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

Nedeterminizmus v konečných automatoch

práca na študentskú vedeckú konferenciu

Ivana Hudáková

Odbor: Informatika 9.2.1
Vedúci: prof. RNDr. Branislav Rovan, PhD.

Bratislava, 2009

Abstrakt

Nedeterminizmus do konečných automatov zaviedli Rabin a Scott, čím získali zariadenie s rovnakou výpočtovou silou, ktoré je možné jednoduchšie zapísať. Neskôr sa nedeterminizmus začal považovať za prostriedok, ktorý je možné merať.

V prvej časti práce popisujeme viaceré modely konečných automatov, s rôznym množstvom nedeterminizmu a vysvetľujeme, aké sú medzi týmito triedami vzťahy. V druhej časti sa venujeme operáciám, ktoré sa pre modely NFA a DFA skúmali a skúmame tieto operácie pre ostatné triedy konečných automatov. V poslednej kapitole prezentujeme našu novú techniku na oddeľovanie tried konečných automatov. Pomocou tejto techniky dokazujeme tvrdenie SUFA $<_P$ FNA, ktoré bolo doteraz otvorené.

Kľúčové slová: konečné automaty, nejednoznačnosť, stavová zložitosť, operácie na jazykoch

Abstract

The concept of nondeterministic machines was first presented by Rabin and Scott. They showed that NFA has more succinct representation than DFA. Both machines have equal computational power. Later on nondeterminism has been thought as measurable resource.

In first part of this thesis, we describe several model of finite automata with different amount of nondeterminism. We also show relationships between these models. In second part, we provide survey of operations complexity for NFA and DFA. We study these operation for other models of finite automata. In the last chapter, we present our new technique which can be use for separating automata classes. Using this technique we find a solution to an open problem SUFA $<_P$ FNA.

Obsah

Kapitola 1

Úvod

Nedeterminizmus do konečných automatov zaviedli Rabin a Scott, čím získali zariadenie s rovnakou výpočtovou silou, ktoré je však v porovnaní s deterministickými automatmi menej zložité na zápis. Dokázali, že obe zariadenia (DFA aj NFA) akceptujú rovnakú triedu, t.j. regulárne jazyky. Použili na to konštrukciu(subset construction), ktorá nájde k NFA ekvivalentný DFA. Ukázalo sa [17], že táto konštrukcia je optimálna, napriek tomu, že môže viesť k exponenciálnemu nárastu počtu stavov.

Nedeterminizmus sa začal považovať za prostriedok, ktorý je možné merať [12]. Študované miery nedeterminizmu popíšeme v kapitole 2. Tu tiež prinesieme prehľad pojmov z popisnej zložitosti konečných automatov a regulárnych jazykov.

Ťažiskom práce je skúmanie vzťahu medzi množstvom nedeterminizmu v automate a jeho popisnou zložitosťou. V kapitole 3 preto popisujeme modely konečných automatov s rôznym množstvom nedeterminizmu (meraného pomocou nejednoznačnosti). Definované modely porovnáme z hľadiska popisnej zložitosti a ukazujeme existujúcu hierarchiu. V kapitole 4 prinášame definíciu iných modelov, ktoré sú definované pomocou istej štrukturálnej vlastnosti. Modely porovnávame a zaraďujeme ich do hierarchie.

V kapitole 5 sa bližšie venujeme operáciám, ktoré je možné vykonávať na regulárnych jazykoch. Popisujeme známe výsledky o zložitosti opercáií pre triedy DFA a NFA. Skúmame tieto operácie aj pre ostatné modely konečných automatov. Prinášame nové výsledky o zložitosti operácii pre triedy UFA, FNA, LNA, PNA, SUFA, GSUFA a MDFA. Skúmame hlavne operácie zjednotenia, prieniku, zreťazenia, iterácie, reverzu a komplementu regulárnych jazykov.

V poslednej kapitole sa venujeme vysvetleniu našej novej techniky na oddeľovanie rôznych tried konečných automatov. Sformulovali sme a dokázali sme všeobecné tvrdenie, ktoré hovorí o tom, za akých podmienok je možné prehlásiť dve triedy automatov za oddelené. Toto tvrdenie sa opiera o zložitosť realizácie operácií. Pomocou našej novej techniky sme dokázali oddelenie tried SUFA a FNA, čím sme vyriešili poslednú otvorenú otázku v hierarchii tried automatov.

Kapitola 2

Definície základných pojmov

V tejto časti uvedieme základné definície z oblasti konečných automatov. Ďalej popíšeme spôsoby používané na meranie nedeterminizmu v konečných automatoch. Tiež povieme, ako je možné merať zložitosť automatu a tiež zložitosť regulárneho jazyka.

Nedeterministický konečný automat je pätica $A = (K, \Sigma, \delta, I, F)$, kde K je konečná množina stavov, Σ je konečná vstupná abeceda, $I \subseteq K$ je množina počiatočných stavov, $F \subseteq K$ je množina akceptačných stavov a $\delta : K \times \Sigma \to 2^K$ je prechodová funkcia. Jazyk akceptovaný automatom A je množina $L(A) = \{w \in \Sigma^* \mid \exists q \in I : \delta(q, w) \in F\}$. Ďalej uvažujeme len také NFA, ktoré nemôžu používať ε prechody. Ďalšie informácie o konečných automatov možno nájsť napríklad v [5].

2.1 Množstvo nedeterminizmu

Nedeterminizmus (Amount of nondeterminism) v konečnom automate je principiálne možné merať staticky a dynamicky. Statické miery sa vzťahujú na automat a sú to funkcie veľkosti automatu. Tieto miery sa však nepoužívajú. Nedeterminizmus sa meria dynamicky [6], t.j. je to miera vzťahujúca sa na výpočet. Budeme uvažovať nasledujúce tri miery (trochu sa odkloníme od spôsobu definície v [6]).

Miery nedeterminizmu

Nedeterminizmus budeme merať dynamicky a preto si najprv zadefinujeme výpočet.

Pre NFA $A = (K, \Sigma, \delta, I, F)$ a pre vstupné slovo w si môžeme predstaviť **strom výpočtov** automatu A na slove w, označme ho $T_{A,w}$. Tento strom nám pomôže k intuitívnej predstave o mierach nedeterminizmu, ktoré ďalej definujeme. **Výpočet** A na w je jedna cesta v strome $T_{A,w}$ z koreňa do listu. Výpočet budeme označovať ako $C_{A,w}$, prípadne ako C_w alebo C. Formálne: výpočet je postupnosť pohybov $C_{A,w} = m_1 m_2 ... m_n$ taká, že $m_i = (q_{i-1}, w_i, q_i) \in \delta_A$ a $w = w_1 w_2 ... w_n$ pre $1 \leq i \leq n$ a $q_0 \in I$. Nemáme žiadne dodatočné požiadavky na výpočet, ako je tomu v [3] a teda uvažujeme akceptačné aj neakceptačné výpočty.

Počet hádaní a veľkosť rady Pre každý uzol x v strome $T_{A,w}$ definujeme veľkosť rady ako počet bitov, ktoré potrebujeme na uloženie počtu jeho nasledovníkov. Táto miera sa nazýva *guessing*[3, 2]. Formálne: $\forall q \in K\ \forall a \in \Sigma : guessing_A(q,a) = \log_2(|\delta(q,a)|)$.

Ďalej definujeme $guessing_A(C)$ ako veľkosť rady potrebnej na prejdenie výpočtu C v strome $T_{A,w}$. $guessing_A(C) = \sum_{(q,a,p)\in C} guessing_A(q,a)$.

Pre $w \in \Sigma^*$ $guessing_A(w)$ definujeme ako minimálny počet bitov, ktoré potrebujeme, aby sme akceptovali w. $guessing_A(w) = min\{guessing(C_w) \mid C_w$ je akceptačný výpočet$\}$. Ďalšia používaná miera je *branching*, pričom *guessing* je logaritmus tejto miery.

Hromkovič *et al.* [6] berú do úvahy iba počet hádaní (rád), ktoré sú potrebné pri výpočte a zanedbávajú veľkosť rady. Túto mieru nazývajú *advice*

Nejednoznačnosť Ak pre nejaké w je v strome $T_{A,w}$ viac ako jeden akceptačný list, potom je automat nejednoznačný. $ambig_A(w)$ = počet akceptačných výpočtov (listov) v strome $T_{A,w}$. Vplyv množstva nejednoznačnosti na popisnú zložitosť automatu rozoberieme v kapitole 3.

Počet výpočtov Zovšeobecnením nejednoznačnosti je miera $leaf_A(w)$, ktorá je definovaná ako počet všetkých výpočtov A na w a teda je to počet listov v strome $T_{A,w}$. Zrejme ak $\forall w \in \Sigma^* : leaf_A(w) = 1$ potom A je DFA.

2.2 Popisná zložitosť

Popisná zložitosť (Descriptional compexity) každého konečného automatu je konečná. Pre implementáciu je však nevyhnutné optimalizovať počet stavov, prípadne prechodov.

Popisná zložitosť automatu

Popisná zložitosť automatu sa najčastejšie meria jedným z týchto troch spôsobov:

Stavová zložitosť Do úvahy sa berie iba počet stavov automatu. Táto miera sa používa vždy pre deterministické automaty, kde presne vyjadruje veľkosť pamäte potrebnej na uloženie automatu, keďže počet prechodov je $\mathcal{O}$(počet stavov).

Prechodová zložitosť Do úvahy sa berie iba počet prechodov v automate. Táto miera je používaná len pre nedeterministické automaty, keďže počet stavov lineárne neohraničuje počet prechodov a veľkosť potrebnej pamäte na uloženie NFA záleží od počtu prechodov.

Kombinovaná zložitosť Popisná zložitosť sa počíta ako súčet prechodov a stavov. Táto miera bola použitá napríklad v článku [10], v ktorom autori uviedli algoritmus pre nájdenie malého NFA pre zadaný regulárny výraz.

Pokiaľ neuvedieme inak, v ďalšom texte máme pod pojmom popisná zložitosť na mysli stavovú zložitosť. Veľkosť automatu A definujeme ako počet jeho stavov a budeme ju označovať $|A|$.

Popisná zložitosť regulárneho jazyka

Popisnú zložitosť regulárneho jazyka L budeme chápať ako stavovú zložitosť minimálneho automatu, ktorý ho akceptuje. Pre každý model automatu tak dostaneme inú zložitostnú mieru. Známe [6] a používané miery sú:

- **s(L)** je počet stavov minimálneho DFA
- **ns(L)** je počet stavov minimálneho NFA

Dôležité pojmy a označenia

Často budeme o DFA predpokladať, že je *úplný*, t.j. z každého stavu má definované prechody na každý znak zo Σ. Toto vieme dosiahnuť pridaním jedného stavu. Budeme hovoriť, že stav q je *dosiahnuteľný*, ak existuje slovo $w \in \Sigma^*$ a počiatočný stav $q_0 \in I$ tak, že $\delta(q_0, w) = q$. Stav q je *užitočný*, ak je dosiahnuteľný a zároveň z neho možno dosiahnuť akceptačný stav, t.j. existuje $v \in \Sigma^*$ také, že $\delta(q, v) \in F$.

Predpokladáme, že čitateľovi je známa Myhill-Nerodova veta [18, 19, 5], ako aj algoritmus na nájdenie DFA k NFA (subset construction).

Rozšírenie $\delta(q, a)$ funkcie automatu budeme označovať ako $\delta^*(q, w)$, $w \in \Sigma^*$ definované nasledovne.

$$\delta^*(q, \varepsilon) = q$$

$$\delta^*(q, wa) = \bigcup_{q' \in \delta^*(q,w)} \delta(q', a)$$

Ďalej množinu výpočtov pre slovo w začínajúcich v stave q a končiacich v stave p budeme označovať ako

$$\delta(q, w, p) = \{m_1 \dots m_n \mid m_i = (q_{i-1}, w_i, q_i) \in \delta, w = w_1, \dots w_n, 1 \leq i \leq n, q_0 = q, q_n = p\}$$

Na označenie počtu ciest vedúcich zo stavu q do stavu p na slovo w budeme používať symbol $|\delta(q, w, p)|$.

V nasledujúcich častiach popíšeme viaceré varianty konečných automatov a ukážeme tiež, aké sú medzi nimi vzťahy (z hľadiska stavovej zložitosti). Pre každú triedu, ktorou sa budeme zaoberať však platí, že je to nadmnožina DFA a podmnožina NFA. To nám zaručí, že každá trieda bude akceptovať práve všetky regulárne jazyky.

Kapitola 3

Nejednoznačnosť

V kapitole 2 sme popísali spôsoby na meranie nedeterminizmu. Miery *guessing* a *branching* sú definované na akceptačných výpočtoch. Preto je zaujímavé skúmať množstvo akceptačných výpočtov v automate. Toto množstvo je vyjadrené mierou *ambig*, ktorej sa bude venovať v tejto kapitole. Uvedieme tu modely konečných automatov s rôznym stupňom nejednoznačnosti a vysvetlíme, ako je možné tieto modely porovnať. Popíšeme známe výsledky, ktoré viedli k zaradeniu tried do hierarchie.

Definícia 3.0.1. *Nejednoznačnosť konečného automatu A na slove w je počet akceptačných výpočtov v automate A pre slovo w. Toto číslo budeme označovať ako $ambig_A(w)$.*
Nejednoznačnosť konečného automatu A je funkcia $ambig_A : \mathbb{N} \to \mathbb{N}$ definovaná ako

$$ambig_A(n) = \max\{ambig_A(w) \mid w \in \Sigma^n\}$$

3.1 Triedy automatov

Nech f je funkcia $\mathbb{N} \to \mathbb{N}$ a A je NFA. Vo všeobecnosti platí, že A je $f(n)$ nejednoznačný, ak $\forall n \in \mathbb{N} : ambig_A(n) \leq f(n)$. Určením funkcie $f(n)$ dostaneme triedu automatov podľa stupňa nejednoznačnosti.

- **UFA** (Unambiguous finite automata) je trieda automatov, ktorých nejednoznačnosť je 1 ($f(n) = 1$), čiže pre každé slovo existuje najviac jeden akceptačný výpočet. Takýmto automatom hovoríme **jednoznačné**.
- **FNA** (Finitely ambiguous nondeterministic automata) je trieda automatov, ktorých nejednoznačnosť sa dá ohraničiť konštantou ($f(n) = k \in \mathbb{N}$). Automat A je **k–nejednoznačný**, ak $\forall n \in \mathbb{N} : ambig_A(n) \leq k$, pre konštantu k (pripomíname, že k musí byť konštanta vzhľadom na dĺžku slova a môže závisieť od veľkosti automatu).
- **LNA** (Linearly ambiguous nondeterministic automata). V tejto triede je $f(n)$ lineárna funkcia ($f(n) = O(n)$). Takéto automaty budeme volať lineárne nejednoznačné.

- **PNA** (Polynomially ambiguous nondeterministic automata). Pre každý automat z tejto triedy existuje polynóm $p(n)$ taký, že $\forall n \in \mathbb{N} : ambig_A(n) \leq p(n)$.
- **ENA**(Exponentially ambiguous nondeterministic automata). Pre každý $A \in ENA$ platí: $\forall n \in \mathbb{N} : ambig_A(n) \leq k^n$, kde k je konštanta. Keďže na vstupe dĺžky n môže v k–stavovom NFA existovať najviac k^n výpočtov, každý NFA je ENA. Automaty z tejto triedy budeme nazývať exponenciálne nejednoznačné automaty.

Skúmanie práve týchto tried nejednoznačnosti má svoj dôvod. [8] uvádzajú, že nejednoznačnosť je buď ohraničená konštantou, polynomiálna alebo najmenej exponenciálna. Preto ďalej v práci predpokladáme, že ak automat nedosahuje exponenciálnu nejednoznačnosť, potom je jeho nejednoznačnosť najviac polynomiálna. Ak navyše nedosahuje ani polynomiálnu nejednoznačnosť, potom ide o automat z triedy FNA.

Uvedené triedy sa dajú ďalej rozdeliť na podtriedy. Pre ľubovoľné $k \in \mathbb{N}$ existuje trieda k–FNA obsahujúca všetky automaty nejednoznačnosti k. Podobne pre každé k existuje trieda k–PNA, obsahujúca automaty s nejednoznačnosťou $ambig(n) = O(n^k)$.

K triedam automatov je možné definovať *striktnú* triedu. Napríklad do triedy striktných ENA patria práve také automaty, ktoré sú v ENA a nie sú v PNA. Dostaneme teda triedu striktných ENA, PNA, FNA.

3.2 Polynomiálna transformovateľnosť na triedach

Definovanie nasledujúcich pojmov nám umožní bližšie skúmať vzťahy medzi vyššie definovanými triedami. Uvidíme, že na triedach existuje istá hierarchia.

Nech C_1 a C_2 sú dve triedy automatov s rôznym stupňom nejednoznačnosti. Hovoríme, že trieda C_1 je **polynomiálne transformovateľná** na triedu C_2, ak existuje polynóm p taký, že ku každému n–stavovému automatu $A_1 \in C_1$ existuje ekvivalentný $p(n)$–stavový automat $A_2 \in C_2$. Túto skutočnosť značíme $C_1 \leq_P C_2$.

Ak $C_1 \leq_P C_2$ a tiež $C_2 \leq_P C_1$, potom C_1 a C_2 sú **polynomiálne ekvivalentné**, čo značíme ako $C_1 =_P C_2$.

Ak triedy nie sú polynomiálne ekvivalentné, hovoríme, že sú **oddelené** (v niektorých článkoch sa hovorí, že takéto triedy sú oddelené exponenciálnou medzerou, čo je samozrejme pravda). Oddelené triedy označujeme $C_1 \neq_P C_2$. Ak platí $C_1 \leq_P C_2$ a $C_1 \neq_P C_2$, budeme používať skrátený zápis $C_1 <_P C_2$. Vtedy tiež hovoríme, že trieda C_2 je **úspornejšia** než trieda C_1.

3.3 Známe výsledky

Keďže sme triedy definovali podľa rovnakého kľúča (stupňa nejednoznačnosti) a množstvo nejednoznačnosti sme zvyšovali, jasne dostávame $UFA \subseteq FNA \subseteq LNA \subseteq PNA \subseteq NFA$. Stupeň nejednoznačnosti pre modely DFA a UFA je síce rovnaký, ale vieme, že $DFA \subseteq UFA$. Triviálne teda dostávame nasledujúcu hierarchiu:

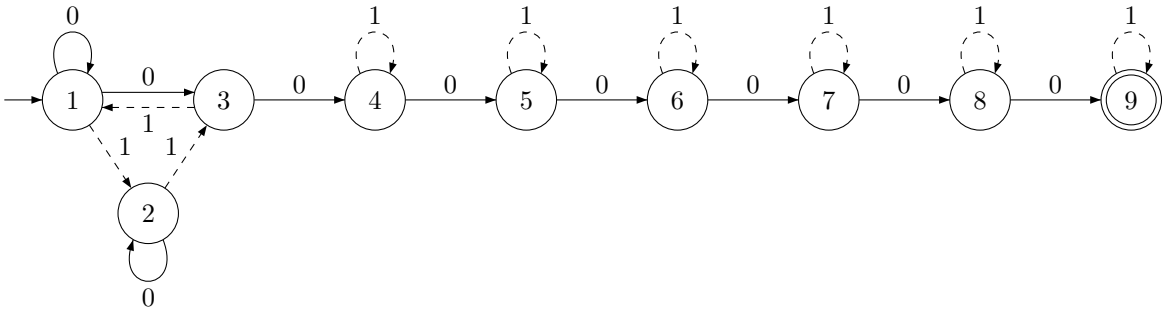
$$DFA \leq_P UFA \leq_P FNA \leq_P LNA \leq_P PNA \leq_P NFA$$

Vzájomné vzťahy medzi týmito triedami boli už vyriešené. Prinášame ich prehľad v nasledujúcej hierarchii.

$$DFA <_P^1 UFA <_P^2 FNA <_P^3 LNA <_P^4 PNA <_P^5 NFA$$

1. Oddelenie tried DFA a UFA bolo uvažované a vyriešené už v roku 1978 [22]. Na oddelenie týchto tried treba nájsť nekonečne veľa takých jednoznačných automatov, pre ktoré najmenšie DFA potrebujú exponenciálne viac stavov. Na obrázku 3.1 je automat $\mathcal{U}_9$[15]. Stav 1 je jeho jediný počiatočný stav a stav 9 je akceptačný. Postupnosť automatov, ktorá oddeľuje DFA od UFA je $\mathcal{U}_n$ pre $n \geq 3$, pričom 1 je vždy počiatočný a n akceptačný stav. Reverz automatu $\mathcal{U}_n$ je DFA, preto $\mathcal{U}_n$ je UFA.

   V [15] je dokázané, že najmenší DFA k $\mathcal{U}_n$ potrebuje presne 2^n stavov. DFA k $\mathcal{U}_n$ je vyrobený klasickou konštrukciou, t.j. stavy deterministického automatu sú podmnožiny množiny stavov daného UFA. Treba ukázať, že takto vzniknutý DFA je minimálny podľa Myhill–Nerodovej vety, a teda, že všetky jeho stavy sú dosiahnuteľné a rozlíšiteľné.

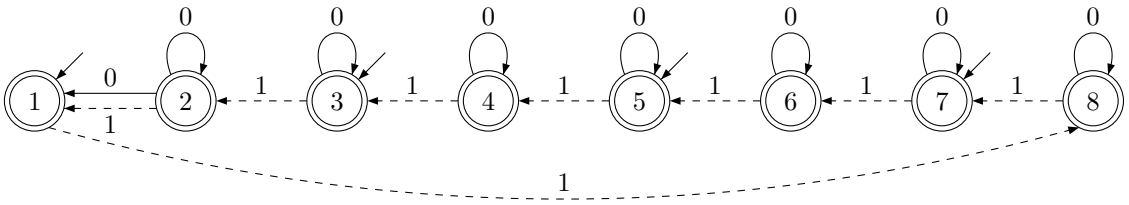


Obr. 3.1: UFA $\mathcal{U}_9$

2. Oddelenie FNA od UFA bolo taktiež vyriešené v [22]. Autor ukázal n–stavový FNA, ku ktorému ekvivalentný UFA má $2^{\Omega(\sqrt{n})}$ stavov. Na dôkaz dolného odhadu pre počet stavov UFA bolo použité nasledujúce tvrdenie.

   Tvrdenie 3.3.1. [22] *Pre daný regulárny jazyk L a slová x_i, y_i pre $1 \leq i \leq n$ definujeme maticu M nasledovne. $M[x_i, y_j] = 1$ práve vtedy, keď $x_i.y_j \in L$, inak $M[x_i, y_j] = 0$. Potom každý UFA pre jazyk L potrebuje aspoň rank(M) stavov.*

   Na obrázku 3.2 vidíme automat $\mathcal{A}_8$. Ľubovoľný stav je označený za akceptačný a nesusedné stavy sú označené ako počiatočné. Tento model uvažoval Leiss [13]. Pri takomto výbere stavov dostaneme na obrázku 3.2 UFA, ku ktorému minimálny DFA potrebuje 2^n stavov. Leung [15] vybral všetky stavy v tomto modeli za akceptačné a počiatočné stavy ponechal. Dostal tak FNA. Ukázal, že najmenší ekvivalentný UFA potrebuje $2^n - 1$ stavov, ktoré zodpovedajú všetkým neprázdnym podmnožinám stavov pôvodného FNA.



Obr. 3.2: FNA $\mathcal{A}_8$

3. Problém $FNA <_P PNA$ [21](1989) bol dlho otvorený. Viaceré jazyky boli navrhované na oddelenie týchto tried. Medzi nimi tiež [6]. Ukázalo sa však [8], že existuje FNA, ktorý nepotrebuje exponenciálne viac stavov.

   Oddelenie tried sa podarilo iba nedávno. Homkovič a Schnitger [7] pomocou komunikačnej zložitosti ukázali, že existuje postupnosť jazykov L_i taká, že pre L_i existuje LNA, ktorého počet stavov je polynomiálny od i, zatiaľčo každý FNA pre L_i potrebuje exponenciálny počet stavov vzhľadom na i.

   Jazyk použitý na oddelenie tried budeme využívať v ďalších častiach práce, preto uvádzame jeho popis tak, ako ho uviedli autori [7].

   Nech Σ_r je abeceda obsahujúca všetky podmnožiny množiny $\{1, \ldots, r^{32}\}$, ktorých mohutnosť je r. Jazyk L_r definujeme nasledovne

$$L_r = \{xy \mid x, y \in \Sigma_r \text{ a } x \cap y \neq \emptyset\}.$$

   Iterovaný jazyk $(L_r)^t$ obsahuje slová tvaru $x_1y_1 \ldots x_ty_t$, kde každé x_iy_i reprezentuje pár prekrývajúcich sa množín. Pre $(L_r)^t$ existuje NFA, ktorý má $poly(r+t)$ stavov. Tento automat budeme označovať N_L, rovnako ako autori [7]. Podarilo sa nám nájsť spôsob, ako akceptovať jazyk $(L_r)^t$ pomocou automatu s $(2t-1)r^{32}+2$ stavmi. Preto budeme ďalej v texte vždy predpokladať, že $|N_L| \leq (2t-1)r^{32}+2$. Tento automat je určite FNA. Vo všeobecnosti totiž platí, že každý automat (ktorý má všetky stavy užitočné) akceptujúci konečný jazyk je FNA.

   Ďalej pre ľubovoľný jazyk L, obsahujúci slová rovnakej dĺžky, definujeme jazyk

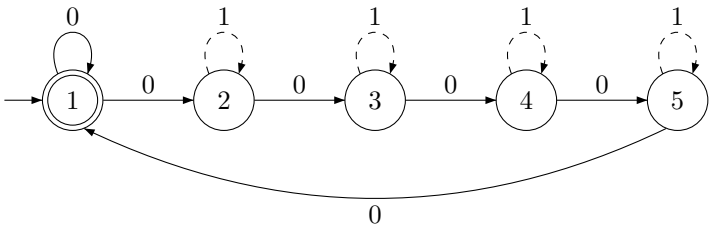
$$\exists_1(L) = \{w_1 \ldots w_m \mid m \in \mathbb{N} \text{ a existuje } 1 \leq i \leq m \text{ také, že } w_i \in L\}.$$

   Podobne môžeme zadefinovať jazyk $\exists_{=0}(L)$ nasledovne

$$\exists_{=0}(L) = \{w_1 \ldots w_m \mid m \in \mathbb{N} \text{ a } w_i \in \Sigma_r^{2t} - L \text{ pre všetky } i\}.$$

   Pre automaty z triedy FNA je zložité rozlíšiť jazyky $\exists_{=0}(L)$ a $\exists_1(L)$. O oddelení tried pomocou jazyka $\exists_1(L)$ hovorí nasledujúce tvrdenie.

   Tvrdenie 3.3.2. *Pre ľubovolné $r \in \mathbb{N}$ položme $t = r^{1/3}$ a $L = (L_r)^t$. Každý FNA pre jazyk $\exists_1(L)$ má aspoň $2^{\Omega(r^{1/3})}$ stavov. Avšak pre jazyk $\exists_1(L)$ existuje LNA, ktorý má $poly(r)$ stavov.*



Obr. 3.3: $\mathcal{A}_5$. 1 je jediný počiatočný aj akceptačný stav.

4. Jazyk $\exists_1(L)$, použitý na oddelenie tried FNA a LNA, je možné zovšeobecniť. Autori [7] tak aj urobili. Zobvšeobcnenie tohoto jazyka je zadefinované takto

$$\exists_k(L) = \{w_1 \dots w_m \mid m \in \mathbb{N} \text{ a } w_i \in L \text{ pre aspoň } k \text{ rôznych pozícií}\}.$$

Tvrdenie 3.3.3. *Pre ľubovolné $r \in \mathbb{N}$ položme $t = r^{1/3}$ a $L = (L_r)^t$. Každý NFA s nejednoznačnosťou $o(n^k)$ pre jazyk $\exists_k(L)$ má aspoň $2^{\Omega((r/k^2)^{1/3})}$ stavov. Avšak pre jazyk $\exists_k(L)$ existuje NFA s nejednoznačnosťou $O(n^k)$, ktorý má $k.poly(r)$ stavov.*

Tvrdenie 3.3.3 vypovedá o hierarchii v rámci triedy PNA. Pre každé $k \in \mathbb{N}$ platí k–PNA $<_P$ $(k+1)$–PNA.

5. NFA A je striktne exponenciálne nejednoznačný, ak $A \in NFA$ a $A \notin PNA$. Pre dôkaz rozhodnuteľnosti problému, či NFA je striktne exponenciálne nejednoznačný [9] uviedli charakterizáciu striktných ENA pomocou nasledujúcej vlastnosti. Automat M je striktne exponenciálne nejednoznačný práve vtedy, keď existuje stav q a slovo w také, že existuje viac ako jedna cesta z q do q na w.

   Leung [14] prezentoval postupnosť automatov $\mathcal{A}_n$. Prechodová logika automatu je znázornená na obrázku 3.3. Automat má jeden počiatočný stav q_1, ktorý je zároveň jediným akceptačným stavom.

   $\mathcal{A}_n$ je striktne exponenciálne nejednoznačný. Napríklad slovo 0^n možno spracovať z q_1 do q_1 dvoma spôsobmi. Tiež nejednoznačnosť slova 0^m je najmenej $2^{m/n}$, pre každé $m = k.n$.

   Leung ďalej ukázal, že minimálny DFA k $\mathcal{A}_n$ má 2^n stavov, každý ekvivalentný UFA potrebuje aspoň $2^n - 1$ stavov a tiež každý ekvivalentný PNA potrebuje $2^n - 1$ stavov.

   Iná postupnosť automatov na oddelenie NFA od PNA bola prezentovaná v [6].

Kapitola 4

Iné modely

V predchádzajúcej kapitole sme sa zaoberali modelmi, ktoré boli definované podľa stupňa nejednoznačnosti. Nevedeli sme toho však príliš veľa usúdiť o tom, ako vyzerá δ funkcia automatu, ani ako sa automat správa na slovách, ktoré nie sú z jazyka. V tejto kapitole si ukážeme modely, ktoré sú definované na základe nejakej štrukturálnej vlastnosti. Tieto modely potom porovnáme s modelmi z predchádzajúcej kapitoly a zaradíme ich do vyššie spomenutej hierarchie. Na konci časti 4.2 uvedieme otvorený problém, ktorým sa budeme ďalej v práci zaoberať, až napokon v kapitole 6 prinesieme jeho riešenie.

4.1 SUFA

Ak chceme pracovať s modelmi definovanými v kapitole 3, potrebujeme ich vedieť charakterizovať na základe nejakej vlastnosti. Preto uvádzame známe [9] štrukturálne vlastnosti modelov definovaných na základe množstva nejednoznačnosti. Predpokladáme, že všetky stavy sú užitočné. Potom:

- ENA – NFA je striktne exponenciálne nejednoznačný práve vtedy, keď existuje stav q a slovo w také, že existuje viac ako jedna cesta z q do q na w.
- PNA – NFA je striktne polynomiálne nejednoznačný práve vtedy, keď nie je striktne exponenciálne nejednoznačný a a existujú rôzne stavy p, q a slovo w, $|w| \geq 1$ tak, že existujú cesty na slovo w z p do p, z q do q aj z p do q.
- FNA – NFA je FNA, ak nie je striktne polynomiálne nejednoznačný.

Tieto vlastnosti sú definované v zmysle prechodovej logiky automatu a sú nezávislé od výberu akceptačných stavov. Leung [16] skúmal štrukturálne vlastnosti UFA. Uviedol nový model (SUFA) a ukázal, že UFA a SUFA sú dve rozdielne triedy automatov(čo znamená, že nenašiel vlastnosť príznačnú pre UFA).

Definícia 4.1.1. [16] *Štrukturálne jednoznačný konečný automat (SUFA – Structurally unambiguous finite automaton) je NFA s jedným počiatočným stavom $A = (K, \Sigma, \delta, q_0, F)$ taký, že pre každé slovo $w \in \Sigma^*$ a stav $q \in K$ existuje najviac jedna cesta z q_0 do q pre slovo w.*

$SUFA \not\subseteq UFA$ – SUFA nemusí byť jednoznačný. Ak máme viac akceptačných stavov, SUFA dovoľuje, aby na jedno slovo w do každého stavu viedla jedna cesta. Na druhej strane však SUFA s jediným akceptačným stavom je UFA.

$UFA \not\subseteq SUFA$ – model UFA nekladie žiadne požiadavky na neakceptačné stavy, preto do nich môže viesť viac ciest, čo porušuje vlastnosť SUFA. Ak však budeme brať do úvahy iba automaty, ktoré majú všetky stavy užitočné, situácia sa zmení. Vyslovili a dokázali sme nasledujúce tvrdenie.

Tvrdenie 4.1.1. *Každý UFA s jedným počiatočným stavov, ktorý má všetky stavy užitočné, je $SUFA$.*

Dôkaz. Nech $A = (\Sigma, K, q_0, F, \delta)$ je UFA bez neužitočných stavov. Nech A nie je SUFA. Potom existuje stav $q \in K$ a slovo $w \in \Sigma^*$ tak, že z q_0 vedú do q aspoň dve cesty na w. Stav q je užitočný a preto existuje slovo $u \in \Sigma^*$ také, že $\delta(q, u) \cap F \neq \emptyset$. Slovo $w.u$ je potom možné v A akceptovať aspoň dvoma cestami, čo je v spore s tým, že A je UFA. □

Toto tvrdenie by sme mohli zapísať ako "UFA nie je úspornejší ako SUFA", keďže k automatom, ktoré majú nejaké neužitočné stavy, vieme ľahko nájsť ekivalentné automaty bez neužitočných stavov. Podmienka na jeden počiatočný stav je nutná kvôli definícií SUFA. Vieme sa s ňou vyrovnať jednoducho pridaním nového počiatočného stavu. V ďalších dôkazoch budeme môcť využívať fakt, že každý UFA vieme rýchlo prerobiť na SUFA.

Pre jazyk *some–register–on* [15] bolo úkázané, že $UFA <_P SUFA$. Popíšeme preto tento jazyk aj $SUFA$, ktorý ho akceptuje.

Majme n registrov, z ktorých každý môže obsahovať hodnotu 0 alebo 1("off" alebo "on"). Na nich definujeme operáciu kopírovania hodnoty z registra i do registra j, ktorú budeme označovať $C_{i,j}$. Operáciu kopírovania registra do seba nepovolíme a tak je možné mať $n.(n-1)$ operácií. Tieto budú tvoriť abecedu jazyka *some–register–on*. Slová budú pozostávať z týchto inštrukcií. Inštrukcie sa budú vykonávať v takom poradí, ako sú na vstupe (sekvenčne).

Na začiatku je v prvom registri hodnota 1 a v ostatných je 0. Pri vykonávaní inštrukcií $C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2}$ bude stav registrov takýto.

register	1	2	3	4
na začiatku	1	0	0	0
po $C_{1,4}$	1	0	0	1
po $C_{4,2}$	1	1	0	1
po $C_{3,1}$	0	1	0	1
po $C_{4,3}$	0	1	1	1
po $C_{1,2}$	0	0	1	1

Slovo inštrukcií patrí do jazyka some-register-on práve vtedy, keď po jeho vykonaní na registroch bude v nejakom registri hodnota 1. Vidíme teda, že slovo $C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2}$ patrí do jazyka, keďže registre 3 a 4 majú na konci hodnotu 1.

$SUFA$ pre tento jazyk sa bude snažiť "uhádnuť", v ktorom registri bude na konci výpočtu hodnota 1. Automat bude mať pre každý register jeden stav a všetky stavy budú akceptačné. Bude platiť, že ak je automat v stave i, potom v registri i je hodnota 1. Ak sa vtedy do registra i skopíruje nejaká hodnota, automat sa zasekne (a teda neakceptuje). Formálnejšie $SUFA$ pre jazyk some-register-on je NFA $A = (K, \Sigma, \delta, q_0, F)$, kde $K = \{1, \ldots, n\}$, $\Sigma = \{C_{i,j} \mid 1 \leq i, j, \leq n, i \neq j\}$, $q_0 = 1$, $F = K$ a $\forall 1 \leq i, j \leq n; i \neq j : j \in \delta(i, C_{i,j})$, $\forall 1 \leq i, j, k \leq n, i \neq j \neq k : k \in \delta(k, C_{i,j})$.

Prvú časť δ funkcie ($j \in \delta(i, C_{i,j})$) možno chápať takto: V registri i bola hodnota 1 a tá sa prekopírovala do registra j, pričom nedeterministicky tipujeme, že v registri j ostane hodnota 1. Druhú časť δ funkcie ($k \in \delta(k, C_{i,j})$) možno chápať takto: V registri k bola hodnota 1, tá sa nezmenila pretože požadujeme $j \neq k$ a preto ostávame v tomto stave. Nedeterministické rozhodnutie nastáva, keď $k = i$. Vtedy máme možnosť buď prejsť do registra j, kde bude hodnota 1, alebo ostať v registri k, kde je tiež hodnota 1.

Je ukázané [16], že A je $SUFA$ a tiež, že najmenší UFA pre jazyk some-register-on s n registrami má najmenej $2^n - 1$ stavov. Z toho vyplýva platnosť vzťahu $UFA <_P SUFA$. Model SUFA sme teda zaradili do predchádzajúcej hierarchie nasledovne:

$$DFA <_P UFA <_P SUFA \leq_P FNA$$

Posledný vzťah $SUFA \leq FNA$ je platný, pretože v SUFA je možné slovo $w \in \Sigma^*$ akceptovať najviac toľkými cestami, koľko má automat akceptačných stavov.

4.2 GSUFA

Ďalej uvádzame definíciu modelu, ktorý je zovšeobecnením modelu SUFA. Nový model nám pomôže porovnať popisnú zložitosť SUFA a modelov s viacerými počiatočnými stavmi.

Definícia 4.2.1. [16] *Zovšeobecnený štrukturálne jednoznačný konečný automat (GSUFA - generalized SUFA) je NFA $A = (K, \Sigma, \delta, I, F)$ taký, že pre každé dva stavy $p, q \in K$ a pre každé slovo $w \in \Sigma^*$ existuje najviac jedna cesta z p do q na slovo w.*

GSUFA s jedným akceptačným stavom už nie je UFA, pretože môžeme akceptovať slovo w z dvoch rôznych počiatočných stavov.

Vlastnosť, ktorú požadujeme je dosť silná. Nepožadujeme jednoznačnosť len na výpočty začínajúce v počiatočných stavoch, ale na všetky výpočty (narozdiel od SUFA). Preto je táto definícia nezávislá na výbere počiatočných stavov. Potrebujeme však ukázať, že každý SUFA spĺňa túto vlastnosť a teda, že SUFA $\subseteq$ GSUFA. Postupujme sporom. Nech $A \in SUFA$ ale nech $A \notin GSUFA$. Potom existujú také stavy p, q a slovo w, že z p do q vedú na w aspoň 2 cesty. Ak je však stav p dosiahnuteľný z počiatočného stavu q_0 na nejaké slovo u, potom vieme stav q dosiahnuť na uw z q_0 aspoň dvoma cestami, čo je v spore s tým, že A je SUFA.

Na druhej strane, vieme k danému GSUFA nájsť ekvivalentný SUFA nasledovne. Nech $A = (K, \Sigma, \delta, I, F)$ je GSUFA a $I = \{q_1, ..., q_k\}$. Vyrobíme k nezávislých kópií

A tak, že $A_i = (K, \Sigma, \delta, s_i, F)$ je SUFA pre $1 \leq i \leq k$. Pridáme nový počiatočný stav $q_0 \notin K$ a vyrobíme ε-prechody z q_0 do q_i. Tieto nahradíme priamymi prechodmi a dostanem tak SUFA, ktorý má $O(|A|^2)$ stavov. Dostávame teda

$$SUFA =_P GSUFA$$

Vieme tiež, že $SUFA \subseteq FNA$, keďže každé slovo môžeme akceptovať najviac toľkokrát, koľko je akceptačných stavov. Podobne $GSUFA \subseteq FNA$, lebo každé slovo môžeme akceptovať najviac $|I|.|F|$ spôsobmi, čo je $O(|A|^2)$ a to je vzhľadom na dĺžku slova konštanta.

$$UFA <_P SUFA =_P GSUFA \leq_P FNA$$

Otázka, či $(G)SUFA <_P FNA$ nebola doposiaľ zodpovedaná. V kapitole 6 ukzaujeme, že nie je možné aby $SUFA =_P FNA$.

4.3 MDFA

V tejto kapitole sa budeme zaoberať zovšeobecnením deterministických automatov. Dovolíme klasickému deterministickému automatu jedno nedeterministické rozhodnutie ale iba na začiatku výpočtu. Túto možnosť pridáme, ak dovolíme automatu mať viacero počiatočných stavov. Celá prechodová logika však ostáva deterministická. Uvedieme známe tvrdenia, ktoré porovnávajú tento model s ostatnými modelmi a tým ho zaradíme do vzniknutej hierarchie.

Definícia 4.3.1. *Deterministický automat s viacerými počiatočnými stavmi ($MDFA$ – Multiple entry finite automaton) je pätica $A = (K, \Sigma, \delta, I, F)$, kde K je konečná množina stavov, Σ je konečná vstupná abeceda, $I \subseteq K$ je množina počiatočných stavov, $F \subseteq K$ je množina akceptačných stavov a $\delta : K \times \Sigma \to K$ je prechodová funkcia.*

Zjavne $MDFA \subseteq FNA$, pretože pre každé slovo existuje toľko výpočtov, koľko je počiatočných stavov. To môže byť síce až $n = |A|$, ale vzhľadom na dĺžku vstupného slova, je to konštanta. Z hora teda vieme $MDFA$ zaradiť do našej hierarchie pod FNA.

Keďže $SUFA =_P GSUFA$ a $MDFA \subseteq GSUFA$, potom zjavne $MDFA \leq_P SUFA$. $SUFA$ sú ale pre niektoré jazyky v porovnaní s $MDFA$ úspornejšie. Na dôkaz tvrdenia $MDFA <_P SUFA$ [16] bol použitý jazyk some–register–on, pre ktorý MDFA potrebuje aspoň $2^n - 1$ stavov. Máme teda vzťah:

$$DFA \leq_P MDFA <_P SUFA$$

Na druhej strane, už pri definovaní modelu, ktorý zahŕňa iba istú podtriedu $MDFA$ [1], bolo ukázané, že $DFA <_P MDFA$. Teda existujú regulárne jazyky, pre ktoré už aj jedno nedeterministické rozhodnutie pomôže ušetriť exponenciálne veľa stavov. Spolu máme teda nasledujúce výsledky:

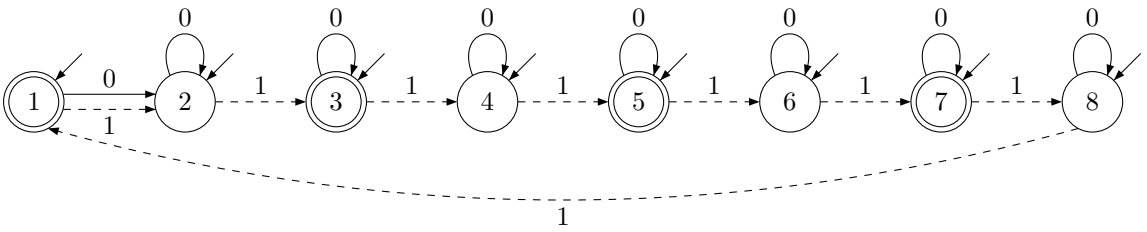
$$DFA <_P UFA <_P SUFA$$

$$DFA <_P MDFA <_P SUFA$$

Ponúka sa teda otázka, aký je vzťah medzi modelmi UFA a $MDFA$. Triviálne nevieme zaradiť tieto modely jeden pod druhý. $UFA \not\subseteq MDFA$, keďže v UFA nemáme zaručenú deterministickú δ funkciu. Podobne $MDFA \not\subseteq UFA$, keďže z rôznych počiatočných stavov môžu pre jedno slovo existovať rôzne akceptačné výpočty. V zostávajúcich častiach tejto kapitoly popíšeme jazyky, pomocou ktorých boli triedy MDFA a UFA oddelené.

4.3.1 MDFA úspornejší než UFA

Na obrázku 4.1 vidíme prechodovú logiku automatu. Vybraním všetkých stavov za počiatočné a označením ľubovoľného stavu za akceptačný dostaneme $MDFA$ [23], o ktorom bolo ukázané, že ekvivalentný DFA má $2^n - 1$ stavov, kde n je počet stavov $MDFA$.



Obr. 4.1: $MDFA$ $\mathcal{M}_8$

Leung [15] poznemil tento model. Počiatočné stavy ponechal a nesusedné stavy vybral za akceptačné a dostal tak automat M_n. Použitím tvrdenia pre dolný odhad veľkosti UFA (tvrdenie 3.3.1 [22]) Leung ukázal, že každý UFA pre jazyk $L(M_n)$ potrebuje $2^n - 1$ stavov. A teda $UFA \neq_P MDFA$.

4.3.2 UFA úspornejší než MDFA

Z predchádzajúcich častí vidíme, že pre jazyk some–register–on UFA aj MDFA potrebujú rovnako veľa stavov. Ak však tento jazyk trochu upravíme, bude už pre UFA akceptovateľný s polynomiálnym počtom stavov (vzhľadom na počet registrov), zatiaľ čo pre $MDFA$ ostane rovnako "ťažký".

Jazyk some–register–on upravíme nasledovne [16]:

Na konci vstupu sa budeme pýtať, či je v nejakom konkrétnom registri hodnota 1. Ak áno, slovo je z jazyka, inak nie je. Inštrukciu otázky (query instruction) pre register i budeme označovať Q_i. Vstup teda bude vyzerať napríklad nasledovne $C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2}Q_3$. Toto slovo je z jazyka, keďže register 3 má po vykonaní hodnotu 1. Takéto slovo $C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2}Q_1$ však do jazyka nepatrí (v registri 1 je hodnota 0).

Na akceptovanie tohoto jazyka upravíme aj automat A pre jazyk some–register–on. Pridáme nový stav f, ktorý sa stane jediným akceptačným stavom. Do každého

stavu i pridáme prechod na symbol Q_i do stavu f, z ktorého už nebudú viesť žiadne prechody. Rozšírime aj abecedu a to o množinu $\{Q_i \mid 1 \leq i \leq n\}$. Dostaneme tak $(n+1)$–stavový $SUFA$ s jedným akceptačným stavom, čo znamená, že je to tiež UFA.

Z toho, že $MDFA$ pre jazyk some–register–on potrebuje 2^n-1 stavov, sa dá ukázať [16], že $MDFA$ potrebuje rovnako veľa stavov aj pre modifikáciu tohoto jazyka.

4.3.3 Zhrnutie

Modely UFA a MDFA nie sú hierarchicky usporiadateľné. Musíme pre ne vyrobiť dve oddelené hierarchie.

$$DFA <_P UFA <_P SUFA =_P GSUGA \leq_P FNA <_P LNA <_P PNA <_P NFA$$

$$DFA <_P MDFA <_P SUFA =_P GSUFA \leq_P FNA <_P LNA <_P PNA <_P NFA$$

Tiež je zrejmé, že neplatí $UFA \leq_P MDFA$ ani $MDFA \leq_P UFA$ a teda platí

$$UFA \neq_P MDFA$$

O tieto vzťahy sa budeme opierať v ďalšej časti tejto práce. Vytvorenie tejto hierarchie je dôsledkom známych vzťahov medzi spomenutými modelmi.

V hierarchii sme uviedli vzťah $GSUGA \leq_P FNA$. Dôvod je ten, že otázka či $GSUGA =_P FNA$ nebola doposiaľ vyriešená.

Kapitola 5

Zložitosť operácií

Je známe, že regulárne jazyky sú uzavreté na zjednotenie, zreťazenie, iteráciu, komplement, prienik, reverz, homomorfizmus aj inverzný homomorfizmus. V praxi sa často stretávame s problémom, ktorého riešenie vyžaduje vykonanie nejakej operácie s regulárnymi jazykmi. Pod vykonaním (realizáciou) operácie rozumieme nájdenie príslušného jazyka. Napríklad vykonanie zjednotenia jazykov L_1 a L_2, znamená nájdenie jazyka $L = L_1 \cup L_2$. V tejto práci používame na definovanie regulárnych jazykov konečné automaty. Teda vykonanie zjednotenia znamená nájdenie automatu A pre jazyk L, pričom predpokladáme, že poznáme automaty A_1, A_2 pre L_1, L_2.

Z predchádzajúcich kapitol už poznáme viacero podtried konečných automatov, ktorými reprezentujeme regulárne jazyky. Budeme sa teraz zaoberať tým, ako efektívne vieme realizovať nejakú operáciu pomocou automatu z konkrétnej triedy.

Aby bolo možné porovnávať naše výsledky ohľadom realizácie operácii, je nutné zadefinovať, kedy je vykonanie operácie ešte efektívne a ako budeme merať zložitosť operácií na jednotlivých triedach. Potrebným definíciám a označeniam sa venujeme v časti 5.1. V ďalších častiach sa venujeme analýze jednotlivých operácii na modeloch uvedeným v predchádzajúcich častiach. V časti 5.8 uvádzame prehľadnú tabuľku, ktorá obsahuje naše výsledky.

5.1 Definície

Označenie. *Nech L je regulárny jazyk a op je operácia, na ktorú sú regulárne jazyky uzavreté. Symbolom $op(L)$ budeme označovať jazyk, ktorý vznikne z jazyka L po vykonaní operácie op. Pre binárne operácie budeme používať označenie $op(L_1, L_2)$.*

Keď budeme hovoriť o konkrétnej operácií, budeme používať štandardné označenie. Napríklad L^*, $L_1.L_2$ a podobne.

Definícia 5.1.1. *Na realizáciu unárnej, resp. binárnej, operácie op pri využití automatov z triedy C postačuje $f(n)$, resp. $f(n, m)$ stavov, ak pre ľubovoľný regulárny jazyk L platí jedna z nasledujúcich možností:*

1. *op je unárna operácia a ku každému $A \in C$ pre jazyk L, existuje $A' \in C$ pre jazyk $op(L)$ tak, že $|A'| \leq f(|A|)$*

2. *op je binárna operácia a ku každému* $A_1, A_2 \in C$ *pre jazyk* L_1 *resp.* L_2*, existuje* $A' \in C$ *pre jazyk* $op(L1, L2)$ *tak, že* $|A'| \leq f(|A_1|, |A_2|)$

Štandardný spôsob, ako ukázať, že na realizovanie nejakej operácie pomocou automatov z vybranej triedy postačuje niekoľko stavov je ukázať, ako nájsť automat pre výsledný jazyk. Tiež je nutné ukázať, akým spôsobom závisí počet stavov vzniknutého automatu od vstupných automatov.

Definícia 5.1.2. *Ak pre každé* $i \in \mathbb{N}$ *existujú jazyky* $L(A_n), L(B_m)$, $n, m > i$ *a* $A_n, B_m \in C$ *tak, že na akceptovanie jazyka* $op(L(A_n), L(B_m))$ *pomocou automatu z triedy* C *je potrebných aspoň* $f(|A_n|, |B_m|)$ *stavov, potom hovoríme, že na realizáciu operácie op pomocou automatov z triedy* C *je nutných* $f(n, m)$ *stavov.*

V prípade unárnej operácie by stačilo nájsť pre každé $i \in \mathbb{N}$ jeden jazyk. Jediný rozdiel oproti definícii 5.1.2 je v tom, že by sme nepotrebovali dvojice jazykov.

Definícia 5.1.3. *Zložitosť operácie op pri využití automatov z triedy* C *budeme chápať ako počet stavov, ktorý je nutný a postačujúci na realizáciu operácie op pri využití automatov z triedy* C*. Tento počet stavov budeme označovať ako* $STATE(C, op)$*.*

Definícia 5.1.4. *Trieda automatov* C *efektívne realizuje operáciu op, ak* $STATE(C, op) = poly(n)$*, resp.* $STATE(C, op) = poly(n, m)$*.*

Pre triedy DFA a NFA bola otázka zložitosti operácií už vyriešená [25, 24, 13, 4]. Medzi skúmané operácie patria zjednotenie, prienik, komplement, reverz, zreťazenie, iterácia a kladná iterácia. Ak je možné operáciu realizovať podľa vybranej triedy (DFA, NFA), potom existuje konštrukcia, ktorá je návodom na jej realizáciu. Z konštrukcie vidíme, aký je postačujúci počet stavov na realizáciu operácie pomocou vybranej triedy automatov. Pre tieto konštrukcie je ukázané, že sú optimálne, t.j. neexistuje všeobecná konštrukcia, na ktorú postačuje menej stavov. Optimalita sa spravidla dokazuje tak, že sa nájde nekonečná postupnosť jazykov L_n (v prípade binárnej operácie treba dve postupnosti), ktorá spĺňa podmienku: pre L_n existuje A_n z danej triedy, ktorý má n stavov, ale každý automat A_n z danej triedy pre $op(L_n)$ potrebuje aspoň toľko stavov, koľko je ich použitých v konštrukcii.

Pre operácie, ktoré nie je možné efektívne realizovať pomocou NFA alebo DFA sú známe postupnosti jazykov, ktoré dokazujú, že na realizáciu operácie na danom jazyku daná trieda potrebuje exponenciálny počet stavov.

V nasledujúcich častiach sa budeme postupne venovať každej z vyššie uvedených operácii. Pre každú operáciu popíšeme buď konštrukciu pre NFA (niekedy aj pre DFA) alebo jazyky, ktoré dokazujú, že nie je možné nájsť polynomiálnu konštrukciu. Naša snaha bude smerovať k tomu, aby sme priniesli podobné výsledky aj pre ostatné triedy automatov, ktoré sme popísali v predchádzajúcich častiach. Pre jednotlivé triedy budeme teda hľadať buď polynomiálne konštrukcie pre realizáciu skúmaných operácií, alebo sa pokúsime nájsť jazyky, ktoré dokazujú, že existencia polynomiálnej konštrukcie je vylúčená.

Aby bolo možné získané výsledky porovnať, budeme predpokladať nasledujúce. Pri realizácii operácií pomocou NFA alebo pomocou tried definovaných na základe množstva nejednoznačnosti (UFA, FNA, LNA, PNA) predpokladáme, že automaty pre dané jazyky majú iba jeden počiatočný stav. Od výsledného automatu požadujeme rovnakú vlastnosť. Poznamenajme, že tento predpoklad je jednoduché splniť. Ku každému UFA, FNA, LNA, PNA aj NFA existuje ekvivalentný automat s jedným počiatočným stavom, ktorý je z rovnakej triedy. Na zostrojenie takéhoto automatu postačuje pridanie jedného nového (počiatočného) stavu. Pre triedy definované na základe štrukturálnej vlastnosti (GSUFA, MDFA) toto požadovať nebudeme, pretože by to viedlo k splynutiu triedy s inou triedou (SUFA, DFA).

Budeme predpokladať, že máme automaty $A = (K_A, \Sigma_A, \delta_A, I_A, F_A)$ a $B = (K_B, \Sigma_B, \delta_B, I_B, F_B)$. V prípade tried UFA, FNA, LNA, PNA a SUFA budeme označovať $I_A = \{q_{0A}\}$ a $I_B = \{q_{0B}\}$. Budeme sa snažiť nájsť automat C tak, aby $L(C) = op(L(A), L(B))$. Ak pôjde o unárnu operáciu, C bude spĺňať $L(C) = op(L(A))$. Hľadaný automat C sa bude skladať z týchto kompomentov $C = (K_C, \Sigma_C, \delta_C, I_C, F_C)$. Pre C musí platiť, že je z rovnakej triedy ako A, B, preto ak A, B majú jeden počiatočný stav, bude $I_C = \{q_{0C}\}$. Naša snaha bude smerovať k tomu, aby stavová zložitosť automatu C bola čo najmenšia.

5.2 Zjednotenie

Máme automaty A, B. Chceme nájsť taký automat C, pre ktorý $L(C) = L(A) \cup L(B)$.

V tejto časti najprv popisujeme známu konštrukciu, ako realizovať zjednotenie pomocou NFA a DFA. Pre modely FNA, LNA, PNA, SUFA, GSUFA a MDFA sme ukázali, že možno použiť rovnakú (alebo podobnú) konštrukciu na realizáciu zjednotenia pomocou NFA.

Zjednotenie pomocou NFA a DFA

Táto časť práce sa venuje dvom konštrukciám pre realizáciu zjednotenia. Najprv uvedieme konštrukciu pre NFA a následne popíšeme, zložitostne náročnejšiu, konštrukciu na realizáciu zjednotenia pomocou DFA.

Tvrdenie 5.2.1. [4] $STATE(NFA, \cup) = n + m + 1$.

Dôkaz. Nerovnosť $STATE(NFA, \cup) \leq n + m + 1$ vyplýva z konštrukcie automatu C, ktorý akceptuje zjednotenie jazykov $L(A)$ a $L(B)$. Idea fungovania automatu C je v tom, že sa nedeterministicky rozhodne, ktorý z automatov A, B bude na vstupnom slove simulovať. Zoberieme teda všetky stavy (aj prechody) automatu A aj automatu B. Pridáme nový počiatočný stav (kvôli podmienke, že C musí mať iba jeden počiatočný stav) a z neho budeme viesť prechody do nasledovníkov q_{0A} a q_{0B} na príslušné znaky. Preto $|C| = |A| + |B| + 1$.

Nerovnosť $STATE(NFA, \cup) \geq n + m + 1$ platí pre postupnosť jazykov $L(A_n) = \{a^n\}^*$ a $L(B_m) = \{b^m\}^*$. Poznamenajme, že A_n aj B_m sú DFA, a že $|A_n| = n$ a

$|B_m| = m$. Bolo dokázané [4], že každý NFA s jedným počiatočným stavom pre $L(A_n) \cup L(B_m)$ potrebuje aspoň $n + m + 1$ stavov. □

V prípade realizácie zjednotenia deterministickými automatmi sa C nemôže rozhodnúť tak, ako sme to videli v konštrukcii pre NFA. Preto na realizáciu zjednotenia treba viac stavov.

Tvrdenie 5.2.2. [25] $STATE(DFA, \cup) = nm$.

Dôkaz. Nerovnosť $STATE(DFA, \cup) \leq nm$ vyplýva z konštrukcie automatu C. Automat bude simulovať A aj B naraz, jeho stavy budú teda dvojice. Bude sa nachádzať akoby paralelne v dvoch stavoch, v jednom pre A a v druhom pre B. C vždy prečíta celé slovo, pretože A aj B sú úplné. Na konci výpočtu C akceptuje práve vtedy, keď aspoň jeden z A, B akceptoval. Stavy automatu C budú kartézskym súčinom stavov automatov A a B. Preto $|C| = |A|.|B|$.

Nerovnosť $STATE(DFA, \cup) \geq nm$ je dokázaná pre postupnosť automatov A_n a B_m takých, že $L(A_n) = \{w \in \{a, b\}^* \mid \#_a(w) \equiv 0 \bmod n\}$ a $L(B_m) = \{w \in \{a, b\}^* \mid \#_b(w) \equiv 0 \bmod m\}$. Každý DFA pre $L(A_n) \cup L(B_m)$ potrebuje aspoň nm stavov. □

Zjednotenie pomocou UFA

Pre triedy UFA sa nám nepodarilo nájsť polynomiálnu konštrukciu na reazlizáciu zjednotenia. V tvrdení 5.2.4 dokazujeme, že UFA nemôžu realizovať zjednotenie tak efektívne ako NFA. Nepodarilo sa nám však dokázať exponenciálnu dolnú hranicu. V tejto časti popíšeme, prečo nie je možné realizovať zjednotenie pomocou UFA rovnako ako pre NFA alebo DFA.

Ak by sme na realizáciu zjednotenia pomocou UFA chceli použiť konštrukciu pre realizáciu pomocou NFA, narazili by sme na nasledujúci problém. Ak jazyky $L(A)$ a $L(B)$ majú neprázdny prienik, potom pre slová z toho prieniku existujú v automate C dva akceptačné výpočty, jeden v automate A a druhý v automate B. Automat C by bol teda 2-FNA. Je stále otvorené či UFA $=_P$ 2-FNA a preto nemôžeme zaručiť, že pre takýto automat vždy existuje ekvivalentný UFA s polynomiálnym počtom stavov.

Konštrukciu pre DFA nemôžeme použiť pre UFA, pretože ňou skonštruovaný automat C tiež nemusí byť UFA. Nech A má jediný akceptačný stav q, ktorým akceptuje slovo w, jedným spôsobom. Nech existujú stavy $q_1, q_2 \in B$ tak, že $q_1, q_2 \in \delta_B(q_{0B}, w)$. Stačí ak aspoň jeden zo stavov q_1, q_2 nie je akceptačný a B je UFA. Potom však C môže akceptovať w prechodom do dvoch rôznych akceptačných stavov, do $[q, q_1]$ a do $[q, q_2]$. Táto konštrukcia funguje, keď máme zaručené, že pre každé slovo z $L(A)$ existuje v B najviac jeden výpočet a naopak.

Nasledujúce tvrdenie prináša horné ohraničenie zložitosti realizácie zjednotenia pri využití automatov z triedy UFA.

Tvrdenie 5.2.3. $STATE(UFA, \cup) \leq 2^{n+m}$.

Dôkaz. Použijeme konštrukciu pre NFA z [4]. Ňou skonštruovaný automat C však nemusí byť UFA. Ak $L(A) \cap L(B) \neq \emptyset$, potom existujú slová s dvoma rôznymi akceptačnými výpočtami v C. Automat C je teda $2 - FNA$. K vzniknutému automatu môžeme nájsť UFA podmnožinovou konštrukciou. Toto však dáva exponenciálny nárast počtu stavov. □

Tvrdenie 5.2.4. *Neexistuje $k \in \mathbb{N}$ také, že $STATE(UFA, \cup) = O(n + m^k)$.*

Dôkaz. Uvažujme jazyk *some–register–on* [15]. V časti 4.1 sme popísali automat A pre tento jazyk. Pripomeňme, že A je SUFA a pre akceptovanie jazyka nad n registrami má n stavov. V tomto automate sú všetky stavy akceptačné.

Označmne symbolom A_i taký automat, ktorý získame z automatu A tak, že iba i –ty stav označíme ako akceptačný. Je zrejmé, že $|A_i| = n$ a tiež, že A_i je UFA. Z popisu automatu A vidíme, že $w \in L(A_i)$ práve vtedy, keď po vykonaní inštrukcií w ostane v i–tom registri hodnota 1. Automat A_i akcpetuje jazyk $L(A_i) = L_i = i\text{th} - \text{register} - \text{on}$.

Teda zjednotenie $L_1 \cup \ldots \cup L_n = L$ obsahuje práve všetky slová z jazyka *some–register–on*. Bolo ukázané [15], že každý UFA pre tento jazyk potrebuje $2^n - 1$ stavov. Ak by existovala taká konštanta $k \in \mathbb{N}$, že na realizáciu zjednotenia pomocou UFA postačuje $O(n + m^k)$ stavov, potom by pre jazyk L existoval UFA s $O(n + (n-1).n^k)$ stavmi. A to je spor. □

Zjednotenie pomocou MDFA, GSUFA a SUFA

V nasledujúcich dvoch tvrdeniach ukážeme, že triedy MDFA, GSUFA a SUFA efektívne realizujú zjednotenie.

Tvrdenie 5.2.5. $STATE(MDFA, \cup) = STATE(GSUFA, \cup) = n + m.$

Dôkaz. Zjednotenie pomocou MDFA alebo GSUFA budeme realizovať podobne ako pomocou NFA. Automat C bude obsahovať všetky stavy aj prechody automatov A, B. Počiatočné stavy automatu C bude tvoriť zjednotenie počiatočných stavov automatov A, B. Ak sú A, B GSUFA, potom aj C je GSUFA, pretože prechodová logika ostala nezmenená. Rovnako ak A, B sú MDFA, potom aj C je MDFA.

Keďže nepožadujeme od MDFA ani GSUFA aby mali jeden počiatočný stav, nemuseli sme pridávať nový počiatočný stav. Na realizáciu zjednotenia preto postačuje $|A| + |B|$ stavov, t.j. $STATE(MDFA, \cup), STATE(GSUFA, \cup) \leq n + m$.

Hranicu $m + n$ stavov dosahuje zjednotenie jazykov $L(A_n) = \{a^n\}^*$ a $L(B_m) = \{b^m\}^*$. Každý automat $C \in NFA$, $L(C) = L(A) \cup L(B)$, potrebuje najmenej $n - 1$ neakceptačných stavov, aby mohol zamietnuť vstupy a^i, $1 \leq i \leq n - 1$ a akceptovať slovo a^m. Podobne C potrebuje najmenej $m-1$ stavov, aby mohol zamietnuť vstupy b^i, $1 \leq i \leq m - 1$. Všetky tieto stavy musia byť užitočné. Pridáme ešte jeden akceptačný stav pre slová tvaru a^{in} a ďalší pre slová b^{im}.

Stavy dosiahnuteľné slovami a^i musia byť rôzne od stavov dosiahnuteľných slovami b^i, inak by C akceptoval slová tvaru $a^i b^j$ alebo $b^i a^j$. C musí mať preto $n + m$ rôznych užitočných stavov a teda $STATE(MDFA, \cup), STATE(GSUFA, \cup) \geq n + m$. □

V prípade realizácie pomocou SUFA nový počiatočný stav neušetríme, ale konštrukciu pre NFA môžeme použiť bez zmeny.

Tvrdenie 5.2.6. $STATE(SUFA, \cup) = n + m + 1$.

Dôkaz. Na dôkaz hornej hranice použijeme konštrukciu pre NFA. Musíme však dokázať, že automat C vznikuntý touto konštrukciou je SUFA ak A, B sú SUFA. Postupujeme sporom. Nech C nie je SUFA. Existuje teda stav $q \in K_C$ a slovo w tak, že z q_{0C} vedú do stavu q dve cesty na slovo w. Do stavu q_{0C} nevchádzajú žiadne prechody, preto $q \neq q_{0C}$. Bez ujmy na všeobecnosti predpokladajme, že q patrí do K_A. Potom podľa konštrukcie vedú dve cesty z q_{0A} do q na slovo w, čo je spor s predpokladom, že A je SUFA. Tým sme dokázali, že $STATE(SUFA, \cup) \leq n + m + 1$.

SUFA dosahujú túto hranicu pre rovnaké jazyky ako NFA. Pre jazyky $L(A_n)$ a $L(B_m)$ z dôkazu tvrdenia 5.2 existuje n – stavový, resp. m – stavový DFA a teda aj SUFA. [4] dokazujú, že každý NFA (s jedným počiatočným stavom) pre $L(A_n) \cup L(B_m)$ potrebuje $n + m + 1$ stavov a teda aj každý SUFA potrebuje toľko stavov. Preto platí $STATE(SUFA, \cup) \geq n + m + 1$. □

Zjednotenie pomocou FNA, LNA a PNA

Zjednotenie je možné efektívne realizovať aj pomocou tried s konečnou a polynomiálnou nejednoznačnosťou. Hovorí o tom nasledujúce tvrdenie.

Tvrdenie 5.2.7. $STATE(FNA, \cup) = STATE(LNA, \cup) = STATE(PNA, \cup) = n + m + 1$.

Dôkaz. Na dôkaz hornej hranice ukážeme, že zjednotenie pomocou FNA, LNA alebo PNA je možné realizovať rovnako ako pomocou NFA. Pre C bude platiť $ambig(C) \leq ambig(A) + ambig(B)$. Rovnosť nastáva, ak existuje slovo z $L(A) \cap L(B)$ s maximálnou nejednoznačnosťou v oboch automatoch.

Ak $A, B \in FNA$ potom $ambig(A)$ aj $ambig(B)$ sú konštanty a teda aj $ambig(C)$ je konštanta.

Ak $A, B \in LNA$ potom $ambig(A)$ aj $ambig(B)$ sú lineárne vzhľadom na dĺžku slova a teda aj $ambig(C)$ je lineárna vzhľadom na dĺžku slova.

Ak $A, B \in PNA$ potom $ambig(A)$ aj $ambig(B)$ sú polynomiálne vzhľadom na dĺžku slova a teda aj $ambig(C)$ je polynomiálna vzhľadom na dĺžku slova.

Dolná hranica, $STATE(FNA, \cup), STATE(LNA, \cup), STATE(PNA, \cup) \geq n + m + 1$, je dosiahnutá pre jazyky $L(A_n)$ a $L(B_m)$ z dôkazu tvrdenia 5.2 pre NFA. Pre $L(A_n)$, $L(B_m)$ existuje n – stavový, resp. m – stavový DFA a teda aj FNA, LNA a PNA. [4] dokazujú, že každý NFA (s jedným počiatočným stavom) pre $L(A_n) \cup L(B_m)$ potrebuje $n + m + 1$ stavov a teda aj každý FNA, LNA alebo PNA potrebuje toľko stavov. □

5.3 Prienik

Máme automaty A, B. Chceme nájsť taký automat C, pre ktorý $L(C) = L(A) \cap L(B)$.

V tejto časti najprv popisujeme známu konštrukciu, ako realizovať prienik pomocou NFA a DFA. Pre modely UFA, FNA, PNA, SUFA, GSUFA a MDFA sme ukázali, že možno použiť rovnakú konštrukciu na realizáciu prieniku pomocou NFA.

Prienik pomocou NFA a DFA

Realizácia prieniku pomocou deterministických a nedeterministických automatov už bola vyriešená [4, 24]. V tejto časti popíšeme optimálnu konštrukciu na realizáciu zjednotenia pomocou týchto modelov a ukážeme ako množstvo nedeterminizmu výsledného automatu závisí od množstva nedeterminizmu pôvodných automatov.

Tvrdenie 5.3.1. [4] $STATE(NFA, \cap) = nm$.

Dôkaz. Hlavné myšlienky dôkazu tohoto tvrdenia v [4] sú nasledujúce.

Pre dôkaz hornej hranice sa použije nasledujúca konštrukcia. Automat C bude simulovať automaty A, B. Vstupné slovo akceptuje práve vtedy, keď A aj B akceptovali. C bude teda obsahovať kartézsky súčin stavov automatov A, B. Preto $|C| = |A|.|B|$ a teda $STATE(NFA, \cap) \leq nm$.

K dôkazu dolnej hranice boli použité postupnosti jazykov $L(A_n) = \{w \in \{a, b\}^* \mid \#_a(w) \equiv 0 \ (mod\ n)\}$ a $L(B_m) = \{w \in \{a, b\}^* \mid \#_b(w) \equiv 0 \ (mod\ m)\}$. Poznamenajme, že A_n aj B_m sú DFA, a že $|A_n| = n$ a $|B_m| = m$. Je ukázané, že každý NFA s jedným počiatočným stavom pre $L(A_n) \cap L(B_m)$ potrebuje aspoň nm stavov, t.j. $STATE(NFA, \cap) \geq nm$. □

Z tvrdenia 5.3.1 vidíme, že existuje konštrukcia pre prienik jazykov, ktorá vyžaduje maximálne polynomiálny nárast stavov. Mohutnosť výsledného automatu je súčinom mohutností vstupných automatov.

Dokázali sme nasledujúce tvrdenie, pomocou ktorého je možné dokázať, že nejednoznačnosť výsledného automatu je súčinom nejednoznačností vstupných automatov.

Tvrdenie 5.3.2. *Nech automat C vznikne z automatov A, B konštrukciou z tvrdenia 5.3.1. Potom* $\forall w \in (\Sigma_A \cup \Sigma_B)^*, \forall q_A, q'_A \in A, \forall q_B, q'_B \in B$:

$$|\delta_C([q_A, q_B], w, [q'_A, q'_B])| = |\delta_A(q_A, w, q'_A)|.|\delta_B(q_B, w, q'_B)|$$

Dôkaz. Budeme postupovať indukciou od dĺžky slova w.

1. Keďže uvažuje automaty bez ε – prechodov, najmenšie w má jeden znak. Z konštrukcie vyplýva, že ak $|\delta_A(q_A, w, q'_A)|, |\delta_B(q_B, w, q'_B)| = 1$ tak potom aj $|\delta_C([q_A, q_B], w, [q'_A, q'_B])| = 1$

2. Nech pre všetky slová u, $|u| \leq n$, platí tvrdenie. Uvažujeme slovo w dĺžky $n+1$. Symbolom u označme prefix slova w dĺžky n. Symbolom a označme také písmeno zo $\Sigma_A \cup \Sigma_B$, že $ua = w$.

   Uvažujme množinu stavov v automate A, ktoré sú dosiahnuteľné zo stavu q_A na slovo u a vedie z nich prechod do stavu q'_A na znak a. Túto množinu označíme $P_A = \delta(q_A, u) \cap \delta^{-1}(q'_A, a)$. Nájdeme analogickú množinu aj v automate B,

$P_B = \delta(q_B, u) \cap \delta^{-1}(q'_B, a)$. Takáto množina existuje aj v automate C, $P_C = \delta([q_A, q_B], u) \cap \delta^{-1}([q'_A, q'_B], a)$. Z konštrukcie vyplýva, že $P_A \times P_B = P_C$, inak (ak sú všetky stavy užitočné) by nemohlo platiť $L(A) \cap L(B) = L(C)$.

Pomocou množiny P_C vyjadríme $|\delta_C([q_A, q_B], w, [q'_A, q'_B])|$ nasledovne

$$|\delta_C([q_A, q_B], w, [q'_A, q'_B])| = \sum_{[p_A, p_B] \in P_C} |\delta_C([q_A, q_B], u, [p_A, p_B])|$$

$$\stackrel{IP}{=} \sum_{[p_A, p_B] \in P_C} |\delta_A(q_A, u, p_A])|.|\delta_B(q_B, u, p_B])|$$

$$= \sum_{p_B \in P_B} |\delta_B(q_B, u, p_B])| \sum_{p_A \in P_A} |\delta_A(q_A, u, p_A])| = |\delta_A(q_A, w, q'_A)|.|\delta_B(q_B, w, q'_B)|$$

Pri poslednom kroku odvodenia sme použili rovnakú myšlienku ako pri prvom kroku. Spočítali sme všetky cesty pre slovo u, ktoré sú predĺžiteľné o 1 krok na písmeno a tým sme dostali počet ciest pre slovo $ua = w$.

□

Tvrdenie 5.3.3. [24] $STATE(DFA, \cap) = nm$.

Konštrukcia použitá na dôkaz tvrdenia 5.3.3 je zhodná s konštrukciou pre NFA. Na dôkaz dolnej hranice je tiež možné použiť jazyky z dôkazu pre NFA.

Prienik pomocou UFA, FNA a PNA

Dokázali sme, že triedy UFA, FNA a PNA efektívne realizujú prienik. Hovorí o tom nasledujúce tvrdenie.

Tvrdenie 5.3.4. $STATE(UFA, \cap) = STATE(FNA, \cap) = STATE(PNA, \cap) = nm$.

Dôkaz. Dokážeme, že konštrukciu pre NFA je možné použiť pre všetky spomenuté triedy. Pre každú triedu T z tvrdenia ukážeme, že ak $A, B \in T$, potom $C \in T$, pričom C vznikol pomocou konštrukcie pre NFA.

Ak budeme v tvrdení 5.3.2 uvažovať iba $q'_A \in F_A$, $q'_B \in F_B$ a $q_A = q_{0A}$, $q_B = q_{0B}$ dostaneme ako dôsledok nasledujúcu rovnosť.

$$ambig_C(n) = ambig_A(n).ambig_B(n)$$

Ak $A, B \in UFA$ potom $ambig_A(n), ambig_B(n) = 1$ a teda aj $ambig_C(n) = 1$.

Ak $A, B \in FNA$ potom $ambig_A(n)$ aj $ambig_B(n)$ sú konštanty a teda aj $ambig_C(n)$ je konštanta.

Ak $A, B \in PNA$ potom $ambig_A(n)$ aj $ambig_B(n)$ sú polynomiálne vzhľadom na dĺžku slova a teda aj $ambig_C(n)$ je polynomiálna vzhľadom na dĺžku slova.

Môžeme teda tvrdiť, že $STATE(UFA, \cap), STATE(FNA, \cap), STATE(PNA, \cap) \leq nm$

Dolná hranica, $STATE(UFA,\cap), STATE(FNA,\cap), STATE(PNA,\cap) \geq nm$, je dosiahnutá pre jazyky $L(A_n)$ a $L(B_m)$ z dôkazu tvrdenia 5.3.1 pre NFA. Pre $L(A_n)$, $L(B_m)$ existuje n – stavový, resp. m – stavový DFA a teda aj UFA, FNA a PNA. [4] dokazujú, že každý NFA s jedným počiatočným stavom pre $L(A_n) \cap L(B_m)$ potrebuje nm stavov a teda aj každý UFA, FNA alebo PNA potrebuje toľko stavov. □

Prienik pomocou LNA

Ako vidíme z tvrdenia 5.3.2, priamočiare použitie konštrukcie pre NFA nevedie k dôkazu, že LNA efektívne realizuje prienik. Automat C by bol 2 - PNA. Je dokázané [7], že $LNA <_P 2 - PNA$.

Na dôkaz toho, že LNA nemôžu efektívne realizovať prienik navrhujeme použiť nasledujúce jazyky.

$$L(A) = \{w_1, ..., w_m \mid \exists i \text{ párne} : w_i \in L\}$$
$$L(B) = \{w_1, ..., w_m \mid \exists i \text{ nepárne} : w_i \in L\}$$

kde $L = (L_r)^t$ z čati 3.3.

Pre tieto jazyky viem nájsť LNA A a B, $|A| = 8t + N_L, |B| = 6t + N_L$. Položme $L(C) = L(A) \cap L(B)$. Je zrejmé, že $L(C)$ obsahuje iba také slová, ktoré obsahujú dve rôzne podslová z jazyka L. Neplatí, že $L(C) = \exists_2(L)$, ale $\exists_2(L) = L(C) \cup L' \cup L''$, kde

$$L' = \{w_1, ..., w_m \mid \exists i \neq j \text{ párne} : w_i, w_j \in L\}$$
$$L'' = \{w_1, ..., w_m \mid \exists i \neq j \text{ nepárne} : w_i, w_j \in L\}$$

Myslíme si, že jazyky $L(C), L', L''$ a $\exists_2(L)$ sú všetky rovnako ťažké pre LNA. Na dôkaz toho, že pre jazyk $\exists_2(L)$ neexistuje LNA, ktorý má $poly(r)$ stavov boli použité výsledky z komunikačnej zložitosti. Preto dôkaz toho, že ani pre jazyk $L(C)$ neexistuje LNA s $poly(r)$ stavmi je nad rámec tejto práce.

Prienik pomocou SUFA

Tvrdenie 5.3.5. $STATE(SUFA,\cap) = nm$.

Dôkaz. Dokážeme, že prienik pomocou SUFA je možné realizovať rovnako ako pomocu NFA. Na to je nutné ukázať, že ak $A, B \in SUFA$, potom $C \in SUFA$, pričom C vznikol pomocou konštrukcie pre NFA.

Nech $A, B \in SUFA$ a nech $C \notin SUFA$. Teda existuje slovo $w \in L(C)$ a stav $[q'_A, q'_B] \in C$ tak, že z $[q_{0A}, q_{0B}]$ vedú aspoň dve cesty do stavu $[q'_A, q'_B]$ na slovo w. Podľa tvrdenia 5.3.2 by museli v automate A alebo B viesť aspoň dve cesty z q_{0A} do q'_A, resp. z q_{0B} do q'_B, čo je spor s tým, že A aj B sú SUFA. Tým sme dokázali $STATE(SUFA,\cap) \leq nm$.

SUFA dosahujú hranicu mn stavov pre rovnaké jazyky ako NFA. Pre jazyky $L(A_n)$ a $L(B_m)$ z dôkazu tvrdenia 5.3.1 existuje n – stavový, resp. m – stavový DFA a teda aj SUFA. [4] dokazujú, že každý NFA s jedným počiatočným stavom pre $L(A_n) \cap L(B_m)$ potrebuje nm stavov a teda aj každý SUFA potrebuje toľko stavov. Preto platí $STATE(SUFA,\cap) \geq nm$. □

Prienik pomocou MDFA a GSUFA

Tvrdenie 5.3.6. $STATE(MDFA, \cap) = STATE(GSUFA, \cap) = nm.$

Dôkaz. Na dôkaz nerovnosti $STATE(MDFA, \cap), STATE(GSUFA, \cap) \leq nm$ použijeme konštrukciu pre NFA. Malá zmena bude v tom, že počiatočné stavy automatu C budú kartézskym súčinom počiatočných stavov automatov A a B. Pre oba modely ukážeme, že konštrukcia zachováva príslušnosť do triedy automatov.

Nech $A, B \in GSUFA$ a nech $C \notin GSUFA$. Teda existuje slovo $w \in L(C)$ a stav $[q'_A, q'_B] \in C$ tak, že z nejakého $[q_A, q_B] \in C$ vedú aspoň dve cesty do stavu $[q'_A, q'_B]$ na slovo w. Podľa tvrdenia 5.3.2 by museli v automate A alebo B viesť aspoň dve cesty z q_A do q'_A, resp. z q_B do q'_B, čo je spor s tým, že A aj B sú GSUFA.

Nech $A, B \in MDFA$ a nech $C \notin MDFA$. Teda existuje stav $[q_A, q_B] \in C$ a písmeno $a \in \Sigma_C$ tak, že pre nejaké dva stavy $[q'_A, q'_B], [q''_A, q''_B] \in C$ platí: $[q'_A, q'_B], [q''_A, q''_B] \in \delta_C([q_A, q_B], a)$. Podľa konštrukcie potom musí tiež platiť $q'_A, q''_A \in \delta_A(q_A, a)$. A to je spor s tým, že A je MDFA.

Na dôkaz dolnej hranice použijeme jazyky $L(A_n) = \{w \in \{a, b\}^* \mid \#_a(w) \equiv 0 \ (mod\ n)\}$ a $L(B_m) = \{w \in \{a, b\}^* \mid \#_b(w) \equiv 0 \ (mod\ m)\}$. Nech má automat C pre $L(A_n) \cap L(B_m)$ menej ako nm stavov. Označme $M = \{(i, j) \mid 0 \leq i \leq n - 1, 0 \leq j \leq m - 1\}$. V M existujú dva prvky $(i, j), (i', j')$ také, že $\{q \in K_C \mid q \in \delta_C(I_C, a^i b^j) \wedge \delta_C(q, a^{n-i} b^{m-j}) \cap F_C\} \cap \delta_C(I_C, a^{i'} b^{j'})$ je neprázdna množina. Potom je slovo $a^{i'} b^{j'} a^{n-i} b^{m-j}$ akceptované automatom C. Platí však, že $i \neq i'$ alebo $j \neq j'$ a preto $i' + n + i \not\equiv 0 \bmod n$ alebo $j' + m + j \not\equiv 0 \bmod m$ a to je spor s tým, že C akceptuje jazyk $L(A_n) \cap L(B_m)$.

□

5.4 Zreťazenie

Najprv popíšeme známu konštrukciu na realizáciu zreťazenia pomocou NFA, o ktorej neskôr ukážeme, že je korektná aj pre model PNA. Túto konštrukciu pozmeníme, aby sme získali konštrukciu na reazlizáciu zreťazenia pomocou SUFA a GSUFA. Ukážeme tak, že je možné efektívne realizovať zreťazenie pomocou SUFA a GSUFA. Pre modely FNA a LNA naopak dokazujeme, že pomocou nich nie je možné efektívne realizovať zreťazenie.

Ukázalo sa, že zreťazenie je operácia, s ktorou majú viaceré modely problémy. Tieto problémy spočívajú v tom, že automat, ktorý akceptuje zreťazenie dvoch jazykov sa pokúša uhádnuť, z akých dvoch podslov je vstupné slovo zložené.

Ukážeme, ako sa zmení zložitosť vykonávania zreťazenia, keď budeme mať k dispozícii informáciu o tom, ako vstupné slovo rozdeliť na dve podslová. Za týmto účelom sa budeme v tejto časti zaoberať dvoma operáciami. Zreťazením, ktoré spočíva v nájdení automatu C, pre ktorý $L(C) = L(A).L(B)$, a značkovaným zreťazením, ktoré spočíva v nájdení automatu C, pre ktorý $L(C) = L(A)\#L(B)$, kde $\# \notin \Sigma_A \cup \Sigma_B$.

Zreťazenie pomocou NFA a DFA

V tejto časti uvádzame dve známe tvrdenia. To prvé hovorí o tom, ako efektívne realizovať zreťazenie pomocou NFA a druhé, že nie je vo všeobecnosti možné efektívne realizovať zreťazenie pomocou DFA.

Tvrdenie 5.4.1. [4] $STATE(NFA, .) = n + m.$

Dôkaz. Hlavné myšlienky dôkazu tohoto tvrdenia v [4] sú nasledujúce.

Pre dôkaz hornej hranice sa použije nasledujúca konštrukcia. Automat C akceptujúci $L(A).L(B)$ spracuje vstupné slovo zo stavu q_{0A} v automate A. Po dosiahnutí niektorého akceptačného stavu automatu A sa nedeterministicky rozhodne, že bude vo výpočte pokračovať v automate B. Overí zvyšok slova v automate B. Ak B akceptuje, akceptuje aj C. Automat C bude teda obsahovať všetky stavy automatov A a B. Akceptačné stavy automatu A budú spojené so stavmi dosiahnuteľnými zo stavu q_{0B} na jeden znak. Preto $|C| = |A| + |B|$.

K dôkazu dolnej hranice boli použité postupnosti jazykov $L(A_n) = \{a^n\}^*$ a $L(B_m) = \{b^m\}^*$. Poznamenajme, že A_n aj B_m sú DFA, a že $|A_n| = n$ a $|B_m| = m$. Je ukázané, že každý NFA s jedným počiatočným stavom pre $L(A_n).L(B_m)$ potrebuje aspoň $n + m$ stavov. □

Tvrdenie 5.4.2. [25] $STATE(DFA, .) = (2m - 1).2^{n-1}.$

Keďže konštrukcia vedie k exponenciálnemu nárastu stavov, nebudeme sa ňou zaoberať. Dolná hranica bola ukázaná pomocou Myhill - Nerodovej vety pre jazyky uvedené v [25].

Značkovanie

Je zrejmé, že značka vo vstupnom slove modelu NFA nepomôže. Pre jazyk $L(C) = L(A_n).L(B_m)$, $L(A_n) = \{a^n\}^*$ a $L(B_m) = \{b^m\}^*$, je totiž rozdelenie vstupného slova jednoznačné aj bez značky.

Vo všeobecnosti nie je možné efektívne realizovať zreťazenie pomocou automatov z triedy DFA. Keď však pridáme značku, potom môžeme toto zreťazenie vykonávať rovnako efektívne ako s automatmi z triedy NFA.

Tvrdenie 5.4.3. $STATE(DFA, \#) = n + m.$

Dôkaz. Konštrukcia, ktorú použijeme, bude veľmi podobná konštrukcii pre zreťazenie pre NFA. Stavy aj prechody automatov A, B ostanú bez zmeny. Pridáme prechody z akceptačných stavov automatu A do počiatočného stavu automatu B na symbol #. Takto vzniknutý automat C akceptuje jazyk $L(C) = L(A)\#L(B)$. Keďže sme nepridali žiadne nové stavy, $|C| = n + m$.

Automat C je DFA práve vtedy, keď automaty A, B sú DFA. Do δ funkcie sme nepridali žiadne nedeterministické prechody, lebo $\# \notin \Sigma_A \cup \Sigma_B$. □

Zreťazenie pomocou SUFA a GSUFA

Pre triedy SUFA a GSUFA sa nám podarilo ukázať, že efektívne realizujú zreťazenie. Pre obe triedy popíšeme, aké problémy vznikli pri aplikovaní konštrukcie pre NFA a ich riešenia.

Nech $A, B \in SUFA$ a nech automat C vznikne konštrukciou pre zreťazenie pre NFA. Nech množina akceptačných stavov automatu A obsahuje k akceptačných stavov, $F_A = \{q_1, ..., q_k\}$. Slovo $w \in L(C)$ je možné zapísať ako $u.v$, kde $u \in L(A)$ a $v \in L(B)$. Nech $p \in \delta_B^*(q_{0B}, v)$. Ak pre slovo w existujú v automate A dva rôzne akceptačné výpočty končiace v stavoch q_i a q_j, potom pre slovo w v automate C existujú dve rôzne akceptačné cesty, končiace v jednom akceptačnom stave p automatu B.

Riešením tohoto problému je vyrobiť k kópií automatu B, jednu pre každý akceptačný stav automatu A. Budeme mať teda automaty $B_1, ..., B_k$. Zo stavu q_i budeme viesť prechody do nasledovníkov počiatočného stavu v automate B_i.

Tvrdenie 5.4.4. $STATE(SUFA, .) \leq n + nm$.

Dôkaz. Formálne popíšeme konštrukciu uvedenú vyššie. Pripomeňme, že máme automaty $A = (K_A, \Sigma_A, \delta_A, q_{0A}, F_A)$ a $B = (K_B, \Sigma_B, \delta_B, q_{0B}, F_B)$, ktoré sú SUFA. Zadefinujme automat $C = (K_C, \Sigma_C, \delta_C, q_{0C}, F_C)$ nasledovne.

Položme $F_A = \{q_1, \ldots, q_k\}$. Potom $K_C \subset K_A \cup K_B \times \mathbb{N}$, t.j. stavy automatu C budú dvojice.

$$\Sigma_C = \Sigma_A \cup \Sigma_B$$

$$K_C = \{[q, 1] \mid q \in K_A\} \cup \{[q, i] \mid q \in K_B, q_i \in F_A\}$$

$$q_{0C} = [q_{0A}, 1]$$

$$F_C = \begin{cases} \{[q, i] \mid q \in F_B, q_i \in F_A\} & q_{0B} \notin F_B \\ \{[q, i] \mid q \in F_B, q_i \in F_A\} \cup \{[q, 1] \mid q \in F_A\} & q_{0B} \in F_B \end{cases}$$

$$\delta_C([q, i], a) = \begin{cases} \{[p, 1] \mid p \in \delta_A(q, a)\} & q \in K_A - F_A, i = 1 \\ \{[p, i] \mid p \in \delta_A(q, a)\} \cup \{[p, j] \mid q = q_j, p \in \delta_B(q_{0B}, a)\} & q \in F_A, i = 1 \\ \{[p, i] \mid q \in \delta_B(q, a)\} & q \in K_B, q_i \in F_A \end{cases}$$

Nech automat C, ktorý vznikne popísanou konštrukciou nie je SUFA. Potom existuje stav $[q, i]$ a slovo w tak, že vedú dve cesty z q_{0C} do $[q, i]$ na w. Ak by $q \in A$ potom by A nemohol byť SUFA. Nech teda $q \in B$. Slovo w rozdelíme na dve časti, $w = u.v$ tak, že $[q_j, 1] \in \delta_C^*(q_{0C}, u)$ a $q \in \delta_C^*([q_j, 1], v)$. Zo stavu q_{0C} do stavu $[q_j, 1]$ na slovo u existuje iba jedna cesta, inak by A nebol SUFA. Musia teda existovať dve rôzne vesty zo stavu $[q_j, 1]$ do stavu $[q, i]$ na slovo v. Potom ale v automate B existujú dve cesty z q_{0B} do q na v, čo je v spore s tým, že B je SUFA.

Automat A použijeme bez zmeny a navyše budeme používať maximálne $|A|$ kópií automatu B. Spolu na realizáciu postačuje $n + nm$ stavov. □

Keďže $SUFA =_P GSUFA$, je možné každý GSUFA previesť na SUFA, aplikovať vyššie popísanú konštrukciu a tým dostať SUFA, ktorý je zároveň aj GSUFA. Takáto konštrukcia je samozrejme správna a vedie k ohraničeniu $STATE(GSUFA, .) \leq n^2 + n^2.m^2$. Konštrukcia pre SUFA bola o niečo úspornejšia, preto sme sa pokúsili túto hranicu vylepšiť. Konštrukcia pre SUFA vytvára kópie automatu B, každú pre jeden akceptačný stav automatu A. Pre GSUFA toto nestačí. Z dvoch počiatočných stavov automatu B môžu viesť do jedného stavu dve rôzne cesty pre slovo v. Preto treba jednoznačne určiť, ktorým počiatočným stavom automatu B bude spracovaná druhá časť slova. Pre každý akceptačný stav automatu A vyrobíme toľko kópií automatu B, koľko má B počiatočných stavov. Automat A bude teda bez zmeny a navyše vytvoríme najviac mn kópií automatu B.

Tvrdenie 5.4.5. $STATE(GSUFA, .) \leq n + nm^2$.

Dôkaz. Formálne popíšeme konštrukciu uvedenú vyššie. Pripomeňme, že máme automaty $A = (K_A, \Sigma_A, \delta_A, I_A, F_A)$ a $B = (K_B, \Sigma_B, \delta_B, I_B, F_B)$, ktoré sú GSUFA. Zadefinujme automat $C = (K_C, \Sigma_C, \delta_C, I_C, F_C)$ nasledovne.
Položme $F_A = \{q_1, \ldots, q_k\}$ a $I_B = \{p_1, \ldots, p_l\}$. Potom $K_C \subset K_A \cup K_B \times \mathbb{N} \times \mathbb{N}$, t.j. stavy automatu C budú trojice.

$$\Sigma_C = \Sigma_A \cup \Sigma_B$$

$$K_C = \{[q, 1] \mid q \in K_A\} \cup \{[q, i] \mid q \in K_B, q_i \in F_A\}$$

$$I_{0C} = \{[q, 1, 1] \mid q \in I_A\}$$

$$F_C = \begin{cases} \{[q, i, j] \mid q \in F_B, q_i \in F_A, p_j \in I_B\} & I_B \cap F_B = \emptyset \\ \{[q, i, j] \mid q \in F_B, q_i \in F_A, p_j \in I_B\} \cup \{[q, 1, 1] \mid q \in F_A\} & I_B \cap F_B \neq \emptyset \end{cases}$$

$$\delta_C([q, i, j], a) = \begin{cases} \{[p, 1, 1] \mid p \in \delta_A(q, a)\} & q \in K_A - F_A, i, j = 1 \\ \{[p, 1, 1] \mid p \in \delta_A(q, a)\} \cup & \\ \{[p, i', j'] \mid q = q_{i'}, p_{j'} \in \delta_B(I_B, a)\} & q \in F_A, i, j = 1 \\ \{[p, i, j] \mid q \in \delta_B(q, a)\} & q \in K_B, q_i \in F_A, p_j \in I_B \end{cases}$$

Nech automat C, ktorý vznikne popísanou konštrukciou nie je GSUFA. Potom existujú stavy $[p, i, j]$ a $[q, i', j']$ a slovo w tak, že existuje viac ciest zo stavu $[p, i, j]$ na slovo w do stavu $[q, i', j']$. Je zrejmé, že $[p, i, j]$ musí byť z automatu A a $[q, i', j']$ musí byť stav v niektorej kópii automatu B (inak by A, alebo B nemohli byť GSUFA). Slovo w môžeme rozdeliť na dve časti, $w = u.v$ tak, že slovo u sa spracuje v automate A a slovo v v automate B.

Predpokladajme, že slovo u možno v automate A spracovať zo stavu q iba jednou cestou do nejakého akceptačného stavu. Potom pre slovo v museli v automate B existovať dve rôzne cesty z počiatočných stavov do stavu q. B je však GSUFA, preto tieto cesty museli začínať z dvoch rôznych počiatočných stavov, nech sú to p_1 a p_2. Z automatu A vedú do jednej kópie automatu B prechody iba cez nasledovníkov jedného počiatočného stavu automatu B. Preto nie je možné slovo v spracovať v automate C cez stavy p_1 a p_2 do toho istého stavu.

Predpokladajme, že slovo u možno v automate A spracovať zo stavu p viacerými cestami. Tieto cesty končia v rôznych akceptačných stavoch automatu A, lebo A je GSUFA. Označme tieto stavy q_1 a q_2. Stavy automatu B dosiahnuteľné v automate C zo stavu $[q_1, 1, 1]$ sú disjunktné so stavmi automatu B dosiahnuteľnými zo stavu $[q_2, 1, 1]$ v automate C. Preto pre slovo w nemôžu v automate C existovať dve cesty medzi danými stavmi. □

Značkovanie

Podobnú konštrukciu ako sme použili pre zreťazenie je možné použiť aj pre značkované zreťazenie. Nepodarilo sa nám ukázať, či je možné realizovať značkované zreťazenie efektívnejšie ako zreťazenie ako také.

Zreťazenie pomocou UFA a MDFA

Pre triedy UFA a MDFA sa nám nepodarilo ukázať, že efektívne realizujú zreťazenie. Pre tieto triedy ostáva táto otázka otvorená. V tejto časti popíšeme problémy, ktoré sú toho príčinou.

Pri aplikovaní konštrukcie použitej na realizáciu zreťaznie pomocou NFA nastáva pre model UFA tento problém. Nech A aj B sú SUFA a nech C vznikne konštrukciou pre NFA z tvrdenia 5.4.1. Pre slovo $w \in L(C)$ môže existovať viacero akceptačných výpočtov. Tento jav nastáva vtedy, ak sa dá slovo w rozdeliť viacerými spôsobmi na $w = u.v$, $u \in L(A), v \in L(B)$.

Vo všeobecnosti je možné každé slovo w rozdlieť na zreťazenie dvoch podslov až $|w| + 1$ spôsobmi. To by znamenalo, že ak $A, B \in UFA$, tak C nemusí byť ani FNA (ale určite je LNA). Vieme však, že každý UFA, ktorý má iba užitočné stavy je zároveň aj SUFA a z tvrdenia 5.4.4 vieme, že SUFA efektívne realizujú zreťazenie. To svedčí o tom, že zreťazením jazykov $L(A), L(B)$ vždy dostaneme jazyk, ku ktorému existuje SUFA s polynomiálnym počtom stavov vzhľadom na $|A|$ a $|B|$. SUFA sú ale vo všeobecnosti úspornejšie ako UFA. Konštrukcia použitá pre model SUFA sa nedá aplikovať na model UFA.

Pre model MDFA vzniká pri realizácii zreťazenia zásadný problém. Automat na rezlizáciu zreťaznia môže urobiť jediné nedeterministiké rozhodnutie na začiatku výpočtu. Ak by sme sa chceli držať myšlienky, že automat uhádne rozdelenie slova w, museli by sme to vedieť urobiť hneď na začiatku výpočtu. Automat má vtedy na výber jednu z konečného množstva možností. Nadrozdiel od MDFA môže automat z triedy SUFA prejsť pri výpočte na slove jedným stavom až lineárne veľa krát. A pri každom prechode môže urobiť iné nedeterministické rozhodnutie. Prikláňame sa k hypotéze, že vo všeobecnosti nie je možné efektívne realizovať zreťazenie pomocou automatov z triedy MDFA.

Značkovanie

Po pridaní značky do vstupného slova sa situácia pre oba modely výrazne zlepšuje. Značkované zreťazenie vieme realizovať pomocou UFA rovnako efektívne ako pomocou

NFA. Pre triedu MDFA sme tiež našli spôsob, ako efektívne vykonávať túto operáciu.

Tvrdenie 5.4.6. $STATE(UFA, \#) = n + m$.

Dôkaz. Na dôkaz hornej hranice použijeme rovnakú konštrukciu ako v tvrdení 5.4.3. Je nutné ukázať, že automat zostrojený touto konštrukciou je UFA, ak A a B sú UFA.

Nech A, B sú UFA a nech $w = u\#v \in L(C)$. Slovo u je možné v automate A akceptovať jednou cestou, ktorá skončí v jednom akceptačnom stave automatu A. Z neho je možné pokračovať na # do počiatočného stavu automatu B a odtiaľ existuje iba jedna akceptačná cesta preslovo v. Preto pre slovo w existuje v C iba jedna akceptačná cesta a teda $C \in UFA$.

Pre jazyk $L(C) = L(A_n).L(B_m)$, $L(A_n) = \{a^n\}^*$ a $L(B_m) = \{b^m\}^*$, každý NFA potrebuje $n + m$ stavov a teda aj každý UFA potrebuje toľko stavov. Slovo z jazyka $L(C)$ je možné jednoznačne rozdeliť na prvú a druhú časť, pretože $\Sigma_A \cap \Sigma_B = \emptyset$. Preto značka medzi jazykmi nepridá žiadnu novú informáciu a teda nemôže ani pomôcť zlepšiť zložitosť realizovania značkovaného zreťazenia pomocou automatov z triedy UFA. □

Pre model MDFA je konštrukcia pre značkované zreťazenie o niečo komplikovanejšia. Ak by sme chceli použiť konštrukciu z dôkazu tvrdenia 5.4.3 dospeli by sme k nasledujúcemu problému. Z akceptačných stavov automatu A by sme viedli prechody do počiatočných stavov automatu B. B má však viacero počiatočných stavov a preto by tieto nové prechody na # boli nedeterministickým rozhodnutím. Výsledný automat má však byť MDFA a preto nemôže obsahovať nedeterministické rozhodnutia inde ako na začiatku, t.j. voľbou počiatočného stavu. O tom, ako tento problém vyriešiť hovorí nasledujúce tvrdenie.

Tvrdenie 5.4.7. $STATE(MDFA, \#) \leq mn + m$

Dôkaz. Keďže automat C nesmie obsahovať nedeterministické rozhodnutia pri spracovávaní vstupu, rozhodnutie o tom, v ktorom počiatočnom stave automatu B začať overovať druhú časť vstupného slova, treba urobiť už na začiatku výpočtu. Automat C teda uhádne, v ktorom stave automatu A začať overovať prvú časť slova a v ktorom počiatočnom stave automatu B začať overovať druhú časť slova. Následne overí prvú časť slova v nejakej kópii automatu A, z ktorej vedú prechody na # iba do toho počiatočného stavu automatu B, ktorý bol zvolený na začiatku.

Automat C bude mať pre každý počiatočný stav automatu B jednu kópiu automatu A. Nech množina počiatočných stavov automatu B obsahuje k stavov $I_B = \{q_1, \ldots, q_k\}$, potom C bude obsahovať k nezávislých automatov $A_1, \ldots, A_k$. Počiatočné stavy automatu C bude tvoriť zjednotenie počiatočných stavov automatov A_1 až A_k. Z každého stavu v automate A_i, ktorý bol v automate A akceptačný, bude viesť prechod na symbol # do stavu q_i v automate B. Akceptačné stavy automatu B budú akceptačnými stavmi automatu C.

Jediné prechody, ktoré sme do automatu pridali, sú prechody na #. Z každého stavu vedie najviac jeden takýto prechod. Preto je C MDFA. V konštrukcii využiijeme vždy najviac $|B|$ kópií automatu A, čo dáva $n.m$ stavov. Tiež potrebujeme všetky stavu automatu B. Spolu nám stačí $mn + m$ stavov. □

Zreťazenie pomocou FNA

V tejto časti ukážeme, že vo všeobecnosti nie je možné efektívne realizovať zreťazenie dvoch jazykov pomocou automatov z triedy FNA. Využijeme na to jazyk $\exists_1(L)$ [7], ktorý oddeľuje triedy FNA a LNA.

Tvrdenie 5.4.8. $STATE(FNA, .) = 2^{\Omega(n^{1/96}+m)}$.

Dôkaz. Nech $L_r^{'}$ je jazyk zadefinovaný v [7], $L_r^{'} = (L_r)^t$, $t = r^{1/3}$. Položme $L(A_r) = \{w_1...w_m \mid w_m \in L_i | w_i| = 2t\}$ a $L(B_r) = \{w_1...w_m \mid |w_i| = 2t\}$.

Je zrejmé, že $L(A_r).L(B_r) = \exists_1(L_r^{'})$. Každé slovo z $\exists_1(L_r^{'})$ možno rozdeliť na dve slová, prvé pred výskytom slova z $L_r^{'}$ a druhé po výskyte tohoto slova. Na druhej strane je garantované, že každé slovo z $L(A_r)$ obsahuje nejaké slovo z $L_r^{'}$, preto aj zreťazenie jazykov $L(A_r)$ a $L(B_r)$ musí obsahovať aspoň jedno slovo.

Zostáva ešte ukázať, že A_r a B_r sú FNA a ich veľkosť je polynomiálna vzhľadom na r.

A_r sa bude skladať z dvoch častí. Bude mať cyklus, ktorý bude kontrolovať dĺžku slov w_i. Potom sa nedeterministicky rozhodne, že slovo w_i, ktoré prichádza je z jazyka $L_r^{'}$ a potom toto slovo overí v automate N_L. V automate A_r neexistujú stavy p a q a slovo w tak, že vedie cesta z p do p na w, a q do q na w a tiež z p do q na w. Preto podľa štrukturálnej vlastnosti automatov z triedy FNA (popísanej v časti 4.1) platí, že $A_r \in$ FNA. $|A_r| = 2t + |N_L| \leq 2r^{1/3} + 2r^{32/3} - r^{32} + 2$.

B_r bude obsahovať iba cyklus na overenie dĺžky slova, rovnako ako automat A_r. B_r je teda deterministický a $|B_r| = 2t = 2r^{1/3}$.

Je dokázané [7], že každý FNA potrebuje na akceptovanie jazyka $\exists_1(L_r^{'})$ aspoň $2^{\Omega(r^{1/3})}$ stavov. Platí, že $\Omega(|A_r|^{1/96} + |B_r|) = r^{1/3}$, preto je na realizáciu zreťazenia pomocou FNA potrebných aspoň $2^{\Omega(n^{1/96}+m)}$ stavov.

□

Značkovanie

Po pridaní značky do vstupného slova vieme realizovať (značkované) zreťazenie pomocou automatov z triedy FNA rovnako efektívne ako pomocou automatov z triedy NFA.

Tvrdenie 5.4.9. $STATE(FNA, \#) = n + m$.

Dôkaz. Nech A, B sú FNA a nech $w = u\#v \in L(C)$. Slovo u je možné v automate A akceptovať najviac k_A cestami. Niektoré z týchto ciest je možné predĺžiť najviac k_B spôsobmi v automate B pri spracovaní slova v. Nejednoznačnosť automatu C je možné ohraničiť konštantou $k_A.k_B$, a preto $C \in FNA$.

Dolná hranica je dosiahnutá pre jazyky z dôkazu tvrdenia 5.4.1. □

Zreťazenie pomocou LNA

V predchádzajúcich častiach sme pri dôkaze polynomiálnych horných hraníc vždy vychádzali z konštrukcie zreťazenia pomocou NFA. Keď A aj B sú LNA potom slová

u a v je možné akceptovať až lineárne veľa spôsobmi. Ak $|v| = |u| = \frac{|w|}{2}$, potom pre w v automate C môže existovať až $|w|^2$ rôznych akceptačných ciest. Podozdrenie, že pomocou LNA nie je vždy možné efektívne realizovať zreťazenie potvrdzujeme v nasledujúcom tvrdení.

Tvrdenie 5.4.10. $STATE(LNA, .) \geq 2^{\Omega(n^{1/96}+m^{1/96})}$.

Dôkaz. Nech $L(A) = L(B) = \exists_1(L)$ [7]. Potom ak $L(C) = L(A).L(B)$, tak $L(C) = \exists_2(L)$.

Pre každé $r \in \mathbb{N}$ teda máme automat $A_r = B_r \in LNA$ taký, že $|A_r| = 2.(2t) + |N_L| = 4.r^{1/3} + 2r^{32/3} - r^{32} + 2$. Je dokázané [7], že každý LNA C_r pre $L(A_r).L(B_r)$ potrebuje aspoň $\Omega((r/4)^{1/3})$.

Platí, že $\Omega(|A_r|^{1/96} + |B_r|^{1/96}) = (r/4)^{1/3}$ a preto je na realizáciu zreťazenia pomocou LNA potrebných aspoň $2^{\Omega(n^{1/96}+m^{1/96})}$ stavov. □

Značkovanie

Nepodarilo sa nám nájsť efektívnu konštrukciu na realizáciu značkovaného zreťazenia pomocou LNA.

Zreťazenie pomocou PNA

Polynomiálna nejednoznačnosť postačuje automatu na to, aby sa vedel jednoducho (t.j. bez veľkého nárastu stavov) vysporiadať so všetkými problémami opísanými v predchádzajúcich častiach.

Tvrdenie 5.4.11. $STATE(PNA, .) = n + m$.

Dôkaz. Použijeme konštrukciu pre NFA. Korektnosť konštrukcie overíme sporom. Nech $A, B \in PNA$ a nech $C \notin PNA$. Potom $C \in sENA$ a teda existuje stav $q \in C$ a slovo w tak, že vedú aspoň dve cesty z q do q na w. Nové prechody, ktoré sme pridali medzi automaty A a B nemôžu byť súčasťou žiadneho cyklu v C, pretože z automatu B nevedú žiadne prechody do automatu A. Preto tie isté cesty z q do q na w museli existovať aj v automate A, resp. B. A to je spor s tým, že A, B sú PNA.

K dôkazu dolnej hranice je možné použiť jazyky $L(A_n) = \{a^n\}^*$ a $L(B_m) = \{b^m\}^*$ z dôkazu pre NFA. □

Značkovanie

Je zrejmé, že značka vo vstupnom slove modelu PNA nepomôže. Pre jazyk $L(C) = L(A_n).L(B_m)$, $L(A_n) = \{a^n\}^*$ a $L(B_m) = \{b^m\}^*$, je totiž rozdelenie vstupného slova jednoznačné aj bez značky.

5.5 Iterácia a kladná iterácia

Popisujeme, ako je možné efektívne realizovať iteráciu (a kladnú iteráciu) pomocou NFA. Medzi známymi výsledkami sme nenašli nutný počet stavov na realizáciu kladnej iterácie pomocou DFA. Dokázali sme, že táto hranica je určite aspoň 3.2^{n-3}. V tejto časti ďalej dokazujeme, že nie je možné efektívne realizovať iteráciu ani kladnú iteráciu žiadnym nami spomenutým modelom okrem NFA.

Máme automat A. Pre realizáciu iterácie chceme nájsť taký automat C, pre ktorý $L(C) = L(A)^*$. V prípade realizácie kladnej iterácie budeme hľadať automat C' taký, že $L(C') = L(A)^+$.

Iterácia pomocou NFA a DFA

V tejto časti uvedieme známe výsledky o zložitosti iterácie pre triedy NFA a DFA.

Tvrdenie 5.5.1. [4] $STATE(NFA,^*) = n + 1$.

Dôkaz. Najprv opíšeme konštrukciu, pre realizáciu iterácie. Máme automat A pre jazyk $L(A)$. Budeme viesť prechody z akceptačných stavov automatu A do nasledovníkov počiatočného stavu automatu A. Tým získame automat C' taký, že $L(C') = L(A)^+$. Ak $\varepsilon \in L(A)$, potom $L(A)^+ = L(A)^*$. Inak (ak $\varepsilon \notin L(A)$) je nutné pridať nový počiatočný stav, ktorý bude zároveň aj akceptačným stavom. Prechody z nového stavu budú viesť do nasledovníkov počiatočného stavu automatu A. Vždy teda postačuje $n + 1$ na realizáciu iterácie.

Bolo dokázané, že pre postupnosť jazykov $L_n = \{w \in \{a, b\}^* \mid \#_a(w) \equiv n - 1 \pmod n\}$ je dokázaná hranica $n + 1$ stavov dosiahnutá. □

Z dôkazu tvrdenia 5.5.1 je zrejmé že je možné realizovať kladnú iteráciu pomocou automatov z triedy NFA s použitím najviac n stavov. Bolo ukázané [4], že $STATE(NFA,^+) = n$.

Tvrdenie 5.5.2. [25] $STATE(DFA,^*) \geq 3.2^{n-2}$.

Jazyky, ktoré dosahujú hranicu z tvrdenia 5.5.2 je možné nájsť v [25]. Autori tu tiež ukazjú, že postačujúci stav na reazlizáciu iterácie pomocou DFA je tiež 3.2^{n-2}. Výnimkou je jazyk $L = \emptyset$, pre ktorý existuje DFA s jedným stavom, ale pre jazyk $L^* = \{\varepsilon\}$ potrebuje každý DFA dva stavy, pretože DFA musí byť úplný.

Kladná iterácia pre triedu DFA nebola doposial presne vyčíslená. Je však zrejmé, že nie je možné, aby DFA vo všeobecnosti efektívne realizovali kladnú iteráciu. Platí totiž $L^* = L^+ \cup \{\varepsilon\}$, takže by sme vedeli efektívne realizovať aj iteráciu pomocou DFA.

Tvrdenie 5.5.3. $STATE(DFA,^+) \geq 3.2^{n-3}$.

Dôkaz. Existuje taká postupnosť DFA A_n [25], $A_n = |n|$, že každý DFA pre jazyk $L(A_n)^*$ potrebuje aspoň 3.2^{n-2} stavov.

Potom pre každý DFA C_n pre jazyk $L(A_n)^+$ musí platiť $3.2^{n-2} \leq 2.|C_n|$. Dôvod je ten, že pomocou DFA vieme realizovať zjednotenie na $n.m$ stavov. Zjednotenie jazykov

$L(A_n)^+ \cup \{\varepsilon\}$ dáva jazyk $L(A_n)^*$. Preto každý DFA pre jazyk $L(A_n)^+$ potrebuje aspoň 3.2^{n-3} stavov. Toľko stavov je teda potrebných na realizáciu kladnej iterácie pomocou DFA. □

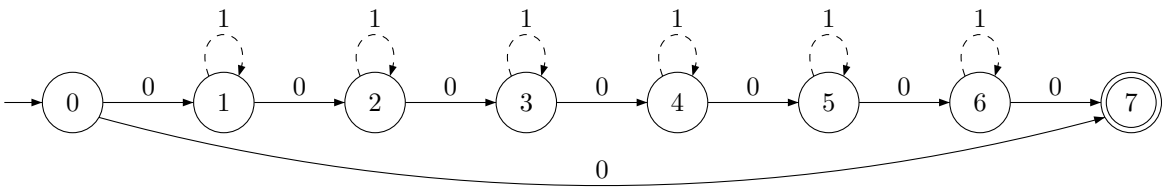
Iterácia pomocou PNA

Všetky operácie, s ktorými sme sa doteraz stretli, mali jednu spoločnú črtu. Dôkazy pre dolné, resp. horné, hranice pre model NFA sa dali priamo aplikovať na model PNA. V prípade iterácie to však nie je možné.

Pokúsili sme sa nájsť spôsob, ako efektívne realizovať iteráciu pomocou PNA. Dospeli sme však k zisteniu, že to nie je vo všeobecnosti možné. Jazyk, ktorý oddeľuje triedy NFA a PNA nám pomôže "dosvedčiť" fakt, že vo všeobecnosti nie je možné efektívne realizovať iteráciu pomocou automatov z triedy PNA.

Tvrdenie 5.5.4. $STATE(PNA,^*) \geq 2^{n-1} - 1$.

Dôkaz. Položme $L_n = 0 + (01^*)^{n-1}0$. Pre L_n existuje UFA A_n s $n+1$ stavmi. Je dokázané [14], že každý PNA s ľubovoľným počtom počiatočných stavov pre L_n^* potrebuje aspoň $2^n - 1$ stavov. Do tejto triedy spáda aj model PNA používaný v tejto práci, ktorý môže mať iba jeden počiatočný stav. Na realizáciu iterácie jazyka L_n pomocou PNA je teda nutných $2^{n-1} - 1$ stavov. □



Obr. 5.1: UFA $\mathcal{A}_7$ pre jazyk $L_7 = 0 + (01^*)^6 0$

Kladná iterácia

Všimnime si, aký je vzťah medzi kladnou iteráciu a iteráciou. Pre každý regulárny jazyk L platí $L^* = L^+ \cup \{\varepsilon\}$. Zjednotenie je možné efektívne realizovať pomocou PNA a pre jazyk $\{\varepsilon\}$ existuje PNA, ktorý má iba jeden stav. Ak by sme teda vedeli efektívne realizovať kladnú iteráciu, vedeli by sme pomocou PNA efektívne realizovať aj iteráciu ako takú.

Pre každú postupnosť jazykov L_n platí nasledujúce. Nech pre L_n^+ existuje automat $A_n \in PNA$. Potom pre L_n^* existuje automat $A_n^{'} \in PNA$ tak, že $|A_n^{'}| \leq |A_n| + 1$. Toto docielime jednoduchým prianím nového počiatočného stavu, ktorý bude zároveň aj akceptačným stavom a budú z neho viesť prechody do nasledovníkov počiatočného stavu automatu A_n.

Tvrdenie 5.5.5. $STATE(PNA,^+) \geq 2^{n-1} - 2$.

Dôkaz. Položme $L_n = 0 + (01^*)^{n-1}0$. Pre L_n existuje UFA s $n+1$ stavmi. Je dokázané [14], že každý PNA pre L_n^* potrebuje aspoň $2^n - 1$ stavov. Potom pre každý PNA C_n pre jazyk L_n^+ musí platiť $2^n - 1 \leq |C_n| + 1$. Teda každý PNA pre jazyk L_n^+ potrebuje aspoň $2^n - 2$ stavov. Na realizáciu kladnej iterácie jazyka L_n pomocou PNA je teda nutných $2^{n-1} - 2$ stavov. □

Iterácia pomocou UFA, FNA, LNA, SUFA a GSUFA

Tvrdenie 5.5.6. *Pre každú triedu* $C \in \{UFA, FNA, LNA, SUFA, GSUFA\}$ *platí* $STATE(C,^*) \geq 2^{n-1} - 1$.

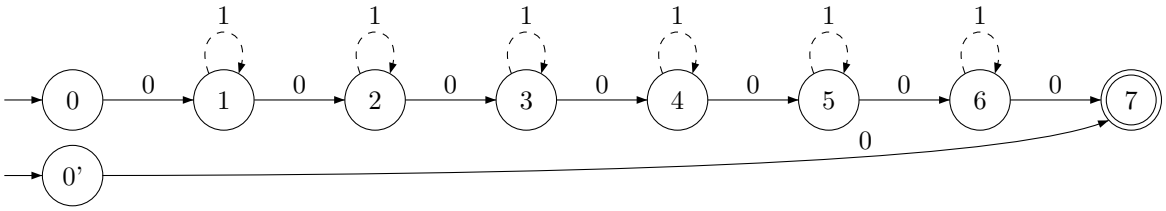
Dôkaz. Položme $L_n = 0 + (01^*)^{n-1}0$. Pre L_n existuje UFA (a teda aj FNA, LNA, SUFA a GSUFA) s $n + 1$ stavmi. Je dokázané [14], že každý PNA s ľubovoľným počtom počiatočných stavov (a teda aj UFA, FNA, LNA, SUFA a GSUFA) pre L_n^* potrebuje aspoň $2^n - 1$ stavov. Na realizáciu iterácie jazyka L_n pomocou UFA, FNA, LNA, SUFA alebo GSUFA je teda nutných $2^{n-1} - 1$ stavov. □

Pre dolnú hranicu kladnej iterácie pre tieto modely sa dá použiť dôkaz tvrdenia 5.5.5. Každý PNA s ľubovoľným počtom počiatočných stavov pre jazyk L_n^+ potrebuje aspoň $2^n - 2$ stavov. Preto aj každý UFA, FNA, LNA, SUFA a GSUFA potrebuje toľko stavov.

Dôsledok 5.5.1. *Pre každú triedu* $C \in \{UFA, FNA, LNA, SUFA, GSUFA\}$ *platí* $STATE(C,^+) \geq 2^{n-1} - 2$.

Iterácia pomocou MDFA

Ani pomocou MDFA nie je možné vo všeobecnosti realizovať iteráciu a kladnú iteráciu. Pre jazyk $L_n = 0 + (01^*)^{n-1}0$ totiž existuje MDFA A'_n s $n + 2$ stavmi. Dolná hranica na realizáciu iterácie pomocou MDFA bude o niečo nižšia ako pre ostatné modely.



Obr. 5.2: MDFA $\mathcal{A}'_7$ pre jazyk $L_7 = 0 + (01^*)^6 0$

Tvrdenie 5.5.7. $STATE(MDFA,^*) \geq 2^{n-2} - 1$.

Situácia sa príliš nemení ani pri kladnej iterácii. Každý PNA s ľubovoľným počtom počiatočných stavov pre jazyk L_n^+ potrebuje aspoň $2^n - 2$ stavov. Tým je dokázaná dolná hranica $2^{n-2} - 2$ stavov.

Tvrdenie 5.5.8. $STATE(MDFA,^+) \geq 2^{n-2} - 2$.

5.6 Reverz

Aj túto časť začneme známym tvrdením, ktoré ukáže ako efektívne realizovať reverz pomocou NFA. Ukazujeme, že rovankú konštrukciu je možné použiť aj pre UFA, FNA, LNA a PNA. Pre triedu GSUFA uvádzame o niečo efektívnejšiu konštrukciu. pre triedu SUFA sme dokázali, že zložitosť reverzu je nanajvýš kvadratická.

Máme automat A. Chceme nájsť taký automat C, pre ktorý $L(C) = L(A)^R$.

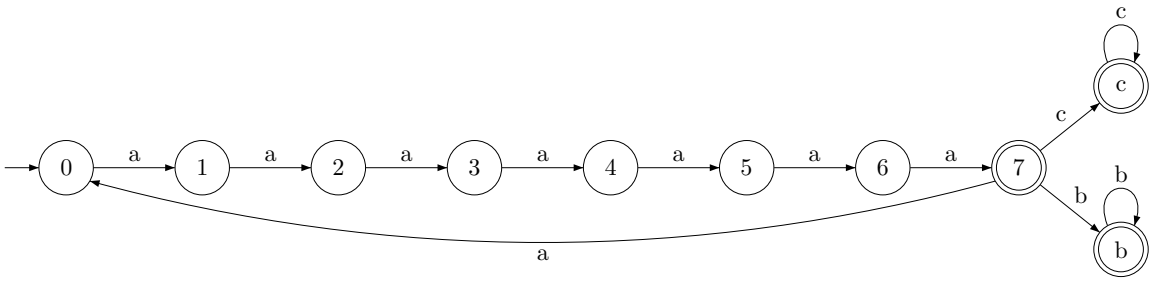
Reverz pomocou NFA a DFA

V tejto časti uvedieme známe výsledky o zložitosti reverzu pre triedy NFA a DFA.

Tvrdenie 5.6.1. [4] $STATE(NFA,^R) = n + 1$.

Dôkaz. Konštrukcia použitá na dôkaz hornej hranice sa opiera o prevrátenie prechodovej logiky pôvodného automatu a zámenu akceptačných stavov s počiatočnými stavmi. Pre automaty, ktoré majú viac akceptačných stavov ale iba jeden počiatočný stav potrebujeme navyše pridať nový počiatočný stav. Spolu teda použijeme maximálne $n + 1$ stavov.

Hranica ukázaná pomocou tejto konštrukcie je dosiahnutá pre postupnosť $L_n = a^n\{a^{n+1}\}^*(\{b\}^* \cup \{c\}^*)$. Pre L_n existuje DFA A_n, ktorý má $n+3$ stavov. Každý automat pre jazyk L_n^R potrebuje dva rôzne stavy q_b a q_c, aby skontroloval, že začiatok slova má tvar $\{b\}^* \cup \{c\}^*$. Žiaden z týchto stavov nemôže byť počiatočný, inak by automat akceptoval slová tvaru b^*c^* alebo c^*b^*. Ďalších $n+1$ stavov je potrebných na overenie suffixu slova, ktorý má mať tvar $\{a^{n+1}\}^*a^n$. Stavy q_b ani q_c nemôžu patriť medzi stavy overujúce suffix, inak by sme akceptovali slová tvaru $c^*a^*b^*a^k$ (alebo podobné). Z rovnakých dôvodov musíme mať počiatočný stav, ktorý nebude súčasťou cyklu. Preto potrebujeme mať nový počiatočný stav a teda nám nestačí $n + 3$ stavov. □



Obr. 5.3: DFA $\mathcal{A}_7$ pre jazyk $L_7 = a^7\{a^8\}^*(\{b\}^* \cup \{c\}^*)$

Tvrdenie 5.6.2. [13] $STATE(DFA,^R) = 2^n$.

Dôkaz. Horná hranica je triviálna. V automate A prevrátime všetky prechody a vymeníme počiatočné a akceptačné stavy. Na vzniknutý automat aplikujeme podmnožinovú konštrukciu.

V [13] tiež možno nájsť jazyky, pre ktoré realizácia reverzu pomocou DFA dosahuje hranicu z tvrdenia. □

Reverz pomocou UFA, FNA, LNA a PNA

Triedy definované na základe množstva nejednoznačnosti realizujú reverz rovnako efektívne ako trieda NFA.

Tvrdenie 5.6.3. $STATE(UFA,^R) = STATE(FNA,^R) = STATE(LNA,^R) = STATE(PNA,^R) = n + 1$.

Dôkaz. Na dôkaz hornej hranice pre tieto modely použijeme konštrukciu, ktorá bola použitá pre NFA. T.j. otočíme prechody a zameníme počiatočné a akceptačné stavy. Automat, ktorý vznikne touto konštrukciou označme A^R. Z konštrukcie vyplýva, že pre každé slovo $w \in \Sigma_A^*$ platí $ambig_A(w) = ambig_{A^R}(w^R)$. Preto má vzniknutý automat rovnakú nejednoznačnosť, ako pôvodný automat.

Hranica ukázaná pomocou tejto konštrukcie je dosiahnutá pre postupnosť $L_n = a^n\{a^{n+1}\}^*(\{b\}^* \cup \{c\}^*)$. Pre L_n existuje $n + 3$ stavový DFA (a teda aj UFA, FNA, LNA a PNA), ale pre jazyk L_n^R potrebuje každý NFA (a teda aj UFA, FNA, LNA a PNA) aspoň $n + 4$ (tvrdenie 5.6.1). □

Reverz pomocou MDFA

Bolo ukázané, že MDFA nemôže vo všeobecnosti efektívne realizovať reverz. Na dôkaz bol použitý jazyk some – register – on definovaný v časti 4.1.

Tvrdenie 5.6.4. [16] $STATE(MDFA,^R) \geq 2^n - 1$.

Dôkaz. Ako sme už spomínali v časti , každý MDFA pre jazyk some–register–on s n registrami potrebuje aspoň $2^n - 1$ stavov, avšak pre tento jazyk existuje SUFA M s n stavmi. Keď prevrátime prechodovú logiku automatu A a zameníme počiatočné a akceptačné stavy dostaneme automat M^R, ktorý je MDFA (pretože prechodová logika automatu M^R je deterministická).

Položme $A = M^R$. Automat A je MDFA a má n stavov. Avšak je dokázané [16], že každý MDFA pre jazyk $L(A)^R$ (some – register – on) potrebuje $2^n - 1$ stavov. □

Reverz pomocou SUFA a GSUFA

Triedy SUFA a GSUFA efektívne realizujú reverz. Keďže ide o triedy definované pomocou štrukturálnej vlastnosti, dokážeme tvrdenie o ich zložitosti tejto realizácie zvlášť.

Tvrdenie 5.6.5. $STATE(GSUFA,^R) = n$.

Dôkaz. Nech A je GSUFA. Keď v automate A otočíme všetky prechody a vymeníme akceptačné stavy s počiatočnými dostaneme automat A^R, ktorý akceptuje jazyk $L(A)^R$. Takýto automat je však GSUFA, pretože pre každé $p, q \in A$ a pre každé $w \in \Sigma_A^*$ platí $|\delta_A(q, w, p)| = |\delta_{A^R}(p, w^R, q)|$.

V tvrdení 5.6.1 sme popísali jazyk L_n, ku ktorému existuje $n + 3$ stavový DFA. Zdôvodnili sme, prečo každý automat s jedným počiatočným stavom potrebuje pre

jazyk L_n^R aspoň $n+4$ stavov. Všetky argumenty okrem potreby pridania nového počiatočného stavu platia aj pre GSUFA. Teda každý GSUFA pre L_n^R potrebuje aspoň $n+3$ stavov.

□

Automaty z triedy SUFA majú iba jeden počiatočný stav. Preto sa pri realizácii reverzu pomocou automatov z tejto triedy môže stať, že potrebujeme pridať nový počiatočný stav (ak má automat viac akceptačných stavov). Po pridaní nového stavu ale výsledný automat nemusí byť SUFA. Ak je možné v automate A akceptovať slovo w dvoma rôznymi cestami (t.j. do dvoch rôznych akceptačných stavov, lebo A je SUFA), potom pre w^R existujú dve cesty z nové počiatočného stavu $q_0^{'}$ do q_0. Riešenie tejto situácie prináša dôkaz nasledujúceho tvrdenia.

Tvrdenie 5.6.6. $STATE(SUFA,^R) \leq n^2+1$.

Dôkaz. Nech A je SUFA. Otočením prechodovej logiky a zámenou počiatočných a akceptačných stavov dostaneme automat pre $L(A)^R$, ktorý je GSUFA a má $|A|$ stavov. K tomuto automatu nájdeme ekvivalentný SUFA. Ten má maximálne $|A|^2+1$ stavov.

□

5.7 Komplement

Pomocou známych výsledkov sme dokázali, že nie je možné efektívne realizovať komplement pomocou PNA.

Máme automat A. Chceme nájsť taký automat C, pre ktorý $L(C) = L(A)^c$.

Komplement pomocou NFA a DFA

V tejto časti uvedieme známe výsledky o zložitosti komplementu pre triedy NFA a DFA.

Tvrdenie 5.7.1. [4] $STATE(NFA,^c) \geq 2^{n-2}$.

Dôkaz. Definujeme postupnosť jazykov $L_n = \{a,b\}^* a \{a,b\}^n b \{a,b\}^*$. Pe jazyk L_n existuje $n+3$–stavový NFA A_n. Ten funguje tak, že uhádne miesto, kde začína stredná časť slova, $a\{a,b\}^n b$ (na to stačí jeden stav). Potom v nasledujúcich $n+1$ stavoch overí strednú časť slova. V poslednom stave spracuje poslednú časť slova, $\{a,b\}^*$.

Ak chce automat C akceptovať komplement jazyka $L(A)$, musí overiť, že vstupné slovo neobsahuje podslovo v tvare $a\{a,b\}^n b$. Automat C si musí po prečítaní znaku a zapamätať ďalších n znakov. Je dokázané [4], že za týmto účelom potrebuje aspoň 2^{n-2} stavov. □

Komplement pomocou DFA vieme realizovať veľmi efektívne a jednoducho.

Tvrdenie 5.7.2. $STATE(DFA,^c) = n$.

Dôkaz. Automat C bude rovnaký ako automat A, ale jeho akceptačné stavy budú neakceptačné stavy automatu A. C teda akceptuje všetky slová, ktoré nie sú z L(A). C je DFA, pretože A je DFA (a od DFA vyžadujeme, aby bol úplný).

Uvažujme ľubovoľný regulárny jazyk L a k nemu najmenší DFA A, ktorý ho akceptuje. Pre L^c existuje automat C. Pripusťme, aby mal C menej ako $|A|$ stavov. Potom ale vieme urobiť komplement jazyka L^c konštrukciou popísanou vyššie a tým získame DFA pre jazyk L, ktorý má menej ako $|A|$ stavov. To je však spor s tým, že A bol minimálny pre L. Preto na realizáciu komplementu potrebujeme aspoň n stavov. □

Komplement pomocou UFA, MDFA, SUFA, GSUFA, FNA a LNA

Pre tieto triedy sa nám nepodarilo ukázať, či môžu efektívne realizovať komplement. Prikláňame sa skôr k názoru, že to nie je možné. V tejto časti vedieme vzťahy medzi otvorenými otázkami o zložitosti kompelmentu a iných operácii (alebo jazykov).

Pre triedu UFA nie je stále vyriešená realizácia zjednotenia. Ak sa ukáže, že nie je možné efektívne realizovať zjednotenie pomocou UFA, bude to znamenať, že nie je možné efektívne realizovať ani komplement pomocou tejto triedy.

Pre triedu LNA nie je stále vyriešená realizácia prieniku. Ak sa ukáže, že nie je možné efektívne realizovať prienik pomocou LNA, bude to znamenať, že nie je možné efektívne realizovať ani komplement pomocou tejto triedy.

Komplement pomocou PNA

Bolo ukázané, že vo všeobecnosti nie je možné efektívne realizovať komplement pomocou NFA. Na dôkaz toho, že ani PNA nemôže vo všeobecnosti efektívne realizovať kompelement použijeme rovnaký jazyk, ako v tvrdení 5.7.1.

Tvrdenie 5.7.3. $STATE(PNA,^c) \geq 2^{n-2}$.

Dôkaz. Pre jazyk $L_n = \{a,b\}^* a \{a,b\}^n b \{a,b\}^*$ existuje $n+3$–stavový NFA A_n. Je dokázané [4], že každý NFA (a teda aj PNA) pre L_n^c potrebuje aspoň 2^{n-2} stavov. Zostáva ukázať, že A_n je PNA.

Podľa štrukturálnej vlastnosti automatov PNA z časti 4.1 vieme, že A_n je PNA ak v ňom neexistuje stav q, z ktorého sa možno dostať na slovo w opäť do q viac ako jednou cestou. A_n obsahuje iba dva cykly, jeden na počiatočnom stave a druhý na akceptačnom. Ide o cykly dĺžky 1, ktoré sú vždy jednoznačné. Automat A_n je teda PNA. Preto PNA potrebuje na realizáciu komplementu aspoň 2^{n-2} stavov. □

5.8 Zhrnutie výsledkov zo zložitosti operácií

V kapitolách 3 a 4 sme popísali 7 modelov automatov. Menovite UFA, FNA, LNA, PNA, SUFA, GSUFA a MDFA. V kapitole 5 sme uviedli 7 skúmaných operácií na

modeloch DFA a NFA. Menovite zjednotenie, prienik, komplement, reverz, zreťazenie, iteráciu a kladnú iteráciu. K týmto sme pridali ešte značkované zreťazenie. Pre väčšinu operácií sme vyriešili ich zložitosť na mnohých modeloch.

Toto viedlo k množstvu výsledkov, ktorých prehľad prinášame v tejto časti v tabuľke 5.1, ktorú možno považovať za rozšírenie tabuľky uvedenej v [4].

Tabuľka obsahuje skúmané modely uvedené v stĺpcoch a operácie v riadkoch. Jedna bunka tabuľky vyjadruje zložitosť operácie v riadku na modeli uvedenom v stĺpci. V každom stĺpci predpokladáme, že máme n – stavový automat A pre jazyk $L(A)$ a m – stavový automat B pre jazyk $L(B)$. Pritom A aj B patria do triedy automatov v príslušnom stĺpci. V niektorých bunkách uvádzame iba nutný, resp. postačujúci, počet stavov na realizáciu danej operácie. Polynóm vždy vyjadruje, že sme našli konštruckiu, pomocou ktorej je možné realizovať danú operáciu a teda ide o hornú hranicu. Exponenciálna funkcia v bunke tabuľky vždy vyjadruje dolný odhad. To či je uvedený odhad tesný, je možné nájsť v predchádzajúcich častiach.

Operácia	DFA	UFA	MDFA	SUFA	GSUFA	FNA	LNA	PNA	NFA
$L(A) \cup L(B)$	mn		$\boldsymbol{m+n}$	$\boldsymbol{m+n+1}$	$\boldsymbol{m+n}$	$\boldsymbol{m+n+1}$	$\boldsymbol{m+n+1}$	$\boldsymbol{m+n+1}$	$m+n+1$
$L(A) \cap L(B)$	mn	$\boldsymbol{mn}$	$\boldsymbol{mn}$	$\boldsymbol{mn}$	$\boldsymbol{mn}$	$\boldsymbol{mn}$		$\boldsymbol{mn}$	mn
$L(A)^c$	n							$\boldsymbol{2^{n-2}}$	2^{n-2}
$L(A)^R$	2^n	$\boldsymbol{n+1}$	2^{n-1}	$\boldsymbol{n^2+1}$	$\boldsymbol{n}$	$\boldsymbol{n+1}$	$\boldsymbol{n+1}$	$\boldsymbol{n+1}$	$n+1$
$L(A).L(B)$	$(2m-1)2^{n-1}$			$\boldsymbol{n+nm}$	$\boldsymbol{n(1+m^2)}$	$\boldsymbol{2^{\Omega(n^{1/96}+m)}}$	$\boldsymbol{2^{\Omega(n^{1/96}+m^{1/96})}}$	$\boldsymbol{m+n}$	$m+n$
$L(A)^*$	3.2^{n-2}	$\boldsymbol{2^{n-1}-1}$	$\boldsymbol{2^{n-2}-1}$	$\boldsymbol{2^{n-1}-1}$	$\boldsymbol{2^{n-1}-1}$	$\boldsymbol{2^{n-1}-1}$	$\boldsymbol{2^{n-1}-1}$	$\boldsymbol{2^{n-1}-1}$	$n+1$
$L(A)^+$	$\boldsymbol{3.2^{n-3}}$	$\boldsymbol{2^{n-1}-2}$	$\boldsymbol{2^{n-2}-2}$	$\boldsymbol{2^{n-1}-2}$	$\boldsymbol{2^{n-1}-2}$	$\boldsymbol{2^{n-1}-2}$	$\boldsymbol{2^{n-1}-2}$	$\boldsymbol{2^{n-1}-2}$	n
$L(A)\#L(B)$	$\boldsymbol{m+n}$	$\boldsymbol{m+n}$	$\boldsymbol{mn+m}$	$\boldsymbol{n+nm}$	$\boldsymbol{n(1+m^2)}$	$\boldsymbol{m+n}$		$\boldsymbol{m+n}$	$\boldsymbol{m+n}$

Tabuľka 5.1: Zložitosť operácií na nekonečných jazykoch nad ľubovoľnou abecedou.

Kapitola 6

Oddelenie tried SUFA a FNA

V kapitole 3 sme uviedli viacero spôsobov, ako je možné oddeliť dve triedy automatov. V tejto kapitole popíšeme našu novú techniku na oddeľovanie tried a použijeme ju na dôkaz tvrdenia SUFA $<_P$ FNA.

Techniky uvedené v tejto práci majú jednu spoločnú črtu. Vždy sa zameriavajú na jednu triedu automatov. Ide o techniky ako ukázať, že nejaký model potrebuje na akceptovanie daného jazyka istý počet stavov. Najznámejšou takouto technikou je Myhill - Nerodova veta, ktorá dokáže presne určiť počet stavov najmenšieho DFA pre jazyk L. Pre model UFA existuje tvrdenie [22] (tvrdenie 3.3.1) na dolný odhad stavov potrebných pre UFA na akceptovanie daného jazyka. Na oddelenie tried FNA a LNA (resp. LNA a PNA) boli použité poznatky z komunikačnej zložitosti [7]. Nami navrhnutá technika nie je zviazaná so žiadnou konkrétnou triedou automatov.

V kapitole 5 sme na dôkaz toho, že nie je možné efektívne realizovať nejakú operáciu pomocou vybranej triedy, využili fakt, že vybraná trieda je menej úsporná ako nejaká iná trieda. Príkladom takéhoto postupu sú tvrdenia 5.4.8, 5.4.10, 5.5.4 a ďalšie. Nemenej zaujímavé je pozrieť sa na vzťah medzi zložitosťou operácii a hierarchickým usporiadaním tried z druhej strany. Fakt, že nie je možné efektívne realizovať danú operáciu pomocou automatov z vybranej triedy je možné použiť na oddelenie tejto triedy od inej triedy.

Vyslovili a dokázali sme nasledujúce tvrdenie.

Tvrdenie 6.0.1 (Veta o oddelení)**.** *Nech C_1 a C_2 sú dve ľubovoľné triedy konečných automatov a op je operácia. Potom*

$$STATE(C_1, op) = poly(n) \wedge STATE(C_2, op) \neq poly(n) \Rightarrow C_1 \neq_P C_2$$

Dôkaz. Postupujme sporom. Predpokladajme, že existuje operácie *op* taká, že $STATE(C_1, op) = poly(n)$ a $STATE(C_2, op) \neq poly(n)$. Ďalej predpokladajme, že $C_1 =_P C_2$.

Vezmime ľubovoľný automat $A \in C_2$. Keďže $C_1 =_P C_2$, potom existuje $A' \in C_1$ taký, že $L(A') = L(A)$ a navyše $|A'| = poly(|A|)$. Nájdeme automat $C' \in C_1$, pre ktorý $L(C') = op(L(A'))$. Vieme, že existuje taký C', že $|C'| = poly(|A'|)$, pretože $STATE(C_1, op) = poly(n)$. K automatu C' potom nájdeme ekvivalentný automat C

z triedy C_2. Keďže $C_1 =_P C_2$, existuje taký C, ktorý má polynomiálny počet stavov vzhľadom na počet stavov automatu C'. Celkovo teda dostávame automat C taký, že $L(C) = L(C') = op(L(A')) = op(L(A))$ a tiež $|C| = poly(|C'|) = poly(|A'|) = poly(|A|)$. Pre každý automat A z triedy C_2 sme našli C z triedy C_2 taký, že $L(C) = op(L(A))$ a $|C| = poly(|A|)$. To je ale v spore s tým, že $STATE(C_2, op) \neq poly(n)$. □

Veta od oddelení platí pre unárne aj binárne operácie. Dôkaz vety pre binárne operácie je analogický uvedenému dôkazu pre unárne operácie.

6.1 SUFA $<_P$ FNA

Autor modelu SUFA (Leung [16]) nepopísal presne vzťah medzi triedami SUFA a FNA. Je zrejmé, že $SUFA \leq_P FNA$. Je ale možné, aby $SUFA =_P FNA$? Práve pomocou skúmania zložitosti operácií na týchto modeloch sme schopní dokázať, že nie je možné, aby triedy SUFA a FNA boli polynomiálne ekvivalentné.

Sformulovali sme a dokázali sme všeobecné tvrdenie (vetu o oddelení 6.0.1), ktorého dôsledkom je oddelenie tried SUFA a FNA. Využijeme zistenie, že trieda SUFA efektívne realizuje zreťazenie (tvrdenie 5.4.4) narozdiel od triedy FNA (tvrdenie 5.4.8). Ak by boli tieto triedy polynomiálne ekvivalentné, vedeli by sme efektívne realizovať zreťazenie aj pomocou automatov z triedy FNA.

Ak zvolíme $C_1 = SUFA$ a $C_2 = FNA$, potom operácia zreťazenia "dosvedčuje" oddelenie tried SUFA a FNA.

Tvrdenie 6.1.1. *Trieda FNA je úspornejšia ako trieda SUFA. T.j. SUFA $<_P$ FNA.*

Dôkaz. Je zrejmé, že SUFA $\leq_P$ FNA, pretože slovo v automate SUFA možno akceptovať najviac toľkými spôsobmi, koľko má automat akceptačných stavov.

Vieme, že $STATE(SUFA, .) = poly(n, m)$ (tvrdenie 5.4.4) ale $STATE(FNA, .) \neq poly(n, m)$ (tvrdenie 5.4.8). Potom podľa vety o oddelení (6.0.1) musí platiť SUFA $\neq_P$ FNA. Preto SUFA $<_P$ FNA. □

Doposiaľ používané techniky na oddelenie tried vždy prezentovali jazyk, ktorý dané dve triedy oddeľuje. My sme tiež našli takýto jazyk. Dokazujeme to v nasledujúcom tvrdení.

Tvrdenie 6.1.2. *Nech $L = L_r^t$ [7]. Pre jazyk L existuje FNA, ktorý má poly(r) stavov, ale každý SUFA pre L potrebuje exponenciálne veľa stavov vzhľadom na r.*

Dôkaz. Nech existuje SUFA pre jazyk $L = L_r^t$, ktorý má poly(r) stavov. Položme $L(A_r) = \{w_1...w_m \mid w_m \in L, |w_i| = 2t\}$ a $L(B_r) = \{w_1...w_m \mid |w_i| = 2t\}$. Automat B_r je DFA, ktorý má $2t$ stavov. Je zrejmé, že $L(A_r) = L(B_r).L$. Keďže SUFA efektívne realizujú zreťazenie, tak potom pre $L(A_r)$ existuje SUFA, ktorý má poly(r) stavov. Potom ale z rovnakého dôvodu aj pre jazyk $L(A_r).L(B_r)$ existuje SUFA, ktorý má poly(r) stavov. To je však v spore s tým, že $L(A_r).L(B_r) = \exists_1(L)$. □

Kapitola 7

Záver

V práci sme priniesli zhrnutie pojmov z oblasti popisnej zložitosti konečných automatov a merania nedeterminizmu. Ukázali sme modely s rôznym stupňom nejednoznačnosti (nedeterminizmu) a popísali sme spôsob, akým modely zaraďovať do hierarchie. Na konkrétnych jazykoch a automatoch sme ukázali známe vzťahy medzi triedami automatov. Po zhrnutí známych výsledkov sme sa dopracovali k týmto hierarchiám a vzťahom.

$$DFA <_P UFA <_P SUFA =_P GSUFA \leq_P FNA <_P LNA <_P PNA <_P NFA$$

$$DFA <_P MDF <_P SUFA =_P GSUFA \leq_P FNA <_P LNA <_P PNA <_P NFA$$

$$UFA \neq_P MDFA$$

Ďalej sme sa venovali zložitosti operácii na rôznych modeloch konečných automatov. Rozšírili sme známe výsledky zo zložitosti operácii pre DFA a NFA o nové výsledky pre ďalšie triedy. Takmer pre všetky nami spomenuté triedy sme vyriešili zložitosť zjendotenia a prieniku. Pre všetky triedy sme vyriešili otázku reverzu a tiež sme ukázali, že žiadna trieda okrem NFA nemôže efektívne realizovať iteráciu ani kladnú iteráciu. Našli sme spôsob ako efektívne realizovať zreťazenie pomocou SUFA a GSUFA a ukázali sme, že nie je možné efektívne realizovať zreťazenie pomocou tried FNA a LNA. Venovali sme sa aj realizácii značkovaného zreťazenia. Všetky naše výsledky sme zhrnuli v tabuľke 5.1.

V kapitole 6 sme prezentovali našu novú techniku na oddeľovanie tried konečných automatov. V práci sme spomenuli Schmidtovu vetu [22](tvrdenie 3.3.1), ktorá dáva dolný odhad pre veľkosť UFA. Iné dolné odhady boli dokázané pomocou komunikačnej zložitosti [6, 8]. Naša technika však nie je závislá na žiadnej konkrétnej triede automatov. Pri oddeľovaní tried pomocou tejto techniky sa opierame o poznatky zo zložitosti operácií.

V kapitole 6 tiež prinášame riešenie tohoto otvoreného problému. Je zrejmé, že $SUFA \leq_P FNA$. Autor modelu [16] píše, že triedy $SUFA, FNA, PNA$ a ENA

tvoria hierarchiu, keďže každý model je všeobecnejší a môže byť exponenciálne viac úsporný. Pripomína, že stále nie je vyriešené, či $FNA <_P PNA$ (v čase vzniku článku tento problém ešte nebol vyriešný), ale nezaoberá sa otázkou, či $SUFA <_P FNA$. My sme pomocou našej techniky na oddeľovanie tried automatov ukázali, že $SUFA <_P FNA$. Ukázali sme tiež, že existuje konečný jazyk L, pre ktorý je trieda FNA úspornejšia ako trieda SUFA.

Priestor na ďalší výskum vidíme v určení vzťahu medzi k nejednoznačnými a $k+1$ nejednoznačnými automatmi. [8] navrhujú študovať štruktúru triedy FNA a ukázať, aká hierarchia existuje vrámci tejto triedy.

Literatúra

[1] Arthur Gill and Lawrence T. Kou. Multiple-entry finite automata. *J. Comput. Syst. Sci.*, 9(1):1–19, 1974.

[2] Jonathan Goldstine, Chandra M. R. Kintala, and Detlef Wotschke. On measuring nondeterminism in regular languages. *Inf. Comput.*, 86(2):179–194, 1990.

[3] Jonathan Goldstine, Hing Leung, and Detlef Wotschke. On the relation between ambiguity and nondeterminism in finite automata. *Inf. Comput.*, 100(2):261–270, 1992.

[4] Markus Holzer and Martin Kutrib. State complexity of basic operations on nondeterministic finite automata. In Jean-Marc Champarnaud and Denis Maurel, editors, *CIAA*, volume 2608 of *Lecture Notes in Computer Science*, pages 148–157. Springer, 2002.

[5] John E. Hopcroft and Jeffrey D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1969.

[6] Juraj Hromkovic, Juhani Karhumäki, Hartmut Klauck, Georg Schnitger, and Sebastian Seibert. Measures of nondeterminism in finite automata. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(76), 2000.

[7] Juraj Hromkovic and Georg Schnitger. Ambiguity and communication. In Susanne Albers and Jean-Yves Marion, editors, *STACS*, volume 09001 of *Dagstuhl Seminar Proceedings*, pages 553–564. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2009.

[8] Juraj Hromkovic, Sebastian Seibert, Juhani Karhumäki, Hartmut Klauck, and Georg Schnitger. Communication complexity method for measuring nondeterminism in finite automata. *Inf. Comput.*, 172(2):202–217, 2002.

[9] Oscar H. Ibarra and Bala Ravikumar. On sparseness, ambiguity and other decision problems for acceptors and transducers. In Burkhard Monien and Guy Vidal-Naquet, editors, *STACS*, volume 210 of *Lecture Notes in Computer Science*, pages 171–179. Springer, 1986.

[10] Lucian Ilie and Sheng Yu. Algorithms for computing small nfas. In *MFCS '02: Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science*, pages 328–340, London, UK, 2002. Springer-Verlag.

[11] Martin Kappes. Descriptional complexity of deterministic finite automata with multiple initial states. *Journal of Automata, Languages and Combinatorics*, 5(3):269–278, 2000.

[12] Chandra M. R. Kintala and Detlef Wotschke. Amounts of nondeterminism in finite automata. *Acta Inf.*, 13:199–204, 1980.

[13] Ernst L. Leiss. Succint representation of regular languages by boolean automata. *Theor. Comput. Sci.*, 13:323–330, 1981.

[14] Hing Leung. Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. *SIAM J. Comput.*, 27(4):1073–1082, 1998.

[15] Hing Leung. Descriptional complexity of nfa of different ambiguity. *Int. J. Found. Comput. Sci.*, 16(5):975–984, 2005.

[16] Hing Leung. Structurally unambiguous finite automata. In Oscar H. Ibarra and Hsu-Chun Yen, editors, *CIAA*, volume 4094 of *Lecture Notes in Computer Science*, pages 198–207. Springer, 2006.

[17] F. R. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Trans. Comput.*, 20:1211 – 1214, 1971.

[18] J. Myhill. Finite automata and the representation of events. Technical Report WADD TR-57-624, Wright Patterson Air Force Base, Ohio, 1957.

[19] A. Nerode. Linear automaton transformations. *Proc. Amer. Math. Soc.*, 9:541–544, 1958.

[20] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Develop.*, 3:115–125, 1959.

[21] Bala Ravikumar and Oscar H. Ibarra. Relating the type of ambiguity of finite automata to the succinctness of their representation. *SIAM J. Comput.*, 18(6):1263–1282, 1989.

[22] Erik Meineche Schmidt. *Succinctness of descriptions of context-free, regular and finite languages.* PhD thesis, Ithaca, NY, USA, 1978.

[23] Paulo A. S. Veloso and Arthur Gill. Some remarks on multiple-entry finite automata. *J. Comput. Syst. Sci.*, 18(3):304–306, 1979.

[24] Sheng Yu and Qingyu Zhuang. On the state complexity of intersection of regular languages. *SIGACT News*, 22(3):52–54, 1991.

[25] Sheng Yu, Qingyu Zhuang, and Kai Salomaa. The state complexities of some basic operations on regular languages. *Theor. Comput. Sci.*, 125(2):315–328, 1994.