Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Výpočtové modely s komunikáciou

Bakalárska práca

2024
Lukáš Hudcovský

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Výpočtové modely s komunikáciou

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	doc. RNDr. Dana Pardubská, CSc.

Bratislava, 2024
Lukáš Hudcovský

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Lukáš Hudcovský
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Výpočtové modely s komunikáciou
Computational models with communication

Anotácia: Skúmanie vlastností abstraktných výpočtových modelov s komunikáciou vzhľadom k rôznym modifikáciám a/alebo ohraničeniam.

Vedúci: doc. RNDr. Dana Pardubská, CSc.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 12.10.2023

Dátum schválenia: 13.10.2023 doc. RNDr. Dana Pardubská, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Na tomto mieste by som sa rád poďakoval mojej vedúcej, doc. RNDr. Dane Pardubskej, CSc., za ponúknutie zaujímavej témy, trpezlivosť, pomoc a smerovanie, kedykoľvek som si nevedel rady

Abstrakt

V práci skúmame abstraktný výpočtový model, bezdrôtový paralelný Turingov stroj, ktorý je inšpirovaný bezdrôtovou komunikáciou v mobilných sieťach. Ten umožňuje procesom počas výpočtu vetviť sa na nové procesy, ktoré môžu následne paralelne pracovať. Každý proces má aj špeciálnu kanálovú pásku, ktorej obsah určuje číslo kanála, na ktorý je proces naladený. Procesy naladené na rovnaký kanál spolu vedia komunikovať.

Zameriame sa na variant modelu používajúci na komunikáciu zásobník namiesto kanálovej pásky. Ukážeme simuláciu iného paralelného výpočtového modelu, ktorým je alternujúci Turingov stroj, pomocou nášho modelu. Neskôr porovnáme komunikáciu pomocou zásobníka s komunikáciou pomocou počítadla.

Kľúčové slová: Turingov stroj, bezdrôtová komunikácia, paralelizmus, komunikačný kanál

Abstract

In this thesis we study an abstract computational model, wireless parallel Turing machine, which is inspired by wireless communication in mobile networks. This model allows its processes during computation to branch and create new processes, which can then work in parallel. Every process has also a special channel tape, the content of which determines a channel, to which the process is tuned. Processes tuned to the same channel can communicate with each other.

We will focus on variant of the model, which uses a push-down stack for communication instead of channel tape. We will show a simulation of another parallel model, an alternating Turing machine, using this model. Later we will compare communication using push-down stack with communication using counter.

Keywords: Turing machine, wireless communication, parallelism, communication channel

Obsah

Úvod

Teória zložitosti sa zaoberá výpočtovými problémami z hľadiska potreby prostriedkov (hlavne času a priestoru) na jeho vyriešenie. Cez Turingov stroj, ktorý sa používa ako štandardný model algoritmickej vypočítateľnosti, boli v minulosti definované zložitostné triedy problémov a usporiadané do hierarchií podľa množstva potrebného času a priestoru na ich vyriešenie. Vrámci tejto hierarchie sú ale aj triedy problémov, ktoré sa zdajú byť pre klasické sekvenčné modely, ako je aj Turingov stroj, prakticky neriešiteľné v rozumnom čase, resp. priestore. S cieľom vedieť tieto problémy efektívnejšie riešiť sa začali skúmať rôzne paralelné výpočtové modely, ktoré by vedeli znížiť časové a priestorové nároky na vyriešenie daného problému.

Jedným veľmi známym a pomerne slušne preskúmaným modelom v teórií paralelných výpočtov je alternujúci Turingov stroj (ATM), ktorý bol predstavený v [1] trojicou Chandra, Kozen a Stockmeyer. Tento oproti klasickému nedeterministickému Turingovmu stroju disponuje dvomi typmi stavov - existenčnými a univerzálnymi.

Existenčné stavy zodpovedajú bežným stavom v nedeterministickom Turingovom stroji, a teda keď z nich má stroj možnosť dostať sa do viac konfigurácií, tak si nedeterministicky vyberie jednu a stačí, aby z takého stavu existoval jeden akceptačný výpočet. Naopak, ak je možnosť viacerých rôznych krokov z univerzálneho stavu, tak sa vytvorí viacero kópií stroja a všetky budú súčasne pokračovať, každá v inej konfigurácií. Tieto kópie sa nazývajú aj procesy a na to aby stroj akceptoval, musia všetky tieto procesy akceptovať.

Ukázalo sa, že ATM vedia akceptovať práve triedu rekurzívne vyčísliteľných jazykov, ale s menšími nárokmi na čas a pamäť ako obyčajné Turingove stroje. Procesy vzniknuté pri univerzálnom vetvení ATM sú ale v pôvodnom modeli izolované a nevedia spolu nijako interagovať. Preto bola neskôr snaha vymyslieť spôsoby, akými by mohli tieto procesy spolu komunikovať.

Jedným z modelov, ktorý zabezpečuje istý druh komunikácie, respektíve synchronizácie, je synchronizovaný alternujúci Turingov stroj ($SATM$), opísaný Slobodovou v prácach [4] a [5]. Dá sa chápať ako rozšírenie ATM, kde majú niektoré stavy pridelený navyše synchronizačný prvok a nazývajú sa potom synchronizačné stavy. Keď proces

vojde do takého stavu, tak sa zasekne a čaká kým všetky procesy vojdú do stavu s rovnakým synchronizačným prvkom alebo do finálneho stavu. Rovnako ako pôvodný ATM, je aj tento model nedeterministický.

Nový spôsob komunikácie medzi procesmi predstavili v roku 2005 Pardubská a Wiedermann v [6], kde navrhli model bezdrôtového paralelného Turingovho stroja ($WPTM$), ktorý bol inšpirovaný komunikáciou v bezdrôtových mobilných sieťach. Procesy sa v ňom dokážu vetviť rovnako ako pri univerzálnych stavoch v ATM, ale narozdiel od predošlého je tento model deterministický a neobsahuje teda žiadne existenčné stavy. Okrem toho má stroj pridanú ešte jednu špeciálnu, takzvanú kanálovú pásku, ktorú používa na komunikáciu medzi procesmi a pracuje s ňou rovnako ako s bežnými páskami. Reťazec na páske určuje číslo kanála, na ktorý je daný proces naladený. Procesy s rovnakým obsahom tejto pásky sa potom chápu tak, že sú naladené na rovnaký kanál a vedia si medzi sebou posielať a prijímať správy prostredníctvom špeciálnych komunikačných stavov.

$WPTM$ bol skúmaný ďalej a v [7] bol ukázaný úzky súvis medzi $WPTM$ a ATM. Predovšetkým bolo dokázané, že súčasne časovo a priestorovo ohraničené $WPTM$ presne zodpovedajú podobne časovo a priestorovo ohraničeným ATM.

Ďalej boli varianty tohto modelu skúmané aj v diplomovej práci [2]. Tam boli okrem iného charakterizované priestorovo ohraničené $WPTM$ a ukázané, že stačí konečný automat so slepým počítadlom na komunikáciu, na rozpoznanie ľubovoľného rekurzívne vyčísliteľného jazyka. Takisto bol preskúmaný nedeterministický variant toho modelu a bolo zistené, že má mnohé vlastnosti zhodné s deterministickým variantom.

Neskôr bolo ešte v diplomovej práci [3] ukázané, že jednosmernosť vstupných hláv nezmenší silu priestorovo ohraničených $WPTM$. Taktiež bolo zistené, za akých podmienok vyšší počet procesov zväčší silu priestorovo ohraničených $WPTM$.

V tejto práci budeme skúmať, ako zmení vlastnosti $WPTM$ zmena štruktúry na komunikáciu oproti štandardnej kanálovej páske.

V prvej kapitole zadefinujeme model, ktorý bude namiesto kanálovej pásky používať na komunikáciu zásobník. V druhej kapitole ukážeme simuláciu viachlavého alternujúceho automatu pomocou zadefinovaného $WPTM$ bez pracovných pások a tým aj fakt, že viac vstupných hláv je možné nahradiť viacerými procesmi s komunikáciu. Nakoniec preskúmame, ako ovplyvní vlastnosti modelu veľkosť zásobníkovej abecedy. Konkrétne ukážeme, pri akej kanálovej a priestorovej zložitosti vieme simulovať komunikáciu s počítadlom pomocou komunikácie so zásobníkom.

Kapitola 1

Bezdrôtový paralelný Turingov stroj so zásobníkom

V tejto kapitole definujeme model bezdrôtového paralelného Turingovho stroja. Definícia je prebratá z prác [6] a [7] s jedným hlavným rozdielom. Oproti pôvodnej definícii, kde sa na komunikáciu používala špeciálna kanálová páska, budeme v našej práci za základný model považovať stroj, ktorý má na komunikáciu zásobník, lebo v celej práci budeme pracovať iba s týmto variantom modelu. Takisto definujeme niektoré zložitostné miery a ďalšie súvisiace pojmy, ktoré budeme v práci používať.

1.1 Definícia modelu

Definícia 1.1 *k-páskový bezdrôtový paralelný Turingov stroj so zásobníkom* ($WPTM$) M s oddelenou vstupnou páskou je 12-tica

$$M = (k,\, Q,\, R,\, \Sigma,\, \Gamma,\, \Delta,\, q_0,\, r_0,\, \varepsilon,\, q_{accept},\, q_{reject},\, Z_0)$$

kde

- k je počet pracovných pások stroja
- Q je konečná množina pracovných stavov s počiatočným stavom q_0
- R je konečná množina komunikačných stavov so štyrmi špeciálnymi stavmi: počiatočným stavom r_0, akceptačným stavom q_{accept}, zamietacím stavom q_{reject} a prázdnym stavom ε
- Σ je vstupná abeceda, pričom ¢,\$$\notin \Sigma$ sú ľavá a pravá zarážka v poradí
- Γ je pracovná abeceda, pričom $\sharp \notin \Gamma$ je prázdny symbol, Z_0 je počiatočný zásobníkový symbol a ¢,\$$\notin \Gamma$

- $\Delta : Q \times R \times (\Sigma \cup \{¢,\$\}) \times (\Gamma \cup \{\sharp,¢\})^k \times \Gamma \to Q \times R \times (\Gamma \cup \{¢\})^k \times \Gamma^* \times \{-1,0,1\}^{k+1}$ je prechodová relácia a jej prvky budeme nazývať prechody

Podľa tejto definície má stroj jednu vstupnú pásku s jednou čítacou hlavou, pričom páska obsahuje ľavú zarážku ¢, pravú zarážku \$ a medzi nimi vstupné slovo. Okrem toho má stroj k pracovných pások, ktoré sú jednosmerne nekonečné smerom doprava s tým, že na najľavejšej pozícií sa nachádza ľavá zarážka ¢. Na každej tejto páske má stroj hlavu, ktorá vie aj zapisovať a pozície symbolov na páskach sa číslujú od nuly. Nakoniec má stroj ešte zásobník s počiatočným symbolom Z_0, ktorý používa na komunikáciu.

Vezmime $\delta = (q, r, x, a_1, ..., a_k, \gamma, q', r', b_1, ..., b_k, v, d_1, ..., d_{k+1}) \in \Delta$. Tento prechod hovorí, že stroj je v pracovnom stave q, komunikačnom stave r, na vstupnej páske číta symbol x, na i-tej pracovnej páske číta symbol a_i pre všetky $i \in \{1, ..., k\}$ a na zásobníku symbol γ. Následne v jednom kroku prejde do nového pracovného stavu q', komunikačného stavu r', na i-tu pracovnú pásku zapíše b_i pre $i \in \{1, ..., k\}$, zo zásobníka vyberie symbol γ a vloží tam reťazec v, a posunie každú z $k+1$ hláv na páskach o $d_j \in \{-1, 1, 0\}$ pre $j \in \{1, ..., k+1\}$ (hlava sa buď posunie o 1 políčko doľava, o 1 doprava alebo ostane stáť na mieste). Dohodou bude zakázané prepisovanie aj prekračovanie zarážok ¢,\$.

Postupne sa teraz dostaneme k definícií výpočtu a akceptácie stroja, ale najprv si budeme musieť definovať niekoľko pomocných pojmov, ktoré nám to umožnia.

Definícia 1.2 *Konfigurácia $WPTM$ M* je prvok z $Q \times R \times \Sigma^* \times (\Gamma^*)^{k+1} \times \mathbb{N}^{k+1}$ a reprezentuje aktuálny pracovný a komunikačný stav stroja, obsah vstupnej a pracovných pások bez prázdnych symbolov a zarážok, obsah zásobníka a pozíciu hlavy na každej páske.

Definícia 1.3 *Hlavová konfigurácia $WPTM$ M* je prvok z $Q \times R \times (\Sigma \cup \{¢,\$\}) \times (\Gamma \cup \{\sharp,¢\})^k \times \Gamma$ a reprezentuje aktuálny pracovný a komunikačný stav stroja, symboly čítané jednotlivými hlavami a symbol na vrchu zásobníka.

Ak má prechod nový komunikačný stav r', tak hovoríme, že tento prechod *vysiela* stav r'. Pri tomto platí jedno obmedzenie, takzvané *pravidlo súhlasného vysielania*, ktoré hovorí, že prechody s rovnakou hlavovou konfiguráciou musia vysielať ten istý komunikačný stav. Toto obmedzenie je iba syntaktické a prechody sa môžu líšiť vo všetkých ostatných zložkách.

Hovoríme ďalej, že konfigurácia je *naladená na kanál c*, ak je obsah jej zásobníka rovný reťazcu c. Pre $c \neq \varepsilon$ hovoríme reťazcu c aj *číslo kanála*. Ak sa z konfigurácie naladenej na kanál $c \neq \varepsilon$ vykonáva prechod s novým komunikačným stavom $r' \neq \varepsilon$, tak hovoríme, že konfigurácia *vysiela r' na kanáli c*.

Ak má konfigurácia nový komunikačný stav ε, nazýva sa aj nevysielajúca/tichá konfigurácia. Tieto konfigurácie sú užitočné napríklad pri prelaďovaní kanálu alebo keď jednoducho nechceme, aby konfigurácie vysielali nejakú informáciu.

Definícia 1.4 Konfigurácia β sa nazýva δ-nasledovník konfigurácie α vzhľadom na $\delta \in \Delta$ (zapisujeme $\alpha \vdash^{\delta} \beta$), ak vznikla aplikovaním prechodu δ na konfiguráciu α. Prechod $\alpha \vdash^{\delta} \beta$ budeme nazývať *jednoduchý krok* stroja M.

Konfigurácie môžu mať viac než jedného δ-nasledovníka.

Definícia 1.5 *Terminálna* konfigurácia je taká konfigurácia, ktorá nemá δ-nasledovníka pre žiadne $\delta \in \Delta$.

Definujeme si teraz niekoľko pomocných funkcií, aby sme vedeli poriadne definovať výpočet stroja M. Nedefinovanú hodnotu budeme označovať symbolom $\perp$.

Definícia 1.6 Funkcia $Comm\colon Q \times R \times \Sigma^* \times (\Gamma^*)^{k+1} \times \mathbb{N}^{k+1} \to R$ priraďuje konfigurácií jej súčasný komunikačný stav.

Definícia 1.7 Funkcia $Tuned\colon Q \times R \times \Sigma^* \times (\Gamma^*)^{k+1} \times \mathbb{N}^{k+1} \to \Gamma^*$ priraďuje konfigurácií jej aktuálne číslo kanála.

Definícia 1.8 Funkcia $Broadcast\colon Q \times R \times \Sigma^* \times (\Gamma^*)^{k+1} \times \mathbb{N}^{k+1} \to R$ priraďuje konfigurácií komunikačný stav, ktorý je ňou vysielaný po aplikovaní niektorého aplikovateľného prechodu. Vďaka pravidlu súhlasného vysielania je táto funkcia daná jednoznačne. Funkciu vieme nasledovným spôsobom rozšíriť aj na množinu konfigurácií $D \neq \emptyset$:

$$Broadcast(D) = \begin{cases} b & \text{ak } \forall \alpha, \beta \in D : Tuned(\alpha) = Tuned(\beta) \wedge Broadcast(\alpha) = b \\ \perp & inak \end{cases}$$

Nech α je konfigurácia a $r \in R$ je jej komunikačný stav. Potom pre ľubovoľný stav $s \in R$ budeme označovať $\alpha|_{r:=s}$ takú konfiguráciu, ktorá vznikne z konfigurácie α nahradením komunikačného stavu r za stav s.

Definícia 1.9 Nech D je množina konfigurácií a $D_c \subseteq D$ je podmnožina jej konfigurácií naladených na kanál c. Nech ďalej $\alpha, \beta \in D$ sú konfigurácie, kde špeciálne $\beta \in D_c$ a $\alpha \vdash^{\delta} \beta$ je jednoduchý krok. Potom konfiguráciu γ, ktorú budeme nazývať δ_D-nasledovník konfigurácie α vzhľadom na $\delta \in \Delta$, modifikovaný vysielaním D (píšeme

$\alpha \vdash^{\delta_D} \gamma$), definujeme nasledovne:

$$\gamma := \begin{cases} \beta & \text{ak } Broadcast(D_c) = \varepsilon \\ \beta|_{Comm(\beta):=b} & \text{ak } \forall \varphi \in D_c : Broadcast(\varphi) \in \{b, \varepsilon\} \text{ a zároveň} \\ & \exists \varphi \in D_c : Broadcast(\varphi) = b \\ \perp & inak \end{cases}$$

Konfiguráciu, pre ktorú $\exists \delta \in \Delta : \alpha \vdash^{\delta_D} \gamma$, budeme nazývať Δ_D-nasledovník konfigurácie α.

Aj tu si môžeme všimnúť, že konfigurácia môže mať viac nasledovníkov. V tom prípade hovoríme, že konfigurácia α *vytvorí* všetky konfigurácie γ, pre ktoré existuje prechod $\delta \in \Delta$, taký že platí $\alpha \vdash^{\delta_D} \gamma$.

Definícia 1.10 Pre každé vstupné slovo $w \in \Sigma$ stroja M je výpočtový graf $G(w)$ stroja M orientovaný, zakorenený, potenciálne nekonečný acyklický multigraf, ktorého vrcholy sú konfigurácie stroja a hrany zodpovedajú prechodovým a komunikačným hranám. HĺbkA vrchola v grafe je jeho vzdialenosť od koreňa, pričom graf má nasledovnú rekurzívnu definíciu:

(*i*) Počiatočná konfigurácia $c_{init} = (q_0, r_0, w, \underbrace{\lambda, ..., \lambda}_{k}, Z_0, \underbrace{0, ..., 0}_{k+1})$, kde λ je prázdny reťazec, je koreňom grafu $G(w)$ (vrchol v hĺbke $d = 0$).

(*ii*) Nech je C_d množina všetkých vrcholov (konfigurácií) v hĺbke $d \geq 0$. Potom pre všetky neterminálne konfigurácie $\alpha \in C_d$ obsahuje množina C_{d+1} všetkých Δ_{C_d}-nasledovníkov konfigurácie α (teda všetky konfigurácie ktoré α vytvorí). Ak je niektorý z Δ_{C_d}-nasledovníkov nedefinovaný, tak celý graf bude nedefinovaný.

(*iii*) Graf má dva typy hrán:

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

Vrcholy grafu $G(w)$, ktoré nemajú nasledovníkov, budeme nazývať *listy*.

Výpočtový krok má teda dve fázy. V prvej fáze sa na konfiguráciu α aplikujú všetky možné aplikovateľné prechody, čím sa vytvorí niekoľko nových konfigurácií. V druhej fáze sa uskutočňuje komunikácia. Konfigurácia α po aplikovaní nejakého aplikovateľného prechodu vysiela svoj nový komunikačný stav na kanáli, na ktorý je naladená.

Nové vytvorené konfigurácie vedia tento vysielaný stav prijať, ak sú naladené na rovnaký kanál, ako bola konfigurácia α pred aplikovaním prechodu. Vďaka pravidlu súhlasného vysielania konfigurácia vždy vysiela iba jeden stav alebo ε (teda nevysiela).

Nech je teraz D_c podmnožina novo vytvorených konfigurácií, naladených na kanál c. V tejto fáze môžu podľa definície 1.9 nastať tri situácie.

- Prvá je, že na kanáli c žiadna konfigurácia nevysiela a vtedy sa komunikačný stav konfigurácií z D_c nezmení.

- V druhej situácii vysiela nejaká konfigurácia α na kanáli c nejaký komunikačný stav a v tom prípade sa komunikačné stavy konfigurácií z D_c prepíšu na tento stav.

- Posledná možná situácia je, že na c vysielajú dve konfigurácie dva rôzne stavy. Vtedy hovoríme, že na kanáli nastal konflikt a výsledok nedefinujeme.

Môžeme si všimnúť, že v korektnom výpočte môžu na jednom kanáli vysielať aj viaceré konfigurácie naraz, len musia vysielať rovnaký stav.

Môžeme si taktiež všimnúť, že vybudovanie grafu je deterministický proces. Vrcholom výpočtového grafu $G(w)$ stroja M budeme hovoriť aj *procesy*. Pojmy vrcholy, procesy a konfigurácie budeme stotožňovať, ak budeme hovoriť o konkrétnom výpočtovom grafe $G(w)$ stroja M.

Definícia 1.11 Výpočtový graf $G(w)$ stroja M nazveme výpočtový graf *akceptujúci* (*zamietajúci*) vstup w, ak platí:

(*i*) *konečnosť*: $G(w)$ je konečný graf

(*ii*) *akceptačné kritérium*: Všetky listy grafu sú terminálne konfigurácie v rovnakej hĺbke, v komunikačnom stave q_{accept} (q_{reject}) a sú naladené na rovnaký kanál.

Akceptačné kritérium vlastne hovorí, že všetky procesy musia zdieľať informáciu o tom, či ostatné procesy skončili alebo nie.

Definícia 1.12 Hovoríme, že stroj M *akceptuje* vstup w práve vtedy, keď jeho výpočtový graf $G(w)$ ho akceptuje. *Jazyk akceptovaný strojom* M (označujeme $L(M)$) je množina všetkých slov w, ktoré stroj akceptuje.

1.2 Zložitostné miery

Teraz zadefinujeme miery zložitosti pre $WPTM$, ktoré budeme v práci ďalej používať. Definície budú zodpovedať intuitívnym očakávaniam, a teda časová zložitosť výpočtu bude závisieť od hĺbky výpočtového grafu a rovnako intuitívne priestorová zložitosť bude závisieť od maximálneho použitého priestoru na páskach.

Definícia 1.13 *$WPTM$ M pracuje v časovej zložitosti $T(n)$*, ak pre všetky $w \in L(M)$ dĺžky n platí, že hĺbka ľubovoľného vrcholu vo výpočtovom grafe $G(w)$ stroja M akceptujúcom w je najviac $T(n)$.

Definícia 1.14 *$WPTM$ M pracuje v priestorovej zložitosti $S(n)$*, ak pre všetky $w \in L(M)$ dĺžky n platí, že pre všetky konfigurácie vo výpočtovom grafe $G(w)$ stroja M akceptujúcom w platí, že počet použitých políčok na ľubovoľnej z ich pracovných pások je najviac $S(n)$.

Definícia 1.15 *$WPTM$ M pracuje v kanálovej zložitosti $C(n)$*, ak pre všetky $w \in L(M)$ dĺžky n platí, že pre všetky konfigurácie vo výpočtovom grafe $G(w)$ stroja M akceptujúcom w je počet symbolov na ich zásobníku najviac $C(n)$.

1.3 Označenia

Bezdrôtový paralelný Turingov stroj budeme štandardne označovať veľkými písmenami $WPTM$.

Špeciálny prípad $WPTM$ je stroj bez pracovných pások, ktorý budeme nazývať *bezdrôtový paralelný konečný automat* a označovať ho $WPFA$. Pri formálnej definícii potom vynechávame prvú zložku 12-tice, ktorá hovorí o počte pracovných pások.

V práci budeme skúmať aj vplyv veľkosti zásobníkovej abecedy stroja. Budeme teda potrebovať rozlíšiť, či zásobník používa iba jeden symbol (počítadlo) alebo viac.

Stroj, ktorého zásobníková abeceda má iba jeden symbol (okrem počiatočného symbolu) budeme označovať spodným indexom 1, teda napríklad $WPFA_1$. Jeho zásobník budeme prirodzene nazývať počítadlo.

Ak má zásobníková abeceda stroja dva a viac symbolov (okrem počiatočného), budeme ho označovať štandardne alebo na zdôraznenie pridáme index 2, napr. $WPTM_2$.

Kapitola 2

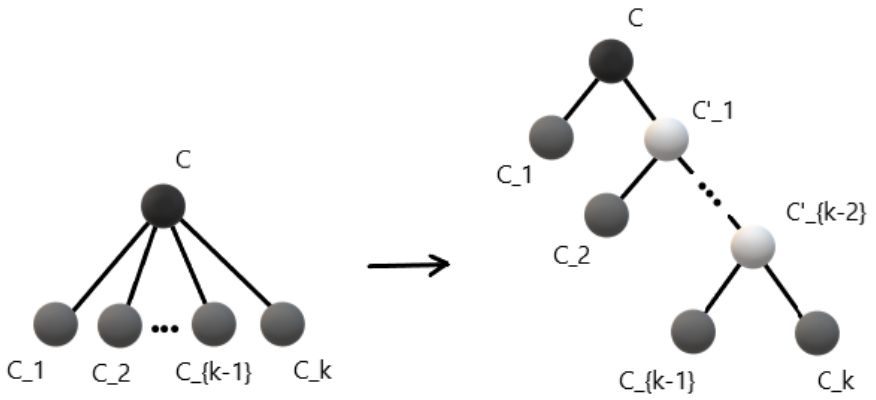
Komunikácia so zásobníkom a alternujúci konečný automat

V prácach, v ktorých bol $WPTM$ v minulosti skúmaný, sa pri niektorých dôkazoch používala kanálová páska stroja, ako keby to bol zásobník. V tejto kapitole ukážeme, že konkrétne simulácia alternujúceho Turingovho stroja pomocou $WPTM$ funguje aj v prípade, že používame model so zásobníkom. Presnejšie, spravíme simuláciu k-hlavého alternujúceho konečného automatu pomocou jednohlavého $WPFA$. Zároveň tým ukážeme, že viac vstupných hláv je možné nahradiť viacerými procesmi s komunikáciou.

Neformálne tu definujeme alternujúci Turingov stroj (ATM), s ktorým budeme v nasledovnom tvrdení pracovať. ATM je Turingov stroj, ktorý má oproti štandardnému modelu dva typy konfigurácií: existenčné a univerzálne, pričom existenčná konfigurácia vedie k akceptácií, ak aspoň jeden z jej nasledovníkov vedie k akceptácií (ako pri NTS) a univerzálna konfigurácia vedie k akceptácií, ak všetci jej nasledovníci vedú k akceptácií (hovoríme o univerzálnom vetvení). Podobne ako pri $WPTM$ aj tu budeme stotožňovať pojmy konfigurácia a proces v jednom výpočte ATM. Dvojsmerným alternujúcim konečným automatom (označ. AFA) budeme rozumieť ATM bez pracovných pások.

V nasledujúcom tvrdení budeme pracovať s ATM v normálnom tvare, pri ktorom sa proces v každom kroku výpočtu rozvetví na najviac dva nové procesy (hovoríme, že každý jeho strom výpočtu má faktor vetvenia najviac dva). To, že je to naozaj normálny tvar, ukážeme formálne iba pre špeciálny prípad AFA, kvôli lepšej prehľadnosti. Pre všeobecný prípad ATM by sa konštrukcia urobila analogicky.

Neformálne, nech C je konfigurácia vo výpočtovom strome AFA A, ktorá má k nasledovníkov $C_1, ..., C_k$ pre $k > 2$. Vytvoríme nové konfigurácie $C'_1, ..., C'_{k-2}$, ktoré budú slúžiť ako prechodné a nový výpočtový strom, ktorý bude vyzerať nasledovne.



Obr. 2.1: Výpočtový strom normálneho tvaru

Konfigurácia C bude mať nasledovníkov C_1 a novú konfiguráciu C'_1. Každá nová konfigurácia C'_x bude mať potom nasledovníkov pôvodnú konfiguráciu C_{x+1} a ďalšiu novú konfiguráciu C'_{x+1}, okrem poslednej konfigurácie C'_{k-2}, ktorej nasledovníci budú posledné dve z pôvodných konfigurácií. Všetkým prechodným konfiguráciám priradíme rovnaký typ (existenčná/univerzálna) ako mala konfigurácia C.

Formálnejšie nech A je $(Q, \Sigma, \delta, q_0, g)$, kde Q je množina stavov, Σ je vstupná abeceda, $\delta : Q \times (\Sigma \cup \{¢,\$\}) \to Q \times \{-1, 0, 1\}$ je prechodová funkcia, q_0 je počiatočný stav a g je funkcia, priraďujúca stavom ich typ, pričom $g : Q \to \{\wedge, \vee, q_{accept}, q_{reject}\}$. Skonštruujeme $A' = (Q', \Sigma, \delta', q_0, g')$, pričom ak nebude povedané inak, tak pokiaľ v nasledujúcej konštrukcii použijeme značenie $\forall q$, budeme tým rozumieť $\forall q \in Q$. Podobne $\forall b$ budeme chápať ako $\forall b \in \Sigma \cup \{¢,\$\}$. Hlavovou konfiguráciou AFA budeme rozumieť jeho aktuálny stav a symbol čítaný jeho vstupnou hlavou.

Zadefinujeme si maximum nasledovníkov z konfigurácie ako

$$m := max\{|\delta(q, b)| \,|\, q \in Q,\, b \in \Sigma \cup \{¢\}\}$$

Množina stavov bude pozostávať z množiny pôvodných stavov q automatu A a nových stavov q'_x, pridaných pre prechodné konfigurácie.

$$Q' = Q \cup \{q'_x |\, q \in Q,\, x \in \{1, ..., m-2\}\}$$

Pôvodné stavy budú mať rovnaký typ (existenčné/univerzálne) ako v stroji A a nové stavy budú mať rovnaký typ ako stavy, z ktorých vznikli.

$$\forall q :\, g'(q) = g(q)$$

$$\forall q \, \forall x \in \{1, ..., m-2\} :\, g'(q'_x) = g(q)$$

Ak mala pôvodná prechodová funkcia dva a menej prechodov z nejakej hlavovej konfigurácie, tak nová funkcia zostane rovnaká.

$$\forall q, b \text{ t.ž. } |\delta(q,b)| \leq 2 : \delta'(q,b) = \delta(q,b)$$

Ak mala pôvodná funkcia z nejakej hlavovej konfigurácie viac ako dva prechody, tak si ich vieme usporiadať a z pôvodnej hlavovej konfigurácie bude v δ' funkcii jeden prechod podľa prvého z nich a druhý prechod do nového stavu q'_1.

$$\forall q, b \text{ t.ž. } \delta(q,b) = \{(p_1,d_1),...,(p_k,d_k) \,|\, k > 2; \forall i \in \{1,...,k\} : p_i \in Q \wedge d_i \in \{-1,0,1\}\} :$$

$$\delta'(q,b) = \{(q'_1, 0), (p_1, d_1)\}$$

Podobne aj z každého nového pridaného stavu q'_x bude jeden krok do pôvodného stavu p_{x+1} a druhý do ďalšieho nového stavu q'_{x+1}.

$$\forall k > 3 \, \forall x \in \{1,...,k-3\} :$$

$$\delta'(q'_x, b) = \{(q'_{x+1}, 0), (p_{x+1}, d_{x+1})\}$$

Okrem posledného nového stavu q'_{k-2}, z ktorého budú oba prechody do posledných dvoch pôvodných stavov.

$$\delta'(q'_{k-2}, b) = \{(p_{k-1}, d_{k-1}), (p_k, d_k)\}$$

Z konštrukcie je zrejmé, že A' akceptuje práve vtedy, keď akceptuje stroj A. Takisto každý výpočtový strom A' bude mať hĺbku najviac $(d-1) \cdot h$, kde h je hĺbka ekvivalentného výpočtového stromu stroja A a d je konštanta, závislá od maximálneho vetvenia vo výpočtovom strome A. Teda ak A pracoval v časovej zložitosti $T(n)$, tak A' bude pracovať v časovej zložitosti $O(T(n))$.

Tvrdenie 2.1: Nech A je k-hlavý AFA s časovou zložitosťou $T(n)$. Potom existuje $WPFA$ M simulujúci A v časovej aj kanálovej zložitosti $O(T(n))$.

Dôkaz: Idea dôkazu je prebratá z dôkazu vety 3.1 o simulácii ATM pomocou $WPTM$ v práci [6], kde sa dokázalo, že ak ATM A' pracuje v časovej zložitosti $T(n)$, pričom $T(n)$ je časovo konštruovateľná, tak $WPTM$ M' ho dokáže simulovať v čase $O(T(n))$. Pôvodný dôkaz pracuje so strojom v normálnom tvare uvedenom vyššie a má dve fázy. V prvej fáze sa "zostupuje dolu" a postupne sa buduje strom výpočtu stroja A' tak, že pre každý proces stroja A' má stroj M' jeden proces, ktorý ho simuluje, pričom na kanálovú pásku si pri každom kroku ukladá informáciu o type jemu prislúchajúcej konfigurácie stroja A' a o tom, či je jeho konfigurácia po poslednom vetvení v ľavej alebo pravej vetve. Týmto spôsobom má proces na páske uloženú informáciu o ceste

z počiatočnej do jemu prislúchajúcej konfigurácie vo výpočtovom strome A'. Vďaka konštruovateľnosti $T(n)$ vie takto M' odsimulovať presne $T(n)$ krokov stroja A', s tým že všetky jeho procesy skončia naraz.

V druhej fáze sa zase "vystupuje naspäť hore" vo výpočtovom strome A' a od koncových až po počiatočnú konfiguráciu sa overuje akceptačné kritérium stroja A'. Robí sa to tak, že každej konfigurácii vo výpočtovom strome A' sa priradí tzv. *kvalita*, čo je hodnota z $\{Y, N, \perp\}$ podľa toho, či je výpočet začínajúci v nej akceptačný, zamietajúci alebo neukončený. Procesy stroja M' začnú tým, že podľa stavu, v ktorom skončila im prislúchajúca konfigurácia, jej priradia kvalitu. Následne si procesy vypočítajú kvalitu konfigurácie C, ktorá je predchodcom im prislúchajúcej konfigurácii nasledovne. Procesy, ktorých konfigurácie sú nasledovníkmi C, majú rovnakú časť cesty vo výpočtovom strome A' až po konfiguráciu C. Teda ak si posunú kanálovú hlavu o dve miesta doľava, tak budú mať rovnaký obsah kanálovej pásky a budú si vedieť postupne poslať informáciu o kvalite ich konfigurácií. Z toho si potom podľa typu konfigurácie C, ktorý majú uložený na páske, vedia vypočítať jej kvalitu. Takto sa pokračuje, až kým všetky procesy nemajú vypočítanú kvalitu počiatočnej konfigurácie, podľa ktorej akceptujú/zamietajú.

Môžeme si všimnúť, že v pôvodnom dôkaze sa v skutočnosti pracuje s kanálovou páskou, ako keby to bol zásobník, teda keď posúvame kanálovú hlavu doľava, tak symboly napravo od nej už v nasledujúcom výpočte nebudeme navštevovať a v podstate ich vymazávame. Základnú myšlienku dôkazu teda vieme aplikovať aj v našom prípade, keď uvažujeme model, ktorý má na komunikáciu zásobník.

Pri aplikovaní dôkazu na náš prípad ale musíme vyriešiť dva problémy:

(1) Simulovaný AFA má k vstupných hláv, zatiaľ čo simulujúci $WPFA$ má pre každý proces iba jednu hlavu.

(2) Nevieme skonštruovať funkciu $T(n)$, keďže $WPFA$ nemá pracovné pásky.

Prvý problém budeme riešiť tak, že M bude mať pre každý proces p automatu A až k procesov, pričom každý bude reprezentovať jednu vstupnú hlavu procesu p. Tieto procesy budú pri simulácií každého kroku automatu A spolu komunikovať a posielať si informácie o tom, aké symboly čítajú im prislúchajúce hlavy.

Na vyriešenie druhého problému musíme prispôsobiť druhú fázu simulácie tomu, že procesy môžu mať rôzne dlhú prvú fázu výpočtu (skončia v rôznej hĺbke výpočtového stromu A). Spravíme to tak, že keď bude skupina procesov, ktorým prislúcha konfigurácia C, chcieť vypočítať kvalitu jej predchodcu C', tak budú striedavo vysielať na kanáli kvalitu C a počúvať, či na kanáli nevysiela informáciu druhá skupina procesov a až keď všetci prijmú informáciu od druhej skupiny, tak vypočítajú kvalitu C' a pokračujú vo výpočte. Prvá a druhá fáza simulácie sa teda môžu v našom prípade prelínať.

Formálnejšie to môžeme zapísať takto. Nech AFA A je 6-tica $(k, Q, \Sigma, \delta, q_0, g)$ kde $g : Q \to \{\wedge, \vee, q_{accept}, q_{reject}\}$. Skonštruujeme ekvivalentný $WPFA$ M.

$$M = (Q',\ R,\ \Sigma,\ \Gamma,\ \Delta,\ q_0',\ r_0,\ \varepsilon,\ q_{accept},\ q_{reject}, Z_0)$$

Zásobníková abeceda bude obsahovať symboly reprezentujúce typ konfigurácie a takisto symboly 0 a 1, kde 0 označuje ľavého syna a 1 označuje pravého syna vrcholu.

$$\Gamma = \{0, 1\} \cup \{\forall, \exists, |\} \cup \{Z_0\}$$

Množina komunikačných stavov bude okrem špeciálnych stavov obsahovať všetky symboly zo $\Sigma \cup \{¢,\$\}$, aby si procesy vedeli posielať informácie o čítaných symboloch a symboly z $\{Y, N\}$ na posielanie informácií o kvalite konfigurácií.

$$R = \Sigma \cup \{¢,\$\} \cup \{Y, N\} \cup \{(Y, *), (N, *)\} \cup \{r_0, \varepsilon, q_{accept}, q_{reject}\}$$

Množina pracovných stavov bude vyzerať nasledovne

$$Q' = \{q_0'\} \cup \{0,1\}^2 \times \{1, ..., k\} \times Q \times (\Sigma \cup \{¢, \$, \#\})^k \times \{0, ..., k+1\} \times \{Y, N, \perp, -\}^2 \times \{0, 1, -\}$$

Význam jednotlivých zložiek vysvetlíme na príklade. Ak sa teda proces p' stroja M nachádza v stave

$$q' = [f, sig, x, q, a_1, ..., a_k, n, qual1, qual2, b]$$

znamená to nasledovné. Proces p' reprezentuje x-tú hlavu nejakého procesu p automatu A, pričom ten je v stave q a svojimi hlavami číta postupne symboly $a_1, ..., a_k$. Ďalej f určuje, či sme v prvej alebo druhej fáze simulácie, sig hovorí, či je aktuálne párny alebo nepárny krok výpočtu a n hovorí, ktorý z k procesov má aktuálne vysielať symbol, ktorý číta jemu prislúchajúca hlava (relevantné iba pre prvú fázu simulácie). Nakoniec $qual1$ je posledná vypočítaná kvalita nejakej konfigurácie, $qual2$ je hodnota kvality poslaná nejakou susednou skupinou procesov a b hovorí, či sa aktuálna konfigurácia nachádza v pravej alebo ľavej vetve ($qual1$, $qual2$ aj b sú relevantné iba pre druhú fázu simulácie).

Keďže stroj nemá pracovné pásky, tak Δ relácia bude tvaru

$$Q' \times R \times (\Sigma \cup \{¢,\$\}) \times \Gamma \ \to\ Q' \times R \times \Gamma^* \times \{-1, 0, 1\}$$

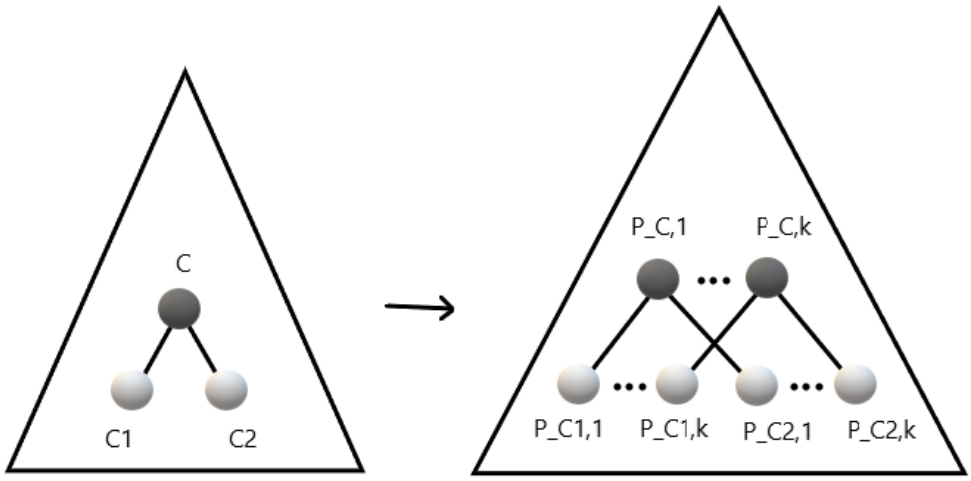
Aby sa nemýlili značenia prázdneho stavu a prázdneho slova, budeme označovať prázdne slovo symbolom λ. Taktiež, keď použijeme v druhej zložke stavu zápis $sig + 1$, myslíme tým sčítanie modulo 2.

B.u.n.v môžeme predpokladať, že AFA A je v normálnom tvare popísanom vyššie, a teda každý jeho strom výpočtu má faktor vetvenia najviac dva. Stroj M začne v počiatočnej konfigurácii, z ktorej sa rozvetví na k procesov, kde každý reprezentuje jednu z k hláv automatu A, pričom v stave si každý pamätá číslo $1, ..., k$, podľa toho ktorej hlave prislúcha. Takisto si procesy budú v stave pamätať počiatočný stav q_0 automatu A a k-ticu $(¢, ..., ¢)$, ktorá reprezentuje symboly čítané hlavami automatu A na začiatku výpočtu, čím bude mať zapamätanú celú jeho počiatočnú hlavovú konfiguráciu.

$$\Delta(q_0', r_0, ¢, Z_0) = \{([0, 0, x, q_0, \underbrace{¢, ..., ¢}_{k}, 0, -, -, -], \varepsilon, Z_0, 0) \mid x \in \{1, ..., k\}\}$$

M odsimuluje jeden krok automatu A v jednom kole, ktoré bude vyzerať nasledovne.

Každý proces M sa podľa δ-funkcie A rozvetví na ďalšie procesy, s tým že v každej vetve realizuje konkrétny prechod podľa konfigurácie A, zapamätanej v stave, nasledovným spôsobom.



Obr. 2.2: Simulácia jedného kroku AFA

Proces zmení zapamätaný stav A na nový podľa použitého prechodu. Ak má v prvej zložke stavu zapamätanú hodnotu x, tak svoju vstupnú hlavu posunie podľa posunu x-tej hlavy automatu A a keďže ešte nevie aké symboly budú čítať vstupné hlavy A v nasledujúcom kroku, tak do zložky stavu, ktorá si ich pamätá, si zatiaľ uloží hodnotu $\underbrace{\#, ..., \#}_{k}$.

Okrem toho si každý proces v tomto kroku vloží do zásobníka typ konfigurácie C automatu A, ktorú má zapamätanú, čo bude hodnota $\square \in \{\forall, \exists, |\}$ a hodnotu $b \in \{0, 1\}$, ktorá hovorí o tom, ktorý prechod (sú maximálne 2) δ-funkcie vykonal, resp. ktorou vetvou výpočtového stromu automatu A bude pokračovať (ak je konfigurácia

jediný nasledovník, tak to bude vždy 0). Týmto spôsobom bude mať každý proces po n odsimulovaných krokoch na zásobníku uložený reťazec $\square_1 b_1 \square_2 b_2 ... \square_n b_n$, ktorý reprezentuje cestu z počiatočnej konfigurácie do C vo výpočtovom strome A. Tomuto reťazcu budeme hovoriť *výpočtová cesta* konfigurácie C.

Ak má δ-funkcia dva prechody z danej konfigurácie, proces sa rozvetví na dva nové procesy, pričom každý z nich si podľa svojho prechodu uloží nový stav p_j, posunie vstupnú hlavu o $d_{j,x}$, do zásobníka vloží typ konfigurácie podľa funkcie f_{type} a symbol z $j \in \{0,1\}$ podľa toho, ktorý prechod realizuje.

$$\forall x \in \{1,...,k\},\ \forall q \in Q,\ \forall sig \in \{0,1\},\ \forall a_1,...,a_k \in \Sigma \cup \{¢,\$\},\ \forall \gamma \in \Gamma$$

$$\text{t.ž. } \delta(q,a_1,...,a_k) = \{(p_0,d_{0,1},...,d_{0,k}),(p_1,d_{1,1},...,d_{1,k})\} :$$

$$\Delta([0,sig,x,q,a_1,...,a_k,0,-,-,-],\varepsilon,a_x,\gamma) =$$
$$\{([0,sig+1,x,p,\underbrace{\#,...,\#}_{k},1,-,-,-],\varepsilon,\gamma f_{type}(q,a_1,...,a_k)j,d_{j,x}) \mid j \in \{0,1\}\}$$

Analogicky ak má δ-funkcia iba jeden prechod, akurát v tom prípade vloží do zásobníka ako číslo vetvy vždy 0.

$$\delta(q,a_1,...,a_k) = \{(p,d_1,...,d_k)\} :$$

$$\Delta([0,sig,x,q,a_1,...,a_k,0,-,-,-],\varepsilon,a_x,\gamma) =$$
$$\{([0,sig+1,x,p,\underbrace{\#,...,\#}_{k},1,-,-,-],\varepsilon,\gamma f_{type}(q,a_1,...,a_k)0,d_x)\}$$

kde $f_{type} : Q \times (\Sigma \cup \{¢,\$\})^k \to \{\exists,\forall,|\}$ je funkcia priraďujúca konfigurácii stroja A jej typ, definovaná ako

$$f_{type}(q,a_1,...,a_k) = \begin{cases} \forall & \text{ak } g(q) = \wedge \text{ a zároveň } |\delta(q,a_1,...,a_k)| = 2 \\ \exists & \text{ak } g(q) = \vee \text{ a zároveň } |\delta(q,a_1,...,a_k)| = 2 \\ | & \text{inak} \end{cases}$$

Teraz si procesy musia ešte zistiť aktuálne symboly čítané hlavami automatu A. Každý má informáciu aký symbol číta jemu prislúchajúca hlava A, lebo je to práve ten symbol, ktorý momentálne číta jeho vstupná hlava a zároveň procesy zodpovedajúce vstupným hlavám rovnakej konfigurácie automatu A majú rovnaký obsah zásobníka, lebo simulovali kroky podľa rovnakých prechodov. To znamená, že sú v tomto kroku naladení na rovnaký kanál a najbližších k krokov v poradí ako prislúchajú jednotlivým hlavám, vždy jeden proces vyšle na kanál informáciu, aký symbol číta jeho hlava a ostatné procesy zatiaľ počúvajú a zapamätajú si ho. Týmto spôsobom budú mať po k krokoch všetky procesy informáciu o čítaných symboloch a simulácia jedného kroku automatu A je ukončená.

V prvom kroku bude prvý proces vysielať svoj čítaný symbol a, ostatné procesy budú vysielať ε (počúvať) a všetci si zvýšia o 1 počítadlo krokov, ktoré hovorí o tom, kto má vysielať.

$$\forall q \in Q,\ \forall sig \in \{0,1\},\ \forall a \in \Sigma \cup \{¢,\$\},\ \forall \gamma \in \Gamma :$$

$$\Delta([0, sig, 1, q, \underbrace{\#, ..., \#}_{k}, 1, -, -, -], \varepsilon, a, \gamma) = \{([0, sig{+}1, 1, p, \underbrace{\#, ..., \#}_{k}, 2, -, -, -], a, \gamma, 0)\}$$

$$\forall x \in \{2, ..., k\} :$$

$$\Delta([0, sig, x, q, \underbrace{\#, ..., \#}_{k}, 1, -, -, -], \varepsilon, a, \gamma) = \{([0, sig{+}1, x, p, \underbrace{\#, ..., \#}_{k}, 2, -, -, -], \varepsilon, \gamma, 0)\}$$

Následne vždy v x-tom kroku si každý proces na $(x-1)$-vú pozíciu uloží prijatý symbol z predošlého kroku, x-tý proces začne vysielať svoj čítaný symbol a ostatné budú počúvať.

$$\forall x \in \{2, ..., k\},\ \forall q \in Q,\ \forall sig \in \{0,1\},\ \forall a_1, ..., a_k, a \in \Sigma \cup \{¢,\$,\#\},\ \forall \gamma \in \Gamma,\ \forall r \in R :$$

$$\Delta([0, sig, x, q, a_1, ..., a_k, x, -, -, -], r, a, \gamma) = \{([0, sig+1, x, p, a_1, ..., a_{x-2}, r, a_x, ..., a_k, x+1, -, -, -], a, \gamma, 0)\}$$

$$\forall x \in \{1, ..., k\}, \forall y \in \{2, ..., k\} : x \neq y :$$

$$\Delta([0, sig, x, q, a_1, ..., a_k, y, -, -, -], r, a, \gamma) = \{([0, sig+1, x, p, a_1, ..., a_{y-2}, r, a_y, ..., a_k, x+1, -, -, -], \varepsilon, \gamma, 0)\}$$

V $(k+1)$-vom kroku si každý proces iba uloží posledný prijatý symbol a do počítadla krokov si nastaví 0, čím ukončí jedno kolo a dostane sa do stavu na simulovanie ďalšieho kroku.

$$\forall x \in \{1, ..., k\},\ \forall q \in Q,\ \forall sig \in \{0,1\},\ \forall a_1, ..., a_k, a \in \Sigma \cup \{¢,\$,\#\},\ \forall \gamma \in \Gamma,\ \forall r \in R :$$

$$\Delta([0, sig, x, q, a_1, ..., a_k, k+1, -, -, -], r, a, \gamma) = \{([0, sig+1, x, p, a_1, ..., a_{k-1}, r, 0, -, -, -], \varepsilon, \gamma, 0)\}$$

Môžeme b.u.n.v predpokladať, že procesy automatu A nerobia po dosiahnutí akceptačného/zamietajúceho stavu už žiadne kroky a teda týmto spôsobom vieme simulovať každý proces vo výpočte automatu A, kým nedosiahne jeden z týchto stavov (alebo sa bude cykliť donekonečna).

Akonáhle sa proces na začiatku kola dostane do stavu q_{accept}/q_{reject}, tak si v prvej zložke stavu vymení hodnotu 0 za 1, čo značí, že začal druhú fázu simulácie. Takisto k-ticu čítaných symbolov nahradí za $\underbrace{\#,...,\#}_{k}$, lebo ich už nebude potrebovať.

$$\forall x \in \{1,...,k\},\ \forall q \in Q\, \forall sig \in \{0,1\},\ \forall a_1,...,a_k, a \in \Sigma \cup \{¢,\$\},\ \forall \gamma \in \Gamma :$$

$$\text{ak } g(q) \in \{q_{accept}, q_{reject}\} :$$

$$\Delta([0, sig, x, q, a_1, ..., a_k, 0, -, -, -], \varepsilon, a, \gamma) = \{([1, sig{+}1, x, q, \underbrace{\#,...,\#}_{k}, 0, -, -, -], \varepsilon, \gamma, 0)\}$$

V druhej fáze budú procesy stroja M postupne priraďovať konfiguráciám automatu A ich kvalitu, teda hodnotu z $\{Y, N\}$ od koncových až po počiatočnú. Procesy zodpovedajúce rôznym hlavám rovnakej konfigurácie budú odteraz pracovať ako jeden proces. Zabezpečíme, aby po vypočítaní kvality pre konfiguráciu C automatu A platil nasledujúci invariant.

(i) Procesy, ktoré majú vypočítanú kvalitu konfigurácie C, sú práve procesy zodpovedajúce terminálnym konfiguráciám v podstrome zakorenenom v C a na zásobníku majú všetky uloženú výpočtovú cestu do konfigurácie C.

Procesy začnú tým, že podľa stavu konfigurácie automatu A v ktorej skončili, si uložia do stavu hodnotu $qual \in \{Y, N\}$. V tomto prípade je invariant splnený.

$$\forall x \in \{1,...,k\},\ \forall q \in Q : g(q) \in \{q_{accept}, q_{reject}\}\, \forall sig \in \{0,1\},\ \forall a \in \Sigma \cup \{¢,\$\},\ \forall \gamma \in \Gamma :$$

$$\Delta([1, sig, x, q, \underbrace{\#,...,\#}_{k}, 0, -, -, -], \varepsilon, a, \gamma) =$$

$$\{([1, sig+1, x, q, \underbrace{\#,...,\#}_{k}, 0, f_{end}(q), -, -], \varepsilon, \gamma, 0)\}$$

kde $f_{end} : Q \to \{Y, N\}$ je funkcia, ktorá priraďuje koncovej konfigurácii kvalitu podľa jej stavu a je definovaná ako

$$f_{end}(q) = \begin{cases} Y & \text{ak } g(q) = q_{accept} \\ N & \text{ak } g(q) = q_{reject} \end{cases}$$

Nech teraz invariant platí pre nejakú konfiguráciu $C1$ a nech $S1$ je skupina procesov, čo má vypočítanú jej kvalitu, pričom každý z nich má obsah zásobníka $\square_1 b_1 ... \square_j b_j$. Každý proces z $S1$ vypočíta kvalitu konfigurácie C, ktorá je predchodcom $C1$ nasledovne.

Najprv si zo zásobníka vyberie najvrchnejší symbol $b_j \in \{0,1\}$, ktorý určuje, v ktorej je konfigurácia $C1$ vetve a uloží si ho do stavu.

$$\forall x \in \{1,...,k\},\ \forall q \in Q,\ \forall sig \in \{0,1\},\ \forall a \in \Sigma \cup \{¢,\$\},\ \forall b \in \{0,1\},\ \forall qual \in \{Y, N\} :$$

$$\Delta([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual, -, -], \varepsilon, a, b) =$$

$$\{([1, sig + 1, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual, -, b], \varepsilon, \lambda, 0)\}$$

Následne sa pozrie na ďalší symbol $\square_j$ na zásobníku a ak má hodnotu $|$, tak to znamená, že $C1$ je jediný nasledovník konfigurácie C a má rovnakú kvalitu ako C. Takže proces si zapamätanú hodnotu *qual* v stave nemení, odstráni symbol $\square_j$ zo zásobníka a vymaže si v stave predtým zapamätaný symbol b_j. Invariant je očividne splnený.

$$\forall x \in \{1, ..., k\},\ \forall q \in Q,\ \forall sig, b \in \{0, 1\},\ \forall a \in \Sigma \cup \{¢,\$\},\ \forall qual \in \{Y, N\}:$$

$$\Delta([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual, -, b], \varepsilon, a, |) =$$

$$\{([1, sig + 1, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual, -, -], \varepsilon, \lambda, 0)\}$$

Ak ale $\square_j$ má hodnotu $\forall$ alebo $\exists$, tak konfigurácia C má aj druhého nasledovníka $C2$ a existuje skupina procesov $S2$, zodpovedajúcich konfiguráciám v podstrome zakorenenom v $C2$, ktoré by mali mať v nejakom čase vypočítanú kvalitu $C2$ (ak nie sú zacyklené). A tým, že $C1$ a $C2$ majú rovnakú výpočtovú cestu až po konfiguráciu C, tak si ich skupiny procesov budú vedieť poslať informácie na kanáli $\square_1 b_1 ... b_{j-1} \square_j$.

Procesy si od začiatku výpočtu budú v druhej zložke stavu uchovávať informáciu o tom či sa vykonáva párny alebo nepárny krok (premenná *sig*). Teraz bude každý proces z $S1$ podľa uloženej hodnoty b_j v každom párnom (nepárnom) kroku vysielať na kanáli $\square_1 b_1 ... b_{j-1} \square_j$ svoju uloženú kvalitu $qual1$ a v každom nepárnom (párnom) počúvať, či na kanáli nevysiela svoju hodnotu nejaký proces zo skupiny $S2$. Keďže procesy nemusia skončiť prvú fázu simulácie v rovnakom čase, tak týmto čakaním je zabezpečená synchronizácia druhej fázy.

Ak sa zhoduje parita kroku a uložená hodnota b_j, tak proces bude vysielať svoju hodnotu *qual*.

$$\forall x \in \{1, ..., k\},\ \forall q \in Q,\ \forall sig \in \{0, 1\},\ \forall a \in \Sigma \cup \{¢,\$\},\ \forall \square \in \{\forall, \exists\},\ \forall qual \in \{Y, N\}:$$

$$\Delta([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual, -, sig], \varepsilon, a, \square) =$$

$$\{([1, sig + 1, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual, -, sig], qual, \square, 0)\}$$

Ak sa tieto hodnoty nezhodujú, tak proces bude počúvať.

$$\Delta([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual, -, sig + 1], qual1, a, \square) =$$
$$\{([1, sig + 1, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual, -, sig + 1], \varepsilon, \square, 0)\}$$

Ak proces z $S1$ alebo $S2$ prijíme od druhej skupiny hodnotu $qual2$, tak si ju uloží do stavu a v ďalšom kroku pošle znovu svoju hodnotu $qual1$, ale so špeciálnym príznakom $*$, ktorý hovorí o tom, že v ďalšom kroku budú mať obe skupiny procesov k dispozícií kvality oboch konfigurácií $C1$ a $C2$, a teda môžu naraz vypočítať kvalitu konfigurácie C.

$\forall x \in \{1, ..., k\}$, $\forall q \in Q$, $\forall sig \in \{0, 1\}$, $\forall a \in \Sigma \cup \{¢, \$\}$, $\forall \square \in \{\forall, \exists\}$, $\forall qual1, qual2 \in \{Y, N\}$:

$$\Delta([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual1, -, sig], qual2, a, \square) =$$
$$\{([1, sig + 1, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual1, qual2, sig], (qual1, *), \square, 0)\}$$

Ak ľubovoľný proces z $S1$ alebo $S2$ vyšle alebo prijme správu tvaru $(qual1, *)$, tak v ďalšom kroku si z hodnôt $qual1$, $qual2$ a hodnoty $\square_j$ na zásobníku vypočíta novú hodnotu $qual$, ktorú si uloží do stavu, premaže uložené hodnoty $qual2$ a b_j, a $\square_j$ odstráni zo zásobníka. Tým pádom budú všetky procesy z $S1 \cup S2$ mať vypočítanú kvalitu konfigurácie C a na zásobníku budú mať reťazec $\square_1 b_1 ... \square_{j-1} b_{j-1}$, teda invariant je znovu splnený.

$\forall x \in \{1, ..., k\}$, $\forall q \in Q$, $\forall sig \in \{0, 1\}$, $\forall a \in \Sigma \cup \{¢, \$\}$, $\forall \square \in \{\forall, \exists\}$, $\forall qual1, qual2 \in \{Y, N\}$:

$$\Delta([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual1, qual2, sig], (qual1, *), a, \square) =$$
$$\{([1, sig + 1, x, q, \underbrace{\#, ..., \#}_{k}, 0, f_{parent}(qual1, qual2, \square), -, -], \varepsilon, \lambda, 0)\}$$

$$\Delta([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, qual1, -, sig + 1], (qual2, *), a, \square) =$$
$$\{([1, sig + 1, x, q, \underbrace{\#, ..., \#}_{k}, 0, f_{parent}(qual1, qual2, \square), -, -], \varepsilon, \lambda, 0)\}$$

kde $f_{parent} : \{Y, N\} \times \{Y, N\} \times \{\forall, \exists\} \to Y, N$ je funkcia, počítajúca kvalitu predchodcu podľa kvalít jej nasledovníkov a je definovaná ako

$$f_{parent}(qual1, qual2, \square) = \begin{cases} Y & \text{ak} \quad (qual1 = Y \wedge qual2 = Y \wedge \square = \forall) \\ & \quad \vee (qual1 = Y \wedge \square = \exists) \vee (qual2 = Y \wedge \square = \exists) \\ N & \text{inak} \end{cases}$$

Takto budú všetky procesy pokračovať, až kým nebudú mať na zásobníku iba počiatočný symbol Z_0. Ak sa tak stane, znamená to, že všetky procesy majú vypočítanú kvalitu počiatočnej konfigurácie a sú naladené na rovnaký kanál. Následne teda, ak platí $qual = Y$ ($qual = N$), tak všetky procesy vyšlú na kanáli q_{accept} (q_{reject}) a ukončia tým výpočet.

$$\forall x \in \{1, ..., k\}, \forall q \in Q, \forall sig \in \{0, 1\}, \forall a \in \Sigma \cup \{¢,\$\} :$$

$$\Delta([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, Y, -, -], \varepsilon, a, Z_0) = \{([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, Y, -, -], q_{accept}, Z_0, 0)\}$$

$$\Delta([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, N, -, -], \varepsilon, a, Z_0) = \{([1, sig, x, q, \underbrace{\#, ..., \#}_{k}, 0, N, -, -], q_{reject}, Z_0, 0)\}$$

Z konštrukcie je zrejmé, že M akceptuje (zamieta) práve vtedy, keď akceptuje (zamieta) A. HĺBka zásobníka bude počas výpočtu vo všetkých procesoch najviac $c \cdot d$, kde d je hĺbka im prislúchajúcej konfigurácie v strome výpočtu automatu A, pre vhodnú konštantu c. HĺBka konfigurácie stroja A môže byť najviac $T(n)$, čiže kanálová zložitosť stroja M je naozaj $O(T(n))$.

Takisto, v prvej fáze simulácie stačí konštantný počet krokov na odsimulovanie jedného kroku A, teda na prvú fázu stačí celkovo $O(T(n))$ krokov. Podobne, čas na druhú fázu je zas lineárne úmerný veľkosti zásobníka, čo je $O(T(n))$. Teda stroj M naozaj pracuje v časovej zložitosti $O(T(n))$.

□

Kapitola 3

Komunikácia pomocou zásobníka a počítadla

Keďže pri používaní zásobníka na komunikáciu je počet rôznych možných kanálov závislý od veľkosti zásobníkovej abecedy, tak je celkom prirodzené porovnať komunikáciu pomocou zásobníka s komunikáciou pomocou počítadla. Zrejme najprirodzenejšie je pozrieť sa, či vieme komunikáciu pomocou počítadla veľkosti n nahradiť komunikáciou so zásobníkom veľkosti $logn$.

3.1 Priama simulácia

Prvá idea ako odsimulovať komunikáciu s počítadlom pomocou komunikácie so zásobníkom, by mohla byť, spraviť to priamo. Teda obsah počítadla 1^n by bol na zásobníku uložený ako binárne zapísané číslo n s najmenej významnými ciframi na vrchu zásobníka. Pri tomto prístupe, ale narazíme na problém. Ak stroj nemá pracovné pásky, tak sa zdá, že na samotnom zásobníku nevieme v logaritmickej pamäti rozumným spôsobom simulovať inkrementáciu a dekrementáciu počítadla. Ak by sme sa totiž počas inkrementácie hodnoty na zásobníku potrebovali vnoriť hlbšie do zásobníka, aby sme zmenili symbol na nejakej významnejšej pozícií, tak všetky symboly predtým by sme museli zo zásobníka odstrániť a nemali si kde zapamätať ich hodnoty ani počet. Nezdá sa ani, že by pomáhala možnosť viacerých procesov a komunikácie, keďže procesy musia mať rovnaký obsah zásobníka, aby mohli spolu komunikovať.

Dokážeme teda o niečo slabší výsledok, a to že okrem zásobníka na komunikáciu pridáme stroju aj nejakú čo najmenšiu pamäť a pomocou tohto upraveného modelu spravíme simuláciu stroja, ktorý má na komunikáciu pridané počítadlo.

Nech je maximálna veľkosť počítadla $C(n)$ a chceme ho odsimulovať zásobníkom veľkosti $O(log(C(n)))$. Ak by sme stroju pridali okrem zásobníka aj pamäť veľkosti

$O(log(C(n)))$, vedeli by sme si všetky symboly vybraté zo zásobníka uložiť na pásku a simulácia práce s počítadlom by bola v tomto prípade priamočiara.

My sa ale pokúsime pridanú pamäť čo najviac minimalizovať. V skutočnosti si stroj nepotrebuje pamätať hodnotu každého symbolu, ktorý vybral zo zásobníka, ale stačí ich počet. Ak máme totiž pri inkrementácii hodnoty na zásobníku situáciu, že niekoľko najvrchnejších symbolov zásobníka predstavuje cifru 1, tak potrebujeme tieto symboly vybrať von, zmeniť prvý symbol na zásobníku čo predstavuje 0 na 1 a doplniť naspať toľko symbolov 0, koľko bolo na začiatku vybraných symbolov 1. Počet vybraných symbolov zo zásobníka si vieme zapamätať v logaritmickom priestore od veľkosti zásobníka, čiže nám stačí pridať pamäť veľkosti $O(log(log(C(n))))$.

Je tu ale ešte jeden problém, a to že oproti pôvodnému stroju, kde inkrementácia a dekrementácia trvajú vždy jeden krok, pri tomto prístupe počet krokov potrebných na simuláciu jednej zmeny obsahu zásobníka závisí od aktuálnej veľkosti zásobníka. Mohlo by sa teda stať, že vo výpočte na pôvodnom stroji mali byť v konkrétnom čase dva procesy naladené na rovnaký kanál a pri pokuse o simuláciu tohto výpočtu na novom stroji by kvôli rozdielnosti dĺžky operácií tomu tak nebolo a výpočty by neboli ekvivalentné. Aby uvedená idea simulácie fungovala, potrebovala by nejakú formu synchronizácie týchto operácií. Na to aby sme ju zabezpečili, pridáme do nasledujúceho tvrdenia ešte jeden predpoklad navyše a to konštruovateľnosť funkcie $log(log(C(n)+1)$.

Tvrdenie 3.1: Nech $C(n)$ je ľubovoľná funkcia taká, že $log(log(C(n)+1)$ je páskovo konštruovateľná a nech M je $WPFA_1$ pracujúci v kanálovej zložitosti $C(n)$. Potom existuje $WPTM_2$ M', simulujúci M v kanálovej zložitosti $O(log(C(n)))$ a priestorovej zložitosti $O(log(log(C(n))))$.

Dôkaz: Nech $M = (Q, R, \Sigma, \Gamma, \Delta, q_0, r_0, \varepsilon, q_{accept}, q_{reject}, Z_0)$ je $WPFA_1$ a σ je jeho jediný zásobníkový symbol okrem Z_0. Nech jeho prechodová relácia má tvar $\Delta : Q \times R \times (\Sigma \cup \{¢, \$\}) \times \{\sigma, Z_0\} \to Q \times R \times \{-1, 0, 1\}^2$, kde posledná zložka značí prácu s počítadlom. Skonštruujeme k nemu ekvivalentný $WPTM_2$ M' s dvomi pracovnými páskami ako $M' = (2, Q', R, \Sigma, \Gamma', \Delta', q'_0, r_0, \varepsilon, q_{accept}, q_{reject}, Z_0)$.

Nech a_0, a_1 sú symboly, ktoré budú spolu s počiatočným symbolom Z_0 tvoriť pracovnú abecedu stroja M'. Ak teraz budeme chápať symbol a_0 ako 0 a a_1 ako 1, tak obsah zásobníka aj pracovných pások vieme interpretovať ako číslo v binárnej sústave. Tým pádom vieme mať na zásobníku uloženú aktuálnu hodnotu počítadla stroja M.

Obsah zásobníka budeme chápať tak, že na vrchu budú najmenej významné cifry binárneho čísla, ktoré má uložené. Potrebovali by sme ale rozlíšiť situáciu, keď má

počítadlo hodnotu 0, takže zásobník obsahujúci iba počiatočný symbol by mal reprezentovať počítadlo s hodnotou 0. Preto budeme mať binárnu reprezentáciu čísel "o jedno posunutú". Čiže zásobník obsahujúci iba prázdny symbol bude znamenať hodnotu počítadla 0 a ak bude zásobník obsahovať binárny zápis čísla n, tak to bude znamenať hodnotu počítadla $n + 1$. Teda napríklad obsah zásobníka $Z_0a_1a_0a_0$ reprezentuje hodnotu 5. Podobne posunutú budeme mať aj reprezentáciu čísel na pracovných páskach, teda prázdna páska bude reprezentovať hodnotu 0, a_0 bude reprezentovať 1, atď. Zároveň budeme obsah pásky chápať ako binárne číslo s najmenej významnými ciframi naľavo, čiže naopak ako je zvyk písať binárne čísla. Potom vie M' spraviť inkrementáciu zásobníka nasledovne (dekrementácia je obdobná).

- Ak je na vrchu zásobníka symbol Z_0, tak M' iba pridá do zásobníka symbol a_0.
- Ak je na vrchu symbol a_0, tak ho vyberie a vloží tam symbol a_1.
- Ak tam je symbol a_1, tak ho vyberie a inkrementujeme číslo na prvej pracovnej páske. Toto opakuje až kým na vrchu zásobníka nebude symbol a_0 alebo počiatočný symbol. V prvom prípade a_0 vyberie a vloží tam a_1, v druhom prípade iba pridá do zásobníka symbol a_1. Následne bude postupne dekrementovať číslo na prvej pracovnej páske a pri každej dekrementácií do zásobníka pridá symbol a_0, až kým na pracovnej páske nebude hodnota 0, čím je inkrementácia hotová. Pracovná páska tu funguje ako počítadlo odstránených symbolov.

Je potrebné ešte vyriešiť problém s nerovnakými dĺžkami operácií zmeny zásobníka. Najdlhšia možná inkrementácia (dekrementácia) je v prípade, keď musíme vybrať všetky symboly zo zásobníka, keď má maximálnu možnú veľkosť, ktorá je $log(C(n))+1$. Vtedy operácia trvá tak dlho, koľko trvá na pracovnej páske počítať od 0 po $log(C(n))+1$ a zas naspäť. Stačí teda zabezpečiť, aby každý simulovaný krok stroja M trval takto dlho. Tu vieme využiť konštruovateľnosť $log(log(C(n)) + 1)$.

Stroj M' si na začiatku výpočtu na druhej pracovnej páske vyhradí $log(log(C(n))+1)$ miesta. Následne bude simulovať kroky stroja M tak, že si vždy podľa prechodu pôvodnej Δ relácie uloží príslušný stav stroja M, spraví posun vstupnej hlavy a pomocou zásobníka a prvej pracovnej pásky vyššie uvedeným spôsobom odsimuluje prácu s počítadlom. Zároveň bude celý čas na druhej páske počítať binárne od 0 po najvyššie možné číslo vo vyhradenom priestore (čo je $log(C(n)) + 1$) a zas naspäť. Až keď M' dokončí tento cyklus na druhej páske, tak začne simulovať ďalší krok. Komunikačný stav bude vysielať vždy až v poslednom kroku tohto cyklu, aby sa vyhol konfliktom počas prelaďovania kanálu, ktoré neboli v pôvodnom výpočte.

Popíšeme konštruovaný stroj M' formálnejšie.

Pracovná abeceda bude množina $\Gamma' = \{a_0, a_1, \bar{\#}, Z_0\}$, kde # je falošný blank.

Množina stavov M' bude $Q \times R \times Q_{stack} \times Q_{sync}$, kde Q_{stack} sú stavy na prácu so zásobníkom a Q_{sync} sú stavy na prácu s druhou pracovnou páskou, ktorou zabezpečujeme synchronizáciu. Zložka $END \in Q_{stack}$ (Q_{sync}) znamená, že bola ukončená práca so zásobníkom (páskou) a môže sa začať simulovať ďalší krok. Zložka $START \in Q_{sync}$ resp. $START_{inc}, START_{dec} \in Q_{stack}$ značí začiatok práce s páskou, resp. začiatok inkrementácie/dekrementácie zásobníka.

Počiatočný stav M' bude $q'_0 = [q_0, r_0, END, END]$.

Keď sa druhá aj tretia zložka v stave rovná END, tak to znamená, že stroj môže simulovať ďalší krok. Takže podľa prvých dvoch zložiek stavu a čítaného symbolu a pôvodnej prechodovej relácie si proces uloží nové stavy, pohne správne vstupnou hlavou a do tretej zložky si nastaví hodnotu $START$. Zároveň podľa toho či sa v danom prechode počítadlo inkrementuje/dekrementuje/ostane rovnaké si proces do druhej zložky stavu nastaví hodnotu $START_{inc}/START_{dec}/END$. Tým sa začne práca s so zásobníkom a synchronizačnou páskou.

$\forall q, p \in Q$, $\forall r, r' \in R$, $\forall \gamma \in \Gamma$, $\forall a \in \Sigma \cup \{¢, \$\}$ $\forall b_1, b_2 \in \{\#, \bar{\#}\}$ $\forall d \in \{-1, 0, 1\}$:

- $\Delta(q, r, a, \sigma) = (p, r', d, 1)$:

  $\Delta'([q, r, END, END], r, a, b_1, b_2, \gamma) = ([p, r', START_{inc}, START], \varepsilon, b_1, b_2, \gamma, d, 0, 0)$

- $\Delta(q, r, a, \sigma) = (p, r', d, -1)$:

  $\Delta'([q, r, END, END], r, a, b_1, b_2, \gamma) = ([p, r', START_{dec}, START], \varepsilon, b_1, b_2, \gamma, d, 0, 0)$

- $\Delta(q, r, a, \sigma) = (p, r', d, 0)$:

  $\Delta'([q, r, END, END], r, a, b_1, b_2, \gamma) = ([p, r', END, START], \varepsilon, b_1, b_2, \gamma, d, 0, 0)$

Formálne popíšeme ako bude vyzerať inkrementácia hodnoty na zásobníku. Kvôli prehľadnosti vynecháme v konštrukcii zložky stavu nepodstatné pre túto časť simulácie a takisto prácu so vstupnou hlavou a druhou pracovnou páskou. Symbolom γ budeme spravidla označovať symbol na vrchu zásobníka, b budeme označovať čítaný symbol na pracovnej páske a prázdne slovo budeme značiť symbolom λ. Zjednodušená prechodová relácia bude tvaru

$$Q_{stack} \times (\Gamma' \cup \{\#\}) \times \Gamma' \to Q_{stack} \times \Gamma' \times \Gamma'^{*} \times \{-1, 0, 1\}$$

Ak sa na vrchu zásobníku nachádza počiatočný symbol, tak tam M' iba pridá symbol a_0 a podobne ak je na vrchu a_0, tak ho len vyberie a vloží tam symbol a_1. V oboch prípadoch je tým inkrementácia ukončená a stroj prejde do stavu END.

$$\forall b \in \{a_0, a_1, \#, \bar{\#}\} : \quad \Delta'(START_{inc}, b, Z_0) = (END, b, Z_0 a_0, 0)$$
$$\forall b \in \{a_0, a_1, \#, \bar{\#}\} : \quad \Delta'(START_{inc}, b, a_0) = (END, b, a_1, 0)$$

Ak M' na zásobníku číta symbol a_1, tak ho odstráni a začne inkrementovať hodnotu na pracovnej páske, reprezentujúcu počet vyhodených symbolov zo zásobníka.

$$\forall b \in \{a_0, a_1, \#, \bar{\#}\} : \quad \Delta'(START, b, a_1) = (INC, b, \lambda, 0)$$

Znovu ak je prvý čítaný symbol na páske *blank* alebo a_0, tak ho iba stroj prepíše symbolom a_0 (v druhom prípade a_1), čím je inkrementácia pásky hotová.

$$\begin{aligned} \forall \gamma \in \{a_0, a_1\} \forall b \in \{\#, \bar{\#}\} : \quad & \Delta'(INC, b, \gamma) = (REMOVE, a_0, \gamma, 0) \\ \forall \gamma \in \{a_0, a_1\} : \quad & \Delta'(INC, a_0, \gamma) = (REMOVE, a_1, \gamma, 0) \end{aligned}$$

V opačnom prípade stroj prechádza pásku smerom doprava, kým nenarazí na symbol iný ako a_1. Keď taký symbol prečíta, prepíše ho na a_1 a následne prechádza pásku naspäť doľava a všetky symboly a_1 po ceste prepisuuje na a_0, až kým nenarazí na začiatok pásky.

$$\begin{aligned} \forall \gamma \in \{a_0, a_1\} : \quad & \Delta'(INC, a_1, \gamma) = (RIGHT, a_1, \gamma, 1) \\ \forall \gamma \in \{a_0, a_1\} : \quad & \Delta'(RIGHT, a_1, \gamma) = (RIGHT, a_1, \gamma, 1) \\ \forall \gamma \in \{a_0, a_1\} \forall b \in \{a_0, \#, \bar{\#}\} : \quad & \Delta'(RIGHT, b, \gamma) = (LEFT, a_1, \gamma, -1) \\ \forall \gamma \in \{a_0, a_1\} : \quad & \Delta'(LEFT, a_1, \gamma) = (LEFT, a_0, \gamma, -1) \\ \forall \gamma \in \{a_0, a_1\} : \quad & \Delta'(LEFT, ¢, \gamma) = (REMOVE, ¢, \gamma, 1) \end{aligned}$$

Po dokončení inkrementácie M' prejde do stavu $REMOVE$, kde sa znovu pozrie na vrchný symbol zásobníka a ak je tam symbol a_1, tak ho znovu odstráni a v tomto cykle pokračuje, až kým na zásobníku nebude čítať symbol a_0 alebo Z_0.

$$\forall b \in \{a_0, a_1, \#, \bar{\#}\} : \quad \Delta'(REMOVE, b, a_1) = (INC, b, \lambda, 0)$$

Keď stroj prečíta symbol a_0 alebo Z_0, tak vloží do zásobníka a_1 a prejde do stavu DEC, kde sa začne dekrementácia hodnoty na páske.

$$\begin{aligned} \forall b \in \{a_0, a_1, \#, \bar{\#}\} : \quad & \Delta'(REMOVE, b, a_0) = (DEC, b, a_1, 0) \\ \forall b \in \{a_0, a_1, \#, \bar{\#}\} : \quad & \Delta'(REMOVE, b, Z_0) = (DEC, b, Z_0 a_1, 0) \end{aligned}$$

Podobne ako pri inkrementácií aj tu ak je prvý čítaný symbol a_1, tak ho M' len prepíše na a_0 a dekrementácia je hotová.

$$\begin{aligned} \forall \gamma \in \{a_0, a_1\} : \quad & \Delta'(DEC, a_1, \gamma) = (ADD, a_0, \gamma, 0) \\ \forall \gamma \in \{a_0, a_1\} : \quad & \Delta'(DEC, a_0, \gamma) = (RIGHT', a_0, \gamma, 1) \end{aligned}$$

Ak je ale prvý symbol a_0, tak znovu bude stroj prechádzať pásku doprava až kým neprečíta symbol a_1 alebo *blank* (resp. falošný *blank*). Ak prečíta a_1, tak si skontroluje či je na ďalšej pozícií *blank* alebo nejaký iný symbol

$$\begin{aligned} \forall \gamma \in \{a_0, a_1\} : \quad & \Delta'(RIGHT', a_0, \gamma) = (RIGHT', a_0, \gamma, 1) \\ \forall \gamma \in \{a_0, a_1\} : \quad & \Delta'(RIGHT', a_1, \gamma) = (CHECK, a_1, \gamma, 1) \end{aligned}$$

V prvom prípade predchádzajúci symbol a_1 prepíše na falošný *blank*, v druhom prípade na a_0 a v oboch prípadoch začne prechádzať pásku smerom doľava a všetky symboly a_0 prepisovať na a_1, až kým nenarazí na začiatok pásky a ukončí dekrementáciu pásky.

$$\begin{aligned}
\forall\gamma \in \{a_0, a_1\}\, \forall b \in \{\#, \bar{\#}\}: &\quad \Delta'(CHECK, b, \gamma) = (BLANK, \bar{\#}, \gamma, -1) \\
\forall\gamma, b \in \{a_0, a_1\}: &\quad \Delta'(CHECK, b, \gamma) = (OTHER, b, \gamma, -1) \\
\forall\gamma \in \{a_0, a_1\}: &\quad \Delta'(BLANK, a_1, \gamma) = (LEFT', \bar{\#}, \gamma, -1) \\
\forall\gamma \in \{a_0, a_1\}: &\quad \Delta'(OTHER, a_1, \gamma) = (LEFT', a_0, \gamma, -1) \\
\forall\gamma \in \{a_0, a_1\}: &\quad \Delta'(LEFT', a_0, \gamma) = (LEFT', a_1, \gamma, -1) \\
\forall\gamma \in \{a_0, a_1\}: &\quad \Delta'(LEFT', ¢, \gamma) = (ADD, ¢, \gamma, 1)
\end{aligned}$$

Ak stroj pri prechádzaní pásky doprava narazí najskôr na *blank*, tak to musí znamenať, že sa na páske nachádza už iba jeden symbol a_0. Takže stroj iba posunie hlavu doľava, prepíše symbol na falošný *blank* a ukončí dekrementáciu.

$$\begin{aligned}
\forall\gamma \in \{a_0, a_1\}\, \forall b \in \{\#, \bar{\#}\}: &\quad \Delta'(RIGHT', b, \gamma) = (LEFT'', \bar{\#}, \gamma, -1) \\
\forall\gamma \in \{a_0, a_1\}: &\quad \Delta'(LEFT'', a_0, \gamma) = (ADD, \bar{\#}, \gamma, 0)
\end{aligned}$$

Po ukončení dekrementácie hodnoty na páske, stroj prejde do stavu ADD, kde pridá do zásobníka jeden symbol a_0 a znovu začne dekrementáciu pásky. Takto pokračuje, kým je možné dekrementovať hodnotu na páske.

$$\forall\gamma \in \{a_0, a_1\}\, \forall b \in \{a_0, a_1, \#, \bar{\#}\}: \quad \Delta'(ADD, b, \gamma) = (DEC, b, \gamma a_0, 0)$$

Ak na začiatku dekrementácie pásky prečíta stroj *blank* resp.falošný *blank*, tak na páske sa nachádza hodnota 0, teda bol do zásobníka vložený potrebný počet symbolov a inkrementácia zásobníka je ukončená.

$$\forall\gamma \in \{a_0, a_1\}\, \forall b \in \{\#, \bar{\#}\}: \quad \Delta'(DEC, b, \gamma) = (END, \bar{\#}, \gamma, 0)$$

To, že stroj M' akceptuje práve vtedy keď M je zrejmé z konštrukcie. Maximálna veľkosť zásobníka počas výpočtu je $log(C(n)) + 1$ a maximálna veľkosť oboch pások je $log(log(C(n)) + 1) + 1$. Teda M' naozaj pracuje v kanálovej zložitosti $O(log(C(n)))$ a priestorovej zložitosti $O(log(log(C(n))))$.

□

Poznámka 3.2 Simulácia sa dá priamočiaro rozšíriť aj na stroje s pracovnými páskami. Ak máme $WPFA_1$ M s k pracovnými páskami a priestorovou zložitosťou $S(n) \geq log(log(C(n)) + 1) + 1$, tak k nemu vieme zostrojiť $WPTM_2$ M' s $k + 2$ pracovnými páskami a priestorovou zložitosťou $S(n)$. Prvé dve pásky bude M' používať na prácu so zásobníkom rovnako ako v predošlom dôkaze (vďaka podmienke $S(n) \geq log(log(C(n)) + 1) + 1$ sa manipulácia so zásobníkom vojde do priestoru $S(n)$). Na zvyšných k páskach bude M' priamo simulovať pracovné pásky stroja M.

Poznámka 3.3 Pri simulácii sme sa nezaoberali tým, nakoľko sa spomalí výpočet stroja. V každom simulovanom kroku počítame na pracovnej páske od 0 po $log(C(n))+1$ s tým, že v najhoršom prípade musíme pri jednej zmene obsahu pásky prejsť celú pásku veľkosti $O(log(log(C(n))))$. Ak teda časová zložitosť pôvodného stroja M bola $T(n)$, tak pre časovú zložitosť M' dostávame netesný odhad $O(T(n)log(C(n))log(log(C(n))))$.

3.2 Nepriama simulácia

Teraz si ukážeme, že v tvrdení 3.1 sa v skutočnosti vieme zbaviť aj predpokladu konštruovateľnosti, ale simuláciu už nebudeme robiť priamo. Využijeme techniku, kde najprv pomocou $WPTM$ so zásobníkom na komunikáciu a pridanou malou pamäťou odsimulujeme TS určitej pamäťovej zložitosti a následne pomocou neho odsimulujeme $WPFA$ s počítadlom na komunikáciu. Skôr ako sformulujeme nové tvrdenie, si dokážeme dve pomocné lemy, z ktorých dôkaz tvrdenia priamo vyplynie. Myšlienky oboch nasledujúcich dôkazov sú prebraté z dôkazu vety 3.9 v práci [3].

Lema 3.4: Nech A je DTS taký, že $L(A) \in DSPACE(nC(n))$, kde $C(n)$ je ľubovoľná funkcia. Potom existuje $WPTM$ M, simulujúci A v kanálovej zložitosti $O(log(C(n)))$ a pamäťovej zložitosti $O(log(log(C(n))))$.

Dôkaz: Nech $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$ je DTS priestorovej zložitosti $c \cdot n \cdot C(n)$, pre nejakú konštantu c. Keďže nás nezaujíma časová zložitosť, môžeme b.u.n.v. predpokladať, že stroj má len jednu, jednosmerne nekonečnú pásku. Skonštruujeme k nemu ekvivalentný $WPTM$ M s jednou pracovnou páskou.

M si bude v jednom kroku pamätať konfiguráciu stroja A nasledovným spôsobom: pre každé políčko pásky stroja A bude mať vytvorený jeden proces, ktorý si v stave bude pamätať obsah daného políčka. Páska stroja A bude rozdelená na $d \cdot C(n)$ častí, každá s veľkosťou $n + 2$, kde d je konštanta závislá od c, pričom procesy zodpovedajúce políčkam v i-tej časti budú naladené na kanál i (na zásobníku budú mať uloženú hodnotu reprezentujúcu i rovnako, ako v predošlom dôkaze).

Nech sa pre nejaké vstupné slovo w pracovná hlava stroja A po t-tom kroku nachádza na j-tom políčku v časti i na páske kde $0 \leq j < n+2$ a $0 \leq i < C(n)$. Zabezpečíme aby pre stroj M platil po odsimulovaní t-tého kroku nasledujúci invariant (ii):

- Ak procesu prislúcha políčko v časti i, tak jeho vstupná hlava si bude pamätať vzdialenosť jemu prislúchajúceho políčka od j-teho políčka. To znamená, že ak je ich vzdialenosť k, tak hlava procesu sa bude nachádzať na k-tom symbole vstupu a v stave si bude proces pamätať či sa políčko nachádza naľavo alebo napravo od j.

- Ak procesu prislúcha políčko v jednej z častí $0, ..., i-1$, tak si jeho vstupná hlava bude vyššie uvedeným spôsobom pamätať vzdialenosť od najpravejšieho políčka v danej časti.

- Podobne ak procesu prislúcha políčko v jednej z častí $i+1, ..., d \cdot C(n)$, tak si jeho vstupná hlava bude vyššie uvedeným spôsobom pamätať vzdialenosť od najľavejšieho políčka v danej časti.

Proces, ktorý zodpovedá práve skúmanému políčku budeme nazývať aktívny a časti pásky v ktorej sa nachádza budeme hovoriť aktívna časť (označenia sme takisto prebrali z práce [3]).

Formálnejšie nech $M = (1, Q', R, \Sigma, \{a_0, a_1, \bar{\#}, Z_0\}, \Delta, q'_0, r_0, \varepsilon, q_{accept}, q_{reject})$ pričom stav stroja M bude 8-tica

$$[faze, q, b, active, side, position, highest, stack]$$

kde prvá zložka hovorí o fáze výpočtu, v ďalších dvoch zložkách sú zapamätané informácie o stroji A, pričom q je aktuálny stav A a b je symbol práve čítaný jeho hlavou. Ďalej v zložke *active* si proces pamätá, či je jeho políčko v aktívnej časti alebo nie, *side* hovorí či je proces naľavo alebo napravo od aktívneho procesu (relevantné len pre aktívnu časť), *position* hovorí, či je proces naľavo, napravo alebo uprostred svojej časti pásky stroja A, *highest* určuje, či je kanál procesu aktuálne najvyšší vytvorený a *stack* je stav na prácu so zásobníkom pri prelaďovaní kanálu. Formálne zapísaná množina stavov môže vyzerať takto.

$$Q' = Q_{faze} \times Q \times (\Sigma \cup \{¢, \#, -\}) \times \{0, 1, -\}^2 \times \{-1, 0, 1, -\} \times \{0, 1, -\} \times Q_{stack}$$

kde $Q_{faze} = \{q_{make}, q_{wait}, q_{ready}, q_{request}, q_{final}\}$ a Q_{stack} je rovnaká množina stavov ako v predošlom dôkaze.

Stroj M bude pracovať tak, že v danom momente bude mať vytvorených iba toľko skupín procesov, koľko je potrebných. Ak by pôvodný stroj A vyšiel svojou vstupnou hlavou do časti pásky, pre ktorú M ešte nemá vytvorené procesy, tak si M vytvorí novú skupinu procesov naladenú na nový najvyšší kanál. Nižšie popíšeme, akým spôsobom bude M vytvárať nové procesy.

V nasledujúcej konštrukcii pre jednoduchosť vynecháme prácu so zásobníkom a pracovnou páskou stroja, keďže sa s nimi pracuje iba pri prelaďovaní kanálov. Zjednodušená prechodová relácia bude mať teda tvar:

$$Q' \times R \times (\Sigma \cup \{¢, \$\}) \to Q' \times R \times \{-1, 0, 1\}$$

Na začiatku sa počiatočný proces inicializuje, teda si nastaví prvú zložku stavu na q_{make}, keďže ide vytvárať nové procesy a zapamätá si v stave počiatočný stav stroja A.

$$\Delta(q'_0, r_0, ¢) = ([q_{make}, q_0, -, -, -, -, -], \varepsilon, 0)$$

Následne proces vyrobí prvých $n+2$ procesov tak, že bude posúvať vstupnou hlavou a postupne vyrábať svoje kópie. Presnejšie sa vždy proces v stave q_{make} rozvetví na dva procesy, z čoho jeden posunie vstupnú hlavu doprava a ostáva v stave q_{make}, a druhý ostane stáť na mieste, stav si zmení na q_{wait}, zapamätá si v stave čítaný symbol a podľa neho si správne nastaví ostatné zložky stavu.

$$\Delta([q_{make}, q_0, -, -, -, -, -], \varepsilon, ¢) = \\ \{([q_{make}, q_0, -, -, -, -, -], \varepsilon, 1), ([q_{wait}, q_0, ¢, 1, 1, -1, 1, -], \varepsilon, 0)\}$$

$$\forall a \in \Sigma : \Delta([q_{make}, q_0, -, -, -, -, -], \varepsilon, a) = \\ \{([q_{make}, q_0, -, -, -, -, -], \varepsilon, 1), ([q_{wait}, q_0, a, 1, 1, 0, 1, -], \varepsilon, 0)\}$$

Procesy v stave q_{wait} len čakajú na správu q_{ready}, signalizujúcu ukončenie vytváracej fázy a ich hlavy ostávajú na mieste.

$$\forall a \in \Sigma \cup \{¢\} \forall pos \in \{-1, 0\} :$$

$$\Delta([q_{wait}, q_0, a, 1, 1, pos, 1, -], \varepsilon, a) = \{([q_{wait}, q_0, a, 1, 1, pos, 1, -], \varepsilon, 0)\}$$

Keď proces v stave q_{make} prečíta symbol \$, tak vie, že došiel na koniec vstupu, do stavu si zapamätá symbol # (blank) a na kanáli vyšle správu q_{ready} (uvedomme si, že všetky doteraz vytvorené procesy sú naladené na ten istý kanál).

$$\Delta([q_{make}, q_0, -, -, -, -, -], \varepsilon, \$) = \{([q_{wait}, q_0, \#, 1, 1, 1, 1, -], q_{ready}, 0)\}$$

Ak procesy v stave q_{wait} prijmú správu q_{ready}, tak vedia, že všetky procesy sú pripravené, do prvej zložky stavu si uložia stav q_{ready} a môžu začať simulovať krok stroja A. Môžeme vidieť, že invariant (ii) je po tejto fáze splnený.

$$\forall a \in \Sigma \cup \{¢, \$\} \forall b \in \Sigma \cup \{¢, \#\} \forall pos \in \{-1, 0, 1\} :$$

$$\Delta([q_{wait}, q_0, b, 1, 1, pos, 1, -], q_{ready}, a) = \{([q_{ready}, q_0, b, 1, 1, pos, 1, -], \varepsilon, 0)\}$$

Predpokladajme teraz, že platí invariant (ii) a ukážme ako M odsimuluje jeden krok výpočtu stroja A. Aktívny proces pozná stav aj symbol čítaný strojom A a vie teda určiť jeho ďalší krok. Uloží si do stavu zapísaný symbol aj nový stav stroja a na kanáli v aktívnej časti pošle správu o novom stave a pohybe hlavy stroja A. Procesy podľa toho následne upravia pozície svojich vstupných hláv.

Formálne popíšeme iba pohyb hlavy smerom doprava. Ak je aktívny proces (proces so vstupnou hlavou na ľavej zarážke) naľavo alebo niekde uprostred jemu prislúchajúcej časti pásky stroja A, tak pri pohybe hlavou doprava z nej A určite nevyjde. Simulácia jedného kroku stroja A bude mať v tomto prípade dva kroky. V prvom kroku aktívny proces podľa zapamätaného stavu a symbolu čítaného strojom A, vyšle na kanáli informáciu o novom stave a pohybe hlavy doprava. Zároveň bude po simulácii tohto kroku určite naľavo od aktívneho procesu, a teda si zmení príslušnú zložku v stave na 0.

$$\forall q \in Q \backslash F \, \forall p \in Q \, \forall a, b \in \Gamma \, \forall pos \in \{-1, 0\} \, \forall sid, hig \in \{0, 1\} \text{ t.ž. } \delta(q, a) = (p, b, 1) :$$

$$\Delta([q_{ready}, q, a, 1, sid, pos, hig, -], \varepsilon, ¢) = ([q_{ready}, p, b, 1, 0, pos, hig, -], (p, 1), 0)$$

Ostatné procesy v aktívnej časti v tomto kroku iba čakajú na prijatie informácie od aktívneho procesu.

$$\forall q \in Q \, \forall a \in \Gamma \, \forall pos \in \{-1, 0, 1\} \, \forall sid, hig \in \{0, 1\} \, \forall b \in \Sigma \cup \{\$\} :$$

$$\Delta([q_{ready}, q, a, 1, sid, pos, hig, -], \varepsilon, b) = ([q_{ready}, q, a, 1, sid, pos, hig, -], \varepsilon, 0)$$

V druhom kroku, potom čo procesy prijmú informáciu o novom stave a pohybe hlavy, v tomto prípade doprava, si nový stav uložia a správne posunú svoju vstupnú hlavu. Teda ak boli vľavo od aktívneho procesu a majú zapamätanú hodnotu 0, tak sa ich vzdialenosť od aktívneho procesu zväčšila a posunú svoju vstupnú hlavu smerom doprava. Naopak, ak boli od neho vpravo, tak posunú svoju vstupnú hlavu doľava.

$$\forall q, p \in Q \, \forall a \in \Gamma \, \forall pos \in \{-1, 0, 1\} \, \forall hig \in \{0, 1\} \, \forall b \in \Sigma \cup \{\$\} :$$

$$\Delta([q_{ready}, q, a, 1, 0, pos, hig, -], (p, 1), b) = ([q_{ready}, p, a, 1, 0, pos, hig, -], \varepsilon, 1)$$

$$\Delta([q_{ready}, q, a, 1, 1, pos, hig, -], (p, 1), b) = ([q_{ready}, p, a, 1, 1, pos, hig, -], \varepsilon, -1)$$

Všetky procesy v neaktívnych častiach budú iba čakať v rovnakom stave a nebudú hýbať vstupnou hlavou.

$$\forall q \in Q \, \forall a \in \Gamma \, \forall pos \in \{-1, 0, 1\} \, \forall sid, hig \in \{0, 1\} \, \forall b \in \Sigma \cup \{¢, \$\} :$$

$$\Delta([q_{ready}, q, a, 0, sid, pos, hig, -], \varepsilon, b) = ([q_{ready}, q, a, 0, sid, pos, hig, -], \varepsilon, 0)$$

Ak by sa stalo, že hlava stroja A pri pohybe prejde do novej časti, tak sa aktívny proces musí preladiť na susedný kanál, vyslať tam informáciu, že táto oblasť je odteraz aktívna a preladiť sa späť. Podrobnejšie to opäť spravíme iba pre prípad pohybu hlavy doprava.

Nech je aktívny proces pravý krajný v aktívnej časti a podľa δ-funkcie stroja A sa má pracovná hlava A posunúť doprava. Takisto, nech kanál v aktívnej časti nie je aktuálne najvyšší, teda proces sa má kam preladiť. Proces si poslednú zložku stavu nastaví na $START_{inc}$ a následne rovnako ako v predošlom dôkaze zvýši pomocou pracovnej pásky hodnotu na svojom zásobníku o 1.

$$\forall q \in Q \backslash F \, \forall p \in Q \, \forall a, b \in \Gamma \, \forall sid \in \{0,1\} \text{ t.ž. } \delta(q,a) = (p,b,1) :$$

$$\Delta([q_{ready}, q, a, 1, sid, 1, 0, -], \varepsilon, ¢) = ([q_{ready}, q, a, 1, sid, 1, 0, START_{inc}], \varepsilon, 0)$$

Keď dokončí inkrementáciu a dostane sa do stavu END_{inc}, tak vyšle na tomto kanáli správu o aktuálnom stave stroja a zároveň prejde do stavu $START_{dec}$ a začne dekrementovať zásobník.

$$\forall q \in Q \backslash F \, \forall p \in Q \, \forall a, b \in \Gamma \, \forall sid \in \{-1,1\} \text{ t.ž. } \delta(q,a) = (p,b,1) :$$

$$\Delta([q_{ready}, q, a, 1, sid, 1, 0, END_{inc}], \varepsilon, ¢) = ([q_{ready}, q, b, 1, sid, 1, 0, START_{dec}], p, 0)$$

Keď procesy v neaktívnej časti prijmú správu o stave stroja, v stave si nastavia zložku, ktorá hovorí, či je časť aktívna na 1 a sú pripravené simulovať ďalší krok stroja A.

$$\forall p, q \in Q \, \forall a \in \Gamma \, \forall pos \in \{-1,0,1\} \, \forall sid, hig \in \{0,1\} \, \forall b \in \Sigma \cup \{¢, \$\} :$$

$$\Delta([q_{ready}, q, a, 0, sid, pos, hig, -], p, b) = ([q_{ready}, p, a, 1, sid, pos, hig, -], \varepsilon, 0)$$

Potom čo aktívny proces ukončí dekrementáciu a dostane sa naspäť na svoj pôvodný kanál, vyšle na ňom správu q_{exit}, ktorú keď procesy prijmú, tak si všetky zmenia príslušnú zložku stavu na 0, čím určia, že sú odteraz v neaktívnej časti.

$$\forall q, p \in Q \, \forall a, b \in \Gamma \, \forall x \in \{0,1\} :$$

$$\Delta([q_{ready}, q, a, 1, sid, 1, 0, END_{dec}], \varepsilon, ¢) = ([q_{ready}, q, b, 1, sid, 1, 0, -], q_{exit}, 0)$$

$$\forall q \in Q \, \forall a \in \Gamma \, \forall pos \in \{-1,0,1\} \, \forall sid, hig \in \{0,1\} \, \forall b \in \Sigma \cup \{¢, \$\} :$$

$$\Delta([q_{ready}, q, a, 1, sid, pos, hig, -], q_{exit}, b) = ([q_{ready}, q, a, 0, sid, pos, hig, -], \varepsilon, 0)$$

Ak by sa stalo, že sa hlava pôvodného stroja má posunúť do novej časti, ale aktuálna aktívna časť je zároveň naladená na najvyšší vytvorený kanál, tak sa musí vytvoriť nová skupina procesov, naladená na o jedno vyšší kanál. Aktívny proces si rovnako ako vyššie inkrementuje zásobník, ale namiesto toho aby poslal informáciu o stave, bude teraz rovnakým spôsobom ako na začiatku výpočtu, vytvárať novú skupinu procesov s tým, že každý bude mať obsah políčka # a následne sa preladí naspäť. Potom okrem informácie, že táto časť už nie je aktívna, pošle na kanáli informáciu, že ani číslo ich kanála už nie je aktuálne najvyššie.

Stačí už len popísať ako bude M' akceptovať. B.u.n.v môžeme predpokladať, že A nerobí z akceptačného stavu už žiadne kroky. Ak stroj A prejde počas výpočtu do akceptačného alebo zamietajúceho stavu (zasekne sa), tak procesy v aktívnej časti budú mať o tom informáciu.

Keď sa teda aktívny proces dostane do stavu, v ktorom má uložený stav $q_F \in F$ stroja A, tak sa začne fáza akceptácie. Konkrétne sa proces preladí o kanál vyššie, vyšle tam správu $q_{request}$ a preladí sa naspäť. Pre zamietnutie by nasledujúca fáza prebiehala analogicky.

$$\forall q_F \in F\, \forall a \in \Gamma\, \forall sid \in \{0,1\}\, \forall pos \in \{-1,0,1\} :$$

$$\Delta([q_{ready}, q_F, a, 1, sid, pos, 0, -], \varepsilon, ¢) = ([q_{ready}, q_F, a, 1, sid, pos, 0, START_{inc}], \varepsilon, 0)$$

$$\Delta([q_{ready}, q_F, a, 1, sid, pos, 0, END_{inc}], \varepsilon, ¢) = ([q_{ready}, q_F, a, 1, sid, pos, 0, START_{dec}], q_{request}, 0)$$

Keď proces s hlavou na ľavej zarážke zachytí správu $q_{request}$, prepne sa do stavu $q_{request}$ a takisto sa preladí o kanál vyššie a pošle tam $q_{request}$.

$$\forall q \in Q\, \forall a \in \Gamma\, \forall sid \in \{0,1\}\, \forall pos \in \{-1,0,1\} :$$

$$\Delta([q_{ready}, q, a, 0, sid, pos, 0, -], q_{request}, ¢) = ([q_{request}, q, a, 0, sid, pos, 0, START_{inc}], \varepsilon, 0)$$

$$\Delta([q_{request}, q, a, 0, sid, pos, 0, END_{inc}], \varepsilon, ¢) = ([q_{request}, q, a, 0, sid, pos, 0, START_{dec}], q_{request}, 0)$$

Takto to pokračuje, až kým sa správa nedostane do časti s najvyšším kanálom. Keď procesy v nej zachytia $q_{request}$ (alebo ak sa priamo v tejto časti akceptuje), tak sa všetky prepnú do stavu q_{final} a preladia sa o kanál nižšie.

$$\forall q \in Q\, \forall a \in \Gamma\, \forall b \in \Sigma \cup \{¢, \$\} \forall sid \in \{0,1\}\, \forall pos \in \{-1,0,1\} :$$

$$\Delta([q_{ready}, q, a, 0, sid, pos, 1, -], q_{request}, b) = ([q_{final}, q, a, 0, sid, pos, 1, START_{dec}], \varepsilon, 0)$$

$$\forall q_F \in F :$$

$$\Delta([q_{ready}, q_F, a, 1, sid, pos, 1, -], \varepsilon, b) = ([q_{final}, q, a, 1, sid, pos, 1, START_{dec}], \varepsilon, 0)$$

Následne vždy po preladení sa na nižší kanál, procesy v stave q_{final} vyšlú na kanáli správu q_{final} a keď ju ostatné procesy zachytia, tak sa všetci prepnú do stavu q_{final}, preladia sa o kanál nižšie a takto sa pokračuje ďalej cez všetky kanály.

$$\forall q \in Q\, \forall a \in \Gamma\, \forall b \in \Sigma \cup \{¢, \$\} \forall pos \in \{-1, 0, 1\}\, \forall sid, hig, act \in \{0, 1\} :$$

$$\Delta([q_{final}, q, a, act, sid, pos, hig, END_{dec}], \varepsilon, b) = ([q_{final}, q, a, act, sid, pos, hig, -], q_{final}, 0)$$

$$\Delta([q_{ready}, q, a, act, sid, pos, hig, -], q_{final}, b) = ([q_{final}, q, a, act, sid, pos, hig, START_{dec}], \varepsilon, 0)$$

$$\Delta([q_{final}, q, a, act, sid, pos, hig, -], q_{final}, b) = ([q_{final}, q, a, act, sid, pos, hig, START_{dec}], \varepsilon, 0)$$

Keď budú mať procesy v stave q_{final} na zásobníku iba symbol Z_0, tak vedia, že sú všetky naladené na kanál Z_0 a môžu naraz vyslať na kanáli q_{accept} a ukončiť tým výpočet.

Rôzne dlhá manipulácia so zásobníkom pri operáciach inkrementácie a dekrementácie v tomto prípade nebude problém, lebo prácu so zásobníkom vykonáva vždy iba proces v aktívnej časti a procesy v ostatných častiach iba čakajú. Je zrejmé, že M akceptuje vstup práve vtedy keď akceptuje A. Procesy budú mať počas výpočtu hodnotu kanála najviac $d \cdot C(n)$, čo si podľa tvrdenia 3.1 vedia zapamätať v zásobníku hĺbky $log(c \cdot C(n)) + 1 = O(log(C(n)))$ a stačí im pamäť $O(log(log(C(n))))$ aby s ním vedeli pracovať.

□

Lema 3.5: Nech M je $WPFA_1$ pracujúci v kanálovej zložitosti $C(n)$, potom existuje Turingov stroj A taký, že $L(A) \in DSPACE(nC(n))$ simulujúci M.

Dôkaz: Dôkaz je skoro identický s dôkazom druhej inklúzie vety 3.9 v práci [3], akurát prispôsobený tomu, že simulujeme $WPFA$ s počítadlom, narozdiel od pôvodného dôkazu, kde bol model s kanálovou páskou.

Majme $WPFA_1$ $M = (Q, R, \Sigma, \Gamma, \Delta, q_0, r_0, \varepsilon, q_{accept}, q_{reject}, Z_0)$ a skonštruujeme k nemu ekvivalentný DTS A.

Stroj A bude mať jednu vstupnú a dve pracovné pásky. Na pracovných páskach si bude pamätať všetky konfigurácie M, ktoré sa v aktuálnom kroku výpočtu stroja M vyskytujú nasledovným spôsobom.

Obe pásky, budú mať po odsimulovaní jedného kroku M tvar

$$¢c_0 \bullet c_1 \bullet c_2 \bullet ...$$

kde c_i je popis konfigurácií, ktoré majú hodnotu počítadla rovnú i.

c_i bude tvaru

$$k_0 \star k_1 \star k_2 \star ... \star k_{n+1},$$

kde k_j je popis konfigurácií s hodnotou počítadla rovnou i a so vstupnou hlavou na pozícii j.

k_j bude obsahovať pracovné a komunikačné stavy všetkých konfigurácií aktuálne sa nachádzajúcich vo výpočte M a bude mať tvar

$$q_0 r_0 * q_1 r_1 * ...$$

Ak by sa v danom momente nevyskytovala vo výpočte žiadna konfigurácia s hodnotou počítadla i a vstupnou hlavou na pozícií j, tak namiesto popisu konfigurácie k_j budú na páske dva symboly $--$.

Na jednej páske bude mať stroj A zaznamenaný stav výpočtu M pred vykonaním kroku a na druhú páske bude postupne zapisovať stav po jeho vykonaní. V stave si bude pamätať, ktorá páska zodpovedá ktorému stavu výpočtu a vždy po odsimulovaní kroku stroja M význam pások vymení. Nech je teraz na prvej páske stav výpočtu pred vykonaním kroku. Ukážeme, ako A odsimuluje jeden krok stroja M a správne zmení obsah druhej pásky.

Najprv A vymaže obsah druhej pásky (zapíše na ňu falošné blanky) a následne na ňu prepíše obsah prvej pásky s tým ale, že namiesto každého popisu konfigurácie k_j zapíše na pásku reťazec $--$. Následne bude postupne prechádzať prvú pásku a podľa konfigurácií zapísaných na nej bude počítať nové konfigurácie a zapisovať ich na druhú pásku.

Vždy keď bude stroj čítať na prvej páske časť k_j v sekcii c_i pre nejaké i a j, tak zabezpečíme aby jeho hlava na druhej páske bola na začiatku časti k_j v sekcii c_i a vstupnú hlavu mal na pozícii j.

Ak bude A na prvej páske čítať symbol $-$, tak sa na oboch pracovných páskach iba posunie do ďalšej časti a posunie o jednu pozíciu doprava aj svoju vstupnú hlavu.

V opačnom prípade vždy prečíta pracovný a komunikačný stav zapísaný na páske, a oba si uloží do svojho stavu. Spolu s čítaným symbolom na vstupnej páske má všetky informácie na určenie kroku stroja M z tejto konfigurácie. Nech má stroj A zapamätané stavy q,r a na vstupnej páske číta symbol a. Ďalej nech pre Δ-funkciu M platí

$$\Delta(q, r, a, \sigma) = \{(q'_1, r', d_1, e_1), ..., (q'_k, r', d_k, e_k)\}$$

kde σ je zásobníkový (počítadlový) symbol, $d_i \in \{-1, 0, 1\}$ je pohyb vstupnej hlavy a $e_i \in \{-1, 0, 1\}$ je práca s počítadlom. Potom A bude postupne zapisovať na druhú pásku dvojice (q'_i, r') na správnu pozíciu, podľa hodnôt d_i, e_i.

- Ak pre konkrétne i platí $d_i = e_i = 0$, tak sa pozícia vstupnej hlavy ani hodnota počítadla M nezmenila a stroj A môže dvojicu zapísať do časti k_j v ktorej sa nachádza. Ak sú v danej časti zapísané symboly $--$, tak ich iba prepíše stavmi $q'_i r'$. Ak sú v nej už ale zapísané nejaké iné stavy, tak stroj skontroluje, či sa tam už daná dvojica nenachádza a ak nie, tak posunie obsah pásky od začiatku danej časti o tri miesta a zapíše na vytvorené miesto reťazec $q'_i r' *$.

- Ak platí $d_i = 1$ a $e_i = 0$, tak vstupná hlava M sa posunula doprava, ale hodnota počítadla ostala rovnaká. Vtedy stroj A posunie hlavu na druhej páske do susednej časti k_{j+1} (bude posúvať hlavu na druhej páske doprava až kým nenarazí na symbol $\star$) a rovnakou procedúrou ako predtým na pásku zapíše dvojicu stavov $q'_i r'$.

- Nakoniec ak $e_i = 1$, tak sa zmenila hodnota počítadla, a stroj A bude musieť dvojicu stavov zapísať do vedľajšej sekcie c_{i+1}. Podobne ako vyššie, bude posúvať hlavu na druhej páske, až kým nenarazí na symbol $\bullet$. Potom si na prvej páske urobí značku na začiatku časti k_j v sekcii c_i a prejde na začiatok sekcie c_i. Následne bude obe hlavy na pracovných páskach posúvať doprava a počítať symboly $\star$, až kým na prvej páske nenarazí na značku, ktorú spravil. Tým dostane hlavu na druhej páske do časti k_j v sekcii c_{i+1}. Potom podľa pohybu vstupnej hlavy M zapíše na pásku dvojicu stavov tak ako predtým.

Ak by sa stalo, že počítadlo stroja M sa inkrementuje na hodnotu $i + 1$ aká ešte vo výpočte predtým nebola, tak A musí na druhej páske vytvoriť novú sekciu c_{i+1}. Spraví to tak, že bude hýbať vstupnou hlavou a zapisovať na koniec pásky symboly $--\star$, až kým nedôjde na koniec vstupu. Potom zapíše na pásku nové stavy spôsobom popísaným vyššie.

Následne ešte musí A odsimulovať komunikáciu v danom kroku. Takže pre každý kanál c_i na prvej páske prečíta znovu všetky stavy konfigurácií, nachádzajúcich sa v ňom a v stave si zapamätá, aký komunikačný stav vysielajú. Ak všetky vysielajú iba jeden stav, tak ním stroj prepíše všetky komunikačné stavy v sekcii c_i na druhej páske. Ak by ale narazil na dva rôzne vysielané stavy, tak vo výpočte M nastal konflikt a stroj A sa zasekne.

Na záver simulácie jedného kroku ešte skontroluje, či nie je splnené akceptačné kritérium. Musí overiť dve podmienky.

- Overí, či sú všetky aktuálne konfigurácie naladené na rovnaký kanál, čo sa prejaví tým, že iba v jednej sekcii c_i budú zapísané pracovné a komunikačné stavy konfigurácií, a v ostatných budú iba symboly $-$.

- Skontroluje, či všetky konfigurácie vysielajú akceptačný stav q_{accept}, čo sa dá jednoducho skontrolovať, keďže stavy sú zapísané na páske.

Ak sú obe podmienky splnené, tak stroj A akceptuje.

Počet použitých kanálov počas výpočtu M je najviac $C(n)$, počet pozícií vstupnej hlavy je $n+2$ a konfigurácií naladených na konkrétny kanál s hlavou na danej pozícií je najviac $|Q| \cdot |R|$, čo je konštanta. Veľkosť pások stroja A teda bude $O(n \cdot C(n))$.

□

Teraz už môžeme sformulovať naše tvrdenie.

Tvrdenie 3.6: Nech $C(n)$ je ľubovoľná funkcia a nech M je $WPFA_1$ pracujúci v kanálovej zložitosti $C(n)$. Potom existuje $WPTM_2$ M', simulujúci M v kanálovej zložitosti $O(log(C(n)))$ a priestorovej zložitosti $O(log(log(C(n))))$.

Dôkaz: Majme $WPFA_1$ M, pracujúci v kanálovej zložitosti $C(n)$. Podľa lemy 3.4 vieme skonštruovať DTS A s dvomi pracovnými páskami simulujúci M taký, že $L(A) \in DSPACE(nC(n))$. Ten vieme následne štandardnou simuláciou prerobiť na jednopáskový stroj DTS A' s tým, že priestorová zložitosť narastie iba konštantne, takže platí $L(A') \in DSPACE(nC(n))$. Teraz vieme priamo použiť lemu 3.3 a teda vieme skonštruovať $WPTM_2$ M' simulujúci A' (teda aj M) v kanálovej zložitosti $O(log(C(n)))$ a priestorovej zložitosti $O(log(log(C(n))))$.

□

Poznámka 3.7 Pri tejto simulácii sme sa síce zbavili predpokladu konštruovateľnosti, ale oproti priamej simulácii sme ešte viac spomalili výpočet. Nech je $T(n)$ časová zložitosť výpočtu stroja M. Simulujúci DTS A musí v najhoršom prípade, pri simulácii jedného kroku, pre každú dvojicu stavov na prvej páske posunúť celý obsah druhej pásky. Teda simulácia jedného kroku M na A má najviac kvadratickú zložitosť od veľkosti pásky A. Páska stroja A má veľkosť $O(nC(n))$, teda časová zložitosť výpočtu na A je $O(T(n)(nC(n))^2)$.

Štandardná simulácia viacpáskového DTS na jednopáskovom má kvadratickú zložitosť, čiže časová zložitosť výpočtu stroja A' bude $O((T(n))^2(nC(n))^4)$. Nakoniec simulácia jedného kroku A' na stroji M' trvá najdlhšie vtedy, keď je potrebné zmeniť hodnotu na zásobníku, čo podľa predošlej simulácie trvá najviac $O(log(C(n))log(log(C(n))))$. Vo výsledku teda dostávame opäť nie úplne tesný odhad na časovú zložitosť simulácie $O((T(n))^2(nC(n))^4 log(C(n))log(log(C(n))))$.

Takisto výrazne narástol počet krokov, ktorých sa uskutočňuje komunikácia. V priamej simulácii bol počet komunikačných krokov vo výpočte rovnaký, ako na pôvodnom simulovanom stroji. Tu procesy stroja M' musia komunikovať pri každom kroku simulovaného stroja A'. Teda vo výsledku vykoná M' až $O((T(n))^2(nC(n))^4)$ komunikačných krokov.

Záver

V práci sme skúmali variant výpočtového modelu $WPTM$, ktorý používa na komunikáciu namiesto kanálovej pásky zásobník.

Najprv sme sa venovali simulácii alternujúceho Turingovho stroja pomocou $WPTM$. Ukázali sme, že idea simulácie použitá pri $WPTM$ s kanálovou páskou sa dá použiť aj pri variante modelu, ktorý používa zásobník a podarilo sa nám spraviť simuláciu k-hlavého alternujúceho konečného automatu pomocou jednohlavého $WPFA$. Tým sme zároveň ukázali, že viac vstupných hláv vieme nahradiť väčším počtom komunikujúcich procesov.

V ďalšej kapitole sme sa pozreli na vplyv veľkosti zásobníkovej abecedy na silu modelu. Cieľom bolo zistiť, či vieme odsimulovať stroj komunikujúci s počítadlom veľkosti n pomocou stroja komunikujúceho so zásobníkom veľkosti $O(log(n))$. Túto simuláciu sa nepodarilo spraviť pre stroje bez pracovných pások. Ukázali sme ale, že stačí pamäť veľkosti $O(log(log(n)))$, aby bola táto simulácia možná a odprezentovali sme dva spôsoby, akými je možné simuláciu uskutočniť.

Ostáva ešte viacero otvorených problémov týkajúcich sa tohto modelu. Pri nepriamej simulácii v tretej kapitole výrazne narástol počet krokov, v ktorých sa uskutočňuje komunikácia oproti priamej simulácii. Bolo by teda zaujímavé pozrieť sa, ako by počet komunikačných krokov, resp. počet prenesených symbolov počas výpočtu ovplyvnili silu modelu.

Všetky doterajšie skúmania sa zaoberali modelom, ktorý mal k dispozícii jednu komunikačnú štruktúru. Mohlo by byť zaujímavé preskúmať model, ktorý by mal k dispozícii viacero komunikačných štruktúr, a teda jeho procesy by boli schopné súčasne komunikovať s niekoľkými rôznymi skupinami procesov.

Literatúra

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

[2] M. Mikula. Skúmanie výpočtovej sily WPTM pri ohraničeniach. Master's thesis, Univerzita Komenského v Bratislave, 2014.

[3] P. Mravec. Formálne výpočtové modely s bezdrôtovou komunikáciou. Master's thesis, Univerzita Komenského v Bratislave, 2007.

[4] Anna Slobodová. Alternujúce výpočtové modely s komunikáciou. Master's thesis, Univerzita Komenského v Bratislave, 1988.

[5] Anna Slobodová. *Synchronizované alternujúce výpočty.* PhD thesis, Univerzita Komenského v Bratislave, 1992.

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

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