UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Separácie a hierarchie zložitostných tried pre výpočtové modely konečných automatov

2011 Bc. Peter Koscelanský

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Separácie a hierarchie zložitostných tried pre výpočtové modely konečných automatov

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Andrej Bebják
Evidenčné číslo:	37e20925-2918-44dd-8502-e0b1082797fd

Bratislava, 2011 **Bc. Peter Koscelanský**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Peter Koscelanský
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: Separácie a hierarchie zložitostných tried pre výpočtové modely konečných automatov

Cieľ: V rámci diplomovej práce je nutné naštudovať a prezentovať techniky separácie zložitostných tried pre rôzne výpočtové modely. Aplikovať ich na konečné automaty s cieľom získať nové hierarchie a separácie zložitostných tried.

Vedúci: RNDr. Andrej Bebják

Dátum zadania: 12.11.2009

Dátum schválenia: 18.02.2011

prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci

Ďakujem môjmu vedúcemu RNDr. Andrejovi Bebjákovi za pomoc pri hľadaní zaujímavej témy, za odborné vedenie a cenné pripomienky.

Abstrakt

V práci sa zaujímame o viachlavové dvojsmerné konečnostavové automaty a o pomocou nich definované zložitostné triedy. Zaoberáme sa výpočtovou silou rôznych modelov vzhľadom na počet čítacích hláv a ohraničenie obratovej miery zložitosti. Analyzujeme a prezentujeme najznámejšie výsledky z tejto oblasti, s cieľom porovnať ich s výsledkami na rôznych zoslabených modeloch (jednosmerné automaty, automaty nezávislé od vstupu, zametajúce automaty, atď.). Popisujeme aj všeobecný model – viachlavový alternujúci stroj (MAM) používajúci nekonečne veľa stavov. Odhady zložitosti získané pre tento model sú aplikovateľné aj na automaty. Na deterministickom variante tohto modelu dokážeme dolný odhad na počet obratov. Dôsledkom nami získaného odhadu je separácia deterministických a nedeterministických zariadení. Ukážeme existenciu hierarchie na obratovú mieru zložitosti pre MAM.

Kľúčové slová: viachlavové automaty, obratová miera zložitosti, dolné odhady zložitosti

Abstract

In this thesis we focus on two-way multi-head finite automata and on the complexity classes defined by them. We are concerned with computational power of different models regarding the number of reading heads and the bounds for reversal complexity measure. We analyze and present known results about this area, in order to compare them with results about restricted models (one-way automata, data independent automata, sweeping automata, etc.). Moreover we describe a general model – multi-head alternating machine (MAM) with infinitely many states. Lower bounds for this model are also valid for finite automata. For the deterministic variant of this general model we prove the lower bound for the number of reversals. As a corollary to our result we obtain the separation of deterministic and nondeterministic machines. We show the existence of MAM hierarchy with respect to reversal complexity.

Keywords: multi-head automata, reversal complexity, lower bounds

Obsah

Úvod

Model konečnostavových automatov sa ukázal ako vhodný výpočtový model na reprezentáciu regulárnych jazykov v Chomského hierarchií. Prirodzenými rozšíreniami základného modelu je pridania viacerých čítacích hláv a schopnosti pohybovať sa po vstupnom slove v oboch smeroch. Ukazuje sa, že takto rozšírené automaty dokážu definovať zaujímavé triedy jazykov.

Vzťah determinizmu a nedeterminizmu je jeden z najznámejších konceptov v teórií výpočtovej zložitosti. Otvorený problém P vs. NP je presne takéhoto typu. Ďalším, nemenej dôležitým, otvoreným problémom je vzťah deterministickej (LOG) a nedeterministickej ($NLOG$) logaritmickej priestorovej zložitosti na Turingových strojoch. Je známe, že tieto zložitostné triedy sa dajú opísať aj pomocou dvojsmerných viachlavových automatov. Preto, v snahe odseparovať LOG od $NLOG$, sa začali skúmať dolné odhady zložitosti (počtu hláv) na konkrétnych jazykoch pre viachlavové automaty.

Práca je delená nasledovne. V prvej kapitole zadefinujeme model viachlavových dvojsmerných automatov. Pozrieme sa na vzťah týchto automatov a tried LOG a $NLOG$. Na záver uvedieme výsledky z popisnej zložitosti, ktoré dávajú do vzťahu problém LOG vs. $NLOG$ a nárast počtu stavov pri redukcií z nedeterministického na deterministický dvojsmerný jednohlavový automat.

Druhá kapitola sumarizuje poznatky o vplyve počtu hláv na množinu akceptovaných jazykov a o vzťahu tried jazykov definovaných deterministickými a nedeterministickými modelmi. Samozrejme, aj na modeli viachlavových automatov sa definovali rôzne zoslabenia základného modelu. Ako napríklad automaty, ktorých hlavy nedokážu rozoznávať všetky znaky zo vstupnej abecedy, alebo automaty pracujúce bez stavov. Ak automatom povolíme iba jednu trajektóriu hláv na slovách dĺžky n, potom definujú presne zložitostnú triedu NC^1. Tá sa považuje za triedu efektívne paralelne riešiteľných problémov. Vzťah LOG a NC^1 nie je známy, reprezentácia pomocou automatov dáva iný pohľad na daný problém. Na záver sa zaoberáme zametajúcimi automatmi, ktoré môžu meniť smer čítania vstupného slova iba na koncoch. Ukážeme, že existuje jazyk, ktorý ak sa dá akceptovať na k hlavovom zametajúcom automate, potom tento mód práce neuberá výpočtovú silu. Prezentujeme najnovšie výsledky o všetkých týchto zariadeniach. Ako uvidíme, niektoré ohraničenia nezmenšujú výpočtovú silu, niektoré sa dajú kompenzovať nárastom počtu hláv a niektoré dokážu akceptovať iba jednoduchšie jazyky.

Jednou zo zložitostných mier, definovaných pre zariadenia s obojsmerným pohybom hláv, je počet obratov. Nárast ohraničenia počtu obratov spôsobuje rozšírenie triedy rozpoznávaných jazykov. V tretej kapitole sa pozrieme na túto mieru zložitosti. Prinesieme niektoré známe výsledky o konštruovateľnosti funkcií pomocou automatov s ohraničením počtom obratov. V skúmaní obratovej zložitosti pokračujeme aj v štvrtej kapitole. Pozrieme sa na všeobecnejší model – viachlavový alternujúci stroj (MAM). MAM akceptuje všetky jazyky rozpoznateľné Turingovými strojmi. Študujeme dolné odhady obratovej zložitosti na MAM. Výsledky získané na tomto modeli sú aplikovateľné aj pre viachlavové automaty. Podľa známych techník ukážeme dolný odhad na obratovú zložitosť pre deterministické modely, čím získame separáciu determinizmu a nedeterminizmu pre ohraničený počet obratov. Rovnako ukážeme, že v rámci jednotlivých modelov existuje striktná hierarchia na počet obratov.

V práci predpokladáme, že čitateľ je oboznámený so základnými pojmami z teórie formálnych jazykov a výpočtovej zložitosti [1].

Kapitola 1

Viachlavové konečnostavové automaty

Viachlavové konečnostavové automaty sú zovšeobecneným klasického modelu konečnostavového automatu pridaním viacerých hláv, ktoré sa môžu pohybovať po páske so vstupným slovom. Po prvý krát boli zadefinované v [2][1]. Ukázalo sa, že niektoré známe otvorené problémy z výpočtovej zložitosti sa dajú transformovať do reči viachlavových automatov. Preto sa tento stále ešte jednoduchý koncept v posledných rokoch intenzívne študuje.

Viachlavové automaty majú väčšiu výpočtovú silu oproti jednohlavovým automatom. Rozhodovacie problémy, ktoré sú jednoduché pre jednohlavové automaty sú pre viachlavové automaty nerozhodnuteľné [3]. Medzi takéto problémy patrí prázdnosť jazyka, konečnosť (nekonečnosť) jazyka a viaceré iné.

V tejto kapitole zadefinujeme model viachlavových dvojsmerných konečnostavových automatov. Ďalej uvedieme súvis takýchto automatov so známym otvoreným problémom z teórie zložitosti – vzťah medzi triedami jazykov definovanými pomocou deterministických a nedeterministických Turingových strojov s logaritmickou priestorovou zložitosťou. Na záver kapitoly stručne uvedieme výsledky z popisnej zložitosti, ktoré dávajú do súvisu počet stavov dvojsmerných automatov a problém uvedený vyššie.

1.1 Definície

Viachlavové konečnostavové automaty sa definujú podobne ako klasické jednosmerné jednohlavové automaty. Automat má konečnostavové riadenie, vstupnú abecedu a konečnú prechodovú funkciu. Vstup dostane na jednej páske, ktorá je ohraničená zarážkami. Automat navyše používa niekoľko čítacích hláv. Na začiatku sú nastavené na prvé písmenko vstupu. Môžu sa voľne pohybovať po vstupe. Automat akceptuje vstup, ak sa pri jeho čítaní dostane do akceptujúceho stavu.

[1] Pôvodne sa používal model s viacerými vstupnými páskami, kde na každej bola jedna čítacia hlava.

Definícia 1.1.1: Dvojsmerným nedeterministickým k-hlavovým konečnostavovým automatom A nazveme sedmicu $A = (K, \Sigma, \triangleright, \triangleleft, \delta, q_0, F)$, kde

(1) K je konečná množina stavov.
(2) Σ je konečná množina symbolov (vstupná abeceda).
(3) $\triangleright, \triangleleft$ je ľavá a pravá zarážka (endmarker).
(4) $\delta: (K \times (\Sigma \cup \{\triangleright, \triangleleft\})^k) \to 2^{(K \times \{-1,0,1\}^k)}$ je prechodová funkcia, pre ktorú platí nasledovná vlastnosť. Nech $\big(p, (\gamma_1, \gamma_2, \dots, \gamma_k)\big) \in \delta\Big(\big(q, (a_1, a_2, \dots, a_k)\big)\Big)$, potom pre všetky $j \in \{1, \dots, k\}$ také, že $a_j = \triangleright$ musí platiť $\gamma_j \in \{0, 1\}$ a pre všetky $i \in \{1, \dots, k\}$ také, že $a_i = \triangleleft$ musí platiť $\gamma_i \in \{0, -1\}$.
(5) $q_0 \in K$ je počiatočný stav.
(6) $F \subseteq K$ je množina akceptujúcich stavov.

V prechodovej funkcii -1 znamená pohyb hlavy doľava, 1 pohyb hlavy doprava a 0 hlavu, ktorá ostáva na mieste. Ak w je vstupné slovo z Σ^*, potom automat vykonáva výpočet na páske s $\triangleright\, w \,\triangleleft$. Dodatočnými podmienkami na δ funkciu sme zabezpečili, aby sa automat počas výpočtu nikdy nedostal za zarážky $\triangleright$ a $\triangleleft$. Dvojsmerné nedeterministické k-hlavové konečnostavové automaty budeme označovať ako $2NFA(k)$.

Definícia 1.1.2: Konfiguráciou $2NFA(k)$ nazveme prvok $\big(w, q, (i_1, i_2, \dots, i_k)\big)$ z množiny $\Sigma^* \times K \times \mathbb{N}_0^k$, kde $\forall j \in \{1, \dots, k\}: 0 \leq i_j \leq |w| + 1$. w je vstupné slovo, q je stav automatu a i_j ($j \in \{1, \dots, k\}$) je pozícia j-tej čítacej hlavy na vstupe. Počiatočnou konfiguráciou označíme konfiguráciu $(w, q_0, \{1\}^k)$.

Definícia 1.1.3: Nech A je $2NFA(k)$, krokom výpočtu automatu A nazveme reláciu $\vdash_A$ na konfiguráciách ($\vdash_A \subseteq \big(w, q, (i_1, i_2, \dots, i_k)\big) \times \big(w, p, (i'_1, i'_2, \dots, i'_k)\big)$).
Nech $C_1 = \big(w, q, (i_1, i_2, \dots, i_k)\big)$, $C_2 = \big(w', q', (i'_1, i'_2, \dots, i'_k)\big)$ sú dve konfigurácie a nech δ je prechodová funkcia automatu A. Potom $C_1 \vdash_A C_2$, ak C_2 vznikne z C_1 aplikovaním prechodovej funkcie δ na C_1 t.j. platí $w = w'$ a existuje $\big(q', (\gamma_1, \gamma_2, \dots, \gamma_k)\big) \in \delta\Big(\big(q, (x_1, x_2, \dots, x_k)\big)\Big)$, kde $\forall j \in \{1, \dots, k\}$, ak $i_j = 0$ (resp. $i_j = |w| + 1$), potom $x_j = \triangleright$ (resp. $x_j = \triangleleft$), inak $x_j = w_{i_j}$. Navyše $\forall j \in \{1, \dots, k\}$ platí $i'_j = i_j + \gamma_j$.

Definícia 1.1.4: Jazyk akceptovaný $2NFA(k)$ $A = (K, \Sigma, \triangleright, \triangleleft, \delta, q_0, F)$ (označíme ako $L(A)$) je množina slov

$$L(A) = \big\{w \in \Sigma^* \big| (w, q_0, \{1\}^k) \vdash_A^* \big(w, q, (i_1, i_2, \dots, i_k)\big), q \in F\big\}.$$

Automat A akceptuje jazyk L, ak pre všetky slová $w \in L$ existuje výpočet, ktorý skončí v akceptujúcom stave. Takýchto výpočtov môže byť aj viac, lebo prechodová funkcia sa správa nedeterministicky. Podobne ako pre klasické konečnostavové automaty,

môžeme požadovať aby sa funkcia správala deterministicky, a teda pre každé slovo, ktoré patrí do jazyka, existoval práve jeden akceptujúci výpočet.

Definícia 1.1.5: Dvojsmerným deterministickým k-hlavovým konečnostavovým automatom ($2DFA(k)$) nazveme $2NFA(k)$, ktorého δ funkcia je tvaru $(K \times (\Sigma \cup \{\triangleright, \triangleleft\})^k) \rightarrow (K \times \{-1,0,1\}^k)$. Konfigurácia, krok výpočtu a akceptovaný jazyk $2DFA(k)$ sa definuje analogicky ako pre $2NFA(k)$.

Budeme používať aj ďalšie zoslabenia modelu $2NFA(k)$ (jednosmerné, zametajúce, s ohraničených počtom obratov, atď.). Tie definujeme neskôr.

Definícia 1.1.6: Nech X je ľubovoľný výpočtový model (napr. $2NFA(k)$), potom symbolom $\mathcal{L}(X)$ označíme množinu jazykov akceptovaných nejakým strojom z X. $\mathcal{L}(X) = \{L | \exists A \in X, L = L(A)\}$.

1.2 Problém LOG vs. $NLOG$ a súvis s konečnostavovými automatmi

Významným problémom v teórií výpočtovej zložitosti je vzťah deterministickej a nedeterministickej priestorovej zložitosti. O ich vzťahu hovorí Savitchová veta [4]. Savitch ukázal, že nedeterministický Turingov stroj s priestorovou zložitosťou $f(n)$ sa dá simulovať pomocou deterministického Turingovho stroja s kvadratickým nárastom priestorovej zložitosti ($f^2(n)$).

Veta 1.2.1: [4] Pre každú funkciu $f(n) \geq \log n$ platí

$$NSPACE\big(f(n)\big) \subseteq DSPACE\left(\big(f(n)\big)^2\right)$$

Čo okrem iného znamená, že polynomiálna deterministická a nedeterministická priestorová zložitosť sú ekvivalentné. Keď sa posunieme nižšie v hierarchií funkcií, konkrétne zoberieme $f(n) = \log n$, potom nám táto veta hovorí, že $NSPACE(\log n) \subseteq DSPACE(\log^2 n)$. Platí $\log n \neq O(\log^2 n)$, preto sa Savitchova veta nedá uplatniť na vzťah deterministickej a nedeterministickej logaritmickej priestorovej zložitosti.

Definícia 1.2.1:

$$LOG = DSPACE(\log n)$$
$$NLOG = NSPACE(\log n)$$

Problém vzťahu $LOG =? NLOG$[2] je jedným z najznámejších otvorených problémov v teórií výpočtovej zložitosti. Pozrieme sa teraz na súvis LOG vs. $NLOG$ s viachlavovými konečnostavovými automatmi. To bola jedna z ciest akou sa ľudia uberali pri snahe o separáciu týchto tried.

Veta 1.2.2: [5]**,** [6]

$$LOG = \bigcup_{k\geq 1} \mathcal{L}\big(2DFA(k)\big)$$
$$NLOG = \bigcup_{k\geq 1} \mathcal{L}\big(2NFA(k)\big)$$

Dôkaz: $LOG \supseteq \bigcup_{k\geq 1} 2DFA(k)$, na logaritmickom priestore sa dá ľahko kódovať pozícia hláv. Turingov stroj, ktorý simuluje k hlavový automat, používa k stopovú pracovnú pásku. Na stopách má zapísané pozície čítacích hláv na vstupe (povedzme v binárnej sústave). Stav automatu si uchováva v stave. Jeden krok potom simuluje nasledovne:

(1) Pre každú hlavu nájde na vstupe symbol na ktorý hlava ukazuje.
(2) Podľa δ funkcie automatu upraví čísla na páske zodpovedajúce pozíciám hláv a zmení svoj vnútorný stav.
(3) Vráti sa na (1) a opakuje až pokiaľ automat neakceptuje (resp. nezamietne).

Na prácu s binárnymi číslami stačí logaritmicky veľa miest na páske, preto sa simulácia zmestí do priestorového ohraničenia.

Na opačnú inklúziu sa využije fakt, že na $k \log n$ cifier nemôžeme zapísať číslo väčšie ako n^k. Presný dôkaz tejto inklúzie uvádzame ako súčasť dôkazu vety 2.2.2.

Nedeterministický prípad sa ukazuje analogicky.

1.3 Popisná zložitosť

V práci sa zaoberáme hlavne výpočtovou zložitosťou dvojsmerných viachlavových automatov. Výpočtová zložitosť sa zaoberá mierami, ktoré závisia od vstupu (napríklad počet obratov, touto mierou sa zaoberáme v tretej kapitole). Popisná zložitosť naproti tomu najčastejšie určuje koľko stavov potrebuje automat, aby mohol nejaký jazyk

[2] V literatúre sa tieto triedy označujú aj ako L a NL.

akceptovať. Takáto miera nezávisí od vstupných slov, ale iba od definície automatu. Zaujímavé je sledovať, akým spôsobom súvisí počet stavov s použitím konkrétneho výpočtového modelu. Napríklad sa vie, že pri redukcií z nedeterministického na deterministický jednohlavový automat je nárast stavov až exponenciálny.

V tejto časti stručne zhrnieme výsledky z popisnej zložitosti dvojsmerných jednohlavových automatov (pre viachlavové sú skoro všetky redukcie nerekurzívne [3], [7], čiže neexistuje rekurzívna funkcia, ktorou by sa dal nárast stavov ohraničiť). Ako sa ukazuje, popisná zložitosť takýchto automatov úzko súvisí z problémom LOG vs. $NLOG$.

Sipser [8] dokázal, že ak $LOG = NLOG$, potom existuje polynóm p taký, že pre všetky $m \in \mathbb{N}$ a pre každý $2NFA(1)$ A s počtom stavov n existuje $p(n \cdot m)$ stavový $2DFA(1)$, ktorý akceptuje všetky slová z množiny $\{w \in L(A) || w| \leq m\}$. Preto, ak sa ukáže exponenciálny nárast stavov pri redukcií z $2NFA(1)$ na $2DFA(1)$ pre nejakú postupnosť jazykov, potom $LOG \neq NLOG$. Dôkaz, či taký jazyk existuje je otvorený problém.

Pre špeciálne triedy $2DFA(1)$ sa podarilo exponenciálny nárast stavov ukázať. Sipser [8] našiel postupnosť jazykov $\{B_n\}_{n\geq 1}$, ktorá sa dá akceptovať na $1NFA(1)$ s použitím n stavov, ale nedá sa akceptovať na žiadnom zametajúcom $2DFA(1)$ (automaty, ktoré môžu meniť smer iba na koncoch slova, viac o nich píšeme v časti 2.6) s menej ako 2^n stavmi.

Výsledok pre zametajúce automaty neskôr rozšírili Hromkovič a Schnitger [9]. Ukázali, že jazyk použitý Sipserom sa nedá akceptovať ani na $2DFA(1)$ s menej ako 2^n stavmi, ak mu povolíme na slovách dĺžky n najviac $o(n)$ rôznych trajektórií hlavy. Tento výsledok je vlastne pre špeciálnu triedu automatov, ktoré môžu spraviť najviac polynomiálne veľa obratov. Otvoreným problémom ostáva, či sa dá postupnosť $\{B_n\}_{n\geq 1}$ akceptovať pomocou $2DFA(1)$ s konštantným počtom obratov pri polynomiálnom náraste stavov.

Najnovší výsledok o súvise počtu stavov a počtu obratov, ktoré môže automat urobiť priniesli Balcerzak a Niwiński [10]. Podarilo sa im ukázať existenciu postupnosti jazykov $\{L_n\}_{n\geq 1}$ takej, že pre všetky L_n existuje $2DFA(1)$ s počtom stavov $O(n)$, ktorý akceptuje L_n s použitím dvoch obratov, ale ľubovoľný $2DFA(1)$, ktorý by akceptoval L_n s použitím jedného obratu musí mať $\Omega(2^n)$ stavov.

Kapitola 2

Výpočtová sila rôznych typov automatov

V predchádzajúcej kapitole sme uviedli výsledok, ktorý hovorí o tom, že $2DFA(k)$ sú rovnako výpočtovo silné ako deterministické Turingové stroje s logaritmickou priestorovou zložitosťou (obdobne aj pre nedeterministický prípad). V tejto kapitole sa budeme venovať separáciám a zložitostným hierarchiám v rámci modelov dvojsmerných automatov. Ako sa ukáže niektoré separácie by dali odpoveď na otázku LOG vs. $NLOG$.

Uvedieme výsledky týkajúce sa rozličných zoslabení modelu $2DFA(k)$. Pôjde o jednosmerné viachlavové automaty, automaty, ktorých hlavy nedokážu rozlišovať všetky znaky, bezstavové automaty, automaty nezávislé na vstupe a zametajúce automaty (hlavy musia vždy prejsť celé slovo, otáčať sa môžu len na zarážkach).

Pri každom modeli ukážeme známe výsledky, ako dopad pridania hláv na popisnú silu. Rovnako rozoberieme vplyv zavedenia nedeterminizmu.

2.1 Jednosmerné viachlavové automaty

Stroje podobné jednosmerným automatom sa začali skúmať ešte v päťdesiatych rokoch dvadsiateho storočia. Medzi prvými prácami, ktoré používajú model podobajúci sa automatom s jednou hlavou na vstupe, sú [11] a [12]. Modernú podobu konečnostavových automatov po prvý krát zadefinovali Rabin a Scott vo svojej práci [2]. Neskôr sa začalo používať prirodzené rozšírenie na model s viacerými čítacími hlavami, ktoré sa mohli pohybovať iba jedným smerom. Algoritmy na takýchto zariadeniach sú v istom zmysle praktické, keďže výpočet trvá vždy iba lineárny čas a počas výpočtu sa použije konštantne veľa pamäte.

V tejto časti stručne popíšeme ako sa líši definícia jednosmerných automatov od definície dvojsmerných, ktorú sme uviedli v predchádzajúcej kapitole. Uvedieme známe výsledky o prejave pridanie jednej hlavy na triedu akceptovaných jazykov.

Definícia 2.1.1: Jednosmerným nedeterministickým k-hlavovým konečnostavovým automatom ($1NFA(k)$) A nazveme automat z definície 1.1.1, ktorého prechodová funkcia δ navyše spĺňa nasledujúcu vlastnosť.

Nech $(p, (\gamma_1, \gamma_2, \dots, \gamma_k)) \in \delta\left((q, (a_1, a_2, \dots, a_k))\right)$, potom platí $\forall i \in \{1, \dots, k\}$ $\gamma_i \in \{0,1\}$.

Automat môže svoje hlavy vždy posúvať iba doprava, alebo ich nechať na mieste. Konfigurácie, krok výpočtu a akceptovaný jazyk sa definujú rovnako ako v dvojsmernom prípade. Podobne definujeme deterministický prípad, ktorý budeme označovať $1DFA(k)$. Jednohlavové deterministické (resp. nedeterministické) varianty budeme označovať DFA (resp. NFA).

DFA rozpoznávajú práve regulárne jazyky v Chomského hierarchií. Ukázalo sa, že pridanie nedeterminizmu im neprinesie väčšiu rozpoznávaciu silu a NFA akceptujú tiež práve regulárne jazyky. Tento výsledok sa objavil spolu so zavedením nedeterminizmu v [2]. Technika akou sa ukáže transformácia NFA A na ekvivalentný DFA A' je známa podmnožinová konštrukcia. Kde A' má ako množinu stavov všetky podmnožiny množiny stavov automatu A. A' simuluje výpočet automatu A. Ako stav si udržuje množinu stavov, v ktorých sa v danom okamihu výpočtu mohol A nachádzať. Automat A' akceptuje ak sa A mohol na konci slova nachádzať v akceptujúcom stave. Ľahko vidieť, že takáto konštrukcia funguje. Čiže platí $\mathcal{L}(DFA) = \mathcal{L}(NFA)$.

Pridanie jednej hlavy už ale pomôže. Známy jazyk $\{a^n b^n | n \geq 1\}$ nie je regulárny (nefunguje na ňom pumpovacia lema), čiže neexistuje žiaden NFA, ktorý by tento jazyk rozpoznával. Naproti tomu nie je ťažké nahliadnuť, že existuje $1DFA(2)$, ktorý tento jazyk rozpoznáva. Z toho vyplýva nasledujúca separácia – $\mathcal{L}(NFA) \subset \mathcal{L}\big(1DFA(2)\big)$. Preto bolo zaujímavé skúmať, či naozaj viac hláv viac vie, alebo existuje nejaké n také, že $\mathcal{L}\big(1DFA(n)\big) = \mathcal{L}\big(1DFA(n+1)\big)$.

Rosenberg [13] uviedol hypotézu, že $\mathcal{L}\big(1DFA(k)\big) \subset \mathcal{L}\big(1DFA(k+1)\big)$ pre všetky $k \geq 1$. Na separáciu navrhoval použiť jazyk P_m.

Definícia 2.1.2: Symbolom P_m označíme množinu slov tvaru

$$P_m = \{1^{a_1} * 1^{a_2} * \dots * 1^{a_m} \# 1^{a_m} * 1^{a_{m-1}} * \dots * 1^{a_1} | a_1, a_2, \dots, a_m \in \mathbb{N}\}.$$

Rosenberg ukázal procedúru, ktorou sadá P_m rozpoznať na $1DFA(k)$, ak $m = \frac{1}{2}k(k-1)$.

Veta 2.1.1: [13] Ak $m \leq \frac{1}{2}k(k-1) = \binom{k}{2}$, potom pre jazyk P_m existuje automat A z $1DFA(k)$ taký, že $L(A) = P_m$.

Dôkaz: Automat A pozná číslo m, preto si ho môže pamätať v stave. To umožňuje automatu A pracovať nasledujúcim deterministickým spôsobom.

(1) Prvú hlavu automat nastaví na začiatok podslova $1^{a_{k-1}}$ (v časti slova za #)
(2) Zvyšných $k-1$ hláv môže byť použitých na porovnanie $k-1$ podslov $1^{a_1}, \ldots, 1^{a_{k-1}}$ s podslovami $1^{a_{k-1}}, \ldots, 1^{a_1}$, ktoré číta prvá hlava (tie sú v druhej časti slova).
(3) Prvá hlava sa dostala na koniec slova a teda nemôže byť už ďalej použitá vo výpočte. Ostatných $k-1$ hláv sa presunie na začiatok 1^{a_k}. Ich úlohou je overiť či zvyšok slova $1^{a_k} * \ldots * 1^{a_m} \# 1^{a_m} * \ldots * 1^{a_k}$ patrí do jazykaP_{m-k+1}. Na to automatu ostalo dosť hláv, keďže $m - k + 1 \leq \frac{1}{2}k(k-1) - (k-1) = \frac{1}{2}(k-1)(k-2) = \binom{k-1}{2}$. Čiže automat môže znova spustiť krok (1).

Ľahko vidieť, že automat A naozaj akceptuje jazyk P_m.

Rosenbergova pôvodná myšlienka bola, že P_m sa nedá rozpoznávať žiadnym $1NFA(k)$, pokiaľ $m > \binom{k}{2} = \frac{1}{2}k(k-1)$. Intuitívne sa zdá, že automat rozpoznávajúci P_m potrebuje na každú dvojicu súhlasiacich podslov jeden pár hláv h_1 a h_2. Nech h_1 je hlava v prvej časti slova (pred znakom #). Potom ako takýto pár hláv skontroluje dĺžku súhlasiacich podslov, už sa nedá použiť na iný pár podslov. Keďže h_2 môže byť použitá len na podslová v druhej časti, ku ktorým prislúchajú podslová z prvej časti pred pozíciou hlavy h_1. Všetkých párov hláv je $\binom{k}{2} = \frac{1}{2}k(k-1)$, preto sa zdá, že pre m väčšie ako táto hodnota nemôže existovať $1NFA(k)$, ktorý jazyk P_m rozpoznáva. Nanešťastie neskôr sa jeho informatívny dôkaz ukázal ako nesprávny. O zložitosti jazyka P_m na jednosmerných viachlavových automatoch hovoria nasledujúce vety [14].

Veta 2.1.2: [14] Ak sa obmedzíme na vstupy tvaru $1^{a_1} * 1^{a_2} * \ldots * 1^{a_m} \# 1^{a_{m+1}} * 1^{a_{m+2}} * \ldots * 1^{a_{2m}}$. Kde existuje konštanta c taká, že $a_i \leq c \cdot a_j$ pre všetky $i, j \in \{1, \ldots, 2m\}$. Potom jazyk P_m môže byť rozpoznávaný na $1DFA(k)$ automate, pre k spĺňajúce nerovnosť $m \geq \frac{k^3}{24}$.

Dôkaz: Princíp dôkazu je v tom, že automat si pozíciou hlavy môže pamätať informáciu o dĺžke podslova jazyka P_m v inom podslove. Keďže dĺžka podslov je jediné čo nás pri porovnávaní zaujíma, nemusí platiť, že každý pár hláv môže porovnať len jednu súhlasiacu dvojicu podslov. Spôsob ako sa môžu takto kódovať dĺžky podslov do pozície hlavy (opísaný v [14]) je nasledujúci:

Nech hlava h_3 ukazuje na začiatok bloku 1^{a_j}, ktorého dĺžku chceme umiestniť do bloku 1^{a_i}. Ďalej nech hlavy h_1 a h_2 sú na začiatku 1^{a_i}. Navyše predpokladajme $a_i > a_j$ Naraz sa začnú hýbať hlavy h_3 a h_1. Keď hlava h_3 prejde na koniec 1^{a_j}, hlava h_1 je v bloku 1^{a_i} na pozícií a_j. Teraz sa začnú naraz hýbať hlavy h_1 a h_2, keď h_1 prejde na koniec podslova 1^{a_i}, potom hlava h_2 má do konca podslova 1^{a_i} práva a_j krokov. Čiže môžeme porovnávať 1^{a_j} aj bez toho, aby bola v tejto časti čítacia hlava.

Autor v [14] navyše ukázal aj horný odhad na m aby bol jazyk P_m rozpoznávaný $1DFA(k)$.

Veta 2.1.3: [14] Ak je jazyk P_m rozpoznávaný pomocou $1DFA(k)$, potom platí nerovnosť $m \leq \frac{1}{2}k^3$.

Ďalší výsledok v snahe dosiahnuť separáciu tried podľa počtu hláv sa podarilo dosiahnuť v [15]. Autori dokázali, že $\mathcal{L}\big(1DFA(2)\big) \subset \mathcal{L}\big(1DFA(3)\big)$ a $\mathcal{L}\big(1NFA(2)\big) \subset \mathcal{L}\big(1NFA(3)\big)$. O úplnú separáciu sa postarali až Yao a Rivest [16].

Veta 2.1.4: [16] Pre všetky $k \geq 1$ platí

$$\Big(\mathcal{L}\big(1DFA(k+1)\big) - \mathcal{L}\big(1NFA(k)\big)\Big) \neq \emptyset.$$

Dôkaz: Na separáciu sa použil jazyk

$$L_m = \{w_1 * w_2 * \ldots * w_{2m} | \forall i \in \{1, \ldots, 2m\}: w_i \in \{0,1\}, w_i = w_{2m-i+1}\}.$$

Dôkaz je založený na tvrdení, že jazyk L_m je rozpoznávaný k-hlavovým automatom A práve vtedy, keď $m \leq \binom{k}{2}$. Teda jazyk $L_{\binom{k}{2}}$ sa dá akceptovať pomocou $1DFA(k)$, ale nie je rozpoznávaný žiadnym automatom z triedy $\bigcup_{1 \leq i < k} 1NFA(i)$. Fakt, že $L_{\binom{k}{2}}$ je rozpoznávaný automatom s k hlavami, sa dokazuje úplne rovnako ako vo vete 2.1.1.

Dolný odhad na počet hláv sa dá ukázať sporom – existenciou slova, ktoré automat akceptuje, aj keď by nemal. Pre spor predpokladajme, že $L_{\binom{k}{2}}$ je akceptovaný automatom A z triedy $1NFA(k-1)$. Automat má menej ako k hláv, teda počet súhlasiacich dvojíc podslov, ktoré musí automat skontrolovať je väčší ako počet dvojíc hláv. Čiže pri každom výpočte je aspoň jedna dvojica korešpondujúcich podslov neskontrolovaná. Všetkých slov z jazyka L_m, kde jednotlivé w_i sú dĺžky n je $2^{m \cdot n}$ (túto množinu slov označíme ako L_m^n). Počet rôznych konfigurácií automatu A na slovách z L_m^n je najviac $q\big(2m(n+1)\big)^k$, kde q je počet stavov automatu A. Budeme si všímať pozície vo výpočte A, keď niektorá hlava začne čítať prvé písmenko podslova w_i. Takýchto pozícií vo výpočte je najviac $2k(m-1)+1$ (pripočítanie jednotky je vo výraze preto, lebo na začiatku sú všetky hlavy na prvom písmenku w_1). Počet výpočtov, ktoré sa líšia v konfiguráciách na týchto pozíciách je najviac $\big(q\big(2m(n+1)\big)^k\big)^{2k(m-1)+1} = P(n)$.

Určite existuje aspoň $2^{m \cdot n}/P(n)$ slov, na ktorých automat A neporovná korešpondujúce podslová w_a a w_b a výpočty na týchto slovách navštívia prvé písmenka slov v rovnakom poradí, s rovnakými konfiguráciami. Spomedzi týchto slov je aspoň $2^n/P(n)$ takých, ktoré sa líšia iba v podslovách w_a a w_b. Pre dostatočne veľké n je $2^n/P(n) \geq 2$. Tieto dve slová sa líšia iba v podslovách w_a a w_b. Konfigurácie v ktorých začína A čítať podslová sú rovnaké. Preto vieme tieto dve slová skombinovať do jedného, ktoré nepatrí do L_m^n, ale výpočet na ňom je akceptujúci.

Priamym dôsledok tento vety je hierarchia deterministických a nedeterministických jednosmerných automatov podľa počtu hláv.

Dôsledok 2.1.1: Pre $k \geq 1$ platí

$$\mathcal{L}\big(1DFA(k)\big) \subset \mathcal{L}\big(1DFA(k+1)\big)$$

$$\mathcal{L}\big(1NFA(k)\big) \subset \mathcal{L}\big(1NFA(k+1)\big)$$

Autori v [16] tiež rozriešili ako je to so vzťahom determinizmu a nedeterminizmu pre k-hlavové automaty.

Veta 2.1.5: [16] Pre všetky $k \geq 2$ platí

$$\mathcal{L}\big(1DFA(k)\big) \subset \mathcal{L}\big(1NFA(k)\big).$$

Dôkaz: Myšlienka dôkazu je, že jazyk

$$\{w_1 * \ldots * w_{2m} | m \geq 1, \forall i: w_i \in \{0,1\}, \exists j: w_j \neq w_{2m-j+1}\}$$

(v istom zmysle komplement jazyka L_m z vety 2.1.4) sa dá ľahko akceptovať nedeterministickým trojhlavovým automatom. Na zistenie správnej štruktúry slova stačí jedna hlava, overovanie bude prebiehať simultánne so zvyškom výpočtu. Pomocou troch hláv vie nastaviť dve z nich na začiatok korešpondujúcich podslov w_j, w_{2m-j+1}. Index j nedeterministicky uhádne. Následne overí, či sú tieto podslová rovnaké, ak nie tak akceptuje. Deterministické stroje sú uzavreté na komplement, preto ak by pre tento jazyk existoval $1DFA(k)$ bolo by to v spore s vetou 2.1.4. Týmto sa ukázala platnosť vety pre $k \geq 3$. Použitím komplikovanejšieho jazyka sa dá dôkaz rozšíriť aj pre $k = 2$.

Jazyk použitý vo vete 2.1.4 je celkom komplikovaný. Nemôžeme dúfať v unárny[3] jazyk na dokázanie hierarchie, lebo trieda unárnych jazykov rozpoznávaných pomocou $1NFA(k)$ je práve trieda unárnych regulárnych jazykov (hlavy fungujú nezávisle na vstupe, čiže môžu overovať iba regulárne vlastnosti).

Chrobak v [17] ukázal, že na separáciu sa dá použiť aj jednoduchší jazyk. Konkrétne jazyk $L_n = \{1^m \# 1^{im} | i, m \in \mathbb{N}, i \leq n\}$. Pre všetky $k \geq 1$ existuje jazyk $L_{f(k)+1}$, ktorý sa dá akceptovať pomocou $1DFA(k+1)$, ale nie pomocou $1DFA(k)$.

[3] Nad jednopísmenkovou abecedou.

2.2 Dvojsmerné viachlavové automaty

Dvojsmerné viachlavové konečnostavové automaty sme zadefinovali v prvej kapitole (definícia 1.1.1). Teraz sa pozrieme na známe výsledky, ako sa zmení popisná sila automatov, keď im pridáme viac hláv, ako sa líšia od jednosmerných automatov a či pomôže pridanie nedeterminizmu.

Na porovnanie $1NFA(k)$ a $2DFA(k)$ sa dá použiť jazyk palindrómov $P = \{ww^R | w \in \{0,1\}^*\}$. Takýto jazyk sa dá akceptovať s použitím $2DFA(2)$, ktorý jednu hlavu nastaví na koniec a potom môže jedným prechodom cez slovo, zistiť či má požadovaný tvar. Naproti tomu je dobre známe, že žiadny $1NFA(k)$ nemôže jazyk P akceptovať. Zjednodušene je dôvod taký, že automat si nemôže pamätať celé slovo v stave, preto sa porovnávanie znakov musí udiať pomocou hláv. Navyše, ak dvojica hláv porovná prislúchajúce znaky, už sa nedá použiť na ďalšie porovnanie. Preto keď si zoberieme dostatočne dlhé slová, $1NFA(k)$ nebude mať dosť prostriedkov, aby mohol jazyk P akceptovať.

Dôsledok 2.2.1: Pre $k \geq 2$ platí

$$\mathcal{L}\big(1DFA(k)\big) \subset \mathcal{L}\big(2DFA(k)\big)$$

$$\mathcal{L}\big(1NFA(k)\big) \subset \mathcal{L}\big(2NFA(k)\big).$$

Dôsledok platí pre $k \geq 2$, pre $k = 1$ sa dá ukázať, že triedy jednosmerných a dvojsmerných automatov sú rovnaké.

Veta 2.2.1: [2]

$$\mathcal{L}\big(1DFA(1)\big) = \mathcal{L}\big(2NFA(1)\big).$$

Dôkaz: Budeme uvažovať len nedeterministické dvojsmerné automaty, ktoré pred akceptovaním nastavia hlavu na $\triangleleft$ a v každom kroku sa pohnú hlavou. Ku každému $2NFA(1)$ existuje ekvivalentný automat A s týmito vlastnosťami. Počas najkratšieho výpočtu môže A s hlavou cez každý znak prejsť najviac $|K_A|$ krát. Inak sa mu na konkrétnej pozícii v slove zopakuje stav a časť výpočtu medzi rovnakými stavmi môžeme vynechať (skrátiť výpočet). Princíp konštrukcie jednosmerného automatu A' je v tom, že nedeterministicky si v stave uhádne prechodové postupnosti (v akých stavoch automat A prešiel cez znak počas akceptačného výpočtu a smer akým sa pohol) dvojsmerného automatu A. V jednom kroku môže A' skontrolovať, či nadväzujú (sú v súlade s δ_A) prechodové postupnosti z práve čítaného znaku a predchádzajúceho znaku. Automat A' sa posunie na ďalší znak a takto odsimuluje celý výpočet A. Nakoniec skontroluje, či automat A môže byť na konci vstupného slova v akceptujúcom stave. Ak áno A' akceptuje, inak zamietne. Ľahko vidieť, že automat A' akceptuje vstupné slovo práve vtedy, keď slovo akceptoval A. K $1NFA(1)$ A' existuje $1DFA(1)$ akceptujúci rovnaký jazyk ako A', čím je dokázané tvrdenie vety.

Sudborough [18] ukázal jeden z prvých výsledkov, ako sa prejaví počet hláv na popisnej sile viachlavových automatov.

Definícia 2.2.1: Zložitostnou triedou $DSPACE(f(n),\gamma)$ (resp. $NSPACE(f(n),\gamma)$) označíme triedu jazykov, ktoré sa dajú akceptovať deterministickým (resp. nedeterministickým) Turingovým strojom M s jednou pracovnou páskou. Navyše M použije počas výpočtu na slovách dĺžky n menej alebo rovno ako $f(n)$ políčok na pracovnej pásky a veľkosť pracovnej abecedy stroja M je γ.

Trieda LOG ($NLOG$) je zjednotením tried $DSPACE(\log_2 n,\gamma)$ (resp. $NSPACE(\log_2 n,\gamma)$) cez všetky γ.

Veta 2.2.2: [18] Pre $k \geq 1$ existuje jazyk $L \subseteq \{1\}^*$ taký, že platí

$$L \in \mathcal{L}\big(2DFA(k+4)\big) - \mathcal{L}\big(2DFA(k)\big)$$

$$L \in \mathcal{L}\big(2NFA(k+4)\big) - \mathcal{L}\big(2NFA(k)\big).$$

Dôkaz: Zoberieme ľubovoľný deterministický (pre nedeterministický prípad sa postupuje rovnako) Turingov stroj M_1 z triedy $DSPACE(\log_2 n, 2)$. Tento potom vieme simulovať pomocou $2DFA(4)$ A. Jedna hlava číta vstup rovnako ako čítacia hlava TS. Druhá hlava udržiava svojou pozíciou informáciu o časti pracovnej pásky od začiatku pásky do pozície čítacej hlavy na pracovnej páske. Ak je čítacia hlava M_1 na i-tom políčku pracovnej pásky a obsah pracovnej pásky je $a_0 a_1 \dots a_{i-1} a_i a_{i+1} \dots a_m$ ($m \leq \log_2 n$, $\forall j \in \{0 \dots m\}: a_j \in \{0,1\}$), potom druhá hlava automatu A je na pozícii $2^{i-1}a_0 + 2^{i-2}a_1 + \dots + 2^0 a_{i-1}$. Treťou hlavou udržiava informáciu o obsahu od pozície čítacej hlavy po koniec pracovnej pásky stroja M_1, čiže číslo $a_i + 2^1 a_{i+1} + \dots + 2^{m-i}a_m$. Keďže $2^{\log_2 n} = n$, zmestia sa nám tieto čísla do pozície hlavy. Pomocou štvrtej hlavy sa dajú vykonávať operácie vynásobenie dvoma, vydelenie dvoma a zvyšok po delení dvoma pozícií ostatných hláv. Označme si pozície druhej a tretej hlavy A ako α a β. Ak chceme zistiť, aký je práve čítaný znak M_1 stačí zistiť, akej hodnote sa rovná $\beta \bmod 2$. Posun po páske vľavo sa dosiahne nasledovne $\beta := 2\beta + \alpha \bmod 2$ a $\alpha := \alpha/2$ (kde / je celočíselné delenie). Posun vpravo je analogický. Zápis symbolu 1 (resp. 0) na pásku sa uskutočňuje pomocou operácie $\beta := \beta - \beta \bmod 2 + 1$ (resp. $\beta := \beta - \beta \bmod 2$). Tieto operácie stačia na simuláciu M_1.

Na simuláciu stroja M_2 z triedy $DSPACE(\log_2 n, 2^k)$ technikou popísanou vyššie potrebujeme $2DFA(k+3)$. K M_2 existuje stroj M_2' z triedy $DSPACE(k\log_2 n, 2)$. Jednotlivé páskové symboly môžeme zapísať pomocou binárneho kódovania a teda každý symbol bude zapísaný na k miestach. K M_2' existuje $2DFA(k+3)$ automat A. Pracovnú pásku M_2' si môžeme rozdeliť na k úsekov dĺžky $\log_2 n$. $k-1$ hláv automatu A slúži na uchovávanie obsahu úsekov, ktoré práve nečíta čítacia hlava (pozícia hlavy A kóduje obsah úseku). Zvyšné 4 hlavy sa používajú rovnako ako pri simulácií stroja M_1. Preto platí $DSPACE(\log_2 n, 2^{k+1}) \subseteq 2DFA(k+4)$.

$2DFA(k)$ automat sa dá simulovať pomocou TS z triedy $DSPACE(\log_2 n, 2^k)$ (veta 1.2.2), čiže $\mathcal{L}\big(2DFA(k)\big) \subseteq DSPACE(\log_2 n, 2^k)$.

Platí $DSPACE(\log_2 n, 2^k) \subset DSPACE(\log_2 n, 2^{k+1})$, dokonca aj pre unárne jazyky [19]. Preto platí

$$\mathcal{L}\big(2DFA(k)\big) \subseteq DSPACE(\log_2 n, 2^k) \subset DSPACE(\log_2 n, 2^{k+1}) \subseteq \mathcal{L}\big(2DFA(k+4)\big)$$

Vylepšenie tohto výsledku priniesol Monien [20]. Podarilo sa mu ukázať, že pre $\forall k \in \{2, \dots\}$ existuje jazyk L_k nad dvojpísmenkovou abecedou taký, že $L_k \in \mathcal{L}\big(2DFA(k)\big)$, ale $L_k \notin \mathcal{L}\big(2DFA(k-1)\big)$. Posledné zlepšenie hierarchie publikoval znovu Monien [21].

Veta 2.2.3: [21] Pre $k \geq 1$ platí

$$\mathcal{L}\big(2DFA(k)\big) \subset \mathcal{L}\big(2DFA(k+1)\big)$$

$$\mathcal{L}\big(2NFA(k)\big) \subset \mathcal{L}\big(2NFA(k+1)\big).$$

Navyše svedčiaci jazyk je unárny.

Dôkaz: Nech X je trieda jazykov, potom symbolom $\tilde{X}$ značíme triedu jazykov $L \in X$, ktoré navyše spĺňajú inklúziu $L \subseteq \{0^{2^n} | n \in \mathbb{N}\}$. Ďalej zadefinujeme zobrazenie na unárnych jazykoch $T_k: \{0^{2^n} | n \in \mathbb{N}\} \to \{0^{2^n} | n \in \mathbb{N}\}$ také, že $T_k(0^{2^n}) = 0^{2^{k \cdot n}}$. Dôkaz, že $k+1$ hláv vie viac ako k pre dvojsmerné automaty sa uskutoční pomocou nasledujúcich tvrdení (popíšeme len deterministický prípad).

(1) $\mathcal{L}\big(2\widetilde{DFA}(k)\big) \subset \widetilde{DSPACE}(\log n)$, pre všetky $k \in \mathbb{N}$.

(2) Pre všetky $L \in \widetilde{DSPACE}(\log n)$ existuje $k \in \mathbb{N}$ také, že $T_k(L) \in \mathcal{L}\big(2\widetilde{DFA}(3)\big)$.

(3) Pre všetky $L \in \widetilde{DSPACE}(\log n)$ a pre $l, k \geq 1$ platí $T_k(L) \in \mathcal{L}\big(2\widetilde{DFA}(l)\big) \Rightarrow L \in \mathcal{L}\big(2\widetilde{DFA}(l \cdot k)\big)$.

(4) Pre všetky $L \in \widetilde{DSPACE}(\log n)$ a pre $k > l \geq 2$ platí $T_{k+1}(L) \in \mathcal{L}\big(2\widetilde{DFA}(l)\big) \Rightarrow T_k(L) \in \mathcal{L}\big(2\widetilde{DFA}(l+1)\big)$.

Ukážeme, že $\mathcal{L}\big(2DFA(k)\big) \subset \mathcal{L}\big(2DFA(k+1)\big)$ pre všetky $k \geq 1$. Pre $k = 1$ použijeme jazyk $L = \{0^{2^n} | n \in \mathbb{N}\}$. Jazyk L nie je regulárny, preto sa nedá akceptovať pomocou $2DFA(1)$. Pomocou $2DFA(2)$ sa už akceptovať dá. Dve hlavy stačia na kontrolovanie dĺžok 2^n pre všetky n. Predpokladajme, že prvú hlavu máme na pozícii 2^n a druhá hlava číta $\triangleright$. Prvú hlavu pošleme vľavo a druhú hlavu vpravo tak, že vždy keď prvá hlava prejde jedno políčko druhá prejde dve políčka. Ak prvá hlava dosiahne $\triangleright$, potom druhá hlava ukazuje na pozíciu 2^{n+1}.

Pre spor predpokladajme, že existuje $l \geq 2$ také, že $\mathcal{L}\big(2DFA(l)\big) = \mathcal{L}\big(2DFA(l+1)\big)$. Ak si zoberieme ľubovoľný jazyk $L \in DSPACE(\log n)$, potom podľa (2) platí

$$\exists k \in \mathbb{N}: T_k(L) \in \mathcal{L}\big(2\widetilde{DFA}(3)\big)$$

to znamená, že podľa predpokladu

$$T_k(L) \in \mathcal{L}\big(2\widetilde{DFA(l+1)}\big) = \mathcal{L}\big(2\widetilde{DFA}(l)\big)$$

aplikovaním tvrdenia (4) môžeme znížiť index zobrazenia T

$$T_{k-1}(L) \in \mathcal{L}\big(2\widetilde{DFA(l+1)}\big) = \mathcal{L}\big(2\widetilde{DFA}(l)\big)$$

opakovane použijeme (4), až sa dostaneme na zobrazenie T_{l+1}

$$T_{l+1}(L) \in \mathcal{L}\big(2\widetilde{DFA(l+1)}\big) = \mathcal{L}\big(2\widetilde{DFA}(l)\big)$$

podľa (3)

$$L \in \mathcal{L}\Big(2DFA\widetilde{\big(l \cdot (l+1)\big)}\Big).$$

Preto $DSPA\widetilde{CE}(\log n) \subseteq \mathcal{L}\Big(2DFA\widetilde{\big(l \cdot (l+1)\big)}\Big)$, čo je v priamom rozpore s (1). Čiže ak platia tvrdenia (1) až (4). Platí aj tvrdenie vety.

(1) Definujeme zobrazenie $bi(x): \{0\}^* \to \{0,1\}^*$. $bi(0^n) = \phi$, kde 1ϕ je binárny zápis čísla n. Inverzné zobrazenie k bi označíme ako $un(x): \{0,1\}^* \to \{0\}^*$.

Predpokladajme, že existuje $k \in \mathbb{N}$ také, že $\mathcal{L}\big(2\widetilde{DFA}(k)\big) = DSPA\widetilde{CE}(\log n)$. V nasledujúcom budú všetky triedy jazykov ohraničené na abecedu $\{0\}$.

Ak $L \in SPACE(n)$, potom $un(L) \in SPA\widetilde{CE}(\log n)$ (keďže v L sú len slová tvaru 0^x, slovo $un(0^x) = 0^{2^x}$, preto sa na logaritmickom priestore dá výpočet odsimulovať). Podľa predpokladu, teda $un(L) \in \mathcal{L}\big(2\widetilde{DFA}(k)\big)$. Štandardnú simuláciu $2DFA(k)$ na TS (veta 1.2.2) vieme vykonať s priestorovou zložitosťou $\log n$ a 2^k pracovnými symbolmi. Čiže máme, že $un(L)$ sa dá akceptovať pomocou stroja M z $SPA\widetilde{CE}(\log n, 2^k)$.

Ak zoberieme jazyk $bi\big(un(L)\big)$, ten vieme akceptovať pomocou stroja z triedy $SPACE(n, 2^{k+2})$. Pracovnú pásku si rozdelíme na dve stopy. Na prvej stope simulujeme obsah pracovnej pásky M (z definície bi sa nám celý zmestí). Na druhej stope máme zakódovanú v binárnej sústave pozíciu hlavy M na vstupnom slove. Ešte si potrebujeme zaznačiť pozíciu hlavy na pracovnej páske, to sa nám zmestí do štvornásobného nárastu počtu symbolov pracovnej abecedy. Keďže $L = bi\big(un(L)\big)$, dostaneme, že $L \in SPACE(n, 2^{k+2})$. Čiže $SPACE(n) = SPACE(n, 2^{k+2})$. Čo je v spore s výsledkom z [22].

(2) Nech $L \in DSPA\widetilde{CE}(\log n)$, potom existuje TS M, ktorý tento jazyk akceptuje s použitím $\log n$ miesta na pracovnej páske. K M existuje ekvivalentný TS M', ktorý používa štvorstopovú pásku. Na prvú stopu zapíše $bi(w)$, kde w je vstupné slovo. Na druhej stope si uchováva pozíciu čítacej hlavy na vstupe v binárnej sústave. Tretiu stopu používa rovnako ako M pracovnú pásku. Na poslednej stope si uchováva pozíciu čítacej hlavy M na pracovnej páske. Ľahko vidieť, že takto definovaný M' môže simulovať výpočet M. Navyše M' má vlastnosť, že vstup číta iba na začiatku výpočtu. Hneď ako zapíše reprezentáciu vstupu na pracovnú pásku, už nikdy vstup nepoužíva. K M' existuje ekvivalentný stroj M'', ktorý používa len dva pracovné symboly a $k \cdot \log_2 n$ miesta na pracovnej páske. Jazyk $T_k(L)$ môžeme akceptovať pomocou $2DFA(3)$ podobne ako v dôkaze vety 2.2.2 (štvrtú hlavu nám netreba, lebo M'' používa vstup len na začiatku výpočtu).

(3) Zaujímavý je až prípad, keď $k \geq 2$. Nech teda A je z triedy $2DFA(l)$ akceptujúci jazyk $T_k(L)$. Potrebujeme nájsť automat A' z triedy $2DFA(k \cdot l)$ akceptujúci jazyk L. Pomocou dvoch hláv A' skontroluje, či má vstup tvar O^{2^n}. A' môže jednu hlavu A simulovať pomocou k hláv. Nech α je pozícia hlavy automatu A, potom $0 \leq \alpha < 2^{k \cdot n}$. Ďalej nech A' bude simulovať pohyb tejto hlavy pomocou hláv $1, \ldots, k$. Pozície hláv $1, \ldots, k$ kódujú číslo v 2^n-adickej sústave.

$$\alpha = \alpha_1 + \alpha_2 2^n + \cdots + \alpha_k 2^{n(k-1)}, \text{ kde } \alpha_i \text{ je pozícia } i\text{-tej hlavy } (0 \leq \alpha_i < 2^n)$$

Posun hlavy A sa deje pripočítaním alebo odpočítaním jednotky. Ak niektorá α_i dosiahne 2^n (resp. 0) potom sa táto nastaví na 0 (resp. 2^{n-1}) a hlava $i+1$ (resp. $i-1$) sa posunie o 1 vpravo (resp. vľavo). Test, či je hlava A na začiatku (resp. konci) slova) sa dá vykonať jednoducho.

Číslo α sa teda celé podarí uchovať pomocou k hláv. Automat A' preto môže pomocou $k \cdot l$ hláv simulovať všetkých l hláv A.

(4) Máme $2DFA(l)$ A pre $T_{k+1}(L)$. Chceme zostrojiť $2DFA(l+1)$ A' pre $T_k(L)$. A' má akceptovať vstup $0^{2^{k \cdot n}}$ práve vtedy, ak by A akceptoval $0^{2^{(k+1) \cdot n}}$. A' bude pomocou svojím hláv simulovať hlavy automatu A nasledovne.

Nech α_i je pozícia i-tej hlavy A, $0 \leq \alpha_i \leq 2^{(k+1) \cdot n} + 1$. A' bude túto hlavu simulovať pozíciou svojej i-tej hlavy β_i $(0 \leq \beta_i \leq 2^{k \cdot n} + 1)$ a číslom σ_i $(0 \leq \sigma_i < 2^n)$ tak, aby platilo $\alpha_i = \beta_i + \sigma_i 2^{k \cdot n}$. Čísla $\sigma_1, \ldots, \sigma_l$ bude A ukladať pomocou $(l+1)$-vej hlavy.

$$\beta_{l+1} = \sigma_1 + \sigma_2 2^n + \cdots + \sigma_l 2^{(l-1) \cdot n} + 2^{(k-1) \cdot n}$$

Nakoľko $l < k$ takáto pozícia sa zmestí na vstupné slovo. Ak A pohne i-tou hlavou vpravo $\alpha_i := \alpha_i + 1$. Potom túto skutočnosť A' odsimuluje. Ak $\beta_i < 2^{k \cdot n} + 1$, stačí pohnúť i-tou hlavou $\beta_i := \beta_i + 1$ a σ_i ostane nezmenená.

Problém nastane, ak $\beta_i = 2^{k \cdot n} + 1$, vtedy by sa malo udiať: $\beta_i := 0$, $\sigma_i := \sigma_i + 1$. Na zmenu σ_i by sme potrebovali 3 hlavy, máme k dispozícií iba 2 ($(l+1)$-vú a i-tu – keďže tá práve nič nereprezentuje). Použijeme preto, bez ujmy na všeobecnosti, ešte prvú hlavu (predpokladáme, že $\beta_1 < 2^{k \cdot n}$, neplatnosť vieme ľahko otestovať, nastaviť $\beta_1 := 2^{k \cdot n} - 1$ a rozdiel si uložiť v stave). Aby sme sa dostali k σ_i v β_{l+1} použije takzvanú rotačnú techniku.

1. Začíname z pozície
$$\beta_1 = \phi_1 + \phi_2 2^n,\ \phi_1 < 2^n,\ \phi_2 < 2^{(k-1)\cdot n}$$
$$\beta_{l+1} = \sum_{j=1}^{k-1} \sigma_j 2^{(j-1)\cdot n} + 2^{(k-1)\cdot n},\ \forall j > l : \sigma_j = 0$$
2. Upravíme pozície hláv
$$\beta_1 = \phi_2 + 2^{(k-1)\cdot n}$$
$$\beta_{l+1} = R_n(\phi_1) + \sigma_1 2^n + \cdots + \sigma_{k-1} 2^{(k-1)\cdot n}$$
kde $R_n(x) < 2^n$ je číslo, ktorého binárny zápis (doplnení nulami na dĺžku n) je reverzom binárneho zápisu čísla x (tiež doplneného nulami).
3. V ďalšom kroku
$$\beta_1 = \sigma_{k-1} + \phi_2 2^n$$
$$\beta_{l+1} = R_n(\phi_1) + \sigma_1 2^n + \cdots + \sigma_{k-2} 2^{(k-2)\cdot n} + 2^{(k-1)\cdot n}$$

Čiže po 3. kroku sme v rovnakej situácií ako na začiatku (1.). Preto viacnásobným aplikovaním krokov 2. a 3. vieme previť požadovanú σ_i na β_1, kde pri prenose na β_{l+1} nej ľahko môžeme pripočítať 1.

$$\beta_1 = \sigma_i + \phi 2^n$$
$$\beta_{l+1} = R_n(\sigma_{i+1}) + \cdots + R_n(\sigma_{k-1}) 2^{(k-i-2)\cdot n} + R_n(\phi_1) 2^{(k-i-1)\cdot n} + \sigma_1 2^{(k-i+1)\cdot n} + \sigma_{i-1} 2^{(k-2)\cdot n} + 2^{(k-1)\cdot n}$$

Ďalším aplikovaním tejto techniky sa dostaneme k pôvodným hodnotám β_1 a β_{l+1} s tým, že namiesto σ_i bude uložené $\sigma_i + 1$. Všetky tieto operácie sa dajú vykonať pomocou troch spomínaných hláv, kompletný popis operácií je v [21].

Rovnako sa dá postupovať aj pri pohybe hlavy automatu A vľavo. Testovanie, či je automat na začiatku alebo konci slova vieme tiež urobiť (v stave si môžeme pamätať, ktoré $\sigma_i = 2^n - 1$ alebo 0). Preto A' dokáže odsimulovať automat A čím je tvrdenie (4) dokázané.

Výpočtová sila determinizmu a nedeterminizmu pre dvojsmerné automaty je stále otvorený problém. Súvisí to s vetou 1.2.2, ktorá hovorí o vzťahu tohto problému s problémom LOG vs. $NLOG$. Sudborough [23] dokonca ukázal nasledujúcu vetu.

Veta 2.2.4: [23] Nasledujúce dve tvrdenia sú ekvivalentné

(1) $LOG = NLOG$.
(2) Existuje $k \geq 1$ také, že $\mathcal{L}\big(1NFA(2)\big) \subseteq \mathcal{L}\big(2DFA(k)\big)$.

Podľa tejto vety na dokázanie platnosti inklúzie $\mathcal{L}\big(2NFA(x)\big) \subseteq \mathcal{L}\big(2DFA(y)\big)$, pre nejaké $x \leq y$, nám stačí ukázať, že každý jazyk akceptovaný pomocou $1NFA(2)$ vieme akceptovať pomocou $2DFA(k)$.

2.3 Automaty so slepými hlavami

Doteraz sme sa zaoberali automatmi, ktorých všetky hlavy vedia rozoznávať znaky. Môžeme model viachlavových automatov definovať aj tak, aby len jedna čítacia hlava vedela čítať všetky symboly. Ostatné hlavy potom vedia rozlišovať len zarážky $\rhd, \lhd$, zvyšné znaky sa im javia ako rovnaké. Takéto automaty nazveme čiastočne slepé[4] automaty.

Napriek tomu, že ide o veľmi jednoduché stroje, vedia rozoznávať aj komplikované jazyky. Napríklad bezkontextový jazyk $\{a^n b^n | n \geq 1\}$ sa dá akceptovať pomocou jednosmerného čiastočne slepého automatu s dvoma hlavami. Prvá hlava prečíta všetky a, dostane sa na prvé b. Potom prvá hlava prečíta všetky b, za každé prečítané b sa posunie druhá hlava o dve pozície. Ak slovo patrí do jazyka, obe hlava dočítajú slovo súčasne. Podobne sa dá akceptovať pomocou troch hláv aj jazyk $\{a^n b^n c^n | n \geq 1\}$.

Definícia 2.3.1: Dvojsmerným čiastočne slepým k-hlavovým automatom $2BDFA(k)$ nazveme $2DFA(k)$ A, ktorého prechodová funkcia δ_A je tvaru $(K \times (\Sigma \cup \{\rhd, \lhd\}) \times \{\rhd, \lhd, \$\}^{k-1}) \to (K \times \{-1,0,1\}^k)$. $\$$ je špeciálny symbol, ktorý nepatrí do vstupnej abecedy Σ_A a $k-1$ slepých hláv ho číta namiesto ľubovoľného znaku z Σ_A.

Podobne sa definujú aj nedeterministické $2BNFA(k)$, pripadne jednosmerné $1BDFA(k)$ a $1BNFA(k)$. V prípade jednosmerných má zmysel pýtať sa, či potrebujú čiastočne slepé automaty rozoznávať aj koncovú zarážku. Preto sa uvažuje aj model, kde nie je $\lhd$ na konci slova. Vtedy automat akceptuje, ak je v akceptujúcom stave a ak všetky hlavy vypadli zo slova (sú na pozícií $n+1$). Ak sa niektorá hlava pohne za pozíciu $n+1$, potom automat neakceptuje. Ibarra a Ravikumar [24] ukázali, že pre nedeterministický prípad je jedno, či používame model so zarážkou alebo nie.

[4] angl. partially blind

Veta 2.3.1: [24] Ku každému $1BNFA(k)$ existuje ekvivalentný $1BNFA(k)$, ktorý pracuje bez pravej zarážky ($\lhd$).

Dôkaz: Zoberme ľubovoľný $1BNFA(k)$ A. K nemu exituje ekvivalentný $1BNFA(k)$ A', ktorý pred akceptovaním všetky hlavy nastaví na koniec slova ($\lhd$). K A' teraz skonštruujeme $1BNFA(k)$ B pracujúci bez pravej zarážky, ktorý A' simuluje. B sa, pre každú hlavu raz, počas výpočtu nedeterministicky rozhodne, že znak, ktorý práve daná hlava číta je posledný v slove. Ak toto nastane, potom B si pre danú hlavu v stave poznačí, že už číta len zarážku a daná hlava sa nachádza na pozícií $n+1$. Ak sa táto procedúra vykoná pre všetky hlavy, potom B ešte odsimuluje krok, ktorý urobí A', keď sú všetky hlavy na $\lhd$. Ak automat B urobil nedeterministické rozhodnutia naozaj na konci slova, tak má všetky hlavy na pozícií $n+1$. Preto akceptuje, ak sa nachádza v akceptujúcom stave (t.j. aj A' sa nachádza v akceptujúcom stave). Ak by B urobil jedno rozhodnutie v zlom momente, potom by jedna hlava nebola na pozícií $n+1$ a B by nemohol akceptovať. Keďže A' akceptuje iba, ak má všetky hlavy na ($\lhd$) vieme, že B môže takýmto spôsobom simulovať A'.

Autori v [24] ukázali, že štruktúra jazykov, ktoré $1BNFA(k)$ akceptujú nie je príliš komplikovaná, konkrétne Parikhovo zobrazenie je semilineárna množina.

Definícia 2.3.2: Podmnožina $P \subseteq \mathbb{N}^n$ je lineárna, ak existuje $m \in \mathbb{N}$ a $\alpha_0, \dots, \alpha_m \in \mathbb{N}^n$ také, že $P = \{\beta | \beta = \alpha_0 + \sum_{i=1}^{m} k_i \alpha_i, \forall i: k_i \geq 0\}$. Podmnožina $S \subseteq \mathbb{N}^n$ je semilineárna, ak existujú lineárne množiny $S_1, S_2, \dots, S_k$ také, že $S = \bigcup_{1 \leq i \leq k} S_i$.

Ľahko vidieť, že každá jednoprvková množina je lineárna. Preto každá konečná množina je semilineárna. Semilineárne množiny sú uzavreté na zjednotenie (z definícií).

Definícia 2.3.3: Nech $\Sigma = \{a_1, \dots, a_n\}$ je abeceda. Parikhovým zobrazením ψ nazveme zobrazenie $\Sigma^* \to \mathbb{N}^n$ také, že $\forall w \in \Sigma^*$ $\psi(w) = \left(\#_{a_1} w, \#_{a_2} w, \dots, \#_{a_n} w\right)$. Kde $\#_x w$ označuje počet znakov x v slove w. Pre jazyk $L \subseteq \Sigma^*$ označíme ako $\psi(L) = \{\psi(w) | w \in L\}$.

Veta 2.3.2: [24] Pre každý jazyk $L \in \mathcal{L}\left(1BNFA(k)\right)$ platí, že $\psi(L)$ je semilineárna množina.

Dôkaz: Uvažujme nedeterministické automaty s k lineárne ohraničenými počítadlami (môžu niesť čísla $\leq n$), ktorú majú dovolené len raz zmeniť pripočítavanie na odpočítavanie. Označme túto triedu ako $NCM(k)$. Tieto automaty majú len jednu čítaciu hlavu a k počítadiel. Počítadlá môžu vykonať operácie $+1, -1$ a test na 0. Pre $NCM(k)$ sa vie, že $\forall L \in \mathcal{L}\left(NCM(k)\right)$ je $\psi(L)$ semilineárna množina. Ukážeme, že platí $\bigcup_{k \geq 1} \mathcal{L}\left(NCM(k)\right) \supseteq \bigcup_{k \geq 1} \mathcal{L}\left(1BNFA(k)\right)$, potom platí aj tvrdenie vety.

Ku každému automatu z triedy $1BNFA(k)$ A existuje ekvivalentný $NCM(k)$ A'. Pre $k = 1$ je tvrdenie evidentné. Nech $k \geq 2$. $k - 1$ počítadiel automatu A' bude simulovať $k - 1$ slepých hláv automatu A. Posledné počítadlo bude mať špeciálny význam. Hneď na začiatku výpočtu A' sa začnú všetky počítadlá plniť do nejakej nedeterministicky určenej hodnoty x. Potom sa začne so simuláciou A. Ak sa niektorá slepá hlava pohne doprava, potom sa príslušné počítadlo zmenší o 1. Keď sa čítacia hlava automatu A posunie doprava, potom sa rovnako posunie aj čítacia hlava A', navyše k-te počítadlo sa zmenší o 1. Hneď ako niektoré počítadlo simulujúce slepú hlavu dosiahne hodnotu 0, znamená to, že daná slepá hlava dosiahla $\triangleleft$. Automat A' ešte musí overiť, či nedeterministicky určená hodnota x je naozaj dĺžka slova. Nato slúži k-te počítadlo, ktoré ak dosiahne 0 mala by čítacia hlava byť na konci slova (a naopak), ak nie tak automat A' neakceptuje.

Nasledujúce dôsledky hovoria o tom, že ďalšia čítacia hlava, ktorá vie rozlišovať všetky znaky sa nedá kompenzovať pomocou ľubovoľného počtu slepých hláv a nedeterminizmus je pre slepé automaty silnejší ako determinizmus.

Dôsledok 2.3.1: [24] Existuje jazyk, ktorý sa dá akceptovať pomocou $1DFA(2)$, ale nedá sa akceptovať pomocou žiadneho $1BNFA(k)$ (pre $k \geq 1$).

Dôkaz: Zoberme jazyk $L = \{a^1\#a^2\# \dots \#a^{n-1}\#a^n\#|n \geq 1\}$. Tento jazyk sa dá akceptovať pomocou $1DFA(2)$, ktorý vždy kontroluje nasledujúce dvojice, či sa počet a zvýšil o 1. Parikhovo zobrazenie $\psi(L)$ je množina $\left\{\left(\frac{(n+1)n}{2}, n\right) \middle| n \geq 1\right\}$, čo nie je semilineárna množina.

Dôsledok 2.3.2: [24] Existuje jazyk, ktorý sa dá akceptovať pomocou $1BNFA(2)$, ale nedá sa akceptovať pomocou žiadneho $1BDFA(k)$ (pre $k \geq 1$).

Dôkaz: Nech L je jazyk z dôsledku 2.3.1. Jazyk L^c (komplement L) sa dá akceptovať pomocou $1BNFA(2)$ A. Ak vstupné slovo nemá tvar $a^*\#a^*\# \dots \#a^*\#$, potom túto skutočnosť automat A ľahko pomocou čítacej hlavy overí. Ak má požadovaný tvar, potom A sa súčasne pohybuje čítacou aj slepou hlavou doprava, až pokým sa na znaku # nedeterministicky rozhodne, že dve skupina znakov a ktoré nasledujú nemajú veľkosť líšiacu sa o 1. Budeme vyžadovať aby takéto rozhodnutie urobil práve raz, inak neakceptuje. Nech $w_1\#w_2\# \dots w_m\#$, kde $w_i \in \{a\}^+$, je zvyšok slova od pozície hláv automatu A po nedeterministickom rozhodnutí. Automat posunie slepú hlavu o jedno políčko vpravo. Čítacou hlavou prejde cez podslovo $w_1\#$ a za každý pohyb vpravo o jedno políčko sa posunie slepou hlavou o 2 pozície. Ďalej čítacou hlavou prejde $w_2\#$ s tým, že slepou hlavou nehýbe. Automat dočíta slovo tak, že oboma hlavami sa hýbe vpravo rovnakou rýchlosťou. Ak sú obe hlavy na $\triangleleft$ potom neakceptuje, inak akceptuje. Ľahko vidieť, že A naozaj akceptuje jazyk L^c. Preto $L^c \in \mathcal{L}\big(1BNFA(2)\big)$.

Nech $L^c \in \mathcal{L}(1BDFA(k))$ pre nejaké $k \geq 1$. Pretože $1BDFA(k)$ sú uzavreté na komplement, platí aj $L \in \mathcal{L}(1BDFA(k))$. Potom patrí L aj do triedy $\mathcal{L}(1BNFA(k))$, čo je v spore s dôsledkom 2.3.1.

Ako sme uviedli vo vete 1.2.2 platí, že $\mathcal{L}(1NFA(k)) \subseteq NSPACE(\log n)$, preto platí aj $\mathcal{L}(1BNFA(k)) \subseteq NSPACE(\log n)$. Ostáva ako otvorený problém, či platí $\mathcal{L}(1BNFA(k)) \subseteq DSPACE(\log n)$. Dokonca sa ani nevie, či by takáto inklúzia priniesla riešenie problému LOG vs. $NLOG$. Nevie sa totiž, či existuje úplný jazyk z triedy $NSPACE(\log n)$, ktorý sa dá akceptovať pomocou $1BNFA(k)$ pre ľubovoľné k [24]. Existuje taký $NSPACE(\log n)$ úplný jazyk, ktorý sa dá akceptovať pomocou $1NFA(2)$ [23].

2.4 Bezstavové viachlavové automaty

Bezstavové viachlavové automaty sú automaty, ktoré používajú len jeden stav. Preto si v stave nemôžu pamätať žiadnu informáciu. Napriek tomu si stále zachovávajú rovnakú výpočtovú silu, ako všeobecné viachlavové automaty.

Definícia 2.4.1: Bezstavový dvojsmerný nedeterministický k-hlavový automat (označíme ako $2StNFA(k)$) je $2NFA(k)$, ktorého množina stavov má len jeden prvok. $2StNFA(k)$ slovo akceptuje, ak výpočet dospeje do konfigurácie, kde sú všetky hlavy na konci vstupnej pásky – symbole $\lhd$. Deterministické a jednosmerné varianty sa definujú obdobne.

Prvé hierarchie bezstavových automatov podľa počtu hláv sa objavujú v [25].

Veta 2.4.1: [25] Pre $k \geq 1$ platí

$$\mathcal{L}(1StDFA(k)) \subset \mathcal{L}(1StDFA(k+1))$$

$$\mathcal{L}(1StNFA(k)) \subset \mathcal{L}(1StNFA(k+1))$$

Veta 2.4.2: [25] Pre $k \geq 1$ platí

$$\mathcal{L}(2StDFA(k)) \subset \mathcal{L}(2StDFA(k+4))$$

$$\mathcal{L}(2StNFA(k)) \subset \mathcal{L}(2StNFA(k+4))$$

Ľahko vidieť, že každý $2NFA(k)$ (resp. $2DFA(k)$) A sa dá simulovať pomocou $2StNFA(k + \lceil \log |K_A| \rceil)$ (resp. $2StDFA(k + \lceil \log |K_A| \rceil)$), kde K_A je počet stavov automatu A. $\log |K_A|$ hláv vie svojou pozíciou uchovávať informáciu o stave automatu A, keď sa stavy automatu kódujú binárne (hlava na $\rhd$ reprezentuje 0, hlava na prvom znaku slova znamená 1). Preto triedy $\bigcup_{k\geq 1} \mathcal{L}(2StDFA(k))$ a $\bigcup_{k\geq 1} \mathcal{L}(2DFA(k))$ sú rovnaké (podobne pre nedeterministický prípad). Ibarra, Karhumäki a Okhotin [26] našli spôsob ako simulovať automaty so stavmi najlepšie ako sa dá, čiže bez nárastu počtu hláv.

Veta 2.4.3: [26] Ku každému $2DFA(k)$ existuje $2StDFA(k)$, ktorý ho simuluje (pre všetky $k \geq 2$).

Dôkaz: Konštrukcia bezstavového automatu k automatu so stavmi pri zachovaní počtu hláv je celkom komplikovaná. Autori v [26] ukázali aj jednoduchšiu konštrukciu, ktorá ale potrebuje navyše jednu hlavu. Tento dôkaz teraz uvedieme.

Zoberme $2DFA(k)$ A taký, že má práve jeden akceptujúci stav a prvá hlava sa pohne v každom kroku. Takýto mód práce zjavne nie je obmedzeným na výpočtovú silu $2DFA(k)$, lebo každý krok keď by prvá hlava malo zostať na mieste sa dá nahradiť dvoma takými, že sa prvá hlava pohne doľava a potom hneď doprava (prípadne naopak, ak je na $\triangleright$). Ďalej predpokladajme, že $S_A = \{1,2,\dots,s\}$ je množina stavov automatu A, počiatočný stav bude 1 a akceptujúci stav s. δ_A je prechodová funkcia automatu A a Σ_A je vstupná abeceda A.

Vyrobíme $2StDFA(k+1)$ B nasledovne. Vstupná abeceda automatu B bude $\Sigma_B = \Sigma \cup \{q^1_{x,y}, q^{-1}_{x,y}, q_{x,y}, {q'}_{x,y} | \delta_A(x, a_1, \dots, a_k) = (y, d_1, \dots, d_k)\}$. Nech $w = w_1 \dots w_n$ je vstupné slovo nad abecedou Σ_A, potom definujeme vstupné slovo $w' = p_1 p_2 w$ nad abecedou Σ_B.

p_1 je lexikografické usporiadanie $tr_{x,y}$ ($tr_{x,y}$ ide pred $tr_{x',y}$, iba ak $x < x'$, alebo ak $x = x'$ a $y \leq y'$). $tr_{x,y} = q^{-1}_{x,y} q_{x,y} q^1_{x,y}$ pre všetky riadky prechodovej funkcie $\delta_A(x, a_1, \dots, a_k) = (y, d_1, \dots, d_k)$.

$p_2 = w_1 t' w_2 t' \dots t' w_n t'$, kde t' je lexikografické usporiadanie ${q'}_{x,y}$ pre riadky prechodovej funkcie $\delta_A(x, a_1, \dots, a_k) = (y, d_1, \dots, d_k)$.

Prvú hlavu B označíme ako stavovú hlavu, ostatné hlavy B budú asociované s hlavami $1, \dots, k$ automatu A. Stavová hlava bude počas výpočtu čítať hlavne časť slova p_1, prvá hlava bude čítať hlavne časť p_2, ostatné hlavy budú čítať hlavne časť w slova w'.

Na začiatku výpočtu automat B nastaví stavovú hlavu na $q_{1,y}$ (1 je počiatočný stav A), prvú hlavu nastaví na ${q'}_{1,y}$ v t' po w_1. Zvyšné hlavy nastaví na w_1 v časti w. Pomocou informácie, ktorá je v slove $p_1 p_2 w$ sa to dá bez pomoci stavov urobiť.

Simulácia začína a končí tým, že stavová hlava číta $q_{x,y}$ a prvá hlava príslušné ${q'}_{x,y}$, pre $x, y \in S_A$. Simuláciu prechodu $\delta_A(x, a_1, \dots, a_k) = (y, d_1, \dots, d_k)$ sa vykoná nasledovne. Hlavy 2 až k sa posunú podľa $d_2, \dots d_k$. Predpokladajme, že $d_1 = 1$ (opačný prípad je analogický). Stavová a prvá hlava sa pohnú doprava (v smere d_1).

Teraz stavová hlava číta $q^1_{x,y}$, prvá hlava sa posúva vpravo (ako indikuje horný index $q^1_{x,y}$) až kým nenájde znak zo Σ_A. Prvá hlava sa pohne vpravo, súčasne s tým sa stavová hlava pohne vľavo. Prvá hlava sa ostane pohybovať vpravo, až kým nenájde symbol $q'_{y,z}$. Keď toto nastane stavová hlava sa pohne vľavo (resp. vpravo), ak $y < x$ (resp. $x < y$). Ostane sa takto pohybovať až kým nenájde $q_{y,z}$. Keď toto nastane, prvá hlava sa začne pohybovať vľavo až kým nebude čítať znak z abecedy Σ_A. Takto sa odsimuloval jeden krok výpočtu A.

Automat B akceptuje ak počas simulácie čítajú stavová a prvá hlava $q_{x,s}$ a $q'_{x,s}$. Potom automat v cykle pošle všetky hlavy $2, \dots, k$ vpravo, až kým nebudú čítať $\lhd$. Potom aj prvú a stavovú hlavu presunie na $\lhd$. A teda akceptuje.

Simulácia pomocou k hláv vyžaduje aby sa prvé dve čítacie hlavy A pohybovali po podslovách p_1 a p_2. Navyše podslová p_1 a p_2 sú komplikovanejšie, aby si prvá čítacia hlava ukazovaním na znak mohla pamätať, či už druhá hlava prešla do nového stavu (a naopak). Stále časti v p_1 a p_2 medzi znakmi vstupu sú konštantne dlhé (je to vlastne zakódovaná prechodová funkcia A).

Podobná simulácia sa autorom v [26] podarila urobiť aj pre nedeterministický prípad. Takáto simulácia ale vyžadovala až $k \geq 3$. Či sa dá $2NFA(2)$ simulovať pomocou $2StNFA(2)$ ostáva ako otvorený problém. Dôsledok tejto vety a vety 2.2.3 je separácia tried $2StDFA(k)$ pre rôzne k.

Dôsledok 2.4.1: Pre $k \geq 2$ a pre $l \in \{2,4,5, \dots\}$ platí

$$\mathcal{L}\big(2StDFA(k-1)\big) \subset \mathcal{L}\big(2StDFA(k)\big)$$

$$\mathcal{L}\big(2StNFA(l-1)\big) \subset \mathcal{L}\big(2StNFA(l)\big).$$

Dôkaz: Nech tvrdenie neplatí, teda existuje $k \geq 3$ (pre $k = 2$ je tvrdenie triviálne) také, že $\mathcal{L}\big(2StDFA(k-1)\big) = \mathcal{L}\big(2StDFA(k)\big)$. Nech A_1 je ľubovoľný automat z triedy $2DFA(k)$. K A_1 existuje podľa vety 2.4.3 automat B_1 z $2StDFA(k)$, ktorý ho simuluje. Teda podľa predpokladu existuje aj B_2 z $2StDFA(k-1)$, ktorý simuluje A_1. K B_2 existuje automat A_2 z $2DFA(k-1)$, ktorý akceptuje rovnaký jazyk ako A_1. Automat B_2 pracuje na slovách $p_1 p_2 w$, časti medzi písmenkami abecedy v p_1 a p_2 (v dôkaze vety 2.4.3 sú označené ako t') sú konštantne dlhé, preto keď A_2 dostane ako vstup w, vie pomocou stavov simulovať výpočet B_2 na $p_1 p_2 w$.

Ako otvorený problém ostáva vzťah medzi $2StNFA(2)$ a $2StNFA(3)$.

2.5 Automaty nezávislé na dátach

Automaty nezávislé na dátach[5] sú také automaty, ktoré sa môžu hlavami po všetkých slovách dĺžky n pohybovať iba po jednej konkrétnej trajektórii. Preto pozície hláv počas výpočtu nezávisia na vstupnom slove, ale iba na jeho dĺžke.

Definícia 2.5.1: [3] Dvojsmerným nedeterministickým k-hlavovým automatom nezávislým na dátach ($2DiNFA(k)$) nazveme $2NFA(k)$, ktorého pozícia i tej hlavy po t krokoch na vstupnom slove w je funkcia $f(i,t,|w|)$ (čiže závisí iba od dĺžky w).

Keďže každý $2DiNFA(k)$ má iba jednu možnú trajektóriu ako sa pohybovať hlavami počas výpočtu, jediné čo ostáva hádať nedeterminizmu je nasledujúci stav. Preto automat A z triedy $2DiNFA(k)$ vieme simulovať pomocou $2DiDFA(k)$ B. B bude mať ako množinu stavov všetky podmnožiny stavov automatu A. Simulácia prejde podobne ako pri simulácií NFA pomocou DFA. Rovnako to platí aj pre jednosmerný prípad.

Veta 2.5.1: [27] Pre všetky $k \geq 1$ platí

$$\mathcal{L}\big(2DiDFA(k)\big) = \mathcal{L}\big(2DiNFA(k)\big)$$

$$\mathcal{L}\big(1DiDFA(k)\big) = \mathcal{L}\big(1DiNFA(k)\big).$$

Každý unárny jazyk L rozpoznávaný pomocou $2DFA(k)$ je vlastne rozpoznávaný aj pomocou $2DiDFA(k)$. Keďže počet slov dĺžky n z jazyka L nie je viac ako jeden, existuje na slovách dĺžky n práve jedna trajektória ako sa môžu hlavy hýbať. Svedčiaci jazyk vo vete 2.2.3 je unárny, preto hierarchia vzhľadom na počet hláv je jej triviálnym dôsledkom.

Dôsledok 2.5.1: Pre všetky $k \geq 1$ platí

$$\mathcal{L}\big(2DiDFA(k)\big) \subset \mathcal{L}\big(2DiDFA(k+1)\big).$$

Pre jednosmerné automaty nezávislé na dátach sa dá použiť výsledok z vety 2.1.4. Jeho úpravou sa získa nasledujúca separácia.

Veta 2.5.2: [3] Pre $k \geq 1$ platí

$$\mathcal{L}\big(1DiDFA(k)\big) \subset \mathcal{L}\left(1DiDFA\left(\tfrac{k(k+1)}{2}+3\right)\right).$$

Či platí táto separácia aj pre k a $k+1$ hláv je stále otvorený problém. Ľahko vidieť, že $\mathcal{L}\big(1DiDFA(1)\big) = \mathcal{L}\big(2DiDFA(1)\big)$. Obe triedy automatov rozpoznávajú práve regulárne jazyky. Pre dva a viac hláv sú už triedy rôzne.

Veta 2.5.3: [3] Pre $k \geq 2$ platí

$$\mathcal{L}\big(1DiDFA(k)\big) \subset \mathcal{L}\big(2DiDFA(k)\big).$$

[5] Označujú sa aj ako zabúdacie (oblivious) automaty.

Holzer [27] ukázal zaujímavý vzťah medzi $2DiDFA(k)$ a uniformnou zložitostnou triedou NC^1. Trieda NC^1 sa považuje za triedu efektívne paralelne riešiteľných problémov. Definuje sa ako trieda tých jazykov, ktoré sa dajú rozoznávať na booleovských obvodoch s polynomiálnym počtom hradiel a celkovou hĺbkou $O(\log n)$. Navyše vyžadujeme, aby sa kód takéhoto obvodu dal skonštruovať na TS s priestorovou zložitosťou $O(\log n)$.

Veta 2.5.4: [27]

$$NC^1 = \bigcup_{k\geq 1} 2DiDFA(k)$$

Vzťah tried NC^1 a LOG je otvorený problém. Pomocou vety 2.5.4 sa dá prepísať do reči viachlavových automatov.

Dôsledok 2.5.2: [27]

$NC^1 = LOG$ práve vtedy, keď $\mathcal{L}\big(1DFA(2)\big) \subseteq \bigcup_{k\geq 1} \mathcal{L}\big(2DiDFA(k)\big)$.

$NC^1 = NLOG$ práve vtedy, keď $\mathcal{L}\big(1NFA(2)\big) \subseteq \bigcup_{k\geq 1} \mathcal{L}\big(2DiDFA(k)\big)$.

2.6 Zametajúce automaty

Zametajúce automaty sú dvojsmerné automaty, ktoré môžu meniť smer čítania hlavy iba na koncových zarážkach. V podstate hlavou zametajú vstup z jednej strany na druhú, z toho pochádza ich názov.

Definícia 2.6.1: Zametajúcim[6] dvojsmerným deterministickým k-hlavovým automatom ($2SwDFA(k)$) nazveme $2DFA(k)$, pre ktorý platí nasledovná vlastnosť. Ak sa hlava automatu začne pohybovať vpravo ($+1$) (resp. vľavo (-1)), potom až do príchodu tejto hlavy na $\lhd$ (resp. $\rhd$) musí hlava v jednom kroku buď ostať na mieste alebo sa pohnúť vpravo (resp. vľavo).

Veta 2.6.1: Pre $k \geq 1$ platí

$$\mathcal{L}\big(2DFA(k)\big) \subset \mathcal{L}\big(2SwDFA(k+1)\big)$$

$$\mathcal{L}\big(2NFA(k)\big) \subset \mathcal{L}\big(2SwNFA(k+1)\big).$$

Dôkaz: Ukážeme vetu pre deterministický prípad. Nech automat A je z triedy $2DFA(k)$, ekvivalentný automat B z $2SwDFA(k+1)$ bude pracovať nasledovne. V stave pamätá informáciu, že jeho i-ta hlava práve simuluje j-tu automatu A. Posledná hlava, ktorá práve nezastupuje žiadnu hlavu A slúži na simuláciu zmeny smeru hlavy A, ktorá nastala inde ako na zarážkach slova. Túto hlavu budeme označovať ako H.

[6] angl. sweeping

Nech B má odsimulovať krok automatu A, v ktorom hlava G mení smer a G sa nenachádza na $\rhd$, ani na $\lhd$. Navyše predpokladajme, že zmena smeru je z $+1$ na -1 (v opačnom prípade postupujeme analogicky). B túto skutočnosť odsimuluje nasledovne.

(1) Hlavu H nastaví na $\lhd$.
(2) Hlavou G sa presunie na $\lhd$ a za každý krok vpravo urobí hlava H krok vľavo.
(3) B si v stave poznačí, že odteraz bude hlavu G simulovať H a naopak.

Ak automat A v jednom kroku urobil zmenu smeru viacerých hláv, potom sa vyššie ucedená procedúra vykoná pre každú takúto hlavu raz. Na poradí vykonávania nezáleží.

Dôsledok 2.6.1: Pre $k \geq 1$ platí

$$\mathcal{L}\big(2SwDFA(k)\big) \subset \mathcal{L}\big(2SwDFA(k+2)\big)$$
$$\mathcal{L}\big(2SwNFA(k)\big) \subset \mathcal{L}\big(2SwNFA(k+2)\big).$$

Dôkaz:

$$\mathcal{L}\big(2SwDFA(k)\big) \subseteq \mathcal{L}\big(2DFA(k)\big) \subset \mathcal{L}\big(2DFA(k+1)\big) \subseteq \mathcal{L}\big(2SwDFA(k+2)\big)$$

Druhá inklúzia (striktná) vyplýva z vety 2.2.3.

Ostáva otázka, či sa tvrdenie vety 2.6.1 dá upraviť z $2SwDFA(k+1)$ na $2SwDFA(k)$ alebo sú triedy všeobecných k-hlavových a zametajúcich k-hlavových automatov rôzne. Ukážeme, že existuje jazyk, ktorý ak sa dá akceptovať na k-hlavových zametajúcich automatoch, potom sa tieto triedy rovnajú.

Definícia 2.6.2: Nech $A = \big(K_A, \Sigma_A, \rhd, \lhd, \delta_A, q_{0_A}, F_A\big)$ je $2DFA(k)$, pre $k \geq 2$. Nech $K_A = \{1, \dots, n\}$, $F_A = \{n\}$, $\Sigma_A = \{a, b\}$ a $q_{0_A} = 1$. Kódom $C(A)$ automatu A nazveme reťazec

$$\#+1^1 + 1^2 + \cdots + 1^n\$D_1 * D_2 * \dots * D_{|\delta_A|}\#.$$

Kódom riadku prechodovej funkcie (δ_A) nazveme reťazec z množiny $\Big\{1^p\big\{\overline{a}, \overline{b}, \overline{\rhd}, \overline{\lhd}\big\}^k\{-1,1,0\}^k 1^q \Big| p, q \in K_A\Big\}$.
K $\big(p, (\gamma_1, \gamma_2, \dots, \gamma_k)\big) \in \delta\Big(\big(q, (a_1, a_2, \dots, a_k)\big)\Big)$ bude korešpondovať kód $1^p\overline{a_1 a_2} \dots \overline{a_k}\gamma_1\gamma_2 \dots \gamma_k 1^q$. $D_1 * D_2 * \dots * D_{|\delta_A|}$ sú zakódované všetky riadky δ_A usporiadané lexikograficky.

Definícia 2.6.3: Jazyk $C(k)$ (pre $k \geq 2$) je množina slov tvaru

$$\{C(A)w_1C(A) \dots C(A)w_nC(A) | w = w_1w_2 \dots w_n \in {\Sigma_A}^*, w \in L(A), A \in 2DFA(k)\}.$$

Lema 2.6.1: $C(k) \in \mathcal{L}\big(2DFA(k)\big)$ (pre $k \geq 2$).

Dôkaz: Nech M je $2DFA(k)$ akceptujúci jazyk $C(k)$. Na vstupnej páske $\triangleright\, C_0(A)w_1C_1(A) \dots C_{n-1}(A)w_nC_n(A) \triangleleft$ bude pracovať nasledovne. Pomocou dvoch hláv skontroluje, či ma vstupné slovo správnu štruktúru (kódy automatu medzi znakmi vstupu w_i) a platnosť $C_i(A) = C_j(A)$ pre všetky $0 \leq i, j \leq n$. Potom skontroluje, či má každý kód $C_i(A)$ správny tvar (overí lexikografické usporiadanie D_i a tvar $+1^1 + 1^2 + \cdots + 1^m$, pre nejaké m) a či sa v kódoch D_i nevyskytuje stav dlhší ako 1^m. Teraz stačí automatu M odsimulovať A na $w = w_1w_2 \dots w_n$. Hlavy automatu M budú simulovať hlavy automatu A. Navyše prvé dve hlavy majú špeciálny význam. Prvá hlava automatu M slúži na simuláciu stavu A, bude čítať hlavne časť kódu pred \$ (túto hlavu označíme ako S). Druhá hlava M (označíme D) slúži na hľadanie kódu C_i v $C(A)$ kódujúceho riadok prechodovej funkcie, ktorý sa dá aplikovať na súčasnú konfiguráciu. Tieto hlavy budú počas výpočtu čítať hlavne kódy $C(A)$. Znaky, ktoré by automat A čítal pomocou hláv S a D, si bude M ukladať v stave. Ostatné hlavy automatu M označíme ako $H_3, \dots, H_k$.

Počas výpočtu si budeme udržiavať nasledujúci invariant. Ak je automat A na slove w v stave p a má hlavy na pozíciách $h_1, \dots, h_k$ ($\forall i \in \{1, \dots, k\}\ 0 \leq h_i \leq n+1$), potom automat M bude v nasledovnej konfigurácií. Hlava S bude ukazovať na znak $+$ pred reťazcom 1^p v $C_{h_1}(A)$, hlava D bude na znaku \$ v kóde $C_{h_2}(A)$. Hlava H_i bude na w_{h_i} (resp. $\triangleright$, $\triangleleft$), ak $1 \leq h_i \leq n$ (resp. $h_i = 0$, $h_i = n+1$). M bude mať v stave uložené znaky w_{h_1}, w_{h_2} (resp. príslušné zarážky).

Problém nastáva, ak hlava S alebo D číta $\triangleleft$ a po tomto znaku nenasleduje žiaden kód. V ďalšom budeme predpokladať, že táto situácia nenastane. Ak by nastala, môže to M zistiť, presunúť hlavu na kód pred $\triangleleft$ a v stave si to zapamätať. Neskôr, ak by sa dotyčnou hlavou malo hýbať, môže podľa informácií v stave odsimulovať krok správne.

Na začiatku výpočtu nastaví automat M hlavu S na znak $+$ pred 1^1 v kóde $C_0(A)$. Rovnako hlavu D nastaví na znak \$ v tomto kóde. V stave si M uloží, že hlavy H a D čítajú znak $\triangleright$. Tým je splnený invariant.

Predpokladajme, že je splnený invariant. Ukážeme, že aj po odsimulovaní jedného kroku automatu A bude splnený. Nech hlava S (resp. D) číta kód $C_S(A)$ (resp. $C_D(A)$). Automat M simuláciu A vykoná nasledovne.

(1) Automat M nájde za pomoci hláv S a D v kóde $C_D(A)$ riadok δ_A, ktorý sa dá aplikovať. K tomu potrebuje vedieť aké symboly práve číta A. Symboly prvých dvoch hláv má v stave, ostatné číta hlavami $H_3, \dots, H_k$. Ďalej potrebuje stav A, na ten ukazuje hlava S. Hlava D pri hľadaní prechádza všetky kódy D_i, porovnáva stav s hlavou S (tá sa po každom kóde vyresetuje), potom porovná vstupné symboly. Ak hlava D našla kód riadku δ_A, ktorý sa dá použiť (označíme

ho ako D_C), automat prejde na krok (2), inak automat M neakceptuje (automat A sa zasekol).

(2) Nech D_C kóduje $\big(p, (\gamma_1, \gamma_2, \dots, \gamma_k)\big) \in \delta\Big(\big(q, (a_1, a_2, \dots, a_k)\big)\Big)$. Hlavy $H_3, \dots, H_k$ sa presunú podľa $\gamma_3, \dots, \gamma_k$ s tým, že ignorujú kódy stroja A, teda pre presunú na pozície ako vyžaduje invariant.

(3) Hlava S sa presunie do príslušného kódu stroja $C_{S+\gamma_1}(A)$. Potom sa S nastaví sa na začiatok tohto kódu a prečíta znak $w_{S+\gamma_1}$ (prípadne zarážky). Tento znak si uloží do stavu. Nakoniec sa hlava S pomocou hlavy D nastaví na $+$ pred 1^p v $C_{S+\gamma_1}(A)$.

(4) Podobne ako v bode (3) sa aj hlava D nastaví na \$ do príslušného $C_{D+\gamma_2}(A)$.

Ľahko vidieť, že invariant ostal zachovaný aj po odsimulovaní jedného kroku A. Automat M akceptuje, ak je niekedy po skončení simulácie kroku A hlava S nastavená na poslednom stave A, čiže $+1^m$. Táto vlastnosť sa dá jednoducho overiť. Ľahko vidieť, že M naozaj akceptuje jazyk $C(k)$.

Veta 2.6.2: Ak $C(k) \in \mathcal{L}\big(2SwDFA(k)\big)$, potom $\mathcal{L}\big(2SwDFA(k)\big) = \mathcal{L}\big(2DFA(k)\big)$.

Dôkaz: Nech L je ľubovoľný jazyk z $\mathcal{L}\big(2DFA(k)\big)$, k nemu existuje automat A z triedy $2DFA(k)$. Predpokladajme, že L je nad abecedou $\{0,1\}$. Ak nie je, môžeme použiť kódovanie znakov do binárnej reprezentácie. Nech S_C je automat z $\mathcal{L}\big(2SwDFA(k)\big)$ pre jazyk $C(k)$. Automat S z triedy $\mathcal{L}\big(2SwDFA(k)\big)$ akceptujúci L bude pracovať nasledovne:

Na slove $w = w_1 w_2 \dots w_n$ bude S simulovať výpočet stroja S_C na slove $C(A)w_1C(A) \dots C(A)w_nC(A)$. Časti $C(A)$ majú konštantnú dĺžku a všetky sú rovnaké. Preto si ich S môže pamätať v stave (pre každú hlavu jednu kópiu) a simulovať pohyb hlavy H_i stroja S_C v $C(A)$ pomocou stavu, zatiaľ čo príslušná hlava H_i' stroja S číta w_x (alebo zarážky). Ak hlava H_i prejde za hranicu $C(A)$, potom sa pohne aj hlava H_i' korešpondujúcim smerom.

Aj pre $2NFA(k)$ je možné skonštruovať podobný jakyk ako $C(k)$. Lema 2.6.1 platí aj pre nedeterministický prípad. Jediný rozdiel v dôkaze lemy je v bode (1), kedy hlava D nedeterministicky nájde kód D_i. Aj pre nedeterministický prípad sa dá ukázať vlastnosť podobná tvrdeniu vo vete 2.6.2.

Kapitola 3

Obratová miera zložitosti

Obratová miera zložitosti sa zavádza ako počet zmien smeru čítania čítacej hlavy. Bola prvý krát skúmaná Hartmanisom v [28]. Zaoberal sa dvojpáskovými Turingovými strojmi so vstupnou páskou s jednosmernou hlavou. Ukázal, že existujú jazyky, ktoré sa dajú akceptovať s použitím k obratov, ale nedajú sa akceptovať s použitím $k-1$ obratov (pre všetky $k \geq 1$). Ďalej dokázal, že pre každú konštruovateľnú funkciu $r(n)$ existuje jazyk, ktorý sa nedá akceptovať s použitím menej ako $r(n)$ obratov (diagonalizácia).

V prvej časti uvedieme definíciu obratovej zložitosti. Ďalej sa budeme zaoberať obratovo konštruovateľnými funkciami. Uvedieme známe výsledky, ako vplýva ohraničenie počtu obratov na popisnú silu viachlavových automatov. V nasledujúcej kapitole sa budeme tiež zaoberať obratovou zložitosťou. Nebude to priamo konečnostavovými automatmi, ale všeobecnejším modelom MAM. Výsledky dosiahnuté na tomto modeli sa dajú aplikovať aj na automaty.

3.1 Definície

Počet obratov hláv, ktoré automat urobí počas výpočtu na slove w, definujeme ako súčet počtu obratov na slove w cez jednotlivé hlavy. Počet obratov, ktoré urobí jedna hlava, je počet zmien smeru pohybu hlavy. Definujeme obratovú mieru pre $2DFA(k)$, pre nedeterministický variant, ako aj pre ostatné modely s dvojsmerným pohybom hlavy, sa obratová miera definuje analogicky.

Definícia 3.1.1: Nech A je automat z triedy $2DFA(k)$, H je jedna jeho čítacia hlava a $w \in \Sigma^*$ je vstupné slovo. Ďalej nech $D = C_1, C_2, \ldots$ je výpočet A na w (C_1 je počiatočná konfigurácia a $C_i \vdash_A C_{i+1}$ pre $\forall i \geq 1$). Potom trajektóriou hlavy H počas výpočtu D nazveme postupnosť $\{h_i\}_{i=1} \in \{-1,0,1\}^*$. Kde $h_i = -1$ (resp. $h_i = 1$, $h_i = 0$), ak sa pri prechode A z konfigurácie C_i do konfigurácie C_{i+1} hlava H posunula vľavo (resp. vpravo, zostala na mieste).

Definícia 3.1.2: Nech A je automat z triedy $2DFA(k)$, H je jedna jeho čítacia hlava, $w \in \Sigma^*$ je vstupné slovo, D je výpočet A na w, a $\{h_i\}_{i=1}$ je trajektória hlavy H na D. Počtom obratov hlavy H na D označíme počet zmien znamienok v trajektórii $\{h_i\}_{i=1}$. Nech $H_1, \ldots, H_k$ sú čítacie hlavy A. Počtom obratov automatu A na D (označíme $REVERSAL_A(D)$) nazveme $\sum_{i=1}^{k}$ počet obratov H_i na D.

Napríklad ak je trajektória hlavy $1, 0, 1, 1, -1, 0, 1, -1$, potom je počet obratov tejto hlavy 3.

Definícia 3.1.3: Počtom obratov automatu A na slove w nazveme

$$REVERSAL_A(w) = \min_D\{REVERSAL_A(D) | D \text{ je akceptujúci výpočet na } w\}.$$

Nech L jazyk akceptovaný automatom A, nech $f: \mathbb{N} \to \mathbb{N}$, potom hovoríme, že automat A pracuje s obratovou zložitosťou $f(n)$, ak platí

$$\max_{w \in L, |w|=n}\{REVERSAL_A(w)\} \leq f(n) \text{ pre } \forall n \in \mathbb{N}.$$

Nech X je trieda strojov (napr. $2DFA(k)$), potom symbolom $X - REVERSAL\big(f(n)\big)$ označíme triedu jazykov, ktoré sa dajú akceptovať pomocou stroja z X s obratovou zložitosťou $O\big(f(n)\big)$.

Napríklad jazyk $\{w\#w | w \in \{0,1\}^*\}$ sa dá akceptovať pomocou $2DFA(2)$ bez použitia obratu. Jazyk $\{w\#w^R | w \in \{0,1\}^*\}$ už na $2DFA(2)$ potrebuje aspoň jeden obrat.

3.2 Obratovo konštruovateľné funkcie

Pri dôkazoch majú veľkú úlohu konštruovateľné funkcie. Sú to funkcie, ktoré sa dajú na danom stroji skonštruovať (väčšinou pomocou ohraničených výpočtových zdrojov). Pomocou takýchto funkcii potom vieme zastaviť výpočet do dosiahnutí istého bodu, alebo si ohraničiť oblasť slova, ktorá nás zaujíma.

Definícia 3.2.1: Funkciu $f: \mathbb{N} \to \mathbb{N}$ nazveme $r(n)$-obratovo konštruovateľnou, ak existuje $2DFA(k)$ A, ktorý na vstupe dĺžky n nepoužije viacej ako $O\big(r(n)\big)$ obratov a po zastavení má prvú hlavu nastavenú na políčku $f(n)$. Obratovo konštruovateľnú nazveme $f(n)$, ak je $f(n)$-obratovo konštruovateľná.

Z definície vyplýva, že žiadna funkcia $f(n) > n$ nemôže byť konštruovateľná (jednoducho automat nemá dostatok miesta na páske). Preto ďalej, ak nebude napísané inak, uvažujeme len funkcie $f(n) \leq n$ (pri operáciách s funkciami predpokladáme, že výsledok je $\leq n$).

Obratovo konštruovateľné funkcie sa spomínajú v [29], kde sa pomocou nich podarilo odseparovať $2DFA(k)$ a $2NFA(k)$ s istým ohraničením na obratovú zložitosť.

Lema 3.2.1: Funkcia $f(n) = n$ je 1-obratovo konštruovateľná.

Dôkaz: Stačí použiť $2DFA(1)$, ktorý so svojou hlavou príde na $\triangleleft$ a potom sa pohne o jedno políčko vľavo.

Lema 3.2.2: [29] Nech f_1, f_2 sú obratovo konštruovateľné funkcie. Potom funkcia $f(n) = f_1(n) + f_2(n)$ je tiež obratovo konštruovateľná.

Dôkaz: Nech A_1 je $2DFA(k)$ počítajúci funkciu f_1 a nech A_2 je $2DFA(l)$ počítajúci funkciu f_2. Potom automat A pre funkciu f bude z triedy $2DFA(k+l)$. Označme jeho hlavy ako $H_1, \dots, H_k, H^1, \dots, H^l$. Automat A najprv odsimuluje A_1 s pomocou hláv $H_1, \dots, H_k$. Po skončení simulácie ukazuje hlava H_1 na pozíciu $f_1(n)$, počas tejto simulácie urobil A najviac toľko obratov ako A_1 čo je $O\big(f_1(n)\big)$. Podobne odsimuluje aj automat A_2 s pomocou hláv $H^1, \dots, H^l$. A sa začne hýbať s hlavou H^1 o jedno políčko vľavo a súčasne sa pohne s H_1 o jednu pozíciu vpravo. Toto opakuje až pokým H^1 nečíta $\triangleright$. Teraz je H_1 nastavená presne na pozíciu $f_1(n) + f_2(n)$. Simulácia A_1 a A_2 zabrala dokopy $O\big(f_1(n) + f_2(n)\big)$ obratov, ukončovacie spočítanie využilo iba jeden obrat. Čiže A počas výpočtu použije $O\big(f_1(n) + f_2(n)\big)$.

Ľahko vidieť, že automatu A by stačilo $\max\{l, k\} + 1$ hláv. Navyše funkcie f_1, f_2 môžu byť aj $r_1(n)$- a $r_2(n)$-konštruovateľné, potom funkcia $f(n) = f_1(n) + f_2(n)$ bude $\big(r_1(n) + r_2(n)\big)$-konštruovateľná.

Lema 3.2.3: Pre ľubovoľnú konštantu $c \in \mathbb{N}$ a ľubovoľnú obratovo konštruovateľnú funkciu $f(n)$ platí, že $c \cdot f(n)$ a $\left\lfloor \frac{1}{c} f(n) \right\rfloor$ sú obratovo konštruovateľné funkcia.

Dôkaz: Keďže $c \cdot f(n) = \underbrace{f(n) + f(n) + \cdots + f(n)}_{c}$, potom podľa lemy 3.2.2 je funkcia $c \cdot f(n)$ obratovo konštruovateľná. Nech A je $2DFA(k)$ pre funkciu $f(n)$. Automat B pre $\left\lfloor \frac{1}{c} f(n) \right\rfloor$ bude používať k hláv (ak $k = 1$, potom bude používať 2 hlavy). Označme si prvú a druhú hlavu B ako H a G. V prvej fáze odsimuluje A, čiže hlava H bude nastavená na pozícií $f(n)$. Hlavu G presunie až na $\triangleright$. Hlavou H prejde c políčok na páske vľavo, zatiaľ čo hlava G sa posunie o jedno políčko vpravo (B si pomocou stavu odpočítava číslo c). Tento pohyb opakuje až pokým hlava H nečíta $\triangleright$. Hlava G je teraz nastavená na pozícií $\left\lfloor \frac{1}{c} f(n) \right\rfloor$. B pracuje s počtom obratov $O\big(f(n)\big)$. Keďže $1/c$ je konštanta platí $O\big(f(n)\big) = O\left(\frac{1}{c} f(n)\right)$.

Lema 3.2.4: Nech f_1 a f_2 sú obratovo konštruovateľné funkcie. Potom funkcia $f(n) = f_1(n) \cdot f_2(n)$ je $\big(f_1(n) + f_2(n)\big)$-obratovo konštruovateľná.

Dôkaz: Nech A_1 je $2DFA(k)$ pre funkciu $f_1(n)$ a A_2 je $2DFA(l)$ pre $f_2(n)$. Automat A pre funkciu $f_1(n) \cdot f_2(n)$ bude mať $k + l + 2$ hláv (stačí aj menej, ale pre jednoduchosť dôkazu použijeme tento počet). Označme hlavy automatu A ako $F, G, H_1^1, \dots, H_k^1, H_1^2, \dots, H_l^2$. Automat A bude pracovať nasledovne:

(1) A odsimuluje automat A_1 na hlavách $H_1^1, \dots, H_k^1$.
(2) A odsimuluje automat A_2 na hlavách $H_1^2, \dots, H_l^2$, po tomto kroku je hlava H_1^1 na pozícií $f_1(n)$ a hlava H_1^2 na pozícií $f_2(n)$.
(3) Hlavou H_1^1 sa posunie o jedno políčko vľavo.
(4) Hlavou H_1^2 prejde až na začiatok slova ($\triangleright$), zatiaľ čo hlavami F a G sa rovnakou rýchlosťou posúva vpravo. (V tomto kroku sa k pozícii hlavy F bude pripočítavať hodnota pozícii H_1^2.)
(5) Hlava H_1^2 si vymení pozíciu s hlavou G.
(6) Ak H_1^1 číta $\triangleright$, potom hlava F ukazuje na pozíciu $f_1(n) \cdot f_2(n)$, inak sa pokračuje krokom (3).

Ľahko vidieť, že A naozaj reprezentuje funkciu $f_1(n) \cdot f_2(n)$. Počas (1) a (2) použije A $O\big(f_1(n) + f_2(n)\big)$ obratov. Počas krokov (3)-(5) sa použije konštantne veľa obratov. Cyklus (3)-(6) sa vykoná $f_1(n)$ krát, čiže počas tohto cyklu použije A $O\big(f_1(n)\big)$ obratov. Celý výpočet sa potom zmestí do $O\big(f_1(n) + f_2(n)\big)$ obratov.

Lema 3.2.5: [29] Ak $f(n)$ je obratovo konštruovateľná funkcia, potom aj $f^i(n)$ je $f(n)$-obratovo konštruovateľná.

Dôkaz: Indukcia vzhľadom na exponent i. Pre $i = 1$ je tvrdenie triviálne splnené podľa predpokladu. Predpokladajme, že funkcia $f^{j-1}(n)$ je $f(n)$-obratovo konštruovateľná, ukážeme, že aj funkcia $f^j(n)$ je $f(n)$-obratovo konštruovateľná. Keďže platí $f^j(n) = f^{j-1}(n) \cdot f(n)$, potom podľa lemy 3.2.4 je $f^j(n)$ $f(n)$-obratovo konštruovateľná.

Lema 3.2.6: [29] Funkcia $\lfloor \log_2 n \rfloor$ je obratovo konštruovateľná.

Dôkaz: Použijeme $2DFA(3)$ A, označme jeho hlavy H_1, H_2, H_3. Automat A bude pracovať v nasledujúcom cykle.

(1) H_1 sa posunie o políčko vpravo.
(2) H_2 ide vľavo až po $\triangleright$, hlava H_3 ide rovnakou rýchlosťou vpravo (teda hlavy si vymenia miesta).
(3) Hlava H_3 sa presunie vľavo až na $\triangleright$, zatiaľ čo hlava H_2 sa dvojnásobnou rýchlosťou presúva vpravo.
(4) Ak počas kroku (3) H_2 narazí na $\triangleleft$, tak výpočet končí a H_1 reprezentuje funkciu $\lfloor \log_2 n \rfloor$, inak sa pokračuje krokom (1).

Na začiatku automat A nastaví hlavu H_1 na pozíciu 0, H_2 na 1 a H_3 na 0. Význam pozícií hláv je nasledovný. Ak H_1 je na pozícií i, potom H_2 je na pozícií 2^i. Hlava H_3 slúži na zdvojenie pozície hlavy H_2 (kroky (2) a (3)). Ľahko vidieť, že takto pracujúci automat naozaj realizuje funkciu $\lfloor \log_2 n \rfloor$. Počas jedného opakovania cyklu sa použije

konštantne veľa obratov. Cyklus sa zopakuje najviac $\lfloor \log_2 n \rfloor$-krát, preto je obratová zložitosť automatu A $O(\log n)$.

Lema 3.2.7: [29] Funkcia $\lfloor \sqrt[i]{n} \rfloor$ je obratovo konštruovateľná.

Dôkaz: Použijeme automat z triedy $2DFA(2i+2)$. Označíme ho ako A, jeho hlavy budú $F, G, H_1, \dots, H_i, H'_1, \dots, H'_i$. Hlava G bude slúžiť iba na pomocné výpočty, hlava F bude reprezentovať číslo x (postupne to budú 0,1,2, …), pre $\forall j \in \{1, \dots, i\}$ bude H_j reprezentovať číslo x^j (hlavy $H'_1, \dots, H'_i$ budú slúžiť na zdvojenie hodnôt $H_1, \dots, H_i$). Zjavne, ak budeme tieto pozície hláv udržiavať počas celého výpočtu, tak hneď ako H_i dosiahne $\triangleleft$, bude hlava F nastavená na pozícii $\lfloor \sqrt[i]{n} \rfloor$ (prípadne $+1$, ak dĺžka vstupného slova nie je tvaru n^i). Aktualizácia pozícií hláv po posune hlavy F o jedno políčko je založená na binomickej vete.

$$(x+1)^i = x^i + \binom{i}{1} x^{i-1} + \cdots + \binom{i}{j} x^{i-j} + \cdots + \binom{i}{i-1} x^1 + 1$$

Keďže hodnoty $x^i, x^{i-1}, \dots, x^1$ máme uložené v hlavách, môžeme aktualizovať pozíciu hlavy H_i pomocou sčítania (binomické koeficienty si pamätáme v stave, teda vieme akou rýchlosťou máme poslať hlavu H_i, keď pripočítavame H_j, $j < i$). Aktualizujeme hodnotu H'_i pomocou hlavy G. Keďže sme práve vynulovali pozície $H_1, \dots, H_{i-1}$, navrátime im pôvodnú pozíciu pomocou hláv $H'_1, \dots, H'_{i-1}$ a hlavy G. Hlavu H_{i-1} vieme podobne nastaviť z x^{i-1} na $(x+1)^{i-1}$, podľa binomickej vety nato potrebujeme iba čísla $x^{i-1}, \dots, x^1$, ktoré stále máme uložené v pozíciách hláv $H_1, \dots, H_{i-1}$ (binomické koeficienty si znovu pamätáme v stave). Takto postupne aktualizujeme všetky hlavy. Počet rôznych binomických koeficientov, s ktorými sa vyskytne mocnina x počas aktualizácie je najviac i^2, preto si ich všetky môžeme uložiť do stavu.

Sčítanie aj nastavenie hodnoty pomocou G sa dá urobiť s konštantným počtom obratov. Keďže máme len konštantne veľa hodnôt (konkrétne i), vieme jednu aktualizáciu urobiť s $O(1)$ počtom obratov. Počet aktualizácií je práve $\lfloor \sqrt[i]{n} \rfloor + 1$, preto aj celkový počet obratov, ktorý A použije je $O\left(\sqrt[i]{n}\right)$.

Podľa vyššie uvedenej lemy je funkcia $\sqrt{n}$ $\sqrt{n}$-obratovo konštruovateľná. Aj funkcia $\sqrt[4]{n}$ je $\sqrt[4]{n}$-obratovo konštruovateľná. Lema 3.2.5 hovorí, že ak je funkcia f obratovo konštruovateľná, potom aj ľubovoľná mocnina f je $f(n)$-obratovo konštruovateľná. Preto funkcia $\sqrt{n} = \left(\sqrt[4]{n}\right)^2$ je aj $\sqrt[4]{n}$-obratovo konštruovateľná.

3.3 Hierarchie zložitostných tried

Ak nepovolíme žiaden obrat, z triedy $2DFA(k)$ (resp. $2NFA(k)$) sa stane trieda $1DFA(k)$ (resp. $1NFA(k)$). O týchto triedach sme hovorili v časti 2.1. Je preto jasné, že niektoré jazyky potrebujú obraty hláv, nech už máme hláv ľubovoľne veľa. Hromkovič [30] ukázal, že existuje jazyk, ktorý sa nedá akceptovať s použitím menej ako $\sqrt[3]{n}$ obratov.

Definícia 3.3.1:

$$L_r = \{w_1 * w_2 * \dots * w_r + w_r * \dots * w_1 | \forall i \in \{1, \dots, r\}: w_i \in \{0,1\}^*\}$$

$$L_R = \bigcup_{r \in \mathbb{N}} L_r$$

$$L = \{x_1 \# x_2 \# \dots \# x_k \# | k \geq 0, \forall i \in \{1, \dots, k\}: x_i \in L_R\}$$

Veta 3.3.1: [30] Jazyk L z definície 3.3.1 nepatrí do triedy $2NFA(k) - REVERSAL\big(f(n)\big)$ pre ľubovoľné k, ak $f(n) \leq n^a$, pre $0 < a < 1/3$.

Dôkaz: Jazyk L_r je podobný jazyku z vety 2.1.4, kde sa použil na odseparovanie tried $1NFA(k)$ a $1NFA(k+1)$. Preto pri jazyku L sa budeme snažiť dokázať, že existuje také podslovo x_i, ktoré automat nečíta žiadnou dvojicou hláv pohybujúcich sa v opačnom smere.

Pre spor predpokladajme, že existuje automat A z triedy $2NFA(k)$, ktorý používa najviac $f(n)$ obratov a akceptuje jazyk L. Navyše predpokladajme, že $f(n) < n^{1/3}$. Symbolom L_z označme množinu slov tvaru

$$\left\{x_1 \# x_2 \# \dots \# x_z \# \middle| x_i = w_{i1} * \dots * w_{iz} + w_{iz} * \dots * w_{i1} \in L_z, |w_{ij}| = z, \forall i,j \in \{1, \dots, z\}\right\}$$

Zrejme platí, že $\bigcup_{z \in \mathbb{N}} L_z \subseteq L$. Ukážeme, že ak A akceptuje $\bigcup_{z \in \mathbb{N}} L_z$, potom musí akceptovať aj slovo nepatriace do L, čo bude hľadaný spor. Budeme hovoriť, že A počas výpočtu porovná korešpondujúce podslová w_{ij}, ak sa počas výpočtu vyskytne konfigurácia v ktorej existuje dvojica hláv taká, že jedna hlava číta znak z prvého w_{ij} a druhá znak z druhého w_{ij}.

Zoberme si časť výpočtu, ktorá neobsahuje žiaden obrat. Potom každá dvojica hláv, ktoré sa hýbu opačným smerom, môže porovnať najviac všetky podslová nejakého x_i (po tomto porovnaní obe hlavy čítajú nekorešpondujúce časti slova). Keďže dvojíc hláv je k^2, počet obratov, ktoré používa A je ostro menší ako $n^{1/3}$ a z je $n^{1/3}$ dostávame, že existuje x_i také, že ani jedno z podslov w_{ij} nie je porovnané s prislúchajúcim podslovom. Čiže každá dvojica hláv automatu A čítajúca podslovo x_i sa pohybuje vždy rovnakým smerom. Počet rôznych slov z L_z je 2^{z^3}, preto existuje aspoň $2^{z^3}/z$ slov, v ktorých sa neporovná žiadne podslovo z x_c (pre konkrétne c). Na týchto x_c automat pracuje podobne ako jednosmerný automat, preto dôkaz ďalej prejde podobne ako dôkaz vety 2.1.4. Čiže nájdeme dve slová, kde sa neporovná

konkrétna dvojica w_{cj}, tieto slová potom zložíme do jedného, ktoré nepatrí do L, ale A ho aj tak akceptuje. Ešte musíme zabezpečiť aby pri tom nepoužil viacej ako $f(n)$ obratov, to sa ale dá, lebo pri skladaní výpočtu použijeme také časti výpočtu, kde sa používa menej obratov.

Pre jazyk L z definície 3.3.1 existuje $2DFA(2)$, ktorý ho rozpoznáva. Dvoma hlavami môže postupne porovnávať w_{ij} v jednom bloku x_i, ak ich porovná všetky, presunie sa na ďalší blok x_{i+1}. Preto ako dôsledok vety 3.3.1 dostaneme nasledovnú separáciu.

Dôsledok 3.3.1: Pre $k \geq 2$ a $1/3 > a > 0$ platí

$$2DFA(k) - REVERSAL(n^a) \subset \mathcal{L}\big(2DFA(k)\big)$$

$$2NFA(k) - REVERSAL(n^a) \subset \mathcal{L}\big(2NFA(k)\big)$$

Na základe vety 3.3.1 sa autorom v [29] podarilo odseparovať $2DFA(k)$ a $2NFA(k)$ s ohraničením na počet obratov. Ukázali, že pre pekné (obratovo konštruovateľné) funkcie sú deterministické dvojsmerné automaty uzavreté na komplement, zatiaľ čo nedeterministické nie sú.

Veta 3.3.2: [29] Pre obratovo konštruovateľné funkcie $f(n)$ je trieda $2DFA(k) - REVERSAL\big(f(n)\big)$ uzavretá na komplement.

Dôkaz: Nech $L \in 2DFA(k) - REVERSAL\big(f(n)\big)$, potom existuje automat A akceptujúci jazyk L s počtom obratov najviac $c \cdot f(n)$. Skonštruujeme automat A' akceptujúci jazyk L^c s počtom obratov $O\big(f(n)\big)$. Z predpokladu, že $f(n)$ je obratovo konštruovateľná vyplýva, že existuje automat A_f z triedy $2DFA(l)$ používajúci $d \cdot f(n)$ obratov, ktorý realizuje funkciu $f(n)$.

A' bude používať $l + k$ hláv. Na začiatku výpočtu odsimuluje automat A_f pomocou prvých l hláv. Po tejto simulácií bude prvá hlava automatu A' ukazovať na pozíciu $f(n)$ na vstupnom slove. A' začne simulovať automat A pomocou zvyšných k hláv. Vždy keď automat A urobí c obratov A' posunie prvú hlavu o jedno políčko vľavo (c obratov si odpočítava v stave). Počas takejto simulácie môže nastať jeden z troch prípadov.

(1) Ak sa automat A počas simulácie dostal do akceptujúceho stavu, potom automat A' neakceptuje vstupné slovo.
(2) Prvá hlava automatu A' sa dostala na začiatok slova (⊳), čo znamená, že automat A prekročil počet obratov $c \cdot f(n)$, čiže automat A' môže slovo akceptovať.
(3) Automat A sa zasekne (nemá definovanú δ funkciu), vtedy môže automat A' slovo akceptovať.

Ľahko vidieť, že automat A' akceptuje práve tie slová, ktoré nepatria do L. Navyše simulácia A zaberie $O(f(n))$ obratov. Na výpočet funkcie v prvej časti sa použije tiež $O(f(n))$ obratov. Preto automat A' pracuje s obratovou zložitosťou $O(f(n))$.

Veta 3.3.3: [29] Pre $1/3 > a > 0$ nie je trieda $2NFA(k) - REVERSAL(n^a)$ uzavretá na komplement.

Dôkaz: Podľa vety 3.3.1 sa jazyk L z definície 3.3.1 nedá akceptovať žiadnym $2NFA(k)$ s počtom obratov menším ako $n^{1/3}$. Ukážeme, že jazyk L^c sa dá akceptovať pomocou $2NFA(2)$ A s konštantným počtom obratov. Ak slovo w z L^c nemá nasledovnú štruktúru vieme to pomocou jednej hlavy zistiť a akceptovať.

$$w = x_1\#x_2\# \dots \#x_m\#, m \geq 1, x_i \in \{*, +, 0, 1\}^*$$

Na takomto slovo w potrebujeme nájsť x_i, ktoré nepatrí do jazyka L_R. Automat A uhádne index i a pomocou dvoch hláv skontroluje, či x_i má nasledujúci tvar, ak nemá tak automat akceptuje.

$$x_i = w_1 * w_2 * \dots * w_r + w_r' * \dots * w_1', \text{ pre } r \geq 0 \text{ a } \forall j \in \{1, \dots, r\}: w_j, w_j' \in \{0,1\}^*$$

Automat A nastaví hlavy na korešpondujúce podslová w_b a w_b'. Index b nedeterministicky uhádne. Potom overí či sú slová, na ktoré ukazujú hlavy rôzne, ak áno tak akceptuje, inak neakceptuje.

Dôsledok 3.3.2: [29] Pre ľubovoľnú obratovo konštruovateľnú funkciu $f(n)$, pre ktorú platí $f(n) = O(n^a)$, $0 < a < 1/3$, platí

$$\bigcup_{k \geq 1} 2DFA(k) - REVERSAL(f(n)) \subset \bigcup_{k \geq 1} 2NFA(k) - REVERSAL(f(n))$$

Ďalší výsledok o obratovej zložitosti ukázal Hromkovič [31]. Použil jazyk $\{w\#w | w \in \{0,1\}^*\}$ (ľahko sa dá akceptovať na $2DFA(2)$). Dokázal, že na modeli $2BNFA(k)$ (časť 2.3) potrebujeme na akceptáciu aspoň $n/\log n$ obratov. Pomocou tohto výsledku sa dá ľahko ukázať nasledujúca separácia.

Veta 3.3.4: [31] Pre každú funkciu $f: \mathbb{N} \to \mathbb{N}$ takú, že $f(n) = o(n/\log n)$ platí

$$\mathcal{L}(2DFA(2)) - \bigcup_{k \geq 1} 2BNFA(k) - REVERSAL(f(a)) \neq \emptyset$$

Sudborough [32] dokázal, že pre ohraničené jazyky, ktoré sú akceptovateľné pomocou $2NFA(k)$ s konštantným počtom obratov, je Parikhovo zobrazenie semilineárna množina (definície 2.3.2 a 2.3.3). Vetu uvedieme tak ako je v [3].

Definícia 3.3.3: Nech $\Sigma = \{a_1, \dots, a_n\}$ je abeceda a L je jazyk nad Σ. L nazveme ohraničeným jazykom, ak $L \subseteq a_1^* a_2^* \dots a_n^*$.

Veta 3.3.5: [32] Parikhovo zobrazenie ohraničených jazykov z triedy $2NFA(k) - REVERSAL(1)$ je semilineárna množina (pre $k \geq 1$).

Nesemilineárna množina je napríklad množina $S = \{n^2 | n \in \mathbb{N}\}$, preto sa podľa vety 3.3.4 nedá jazyk $L = \{a^{n^2} | n \in \mathbb{N}\}$ akceptovať žiadnym $2NFA(k)$ s konštantným počtom obratov. Podľa lemy 3.2.7 je funkcia $\lfloor \sqrt{n} \rfloor$ obratovo konštruovateľná. Podľa lemy 3.2.5 je aj funkcia $\lfloor \sqrt{n} \rfloor^2$ $\sqrt{n}$-obratovo konštruovateľná. Slovo $w \in \{a\}^*$ patrí do jazyka L práve vtedy, keď $\left\lfloor \sqrt{|w|} \right\rfloor^2 = |w|$, preto sa jazyk L dá akceptovať na $2NFA(k)$ s použitím $\sqrt{n}$ obratov.

Kapitola 4

Viachlavový alternujúci stroj

Viachlavový alternujúci stroj MAM[7] bol prvý krát použitý v [31]. MAM je zovšeobecnením alternujúceho Turingovho stroja (TS). Takýto stroj má vstupnú pásku iba na čítanie, na ktorej je zapísané vstupné slovo. Po vstupnej páske sa pohybuje niekoľko čítacích hláv. Na rozdiel od TS nemá pracovné pásky a konečnostavové riadenie, ale používa nekonečne (spočítateľne) veľa stavov (obdoba pamäte v počítačoch). Jeden krok výpočtu závisí od obsahu pamäte a čítaných symboloch na vstupnej páske. Rovnako ako TS môže využívať nedeterminizmus (hádanie správneho kroku výpočtu) a alternovanie (paralelné vetvenie výpočtu). Výpočtový model MAM a dolné odhady zložitosti na ňom boli skúmané v [33], [31] a [34].

V prvej a druhej časti definujeme model MAM a zadefinujeme zložitostné miery používané na modeli MAM. V tretej časti dokážeme výsledky týkajúce sa separácie alternácie, nedeterminizmu a determinizmu pre ohraničenú obratovú zložitosť. Na záver ukážeme hierarchiu zložitostných tried v rámci jedného typu MAM. Výsledky dosiahnuté na MAM sú platné pre širokú škálu výpočtových modelov. Preto ako dôsledky vyššie spomínaných tvrdení sú obdobné separácie a hierarchie pre viachlavové konečnostavové automaty.

4.1 Definície

Definujeme k-hlavový alternujúci stroj ($AM(k)$) rovnako ako v [31]. MAM je zjednotenie takýchto $AM(k)$ cez všetky k. Hlavný rozdiel medzi $AM(k)$ a TS je, že $AM(k)$ má možnosť ľubovoľného usporiadania pamäte. Nie iba lineárne usporiadanie na pracovných páskach ako má TS. Ďalej TS číta v jednom kroku z pásky len obmedzený počet symbolov, $AM(k)$ vidí celú pamäť naraz, teda krok výpočtu môže závisieť od celého obsahu pamäte.

Definícia 4.1.1: k-hlavový alternujúci stroj $AM(k)$ je usporiadaná osmica $(K, \Sigma, K_U, \delta, q_0, F, d, k)$, kde

(1) K je neprázdna množina stavov, potenciálne nekonečná spočítateľná.
(2) Σ je konečná neprázdna množina symbolov (vstupná abeceda), dva význačné symboly $\rhd$ a $\lhd$ označujúce ľavý a pravý koniec slova (endmarker).

[7] angl. multi-head alternating machine

(3) $K_U \subseteq K$ je množina univerzálnych stavov, $K_E = K - K_U$ je množina existenčných stavov.

(4) δ je prechodová funkcia, $\delta: K \times \left((\Sigma \cup \{\triangleright, \triangleleft\})^k\right) \to 2^{(K \times \{-1,0,1\}^k)}$, kde -1 (resp. 1) označuje pohyb hlavy vľavo (resp. vpravo) a 0 označuje hlavu bez pohybu. Pre $\left(\left(q \times (a_1, a_2, \dots, a_k)\right), \left(p \times (\gamma_1, \gamma_2, \dots, \gamma_k)\right)\right) \in \delta$ musí platiť nasledovné. Ak $a_j = \triangleright$ pre nejaké $j \in \{1, \dots, k\}$, potom $\gamma_j \in \{0, 1\}$, ak $a_i = \triangleleft$ pre nejaké $i \in \{1, \dots, k\}$, potom $\gamma_i \in \{0, -1\}$.

(5) $q_0 \in K$ je počiatočný stav.

(6) $F \subseteq K$ je množina akceptačných stavov.

(7) d je kladné prirodzené číslo také, že pre $\forall q \in K_U$, $\forall x \in (\Sigma \cup \{\triangleright, \triangleleft\})^k$ existuje najviac d rôznych dvojíc (p, γ), kde $p \in K$ a $\gamma \in \{-1,0,1\}^k$ takých, že $\left((q, x), (p, \gamma)\right) \in \delta$.

(8) k je počet hláv na páske so vstupným slovom.

Táto definícia $AM(k)$ sa príliš nelíši od definície TS. Jediný podstatný rozdiel je v nekonečnom počte stavov a v žiadnej pracovnej páske. Podmienka (7) v definícií hovorí o tom, že výpočet sa nemôže vetviť na priveľa vetiev. Ďalej definujeme konfigurácie výpočtu $AM(k)$ a reláciu krok výpočtu. Znova sú definície podobné ako pri TS. V konfigurácií si pamätáme stav a pozície hláv na vstupnom slove. Krok výpočtu je potom relácia medzi konfiguráciami, ktorá sa riadi pomocou prechodovej funkcie $AM(k)$.

Definícia 4.1.2: Konfiguráciou $AM(k)$ stroja $M = (K, \Sigma, K_U, \delta, q_0, F, d, k)$ nazveme prvok $\left(w, q, (i_1, i_2, \dots, i_k)\right)$ z množiny $\Sigma^* \times K \times \mathbb{N}_0^k$, kde $\forall j \in \{1, \dots, k\}: 0 \leq i_j \leq |w| + 1$. Symbolom $I_M(w) = \left(w, q_0, (0, \dots 0)\right)$ označíme počiatočnú konfiguráciu stroja M na slove w. Konfiguráciu $\left(w, q, (i_1, i_2, \dots, i_k)\right)$ budeme nazývať existenčnou (resp. univerzálnou, akceptujúcou) ak q je existenčný (resp. univerzálny, akceptujúci) stav.

Definícia 4.1.3: Nech $M = (K, \Sigma, K_U, \delta, q_0, F, d, k)$ je $AM(k)$, krokom výpočtu stroja M nazveme reláciu $\vdash_M$ na konfiguráciách $(\vdash_M \subseteq \left(w, q, (i_1, i_2, \dots, i_k)\right) \times \left(w, p, (i_1', i_2', \dots, i_k')\right))$. Nech C_1, C_2 sú dve konfigurácie, potom $C_1 \vdash_M C_2$, ak C_2 vznikne z C_1 aplikovaním prechodovej funkcie δ na C_1.

Do konfigurácií sme zahrnuli aj vstupné slovo w, v ďalšom budeme niekedy používať konfigurácie bez tohto vstupného slova (keďže to sa počas výpočtu nemôže meniť).

Definícia 4.1.4: Výpočtom stroja M na slove w nazveme neprázdny konečný strom T, s ohodnotenými vrcholmi, ktorý spĺňa nasledujúce podmienky:

(1) každý vrchol v stromu T je označený konfiguráciu $l(v)$ stroja M na w
(2) koreň stromu T je označený konfiguráciou $I_M(w)$
(3) ak v je vnútorný vrchol stromu T a $l(v)$ je univerzálna konfigurácia taká, že $\{C|l(v) \vdash C\} = \{C_1, \dots, C_m\}$, potom v má práve m synov označených konfiguráciami $C_1, \dots, C_m$.
(4) ak v je vnútorný vrchol stromu T a $l(v)$ je existenčná konfigurácia taká, že $\{C|l(v) \vdash C\} = \{C_1, \dots, C_m\}$, potom v má práve jedného syna označeného jednou konfiguráciou spomedzi $C_1, \dots, C_m$.

Akceptujúcim výpočtom M na slove w nazveme výpočet, ktorý má všetky listy označené konfiguráciou s akceptujúcim stavom. Hovoríme, že M akceptuje w, ak existuje akceptujúci výpočet na slove w. Jazykom, ktorý akceptuje stroj M nazveme množinu slov $L(M) = \{w \in \Sigma^* | M \text{ akceptuje } w\}$.

Výpočet je definovaný obvyklým spôsobom. Pri univerzálnom stave sa výpočet vetví do všetkých možných pokračovaní výpočtu, pri existenčnom stave sa vyberie jeden z potenciálnych pokračovaní. Slovo je akceptované, ak skončia všetky vetvy výpočtu v akceptujúcom stave.

Pri dokazovaný dolných odhadov sa používa technika, pri ktorej si definujeme význačné konfigurácie na jednotlivých výpočtoch. Takéto konfigurácie musia akosi zachytávať podstatné vlastnosti výpočtu. Podstatná vlastnosť výpočtu je zvyčajné taká, ktorá vyjadruje komplikovanosť vstupu. Ak si tieto význačné konfigurácie zvýrazníme v strome výpočtu dostaneme vzor výpočtu. Počet rôznych vzorov výpočtu je potom počet podstatne odlišných výpočtov (vzhľadom na význačné konfigurácie). Výpočty na dvoch rôznych slovách s rovnakým vzorom výpočtu sa v zásade líšia iba medzi význačnými konfiguráciami. Ak slová rozsekneme v mieste, kde sa stroj dostane do význačnej konfigurácie, a zlepíme do nového slova, potom toto nové slovo bude mať tiež rovnaký vzor výpočtu ako pôvodné slová. Čiže výpočet na novom slove dopadne rovnako ako na pôvodných. Touto technikou môžeme prinútiť stroj aby akceptoval (resp. zamietol) aj slovo, ktoré má zamietnuť (resp. akceptovať). Podobná technika sa použila aj vo vete 2.1.4.

Definícia 4.1.5: Nech V je množina význačných konfigurácii. Vzorom výpočtu stroja M na slove w nazveme podstrom U stromu výpočtu T stroja M na w, ktorý spĺňa nasledujúce vlastnosti:

(1) koreň stromu T je zároveň koreňom stromu U
(2) ďalšie vrcholy U sú vrcholy T, ktorých ohodnotenie sa nachádza v množine konfigurácií V
(3) vrcholy u a v v strome U sú spojené hranou, ak existuje v strome T cesta z u do v, ktorá neobsahuje vrchol s ohodnotením z množiny V

Definícia 4.1.6: Nech $M = (K, \Sigma, K_U, \delta, q_0, F, d, k)$ je $AM(k)$. Hovoríme, že M je k-hlavový nedeterministický stroj ($NM(k)$), ak $K_U = \emptyset$. Hovoríme, že M je k-hlavový deterministický stroj ($DM(k)$), ak $\forall x \in \left(K \times \left((\Sigma \cup \{\triangleright, \triangleleft\})^k\right)\right)$, $|\{y|(x,y) \in \delta\}| \leq 1$.

Ľahko vidieť, že ak je veľkosť množiny stavov konečná, potom $NM(k) = 2NFA(k)$ a $DM(k) = 2DFA(k)$. Podobne definujeme alternujúce automaty.

Definícia 4.1.7: Symbolom $2AFA(k)$ označíme triedu k-hlavových dvojsmerných alternujúcich automatov. Sú to $AM(k)$, ktoré nepoužívajú viacej ako konštantne veľa stavov.

4.2 Zložitostné miery

Na modeli MAM sa definujú štandardné zložitostné miery ako na ostatných výpočtových modeloch. Časová zložitosť stroja M – $T_M(n)$ sa definuje ako dĺžka najdlhšej vetvy v strome, čo intuitívne zodpovedá času výpočtu. Obratová zložitosť $R_M(n)$ ako maximum z počtu obratov v jednotlivých vetvách, čo znovu zodpovedá definícií obratovej zložitosti na iných modeloch. Paralelná zložitosť $P_M(n)$ ako počet univerzálnych (vetviacich) stavov vo výpočte. Intuitívne je paralelná zložitosť skôr počet listov v strome výpočtu. Počet listov je ale menší ako $dP_M(n) = O\left(P_M(n)\right)$.

Problém je s priestorovou (pamäťovou) zložitosťou $S_M(n)$, keďže stroj môže mať až nekonečne veľa stavov. Navyše na rôzne slová môže stroj použiť úplne odlišné stavy. Čiže definícia ako maximum cez slová dĺžky n z počtu stavov použitých na akceptovanie nie je na mieste. Preto sa $S_M(n)$ definuje ako suma všetkých rôznych stavov použitých pri akceptovaní slov dĺžky n. V definícii sa navyše uvádza logaritmus z tejto hodnoty, potom priestorová zložitosť korešponduje s miestom, ktoré by na pracovnej páske použil TS.

Definícia 4.2.1: Nech M je $AM(k)$ akceptujúci jazyk $L(M)$. Priestorová (pamäťová) zložitosť stroja M je funkcia závisiaca od dĺžky vstupu $S_M(n) = \log C_M(n)$. Kde $C_M(n)$ je minimálny počet rôznych stavov použitých pri akceptačných výpočtoch na slovách z $L(M)$ dĺžky n.

Definícia 4.2.2: Nech M je $AM(k)$ akceptujúci jazyk $L(M)$. Pre akceptačný výpočet D označíme symbolom $T_M(D)$ (resp. $R_M(D)$) maximálny počet krokov (resp. obratov hláv), ktoré M vykoná v sekvenčnom výpočte z koreňa D do niektorého z listov D. Časová (resp. obratová) zložitosť stroja M na slove $w \in L(M)$ je definovaná ako

$$T_M(w) = \min\{T_M(D)|D \text{ je akceptačný výpočet na slove } w\}$$

$$R_M(w) = \min\{R_M(D)|D \text{ je akceptačný výpočet na slove } w\}.$$

Časová (resp. obratová) zložitosť $T_M(n)$ (resp. $R_M(n)$) stroja M je

$$T_M(n) = \max\{T_M(w)|w \in L(M), |w| = n\}$$

$$R_M(n) = \max\{R_M(w)|w \in L(M), |w| = n\}.$$

Definícia 4.2.3: Nech M je $AM(k)$ akceptujúci jazyk $L(M)$. Pre akceptačný výpočet D označíme symbolom $P_M(D)$ počet univerzálnych stavov v D. Paralelná zložitosť stroja M na slove $w \in L(M)$ je definovaná ako

$$P_M(w) = \min\{P_M(D)|D \text{ je akceptačný výpočet na slove } w\}.$$

Paralelná zložitosť $P_M(n)$ stroja M je

$$P_M(n) = \max\{P_M(w)|w \in L(M), |w| = n\}.$$

Budeme používať súčiny zložitostných mier. Napríklad $NM(k) - TIME \cdot SPACE(n)$ je trieda jazykov, ktoré sa dajú akceptovať na k-hlavovom nedeterministickom stroji, ktorého súčin časovej a priestorovej zložitosti je $O(n)$. Obratovú zložitosť označíme ako $REVERSAL$ a paralelnú ako $PARALLELISM$.

4.3 Hierarchie determinizmu, nedeterminizmu a alternácie

Prvé výsledky ohľadom dolných odhadov a separácie na mieru $REVESAL \cdot SPACE \cdot PARALLESISM$ sú ukázané v [31]. Pre špecifický jazyk je dokázaný dolný odhad na modeli MAM s hodnotou $R_M(n)S_M(n)P_M(n) = \Omega\left(\sqrt[3]{n}/\log_2 n\right)$.

Vylepšenie odhadu pre iný jazyk sa podarilo v [35]. Pomocou týchto odhadov sa dá ukázať separácia nedeterministického a alternujúceho MAM s istým ohraničením na výpočtové zdroje. Pomocou techniky z [34] ukážeme, že podobná separácia existuje aj pre deterministický a nedeterministický prípad. Na úvod zadefinujeme jazyky použité pri týchto odhadoch.

Definícia 4.3.1: Symbolom $PS(f, g)$ označíme jazyk slov tvaru

$$x_1\#^{g(n)}x_2\#^{g(n)} \dots \#^{g(n)}x_{f(n)}\#^{g(n)+z(n)},$$

kde $n \geq 1$, $x_i \in \{0,1\}^{g(n)}$, $z(n) = n - 2f(n)g(n)$ a navyše je splnená vlastnosť $\exists j \in \{1, \dots, g(n)\}$, $\oplus_{i=1}^{f(n)} x_{ij} \neq 0$.

Definícia 4.3.2: Symbolom $S(f, g)$ označíme jazyk slov tvaru

$$x_1\#^{g(n)}x_2\#^{g(n)} \dots \#^{g(n)}x_{f(n)}\#^{g(n)+z(n)},$$

kde $n \geq 1$, $x_i \in \{0,1\}^{g(n)}$, $z(n) = n - 2f(n)g(n)$ a navyše je splnená vlastnosť $\forall j \in \{1, \dots, g(n)\}$, $\oplus_{i=1}^{f(n)} x_{ij} = 0$.

Intuitívne je vidieť, že jazyk $S(f,g)$ je ľahký pre stroj s možnosťou alternácie. Stačí keď spustí $g(n)$ paralelných výpočtov, ktoré overia vlastnosť, že XOR má byť rovný 0. Jazyk $PS(f,g)$ je zase jednoduchý pre nedeterministický stroj, keďže stačí vlastnosť overiť len na jednom mieste.

Veta 4.3.1: [35] Nech f a g sú funkcie nad prirodzenými číslami také, že $f(n) = \left\lfloor \frac{1}{2}\sqrt{n} \right\rfloor$ a $g(n) = \lfloor \sqrt{n} \rfloor$. Nech A je MAM akceptujúci jazyk $S(f,g)$, potom platí $R_A(n)S_A(n)P_A(n) = \Omega\left(\sqrt{n}/\log n\right)$.

Veta 4.3.2: Nech f a g sú funkcie nad prirodzenými číslami také, že $f(n) = \left\lfloor \frac{1}{2}\sqrt{n} \right\rfloor$ a $g(n) = \lfloor \sqrt{n} \rfloor$. Potom existuje $2AFA(k)$ A (pre nejaké $k \geq 1$), ktorý akceptuje $S(f,g)$ a $R_A(n) = O(1)$.

Dôkaz: V prvej fáze automat A overí, či má vstupné slovo správny tvar. Najprv skontroluje, či podslová x_i majú rovnakú dĺžku a či túto dĺžku majú aj oddeľovače $\#^{g(n)}$ (až na posledný). To sa dá spraviť pomocou dvoch hláv bez obratov tak, že kontrolujú za sebou idúce podslová. Ďalej A overí, či je v slove správny počet podslov x_i. Nato znova stačia dve hlavy bez akéhokoľvek obratu (ak sa druhá hlava posunie na ďalšie podslovo, tak sa prvá hlava posunie na ďalšie písmenko v x_1). Následne treba ešte overiť, či nie je posledná skupina oddeľovačov príliš dlhá, keďže vyžadujeme, aby $|x_i| = \lfloor \sqrt{n} \rfloor$. To sa dá overiť jednoducho, stačí skontrolovať, či je posledná skupina oddeľovačov kratšia ako $3\lfloor \sqrt{n} \rfloor + 1 = 3|x_1| + 1$. Keď automat overí správny tvar vstupného slova, nastaví sa na prvý symbol slova a pokračuje v druhej fáze. Tá spočíva v overení vlastnosti $\bigoplus_{i=1}^{f(n)} x_{ij} = 0$.

(1) Výpočet sa rozdelí na dve vetvy.
(2) Prvá vetva sa posunie prvou hlavou o jeden symbol vpravo. Ak práve číta # prejde do akceptačného stavu. Inak pokračuje ako (1).
(3) Druhá vetva overí či je XOR bitov na mieste kde sa nachádza prvá hlava 0. V stave si pamätá XOR bitov doposiaľ skontrolovaných podslov. Na správny bit ďalšieho podslova sa presunie pomocou druhej hlavy, ktorou si odpočítava o koľko sa má posunúť prvá hlava (posúva sa o $2\lfloor \sqrt{n} \rfloor$, čo je dĺžka x_i spolu s oddeľovačmi). Keď sa dostane na koniec slova, akceptuje ak je XOR 0 inak neakceptuje.

Ľahko vidieť, že takýto automat akceptuje jazyk $S(f,g)$ a nepoužije pri tom viac ako konštantne veľa obratov.

Dôsledok 4.3.1:

$$\left(AM(k) - REVERSAL \cdot SPACE(1)\right) - \left(NM(k) - REVERSAL \cdot SPACE\left(\sqrt{n}/\log_2 n\right)\right) \neq \emptyset$$

Keďže $AM(k)$ použitý v dôkaze vety 4.3.2 je vlastne alternujúci konečný automat a odhad z vety 4.3.1 musí platiť aj pre nedeterministické konečné automaty môžeme dôsledok 4.3.1 prepísať do reči automatov.

Dôsledok 4.3.2:

$$\left(2AFA(k) - REVERSAL(1)\right) - \left(2NFA(k) - REVERSAL\left(\sqrt{n}/\log_2 n\right)\right) \neq \emptyset$$

Pomocou techniky prezentovanej v [34] urobíme rovnaký odhad ako vo vete 4.3.1 na deterministických modeloch pre jazyk $PS(f,g)$.

Veta 4.3.3: Nech $f(n) = \lfloor\sqrt{n}\rfloor$ a $g(n) = \left\lfloor\frac{1}{2}\sqrt{n}\right\rfloor$. Pre ľubovoľné prirodzené číslo k, nech A je $DM(k)$ taký, že $L(A) = PS(f,g)$. Potom $R_A(n)S_A(n) \in \Omega\left(\sqrt{n}/\log_2 n\right)$.

Dôkaz: Sporom. Nech existuje $DM(k)$ A taký, že $L(A) = PS(f,g)$ a nech platí $R_A(n)S_A(n) \notin \Omega\left(\sqrt{n}/\log_2 n\right)$. Preto platí $\forall a \in \mathbb{R}, \exists s \in \mathbb{N}, \forall m \geq s, R_A(m)S_A(m) < a\sqrt{m}/\log_2 m$. Za a zoberme $\frac{1}{16k^3}$. Potom určite platia nasledujúce nerovnosti:

(i) $k^3 R_A(s) < \frac{1}{16}\sqrt{s} = \frac{1}{16}f(s)$

(ii) $8k^3 R_A(s)S_A(s)\log_2 s < \frac{1}{2}\sqrt{s} = g(s)$

Ukážeme, že existuje slovo z $PS(f,g)$, ktoré nie je akceptované strojom A.

Uvažujme množinu $S_s(f,g) = \{w \in S(f,g) || w| = s\}$. Pre každé slovo z $S_s(f,g)$ musí existovať práve jeden (A je deterministický) neakceptujúci výpočet. Označme ako D_y takýto neakceptujúci výpočet A na slove $y \in S_s(f,g)$. Hovoríme, že A porovnáva pár podslov x_i, x_j práve vtedy, keď existuje konfigurácia A v ktorej jedna hlava A číta znak z podslova x_i a iná hlava znak z podslova x_j. Ako i-významnú konfiguráciu nazveme konfiguráciu A v ktorej niektorá hlava ukazuje na prvý alebo posledný znak podslova x_i hneď po tom ako prešla cez celé podslovo $\#^{g(n)}$. Každá i-významná konfigurácia je významná konfigurácia. Množinou V_y označíme množinu významných konfigurácií, do ktorých sa stroj A dostal počas výpočtu D_y s počtom obratov $\leq R_A(n)$. Ak A používa viacej ako $R_A(n)$ obratov (to sa môže stať, lebo môže ísť o neakceptujúci výpočet), potom do V_y pridáme prvú významnú konfiguráciu, do ktorej sa A dostal s počtom obratov $> R_A(n)$.

Tvrdenie 1: Pre každý neakceptujúci výpočet D_y stroja A na slovách $y \in S_S(f,g)$ existuje pár indexov (i,j), $1 \leq i < j \leq f(s)$ taký, že podslová x_i a x_j nie sú porovnané v D_y s použitím $\leq R_A(s)$ obratov.

Dôkaz: Každý pár hláv môže porovnať najviac $2f(s)$ podslov v časti výpočtu bez obratov. Všetkých párov hláv je $\binom{k}{2} \leq k^2$. Preto v časti výpočtu bez obratov sa dá porovnať najviac $k^3 f(s)$ dvojíc podslov. Dovolíme najviac $R_A(n)$ obratov, preto počet porovnaných dvojíc podslov sa dá zhora odhadnúť výrazom $k^3 f(s)R_A(n)$. Podľa (i)

platí $k^3 f(s) R_A(s) < \frac{f^2(s)}{16}$. Počet všetkých párov (x_u, x_v) je $\binom{f(s)}{2} \geq \frac{f^2(s)}{16}$. Týmto sme ukázali tvrdenie 1.

Dôkaz vety 4.3.3: Počet rôznych slov z množiny $S_s(f,g)$ je $2^{g(s)(f(s)-1)}$ ($f(s)-1$ podslov si môžeme vybrať ľubovoľne, posledné je potom určené tak aby XOR vyšiel 0). Počet rôznych dvojíc indexov podslov na slovách z $S_s(f,g)$ je zhora ohraničený $f^2(s)$. Preto podľa Dirichletovho princípu a tvrdenia 1 existuje dvojica indexov (u,v) taká, že v najmenej $2^{g(s)(f(s)-1)}/f^2(s)$ výpočtoch na slovách z $S_s(f,g)$ nie je dvojica (u,v) porovnaná. Spomedzi týchto slov je

$$\frac{2^{g(s)(f(s)-1)}}{f^2(s)2^{g(s)(f(s)-2)}} = \frac{2^{g(s)}}{f^2(s)}$$

takých, ktoré sa líšia iba v podslovách x_u a x_v (ostatné podslová sú rovnaké – člen $2^{g(s)(f(s)-2)}$). Množinu týchto slov označme ako $S'_s(f,g)$. Pre každé slovo $y \in S_s(f,g)$ označme ako V'_y podmožinu V_y, ktorá obsahuje iba u a v-význačné konfigurácie. Ako vzor výpočtu D_y označíme vzor zohľadňujúci množinu V'_y.

Tvrdenie 2: Počet rôznych vzorov na slovách z $S_s(f,g)$ je ohraničený

$$e(s) = 2^{(k\log_2(s+2)+S_A(s))4kR_A(s)}.$$

Dôkaz: Počet rôznych konfigurácií A je $(s+2)^k 2^{S_A(s)} = 2^{(k\log_2(s+2)+S_A(s))}$. (Člen $(s+2)^k$ je počet rozmiestnia hláv, $2^{S_A(s)}$ je počet stavov.) Počas úseku, bez obratov sa môže vyskytnúť najviac $4k$ u alebo v-význačných konfigurácií (keďže hlavy môžu ísť iba jedným smerom). Počas $R_A(s)$ obratov sa teda vyskytne $\leq 4kR_A(s)$ takýchto konfigurácií. Preto je počet vzorov ohraničený výrazom $e(s)$.

Dôkaz vety 4.3.3: Pomocou tvrdenia 2 vieme určiť nasledujúcu nerovnosť

$$\begin{aligned}
e(s)f^2(s) &\leq 2^{(k\log_2(s+2)+S_A(s))4kR_A(s)} f^2(s) \\
&\leq 2^{(k\log_2(s+2)+S_A(s))4kR_A(s)} s \\
&\leq 2^{(k\log_2(s+2)+S_A(s))4kR_A(s)+\log_2 s} \\
&\leq 2^{(k\log_2(s+2)+\log_2 s+S_A(s))4kR_A(s)} \\
&\leq 2^{(k^2\log_2(s+2)+S_A(s))4kR_A(s)} \\
&\leq 2^{(k^2\,2\log_2 s)4kS_A(s)R_A(s)} \\
&\leq 2^{8k^3 S_A(s)R_A(s)\log_2 s} < 2^{g(s)}
\end{aligned}$$

Posledná nerovnosť vyplýva z vlastnosti (ii). Čiže platí $e(s) < 2^{g(s)}/f^2(s)$, čo znamená, že je menej vzorov ako je slov z množiny $S'_s(f,g)$. Preto existujú dve rôzne slová $w, w' \in S'_S(f,g)$, ktoré majú rovnaký vzor.

$$w = x_1 \#^{g(s)} \dots \#^{g(s)} x_{u-1} \#^{g(s)} x_u \dots \#^{g(s)} x_{v-1} \#^{g(s)} x_v \dots x_{f(s)} \#^{g(s)+z(s)}$$

$$w' = x_1 \#^{g(s)} \dots \#^{g(s)} x_{u-1} \#^{g(s)} x'_u \dots \#^{g(s)} x_{v-1} \#^{g(s)} x'_v \dots x_{f(s)} \#^{g(s)+z(s)}$$

Pre tieto slová platia nasledujúce vlastnosti:

(iii) $x_u \neq x_u'$ a $x_v \neq x_v'$
(iv) w a w' majú rovnaký vzor P
(v) pár podslov (x_u, x_v) z w nie je porovnaný vo výpočte D_w a pár $(x_u{}', x_v{}')$ z w' nie je porovnaný vo výpočte $D_w{}'$

Ukážeme, že existuje slovo z $PS(f,g)$, ktoré je neakceptované strojom A. To bude hľadaný spor. Rozlíšime dva prípady.

(1) Posledná význačná konfigurácia K vzoru P bola dosiahnutá s počtom obratov $\leq R_A(n)$ v aspoň jednom z výpočtov D_w a $D_{w'}$ (bez ujmy na všeobecnosti nech je to D_w). Najviac jedno z podslov x_u a x_v je čítané v K (nech je to x_u). Potom výpočet na slove

$$r = x_1 \#^{g(s)} \dots \#^{g(s)} x_{u-1} \#^{g(s)} x_u \dots \#^{g(s)} x_{v-1} \#^{g(s)} x_v' \dots x_{f(s)} \#^{g(s)+z(s)}$$

je identický s D_w po dosiahnutí K, keďže w je neakceptované, musí byť neakceptované aj toto slovo. Slovo r patrí do jazyka $PS(f,g)$, lebo $w \in S(f,g)$ a my sme zamenili len jedno podslovo za iné (x_v'), preto XOR bitov podslov nemôže vyjsť 0. V ostaných prípadoch ($D_{w'}$ a x_v) postupujeme analogicky.

(2) Posledná význačná konfigurácia K vzoru P bola dosiahnutá v oboch výpočtoch D_w a $D_{w'}$ s počtom obratov $> R_A(n)$. Potom výpočty na slovách r a r'

$$r' = x_1 \#^{g(s)} \dots \#^{g(s)} x_{u-1} \#^{g(s)} x_u{}' \dots \#^{g(s)} x_{v-1} \#^{g(s)} x_v \dots x_{f(s)} \#^{g(s)+z(s)}$$

sa dostanú do konfigurácie K. Výpočty D_r a $D_{r'}$ sa skladajú z častí výpočtov D_w a $D_{w'}$. Preto aspoň jeden z D_r a $D_{r'}$ použije $> R_A(n)$ obratov (nech je to D_r). Čiže ak by aj mal A slovo r akceptovať, musí na to použiť viacej ako $R_A(n)$ obratov, čo nemôže.

Veta 4.3.4: Nech f a g sú funkcie nad prirodzenými číslami také, že $f(n) = \left\lfloor \frac{1}{2}\sqrt{n} \right\rfloor$ a $g(n) = \lfloor \sqrt{n} \rfloor$. Potom existuje $2NFA(k)$ A (pre nejaké $k \geq 1$), ktorý akceptuje $PS(f,g)$ a $R_A(n) = O(1)$.

Dôkaz: Automat A začne rovnako ako automat z dôkazu vety 4.3.2. Keď A dokončí prvú fázu nastaví sa hlavami na začiatok vstupného slova. Uhádne, ktorý bit v podslovách x_i nedáva XOR 0. Overovanie sa dá urobiť rovnako ako vetva výpočtu automatu z dôkazu vety 4.3.2, v ktorej sa overuje XOR na 0.

Dôsledok 4.3.3:

$$\big(NM(k) - REVERSAL \cdot SPACE(1)\big) - \Big(DM(k) - REVERSAL \cdot SPACE\big(\sqrt{n}/\log_2 n\big)\Big) \neq \emptyset$$

Dôsledok 4.3.4:

$$\big(2NFA(k) - REVERSAL(1)\big) - \Big(2DFA(k) - REVERSAL\big(\sqrt{n}/\log_2 n\big)\Big) \neq \emptyset$$

Odhady vo vetách 4.3.1 a 4.3.3 sú skoro tesné. Jazyky $S(f,g)$ a $PS(f,g)$ idú ľahko akceptovať pomocou $2DFA(k)$ A s počtom obratov $\sqrt{n}$. Automat z vety 4.3.2 vykonáva prvú fázu deterministicky. Ďalej s použitím konštantne veľa obratov vieme skontrolovať XOR jedného konkrétneho bitu. Počet bitov v podslovách je $\lfloor\sqrt{n}\rfloor$, preto ich automat A môže všetky skontrolovať.

4.4 Hierarchie v rámci jedného modelu

V predchádzajúcej kapitole sme ukázali, že nedeterminizmus je silnejší ako determinizmus, ak položíme ohraničenie na obratovú zložitosť. Dolný odhad z vety 4.3.3 sa dá použiť aj na vybudovanie hierarchie v rámci deterministických, nedeterministických a alternujúcich strojov. Autori v [34] ukázali takúto hierarchiu pre mieru $TIME \cdot SPACE \cdot PARALLELISM$, podobnou technikou dostaneme aj hierarchiu pre $REVERSAL \cdot SPACE \cdot PARALLELISM$.

Rovnako ako v [34] použijeme modifikovaný jazyk $S(f,g)$ z definície 1.3.2. Priamo do definície jazyka zahrnieme ohraničenie, ktorým chceme odseparovať triedy. Tým akoby skrátime úsek slova, ktorý nesie ťažkú vlastnosť. Zvyšok bude len vypchávka, slúžiaca na predĺženie slova, aby mal automat dosť výpočtových zdrojov.

Definícia 4.4.1: Nech a je funkcia $\mathbb{N} \to \mathbb{N}$ taká, že $\forall n \in \mathbb{N}: a(n) \leq n$. Symbolom $S(f,g)(a)$ označíme jazyk slov tvaru

$$x_1 \#^{g(a(n))} x_2 \#^{g(a(n))} \dots \#^{g(a(n))} x_{f(a(n))} \#^{g(a(n))+z(a(n))} u,$$

kde $n \geq 1$, $x_i \in \{0,1\}^{g(a(n))}$, $z(a(n)) = a(n) - 2f(a(n))g(a(n))$, $u \in \{0,1\}^{n-a(n)}$ a navyše je splnená vlastnosť $\forall j \in \{1, \dots, g(a(n))\}$, $\oplus_{i=1}^{f(a(n))} x_{ij} = 0$.

Veta 4.4.1: Nech a je funkcia $\mathbb{N} \to \mathbb{N}$ taká, že $\forall n \in \mathbb{N}: a(n) \leq n$, nech $f(n) = \lfloor\sqrt{n}\rfloor$ a $g(n) = \left\lfloor \frac{1}{2}\sqrt{n} \right\rfloor$. Ďalej nech A je $AM(k)$ (pre nejaké $k \geq 1$) akceptujúci jazyk $S(f,g)(a)$. Potom platí

$$R_A(n)S_A(n)P_A(n) = \Omega\left(\sqrt{a(n)}/\log_2 n\right).$$

Dôkaz: Sporom. Nech existuje $AM(k)$ A taký, že $L(A) = S(f,g)(a)$ a nech platí $R_A(n)S_A(n)P_A(n) \notin \Omega\left(\sqrt{a(n)}/\log_2 n\right)$. Preto platí $\forall c \in \mathbb{R}$, $\exists s \in \mathbb{N}, \forall m \geq s$, $R_A(m)S_A(m)P_A(m) < c\sqrt{a(m)}/\log_2 m$. Za c zoberme $\frac{1}{16dk^3}$. Potom určite platia nasledujúce nerovnosti:

(i) $dk^3 R_A(s)P_A(s) < \frac{1}{16}\sqrt{a(s)} = \frac{1}{16}f(a(s))$

(ii) $8k^3 dR_A(s)S_A(s)P_A(s)\log_2 s < \frac{1}{2}\sqrt{n} = g(a(s))$

Zafixujeme si množinu slov dĺžky s s rovnakým sufixom u. Takúto množinu označíme ako $S_s^u(f,g)(a)$. Význačné konfigurácie definujeme rovnako ako vo vete 1.3.3, symbolom D_y označíme akceptujúci výpočet na slove $y \in S_s^u(f,g)(a)$.

Tvrdenie 1: Pre každý akceptujúci výpočet D_y stroja A na slovách $y \in S_s^u(f,g)(a)$ existuje pár indexov (i,j), $1 \leq i < j \leq f\big(a(s)\big)$ taký, že podslová x_i a x_j nie sú porovnané v D_y.

Dôkaz: Rovnako ako v tvrdení 1 vo vete 4.3.3 vieme počet porovnaných dvojíc v jednej vetve výpočtu odhadnúť výrazom $k^3 f\big(a(s)\big)R_A(s)$. Počet rôznych vetiev výpočtu je najviac $dP_A(s)$. Preto vo všetkých vetvách sa neporovná dokopy viacej ako $dk^3 f\big(a(s)\big)R_A(s)P_A(s)$. Podľa (i) je tento výraz menší ako $\frac{1}{16}f^2\big(a(s)\big)$. Z čoho vyplýva tvrdenie 1.

Dôkaz vety 4.4.1: Počet rôznych slov z množiny $S_s^u(f,g)(a)$ je $2^{g(a(s))(f(a(s))-1)}$. Preto podľa Dirichletovho princípu a tvrdenia 1 existuje dvojica indexov (u,v) taká, že v najmenej $2^{g(a(s))(f(a(s))-1)}/f^2\big(a(s)\big)$ výpočtoch na slovách z $S_s(f,g)$ nie je dvojica (u,v) porovnaná. Z toho najmenej $2^{g(a(s))}/f^2\big(a(s)\big)$ slov sa líši iba v podslovách x_u a x_v. Množinu takýchto slov označme $S_s^{u\prime}(f,g)(a)$. Ako vzor výpočtu D_y pre $\forall y \in S_s^u(f,g)(a)$ budeme uvažovať vzor vzhľadom na u a v-význačné konfigurácie.

Tvrdenie 2: Počet rôznych vzorov na slovách z $S_s^u(f,g)(a)$ je ohraničený

$$e(s) = 2^{(k\log_2(s+2)+S_A(s))4kdR_A(s)P_A(s)}.$$

Dôkaz: Tvrdenie 2 vo vete 4.3.3 hovorí o prípade, keď je výpočet len jedna vetva. Keďže rôznych vetiev výpočtu môže byť len $dP_A(s)$ tvrdenie 2 platí.

Dôkaz vety 4.4.1: Z platnosti tvrdenia 2 a (ii) dostávame $e(s) < 2^{g(a(s))}/f^2\big(a(s)\big)$. Teda existuje menej rôznych vzorov ako je slov z $S_s^{u\prime}(f,g)(a)$. Čiže aspoň dve slová z $S_s^{u\prime}(f,g)(a)$ majú rovnaký vzor. Tieto slová vieme skombinovať podobne ako vo vete 1.3.3 a dostaneme slovo, ktoré nepatrí do $S(f,g)(a)$, napriek tomu ho A musí akceptovať. Čo je hľadaný spor.

Veta 4.4.2: Nech a je funkcia $\mathbb{N} \to \mathbb{N}$ taká, že $\forall n \in \mathbb{N}: a(n) \leq n$, nech $f(n) = \lfloor\sqrt{n}\rfloor$ a $g(n) = \left\lfloor\frac{1}{2}\sqrt{n}\right\rfloor$. Navyše nech a je $\sqrt{a(n)}$-obratovo konštruovateľná. Potom existuje $2DFA(k)$ (pre nejaké $k \geq 1$) A rozpoznávajúci jazyk $S(f,g)(a)$ taký, že $R_A(n) = O\left(\sqrt{a(n)}\right)$.

Dôkaz: Na začiatku automat A vypočíta hodnotu $a(n)$. Tým si ohraničil oblasť ktorá ho zaujíma. Či sú zvyšok len 0 a 1 skontroluje rýchlo bez akéhokoľvek obratu. Správny tvar slova overí rovnako ako automat z vety 1.3.2 v prvej fáze. Ďalej pomocou $\sqrt{a(n)}$ obratov vie skontrolovať všetky pozície bitov, či je ich XOR 0.

Obratovo konštruovateľným funkciám sme sa venovali v časti 3.2. Vieme preto, že existujú funkcie, ktoré spĺňajú predpoklady vety 4.4.2. Ako príklad môžu slúžiť funkcie $\sqrt[i]{n}$, pre $i \geq 2$, ktoré sú $\sqrt[2i]{n}$-obratovo konštruovateľné. Z viet 4.4.1 a 4.4.2 vyplýva výsledok o separácií zložitostných tried v rámci jedného výpočtového modelu. Pretože vo vete 4.4.2 je použitý deterministický konečnostavový automat platí tento dôsledok na rôznych výpočtových modeloch (vrátane viachlavových automatov).

Dôsledok 4.4.1: Nech M je ľubovoľný stroj z pomedzi $2DFA(k)$, $2NFA(k)$, $2AFA(k)$, $AM(k)$, $NM(k)$ a $DM(k)$. Nech $a(n)$ a $b(n)$ sú funkcie $\mathbb{N} \to \mathbb{N}$, $a(n) \leq n$, $b(n) \leq n$. Navyše nech $a(n)$ je $\sqrt{a(n)}$-obratovo konštruovateľná a $b(n) = o\left(\sqrt{a(n)}/\log n\right)$, potom

$$\left(M - REVERSAL \cdot SPACE \cdot PARALLELISM\left(\sqrt{a(n)}\right)\right) - \left(M - REVERSAL \cdot SPACE \cdot PARALLELISM\big(b(n)\big)\right) \neq \emptyset$$

Záver

V práci sme sa zaujímali o výpočtový model viachlavových automatov. Spomenuli sme výsledky dosiahnuté na tomto modeli, uviedli otvorené problémy a načrtli ďalší smer, akým sa môže výskum uberať.

Napriek tomu, že viachlavové automaty predstavujú jednoduchý výpočtový model, definujú širokú škálu zaujímavých doposiaľ nevyriešených otázok. Jedným z najdôležitejších otvorených problémov je vzťah $2DFA(k)$ a $2NFA(k)$. Rozriešenie tohto vzťahu by prinieslo odpoveď na dlhodobo odolávajúci problém LOG vs. $NLOG$. Taktiež ostáva problém relácie medzi $2DFA(k)$ a $2DiDFA(k)$, vyriešenie ktorého by dalo odpoveď na otázku LOG vs. NC^1. Podobne určenie vzťahu medzi $2NFA(k)$ a $2DiDFA(k)$, by vyriešilo problém $NLOG$ vs. NC^1.

V poslednej kapitole sme ukázali niektoré odhady na obratovú zložitosť a z nich vyplývajúce separácie a hierarchie zložitostných tried. Tieto výsledky hovoria stále iba o čiastočnom riešení vzťahu $2DFA(k)$ a $2NFA(k)$.

Dolné a horné odhady z poslednej kapitoly sa líšia o nezanedbateľný faktor. Preto by bolo zaujímavé skúsiť vylepšiť dolné odhady alebo nájsť efektívnejší spôsob rozpoznávania použitých jazykov. Rovnako je potreba nájsť nové techniky prinášajúce lepšie dolné odhady pre konkrétne jazyky. Ich hľadanie býva spravidla náročné.

Literatúra

1. Hopcroft, J., Ullman, J.: Formal languages and their relation to automata. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1969)

2. Rabin, M., Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. 3, 114–125 (1959)

3. Holzer, M., Kutrib, M., Malcher, A.: Complexity of multi-head finite automata: Origins and directions. Theoretical Computer Science(412), 83–96 (2011)

4. Savitch, W.: Relationships between nondeterministic and deterministic tape complexities. J. Comput. Syst. Sci. 4, 177-192 (1970)

5. Sudborough, I.: On deterministic context-free languages, multihead automata, and the power of an auxiliary pushdown store. Proceedings of the eighth annual ACM symposium on Theory of computing, 141 - 148 (1976)

6. Hartmanis, J.: On non-determinancy in simple computing devices. Acta Informatica Volume 1, Number 4, 336-344

7. Kutrib, M.: On the descriptional power of heads, counters, and pebbles. Theoretical Computer Science(330), 311 – 324 (2005)

8. Sipser, M.: Lower bounds on the size of sweeping automata. Proceedings of the eleventh annual ACM symposium on Theory of computing, 360 - 364 (1979)

9. Hromkovič, J., Schnitger, G.: Nondeterminism versus Determinism for Two-Way Finite Automata: Generalizations of Sipser's Separation. ICALP, 439-451 (2003)

10. Balcerzak, M., Niwiński, D.: Two-way deterministic automata with two reversals are exponentially more succinct than with one reversal. Information Processing Letters(110), 396–398 (2010)

11. Moore, E.: Gedanken-experiments on Sequential Machines. Automata Studies, Annals of Mathematical Studies(34), 129–153 (1956)

12. Kleene, S.: Representation of Events in Nerve Nets and Finite Automata. Automata Studies, Princeton University Press, 3–41 (1956)

13. Rosenberg, A.: On Multi-Head Finite Automata. IBM Journal of Research and Development 10(5), 388 - 394 (1966)

14. Kutylowski, M.: Computational power of one-way multihead finite automata. STACS 90 Proceedings of the seventh annual symposium on Theoretical aspects of computer science (1990)

15. Ibarra, O., Kim, C.: On 3-head versus 2-head finite automata. Acta Informatica 4(2), 193-200 (1975)

16. Yao, A., Rivest, R.: k + 1 Heads Are Better than k. (1978)

17. Chrobak, M.: Hierarchies of one-way multihead automata languages. Lecture Notes in Computer Science 194, 101-110 (1985)

18. Sudborough, I.: Some remarks on multihead automata. RAIRO – Informatique théorique 11(3), 181-195 (1977)

19. Seiferas, J.: Nondeterministic Time and Space Complexity Classes. Technical Report (1974)

20. Monien, B.: Transformational methods and their application to complexity problems. Acta Informatica 6(1), 95-108 (1976)

21. Monien, B.: Two-way Multihead Automata over a One-letter Alphabet. R.A.I.R.O. Informatique theoretique/ Theoretical Informatics 14(1) (1980)

22. Seiferas, J.: Nondeterministic Time and Space Complexity Classes. Technical Report (1974)

23. Sudborough, I.: On tape-bounded complexity classes and multihead finite automata. Journal of Computer and System Sciences 10(1), 62-76 (1975)

24. Ibarra, O., Ravikumar, B.: On partially blind multihead finite automata. Theoretical Computer Science(356), 190 – 199 (2006)

25. Ibarra, O.: On two-way multihead automata. Journal of Computer and System Sciences(7), 28–36 (1973)

26. Ibarra, O., Karhumäki, J., Okhotin, A.: On Stateless Multihead Automata: Hierarchies and the Emptiness Problem. Lecture Notes in Computer Science 4957, 94-105 (2008)

27. Holzer, M.: Multi-head finite automata: data-independent versus data-dependent computations. Theoretical Computer Science 6(1), 97-116 (2002)

28. Hartmanis, J.: Tape-reversal bounded turing machine computations. Journal of Computer and System Sciences 2(2), 117-135 (1968)

29. Bebják, A., Štefaneková, I.: Nondeterminism is essential for reversal-bounded two-way multihead finite automata. Kybernetika 1(24) (1988)

30. Hromkovič, J.: Fooling a two-way nondeterministic multihead automaton with reversal number restriction. Acta Informatica 22(5), 589-594 (1985)

31. Hromkovič, J.: Reversals-space-parallelism tradeoffs for language recognition. Mathematica Slovaca 41(2), 121-136 (1991)

32. Sudborough, I.: Bounded-reversal multihead finite automata languages. Inform. Control(25), 317–328 (1974)

33. Hromkovič, J.: Hierarchy of reversal bounded one-way multicounter machines. Kybernetika 22(2) (1986)

34. Bebják, A., Štefanekoná, I.: Separation of deterministic, nondeterministic and alternating complexity classes. Theoretical Computer Sience 88, 297-311 (1991)

35. Hromkovič, J.: Tradeoffs for language recognition on alternating machines. Theoretical Computer Science(63), 203-221 (1989)