budeme simulovať sieťou W'', ktorá bude používať rovnaký počet procesov a hláv, ale bude mať kanálovú zložitosť iba $S(n)$. Sieť W'' použije pracovnú abecedu veľkosti $4|\Sigma_{W'}|^4$, kde $\Sigma_{W'}$ je pracovná abeceda siete W'. Úsek štyroch políčok na kanálovej páske si pamätáme jedným písmenom. Obsah kanálovej pásky ľubovoľného procesu siete W vieme zapísať na kanálovú pásku veľkosti $S(n)$. Pozíciu kanálovej pásky v rámci štvorpolíčkového úseku si nepamätáme stavom, ale vhodným očiarkovaním písmena. Týmto sme zabezpečili, že v prípade simulácie dvoch procesov naladených na kanáloch líšiacich sa iba pozíciou kanálovej hlavy v rámci štvorpolíčkového úseku, potom procesy, ktoré ich simulujú, budú naozaj naladené na rôznych kanáloch.

Sieť W'' je ekvivalentná sieti W' a W, pričom používa k vstupných hláv, $P(n)$ procesov a má kanálovú zložitosť $S(n)$. □

Dôsledok 3.10 *Vo vetách 3.4, 3.5, 3.7, v dôsledku 3.8 a ostatných lemách môžeme, pre všetky konštruovateľné funkcie $S(n)$, nahradiť obmedzenie celkovej priestorovej zložitosti $S(n)$ siete WPTM obmedzením na kanálovú zložitosť $S(n)$.*

Záver

V tejto práci sme sa pokúsili zistiť niektoré vlastnosti modelu WPTM. Skúmali sme, ako sa zmení sila modelu pri pridaní rôznych obmedzení.

V prvej kapitole sme ukázali, že použitie jednosmerných vstupných hláv nezmenší silu bezdrôtových paralelných konečných automatov. Tento výsledok považujeme za najzaujímavejší, a to prevažne použitou originálnou technikou dôkazu. Výsledok sme potom zovšeobecnili na plný model s pracovnou aj kanálovou páskou.

V druhej kapitole sme skúmali vplyv počtu procesov na silu priestorovo ohraničených WPTM. Podarilo sa nám ukázať niekoľko tvrdení, ktoré nám dovolia za určitých podmienok rozhodnúť, či použitie rádovo väčšieho počtu procesov dovolí priestorovo ohraničeným WPTM rozhodovať ťažšie problémy.

Ostáva ešte niekoľko otvorených problémov. V prvej kapitole sme neskúmali, nakoľko sa spomalí sieť WPTM, ak použijeme jednosmerné vstupné hlavy. Predpokladáme, že spomalenie siete pre WPTM s priestorovou zložitosťou $\Omega(\log n)$ nebude výrazné. V druhej kapitole sa nám nepodarilo presne určiť silu priestorovo ohraničených WPTM pre všetky možné ohraničenia počtu procesov. Zaujímave je, že aj v najjednoduchšom prípade, keď skúmame ohraničenia počtu procesov na k-hlavých konečných automatoch, nevieme presne určiť silu modelu s počtom procesov $P(n) \in \omega(n^{k-1})$ a $P(n) \in o(n^k)$. Domnievame sa, že výsledky sa môžu dať zosilniť efektívnejším využívaním vstupných hláv. Zaujímavé by bolo tiež preskúmanie vplyvu počtu procesov na zrýchlenie časovo ohraničených WPTM. Nepreskúmaný zostáva vplyv ďalších zložitostných mier na silu WPTM, napríklad vplyv počtu komunikačných krokov.

Literatúra

[1] A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[2] P. Mravec. Formálne výpočtové modely s bezdrôtovou komunikáciou. Master's thesis, Univerzita Komenského, Bratislava, 2007.

[3] A. Slobodová. Some properties of space-bounded synchronized alternating turing machines with only universal states. In *IMYCS*, pages 102–113, 1988.

[4] A. Slobodová. Communication for alternating machines. *Acta Inf.*, 29(5):425–441, 1992.

[5] A. Slobodová. Alternujúce výpočtové modely s komunikáciou. Master's thesis, Univerzita Komenského, Bratislava, 1998.

[6] H. Vollmer. *Introduction to circuit complexity - a uniform approach.* Texts in theoretical computer science. Springer, 1999.

[7] J. Wiedermann and D. Pardubská. On the power of broadcasting in mobile computing. In S. Cooper, B. Löwe, and A. Sorbi, editors, *New Computational Paradigms*, pages 195–209. Springer New York, 2008.