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

Vzťah veľkosti minimálnych DKA a NKA

(Bakalárska práca)

Pavol Panák

Vedúci: RNDr. Michal Forišek

Bratislava, 2008

Abstrakt

V práci skúmame vzťahy medzi veľkosťou stavovo-minimálneho deterministického a nedeterministického konečného automatu zodpovedajúceho tomu istému jazyku. Uvádzame experimentálne výsledky pre nedeterministické konečné automaty s malým počtom stavov. Okrem toho, nachádzame príklady n-stavových nedeterministických konečných automatov, ktorých ekvivalentné deterministické konečné automaty majú 2^n stavov. Tiež popisujeme rozličné algoritmy na minimalizáciu konečných automatov.
Kľúčové slová: FSA, NFSA, konečné automaty, minimalizácia, zložité automaty

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

.....

Chcel by som sa poďakovať vedúcemu mojej bakalárskej práce Michalovi Foriškovi za výber vhodnej témy práce a za usmernenia, ktoré mi pri vypracovávaní poskytol.

Obsah

Zoznam obrázkov

Úvod

Modely konečných automatov, či už deterministické alebo nedeterministické, sú skúmané intenzívne už desaťročia. Napriek dlhotrvajúcemu výskumu v tejto oblasti existujú otvorené problémy o konečných automatoch, hlavne o prínose nedeterminizmu v konečných automatoch.

Pri konečných automatoch sa neskúma výpočtová zložitosť, pretože každý výpočet na automate trvá lineárny čas. Z tohto dôvodu, zložitosť konečného automatu je určená popisnou zložitosťou automatu. Existuje viacero mier popisnej zložitosti ako počet stavov alebo počet prechodov. V tejto práci budem používať mieru s počtom stavov.

Vie sa, že nedeterministické konečné automaty sa dajú simulovať deterministickými konečnými automatmi. Avšak, keď pôvodný nedeterministický konečný automat ma n stavov, simulujúci deterministický konečný automat môže mat až exponenciálne veľa stavov (2^n). Teda deterministické konečné automaty vedia za exponenciálneho zvýšenia popisnej zložitosti automatu odsimulovať nedeterministické konečné automaty.

V minulosti sa tiež skúmala otázka, že koľko stavov má minimálny deterministický konečný automat ekvivalentný k n-stavovému minimálnemu nedeterministickému konečnému automatu. Neskôr sa v [Jir01] dokázalo, že pre každé dve kladné celé čísla n a α také, že $1 \leq n \leq \alpha \leq 2^n$, existuje minimálny n-stavový nedeterministický konečný automat, ktorého ekvivalentný minimálny deterministický konečný automat má presne α stavov.

V tejto práci som sa zaoberal podobnou otázkou: Koľko je minimálnych α-stavových deterministických konečných automatov ekvivalentných k nejakému minimálnemu n-stavovému nedeterministickému konečnému automatu, kde α je z intervalu $[n, 2^n]$? Samozrejme n-stavových nedeterministických konečných automatov je nekonečne veľa, preto som považoval množinu stavov automatu a aj množinu vstupnej abecedy za pevne danú. Týmto počet nedeterministických konečných automatov klesol na konečný

počet. Experimentálne sa mi podarilo nájsť odpoveď na túto otázku pre nedeterministické konečné automaty s malým počtom stavov. Taktiež uvediem nájdené príklady takých minimálnych n-stavových nedeterministických konečných automatov, ktorých ekvivalentný minimálny deterministický konečný automat je 2^n-stavový.

Túto prácu som rozdelil nasledovne: V kapitole 1 sú základne definície (minimálnych) konečných automatov. V ďalšej kapitole 2 sa je možné dočítať o probléme minimalizácii konečných automatov ako aj o algoritmoch, ktoré s problematikou súvisia a ktoré som implementoval. Hlavné výsledky sú v kapitole 3. V kapitole 4 sú informácie ako som generoval výsledky a bližší popis implementácie programu, ktorý som na tento účel naprogramoval.

Kapitola 1

Základné definície

Na začiatok uvediem základné definície pojmov z oblasti konečných automatov. Väčšina z nich definuje elementárne pojmy, s ktorými sa čitateľ s veľkou pravdepodobnosťou už stretol.

Definícia 1.0.1. ***Deterministický konečný automat*** *(DKA) A je pätica $(K, \Sigma, \delta, g_0, F)$, kde K je konečná množina stavov, Σ je konečná množina vstupnej abecedy, $g_0 \in K$ je začiatočný stav, $F \subseteq K$ je množina akceptačných stavov a $\delta : K \times \Sigma \to K$ je prechodová funkcia.*

Definícia 1.0.2. ***Konfigurácia*** *deterministického konečného automatu A je dvojica $(g, w) \in K \times \Sigma^*$, kde q je stav automatu A a w je ešte nespracovaná časť vstupného slova.*

Definícia 1.0.3. ***Krok výpočtu*** *deterministického konečného automatu A je relácia $\vdash_A$ na konfiguráciách definovaná nasledovne: $(g, av) \vdash_A (p, v) \iff p = \delta(g, a)$.*

Poznámka 1.0.1. *$\vdash^*$ bude označovať reflexívno-tranzitívny uzáver relácie $\vdash$. Σ^* bude označovať množinu všetkých slov nad abecedou Σ. ϵ označuje prázdne slovo.*

Definícia 1.0.4. ***Jazyk*** *akceptovaný deterministickým konečným automatom A je množina $L(A) = \{w | \exists g_F \in F : (g_0, w) \vdash_A^* (g_F, \epsilon)\}$.*

Definícia 1.0.5. ***Nedeterministický konečný automat*** *(NKA) A je pätica $(K, \Sigma, \delta, g_0, F)$, kde K je konečná množina stavov, Σ je konečná množina vstupnej abecedy, $g_0 \in K$ je začiatočný stav, $F \subseteq K$ je množina akceptačných stavov a $\delta : K \times \Sigma \to 2^K$ je prechodová funkcia.*

Poznámka 1.0.2. *Moja definícia prechodovej funkcie pre NKA sa mierne líši od "bežnej" definície prechodovej funkcie, ktorá je:* $\delta : K \times (\Sigma \cup \epsilon) \to 2^K$. 2^K *je množina všetkých podmnožín množiny* K.

Definícia 1.0.6. ***Krok výpočtu*** *nedeterministického konečného automatu* A *je relácia* $\vdash_A$ *na konfiguráciách definovaná nasledovne:* $(g, av) \vdash_A (p, v) \iff p \in \delta(g, a)$.

Poznámka 1.0.3. *Definície konfigurácie a jazyka nedeterministického konečného automatu sú totožné definíciám ako pri deterministickom konečnom jazyku.*

Definícia 1.0.7. ***Veľkosť*** *konečného automatu* A *je číslo* $|K|$ *(t.j. mohutnosť množiny stavov automatu* A*). Označenie:* $|A|$

Definícia 1.0.8. ***Stav*** $g \in K$ ***je dosiahnuteľný****, ak* $\exists w \in \Sigma^*$ *také, že* $(g_0, w) \vdash^* (g, \epsilon)$.

Definícia 1.0.9. *Konečný automat* A *je* ***súvislý****, ak každý stav* $g \in K$ *je dosiahnuteľný.*

Definícia 1.0.10. ***Minimálny deterministický konečný automat*** A *je deterministický konečný automat pre ktorý platí* $\forall$ *DKA* $B : L(A) = L(B) \Rightarrow |A| \leq |B|$

Definícia 1.0.11. ***Minimálny nedeterministický konečný automat*** A *je nedeterministický konečný automat pre ktorý platí* $\forall$ *NKA* $B : L(A) = L(B) \Rightarrow |A| \leq |B|$

Kapitola 2

Algoritmy na DKA, NKA

V nasledujúcej časti popíšem znalosti z oblasti minimálnych deterministických a nedeterministických konečných automatov. Taktiež ukážem základné algoritmy z tejto oblasti, ktoré som aj implementoval.

Treba si uvedomiť, že ak budem spomínať nedeterministický konečný automat mám na mysli automat, ktorý neobsahuje v prechodovej funkcií prechody na ϵ. Kedže ďalej budem uvažovať len o minimálnych nedeteministických konečných automatoch, nemá zmysel uvažovať o automatoch, ktoré obsahujú prechody na ϵ. Je zrejmé, že ak existuje minimálny nedeterministický konečný automat s prechodmi na ϵ, vieme k nemu vytvoriť automat, ktorý neobsahuje prechody na ϵ a má rovnaký počet stavov.

2.1 Realizácia DKA, NKA

Ľubovoľný deterministický aj nedeterministický konečný automat A je určený usporiadanou päticou $(K, \Sigma, \delta, g_0, F)$. Teda automat A môže mať rozličné množiny stavov K, množiny vstupnej abecedy Σ a rozličné začiatočné stavy g_0. Avšak nie je praktické rozlišovať medzi automatmi na základe rozdielnosti množiny stavov, množiny vstupnej abecedy a začiatočných stavov, preto budem predpokladať, že K, Σ a g_0 sú vopred definované: $K = Z_{|K|}$ (množina stavov je množina zvyškových tried modulo $|K|$ alebo množina nezáporných celých čísel menších ako $|K|$), $\Sigma = Z_{|\Sigma|}$ (množina vstupnej abecedy je množina zvyškových tried modulo $|\Sigma|$) a $g_0 = 0$ (začiatočný stav je 0). Potom pre automat A stačí zadefinovať prechodovú funkciu δ a množinu akceptačných stavov F.

Realizácia deterministického konečného automatu je implementovaná v

triede DKA (v súbore DKA.h, DKA.cpp). Konkrétna inštancia tejto triedy reprezentuje deterministický konečný automat. V triede sú uložené informácie o veľkosti množiny stavov, veľkosti množiny vstupnej abecedy a tabuľke, ktorá obsahuje definíciu prechodovej funkcie. Táto tabuľka je realizovaná dynamickým polom integer-ov. Taktiež pre každý stav je uložená informácia, či je akceptačný. Trieda DKA obsahuje aj metódy, z ktorých by som spomenul aspoň dve:

void minimize() : Metóda transformuje inštanciu DKA na inštanciu DKA, ktorej reprezentácia je minimálny deterministický konečný automat ekvivalentný pôvodnému automatu. Podrobnejší popis algoritmu je v časti 2.3.2 na strane 11.

bool is_equal(DKA * dka) : Vstupom metódy (dka) je minimálny deterministický konečný automat, výstupom je odpoveď na otázku, či je deterministický konečný automat ekvivalentný vstupnému automatu (dka). Metóda predpokladá, že obidva deterministické konečné automaty sú minimálne. Metóda hľadá izomorfizmus medzi stavmi obidvoch konečných automatoch, ak taký izomorfizmus nájde, potom sú automaty ekvivalentné.

Realizácia nedeterministického konečného automatu je implementovaná v triede NKA (v súbore NKA.h, NKA.cpp). Konkrétna inštancia tejto triedy reprezentuje nedeterministický konečný automat. V triede, rovnako ako pri DKA, sú uložené informácie o veľkosti množiny stavov, veľkosti množiny vstupnej abecedy a tabuľke, ktorá obsahuje definíciu prechodovej funkcie NKA. Tato tabuľka je realizovaná viac-rozmerným dynamickým polom logických hodnôt(bool). Taktiež pre každý stav je uložená informácia, či je akceptačný. Trieda NKA obsahuje aj metódy:

DKA * convert2DKA() : Metóda vytvorí nový deterministický konečný automat, ktorý je ekvivalentný nedeterministickému konečnému automatu. Bližší popis algoritmu možno nájsť v časti 2.2 na strane 7.

DKA * convert2minDKA() : Je len modifikácia predošlej metódy, ktorej výstupný DKA sa navyše minimalizuje.

int generateNextNKA() : Modifikuje nedeterministický konečný automat na nasledovný NKA, ktorý nasleduje v abecednom usporiadaní.

void generateNextNKA(double prp, double pra) : Vygeneruje nový nedeterministický konečný automat, kde "prp" je hodnota určujúca pravdepodobnosť existencie prechodu medzi stavmi a "pra" určuje pravdepodobnosť, že stav je akceptačný.

void generateNextNKA(double prp, double prp0, double pra) : Vygeneruje nový nedeterministický konečný automat, kde "prp" je hodnota určujúca pravdepodobnosť existencie prechodu medzi stavmi na písmeno z množiny $\Sigma - \{0\}$, "prp0" je hodnota určujúca pravdepodobnosť existencie prechodu na písmeno 0 a "pra" určuje pravdepodobnosť, že stav je akceptačný. Okrem toho sa vytvorí cyklus cez všetky stavy na písmeno 0.

void generateNextNKA(int ohr_d, double pra) : Vygeneruje nový nedeterministický konečný automat, kde "ohr_d" ohraničuje veľkosť prechodovej funkcie a "pra" určuje pravdepodobnosť, že stav je akceptačný.

bool have_cycle() : Metoda zisťuje, či NKA obsahuje cyklus cez všetky stavy.

bool is_minimal(DKA * dka) : Metoda zisťuje, či je NKA ozaj minimálny. V dôsledku veľkej časovej zložitosti, túto metódu možno použiť iba pre NKA, ktorých veľkosť je veľmi malá. Popis algoritmu je v časti 2.4 na strane 12.

2.2 Transformácia NKA na DKA

Definícia 2.2.1. *Konečný automat A je **ekvivalentný** konečnému automatu B, ak $L(A) = L(B)$.*

Veta 2.2.1. *K ľubovoľnému nedeterministickému konečnému automatu A existuje deterministický konečný automat A_d taký, že $L(A) = L(A_d)$.*

Dôkaz vety 2.2.1 je vlastne algoritmus, ako k danému NKA A skonštruovať ekvivalentný DKA A_d:
Vstup: $A = (K, \Sigma, \delta, g_0, F)$ je NKA.
Výstup: $A_d = (K_d, \Sigma, \delta_d, \{g_0\}, F_d)$ je DKA, ktorý je ekvivalentný k A.
Algoritmus:

1. $K_d = 2^K$, t.j. množina stavov DKA je množina všetkých podmnožín množiny stavov NKA.

2. $\delta_d(Q, a) = \{\delta(p, a) | p \in Q\}$, kde $a \in \Sigma$ a $Q \in K_d$.

3. $F_d = \{Q | Q \cap F \neq \emptyset\}$, kde $Q \in K_d$

4. Takto zostrojení DKA A_d môže obsahovať nedosiahnuteľne stavy, ktoré sa následne odstránia.

Časová zložitosť: $O(2^{|K|}) = O(2^n)$, pretože všetkých podmnožín množiny stavov NKA A je $2^{|K|}$.

Predošlý algoritmus som implementoval v triede NKA.cpp v metóde convert2DKA().

2.3 Minimalizácia DKA

Binárna relácia R na množine S je množina dvojíc prvkov zo S. Pre zjednodušenie ak $(x, y) \in R$ zapíšem daný fakt ako xRy. Relácia ekvivalencie(relácia s vlastnosťami: reflexívnosť, symetrickosť, tranzitivnosť) má dôležitú vlastnosť, rozdeľuje množinu S na triedy ekvivalencie, ktorých počet môže byť aj nekonečný. Triedu ekvivalencie do ktorej patrí prvok $x \in S$ označíme $[x]$. V našom prípade za množinu S zoberieme množinu Σ^* a na nej bude definovaná relácia R s konečným počtom tried ekvivalencie.

Definícia 2.3.1. *Binárna relácia R je **sprava invariantná** vzhľadom na operáciu zreťazenia, ak platí: $xRy \iff \forall z : xzRyz$*

Veta 2.3.1. *(Myhill-Nerode) Nasledujúce tvrdenia sú ekvivalentné:*

1. *jazyk $L \subseteq \Sigma^*$ je akceptovateľný nejakým konečným automatom.*

2. *L je zjednotením nejakých tried ekvivalencie nejakej sprava invariantnej relácie ekvivalencie s konečným počtom tried.*

3. *Nech je R relácia ekvivalencie s konečným počtom tried definovaná nasledovne: $xRy \iff \forall z \in \Sigma^* : (xz \in L \iff yz \in L)$*

Dôkaz možno nájsť v knihe [HU69]. V pôvodnom vydaní možno nájsť v [Myh57] a [Ner58].

Poznámka 2.3.1. *Pre jazyk L z vety 2.3.1 a k nemu definovanú reláciu R z časti 3 možno zostrojiť minimálny deterministický konečný automat A' nasledovne: $A' = (K', \Sigma, \delta', g'_0, F')$, kde $K' = \{[x] | x \in \Sigma^*\}$ (t.j. množina stavov DKA A' je množina tried relácie R), $\delta'([x], z) = [xz]$, $g'_0 = [\epsilon]$ a $F' = \{[x] | x \in L\}$.*

Veta 2.3.1 vlastne hovorí, že ak máme daný konečný automat A, ktorý definuje jazyk $L(A) = L$, potom existuje relácia ekvivalencie R z vety 2.3.1 z časti 3. Úloha nájsť minimálny deterministický konečný automat A' k DKA A, je nájsť deterministický konečný automat, ktorého veľkosť nie je väčšia ako počet tried relácie ekvivalencie R. Ak nájdeme DKA A' s veľkosťou rovnou počtu tried relácie R, tak sme našli minimálny DKA.

Veta 2.3.2. *Minimálny deterministický konečný automat A akceptujúci jazyk L je jednoznačný vzhľadom na izomorfizmus a je definovaný v poznámke 2.3.1 ako DKA A'.*

Dôkaz je dôsledkom vety 2.3.1 a jej dôkazom.

Keďže k danému DKA A hľadáme minimálny DKA A' je nepraktické hľadať reláciu ekvivalencie R nad slovami zo Σ^*, zavedieme nasledujúci pojem.

Definícia 2.3.2. *Dva stavy p, q v deterministickom konečnom automate A sú **ekvivalentné** (zapíšeme pRq) $\iff$ $\forall z \in \Sigma^*$: ak A začne dva výpočty v stavoch p a q so vstupom z, potom súčasne oba výpočty akceptujú alebo zamietnu.*

Pre daný automat A, existuje relácia ekvivalencie R na stavoch automatu A, definovaná: $pRq \iff p$ je ekvivalentné q. Problém nájdenia minimálneho deterministického konečného automatu A' je vlastne nájdenie takého automatu, ktorého počet tried relácie ekvivalencie R na stavoch automatu je rovná veľkosti automatu.

Princíp minimalizačných algoritmov je využitie faktu, že ekvivalentné stavy prechádzajú do ekvivalentných stavov pre každý vstup. Uvediem dva algoritmy minimalizácie deterministického konečného automatu, jeden so zložitosťou $O(n^2)$ a Hopcroftov algoritmus so zložitosťou $O(n \log n)$.

2.3.1 Minimalizačný algoritmus so zložitosťou $O(n^2)$

Väčšina minimalizačných algoritmov pre DKA majú zložitosť $O(n^2)$. Predpokladáme, že DKA A, ktorý chceme minimalizovať, je súvislý a kom-

pletný(t.j. neobsahuje stavy v ktorých by sa mohol zaseknúť). Ak množina akceptačných stavov F_A deterministického konečného automatu A je totožná s $\emptyset$ (množinou stavov K_A automatu A), potom minimálny DKA $A' = (\{g_0\}, \Sigma, \delta, g_0, F_{A'})$ je jedno-stavový automat s množinou akceptačných stavov $F_{A'} = \emptyset$ ($F_{A'} = \{g_0\}$) a $\forall a \in \Sigma : \delta(g_0, a) = g_0$. Takýto automat akceptuje jazyk, buď $L(A) = \emptyset$ alebo $L(A) = \Sigma^*$.

Algoritmus konštruuje reláciu R na stavoch automatu $A = (K, \Sigma, \delta, g_0, F)$ s minimálnym možným počtom tried takú, aby ekvivalentné stavy prechádzali do ekvivalentných stavov pre každý vstup. Na začiatku vytvoríme dve triedy relácie ekvivalencie R na stavoch pôvodného DKA A nasledovne (označenie: R_i je i-ta trieda relácie R): R_0 je množina zamietacích stavov a R_1 je množina akceptačných stavov. Následne pre každú triedu relácie R_i hľadáme dvojicu stavov p, q z triedy R_i a písmeno a z abecedy Σ také, že zo stavu p na písmeno a automat A prejde do stavu p' z triedy R_p, zo stavu q na písmeno a automat A prejde do stavu q' z triedy R_q a $R_p \neq R_q$(teda prejdú do stavov, ktoré nie sú ekvivalentné vzhľadom na R). Ak dané stavy p, q nájdeme, rozdelíme stavy triedy R_i na stavy viacero tried podľa toho do akej triedy daný stav na písmeno a prejde. Postup rozdeľovania viackrát zopakujeme (pokiaľ je to možné) až dostaneme reláciu ekvivalencie R, ktorej triedy definujú minimálny DKA A'. Triedy relácie R budú stavmi automatu A':

Vstup: $A = (K, \Sigma, \delta, g_0, F)$ je DKA.

Výstup: $A_m = (K_m, \Sigma, \delta_m, g_m, F_m)$ je minimálny DKA, ktorý je ekvivalenty k A.

Algoritmus:

1. $R_0 = K - F$ a $R_1 = F$
2. Vytvoríme maticu $M_{|\Sigma| \times |K|}$ takú, že prvok matice m_{ij} je množina R_k, pričom stav, do ktorého automat A prejde zo stavu j na písmeno i, patrí do množiny R_k. Formálne, ak $\delta(j, i) = q$ a $q \in R_k$, potom $m_{ij} = R_k$.
3. Zisťujeme, či existuje množina R_i taká, že niektoré stĺpce matice M prislúchajúce stavom z množiny R_i sú rôzne. Ak taká množina R_i existuje, vytvoríme novú sústavu množín $R_k \ldots R_l$ nasledovne: Každá nová množina bude reprezentovať iný stĺpec matice M. Každý stav z množiny R_i priradíme len do jednej z novovytvorených množín podľa toho,

aký stĺpec v matici M mu prislúcha. Teda ak dva stavy z množiny R_i majú rovnaké stĺpce v matici M ,potom budú v rovnakej novovytvorenej množine. Spolu novovytvorených množín bude presne toľko, koľko je rôznych stĺpcov (prislúchajúcich stavom z množiny R_i) v matici M. Množinu R_i vymažeme.

4. Ak sme v kroku 3 našli nejakú množinu R_i s príslušnou vlastnosťou, potom pokračujeme krokom 2.

5. Množiny R_i tvoria stavy minimálneho DKA ($K_m = \{R_i | R_i$ vznikla v priebehu behu algoritmu a nevymazali sme ju $\}$). Prechodová funkcia je definovaná: $\delta_m(R_i, a) = R_j$, kde $p \in R_i$ a $\delta(p, a) = q$ a $q \in R_j$. Počiatočný stav je $g_m = R_i$, pričom $g_0 \in R_i$. Množina akceptačných stavov je $F_m = \{R_i | \exists q : q \in R_i \wedge q \in F\}$.

Časová zložitosť: $O(n^2)$, zrejme krok 2 potrebuje na vykonanie $O(n)$ času, a cyklus 2-4 sa zopakuje najviac n krát. Najväčšiu časovú zložitosť má algoritmus v prípade ak na vstupe automat A už je minimálny.

Spomínaná časová zložitosť je síce kvadratická, ale nie minimálna. V nasledujúcej časti uvediem algoritmus, ktorého časová zložitosť je menšia.

2.3.2 Hopcroftov algoritmus so zložitosťou $O(n \log n)$

Algoritmus pracuje so sústavou množín R a množinou N. V množine N sú dvojice (C, a), kde C je množina z R a a je písmeno z abecedy Σ. Množina R obsahuje množiny stavov, pričom R reprezentuje očakávané rozdelenie stavov do tried relácie ekvivalencie na stavoch.
Vstup: $A = (K, \Sigma, \delta, g_0, F)$ je DKA.
Výstup: $A_m = (K_m, \Sigma, \delta_m, g_m, F_m)$ je minimálny DKA, ktorý je ekvivalentný k A.
Algoritmus:

1. Skonštruujeme inverznú prechodovú funkciu: $\delta^{-1}(q, a) = \{p | \delta(p, a) = q\}$, kde $a \in \Sigma$.

2. Skonštruujeme dve množiny $R_0 = K - F$ a $R_1 = F$. Množinu R definujeme $R = \{R_0, R_1\}$. Do množiny N pridáme pre každé písmeno $a \in \Sigma$ dvojicu (R_i, a), kde R_i je menšia množina z dvoch množín R_0, R_1.

3. Vyberieme dvojicu (R_i, a) z množiny N, ktorú z množiny N aj odstránime.

4. Pre každú množinu R_j z množiny R zistíme, či existuje stav $p \in R_j$ taký, že $\delta(p, a) \in R_i$ (túto vlastnosť možno efektívne zistiť z inverznej prechodovej funkcie). Ak taký stav existuje vykonáme nasledujúce podkroky:

   (a) Rozdelíme množinu R_j na dve množiny $R'_j = \{p | \delta(p, a) \in R_i \wedge p \in R_j\}$ a $R''_j = R_j - R'_j$. Teda na stavy z R_j, ktoré na písmeno a prejdú do stavu z množiny R_i (R'_j) alebo mimo množiny R_i (R''_j).

   (b) Ak niektorá z množín R'_j alebo R''_j je prázdna, preskočíme nasledujúce podkroky.

   (c) Množinu R_j nahradíme v množine R množinami R'_j a R''_j.

   (d) Upravíme množinu N pre každé písmeno $b \in \Sigma$ nasledovne: Ak $(R_j, b) \in N$, potom dvojicu nahradíme dvoma dvojicami (R'_j, b) a (R''_j, b). Ak $(R_j, b) \notin N$, potom do množiny N pridáme dvojicu (C, b), kde C je menšia množina z dvojice R'_j a R''_j.

5. Pokiaľ je N neprázdna, pokračujeme krokom 3.

6. Množiny R_i tvoria stavy minimálneho DKA ($K_m = \{R_i | R_i \in R\}$). Prechodová funkcia je definovaná: $\delta_m(R_i, a) = R_j$, kde $p \in R_i$, $\delta(p, a) = q$ a $q \in R_j$. Počiatočný stav je $g_m = R_i$, pričom $g_0 \in R_i$. Množina akceptačných stavov je $F_m = \{R_i | \exists q : q \in R_i \wedge q \in F\}$.

Časová zložitosť: $O(n \log n)$, dôkaz možno nájsť v [Hop71].

Hopcroftov algoritmus som implementoval v triede DKA.cpp v metóde minimize().

2.4 Minimalizácia NKA

Minimalizácia nedeterministického konečného automatu je na rozdiel od minimalizácie DKA ťažký problém. Pri DKA platí, že minimálny DKA je jednoznačný vzhľadom na izomorfizmus. Takáto vlastnosť pri NKA neplatí, teda môže existovať viacero rozličných síce ekvivalentných NKA, ale nie izomorfných. Pri minimalizácii DKA je v súčastnosti známy algoritmus s najmenšou časovou zložitosťou $O(n \log n)$, avšak pri minimalizácii NKA v

súčastnosti nepoznáme efektívny algoritmus riešiaci tento problém. Síce nepoznáme algoritmus, ale sú známe výsledky o zložitosti nasledujúceho problému, ktorý priamo súvisí s hľadaním minimálneho NKA:
Problém: Pre zadanú dvojicu DKA A a cele číslo k, existuje NKA B s počtom stavov najviac k akceptujúci jazyk $L(A)$?

Veta 2.4.1. *Predchádzajúci problém je PSPACE-úplný.*

Dôkaz možno nájsť v článku [JR93]. Dôsledkom tejto vety je, že problém, či NKA je minimálny, patrí do triedy PSPACE. Dokonca ak by pre tento problém existoval efektívny algoritmus, potom by sme taktiež vedeli efektívne riešiť všetky problémy z triedy PSPACE (vyplýva z toho, že problém je PSPACE-úplný).

Keďže problém minimalizácie NKA je v triede PSPACE, existuje algoritmus riešiaci tento problém v polynomiálnom priestore $O(n^k)$. Avšak časová zložitosť tohto algoritmus je až exponenciálna $O(2^{n^k})$.
Vstup: $A = (K, \Sigma, \delta, g_0, F)$ je NKA.
Výstup: Odpoveď, či je NKA A minimálny.
Algoritmus:

1. K NKA A zostrojím minimálny DKA A_d, ktorý je k nemu ekvivalentný.

2. Vygenerujem nový NKA B, ktorého veľkosť je menšia ako veľkosť NKA A.

3. K NKA B zostrojím minimálny DKA B_d, ktorý je k nemu ekvivalentný.

4. Zistím, či DKA A_d a B_d sú ekvivalentné. Kedže obidva automaty sú minimálne, stačí nájsť izomorfizmus na stavoch automatov A_d a B_d. Ak taký izomorfizmus existuje potom automaty sú ekvivalentné, ináč nie sú.

5. Ak automaty A_d a B_d sú ekvivalentné, potom NKA A nie je minimálny a algoritmus skončí.

6. Ak automaty A_d a B_d nie sú ekvivalentné, potom pokračujem krokom 2, pokiaľ nevygenerujem všetky možné NKA B, s veľkosťou menšou ako NKA A. Ak som už vygeneroval všetky možné NKA B, potom NKA A je minimálny.

Časová zložitosť: $O(2^{n^k})$, zrejme všetkých možných NKA menších ako NKA A je až exponenciálne veľa z čoho vyplynie daná zložitosť.

Overovanie, či NKA je minimálny, som implementoval v triede NKA.cpp v metóde is_minimal(DKA * dka). Vstupom metódy (dka) je minimálny DKA, ktorý je ekvivalentný NKA a o ktorom zisťujem, či je minimálny. Avšak táto metóda je použiteľná len pre veľmi malé NKA, z dôvodu veľkej (až exponenciálnej) časovej zložitosti.

Kapitola 3

Experimentálne skúmanie vzťahov

V nasledujúcej kapitole uvediem experimentálne výsledky, ktoré sa mi podarili nadobudnúť. Základnou otázkou, ktorou možno začať znie: Koľko stavov má minimálny DKA A_d, ktorý je ekvivalentný minimálnemu n-stavovému NKA? Vie sa, že DKA A_d ma α stavov, kde α je z intervalu $[n, 2^n]$. Z tohto vyplynie ďalšia otázka: Existuje číslo β z intervalu $[n, 2^n]$ také, že žiaden minimálny DKA A_d nemá β stavov a DKA A_d je ekvivalentný minimálnemu n-stavovému NKA? Táto otázka bola v minulosti taktiež skúmaná a dokonca bolo v nasledujúcej vete dokázané, že také číslo β neexistuje:

Veta 3.0.2. *Pre každé dve kladné celé čísla n a α také, že $1 \leq n \leq \alpha \leq 2^n$, existuje minimálny n-stavový nedeterministický konečný automat, ktorého ekvivalentný minimálny deterministický konečný automat má presne α stavov.*

Dôkaz možno nájsť v [Jir01].

Z predošlého tvrdenia prirodzene vyplynie nasledujúca otázka, ktorou sa budem aj hlbšie zaoberať: Koľko je minimálnych $\alpha(\in [n, 2^n])$-stavových DKA, ktoré sú ekvivalentné minimálnym n-stavovým NKA? Tento problém nazvem **hľadanie rozdelenia n-stavových NKA**. Čiastočné výsledky tohto problému, možno nájsť v nasledujúcej časti 3.1.

Pre jednoduchšie pomenovanie vlastností nedeterministických konečných automatov ešte uvediem nasledovné označenia. n-stavový NKA označím, že je **zložitý**, ak k nemu ekvivalentný minimálny DKA má presne 2^n stavov. Opačne n-stavový NKA je **jednoduchý**, ak k nemu ekvivalentný minimálny

DKA má presne n stavov. V tomto kontexte zadefinujem ešte dva pojmy, ktoré porovnávajú medzi sebou n-stavové NKA. n-stavový NKA A je **zložitejší** ako n-stavový NKA B, ak k nim ekvivalentné minimálne DKA sú A_d, B_d ($L(A) = L(A_d)$ a $L(B) = L(B_d)$) a navyše veľkosť DKA A_d je väčšia ako veľkosť DKA B_d. Podobne n-stavový NKA A je **jednoduchší** ako n-stavový NKA B, ak k nim ekvivalentné minimálne DKA sú A_d, B_d ($L(A) = L(A_d)$ a $L(B) = L(B_d)$) a navyše veľkosť DKA A_d je menšia ako veľkosť DKA B_d.

Pri hľadaní rozdelenia n-stavových NKA som hľadal aj zložité NKA, ktorých minimálny DKA má 2^n stavov. Príklady takýchto zložitých automatov možno nájsť v časti 3.2.

3.1 Hľadanie rozdelenia automatov

Hľadanie rozdelenia n-stavových NKA realizujem nasledovným výpočtom: Postupne generujeme všetky alebo iba niektoré n-stavové NKA, následne ku každému z nich nájdeme ekvivalentný minimálny DKA A_d a označíme veľkosť A_d. Nakoniec pre všetky $\alpha \in [n, 2^n]$ spočítame koľko α-stavových DKA A_d sme počas výpočtu vygenerovali. Týmto postupom dostaneme (približné) rozdelenie n-stavových NKA.

Pri experimentálnom riešení tohto problému narážame na to, že počet n-stavových NKA rastie superexponenciálne v závislosti od n, konkrétne: $2^{|K|^2|\Sigma|+|K|}$ a keďže $|K| = n$, n-stavových NKA je $2^{n^2|\Sigma|+n}$. Zrejme pre väčšie n už nemôžeme vygenerovať všetky n-stavové NKA, preto v týchto prípadoch zadaný počet krát generujeme náhodné n-stavové NKA.

Ďalší problém sa objavuje pre väčšie n-stavové NKA, pretože nevieme efektívne zistiť, či n-stavový NKA je minimálny. Časovú zložitosť problému minimálnosti NKA som bližšie popísal v časti 2.4 na strane 12. Pri hľadaní rozdelenia n-stavových NKA avšak chceme, aby NKA boli minimálne. O čiastočnom riešení tohto problému hovorí nasledujúca lema:

Lema 3.1.1. *Nech NKA A je n-stavový, k nemu ekvivalentný minimálny DKA A_d je d-stavový a nech $d \in (2^{n-1}; 2^n]$, potom NKA A je minimálny.*

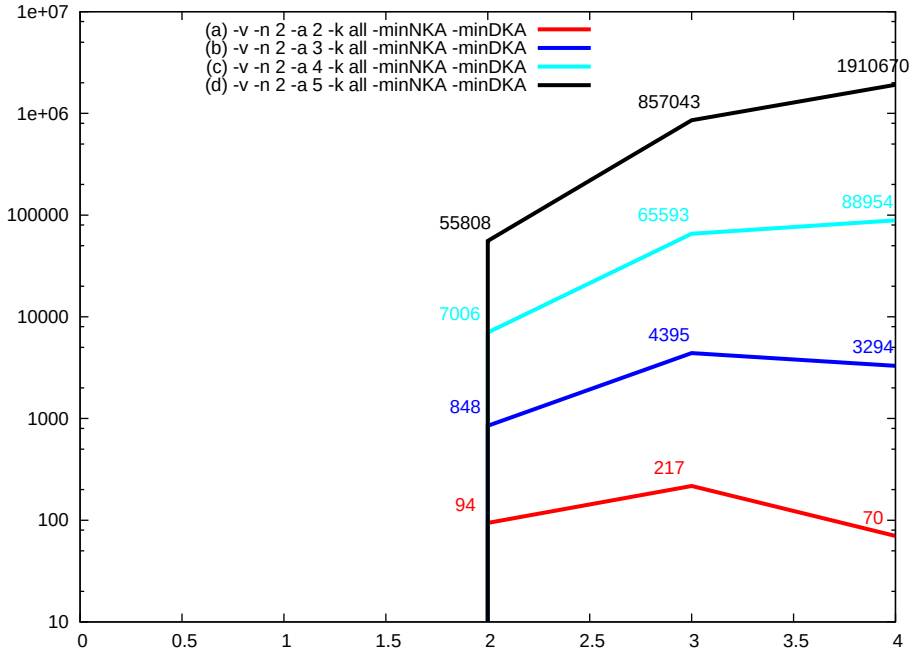
Dôkaz: sporom, nech existuje NKA A', ktorý je ekvivalentný s DKA A_d a menší od A. Bez ujmy na všeobecnosti, nech A' je $(n-1)$-stavový. Avšak k nemu ekvivalentný DKA môže mat maximálne 2^{n-1} stavov a DKA A_d je minimálny s viac ako 2^{n-1} stavmi, čo je spor.

Dôsledok tejto lemy je, že pre väčšie n-stavové NKA dostaneme vyššie spomenutým výpočtom korektné rozdelenie n-stavových NKA len na intervale $(2^{n-1}, 2^n]$, rozdelenie na intervale $(n, 2^{n-1}]$ nemusí byť presné, pretože pôvodné NKA nemusia byť minimálne.

3.1.1 2-stavové NKA

Kedže 2-stavových NKA je spolu až $2^{2^2 \cdot |\Sigma|+2}$, je možné efektívne vygenerovať všetky 2-stavové NKA. Konkrétne ak $|\Sigma| = 2$, potom všetkých 2-stavových NKA je $2^{2^2 \cdot 2+2} = 2^{10}$ alebo ak $|\Sigma| = 5$, potom všetkých 2-stavových NKA je $2^{2^2 \cdot 5+2} = 2^{22}$. Predošlé počty 2-stavových NKA sú dostatočne malé na to, že sa dajú vygenerovať všetky.

Dokonca, o 2-stavovom NKA vieme ešte v realizovateľnom čase zistiť, či je minimálny. Preto pri generovaní všetkých 2-stavových NKA vieme vyradiť tie NKA, ktoré nie sú minimálne.



Obr. 3.1: Rozdelenie 2-stavových automatov pre rôzne množiny vstupnej abecedy: (a) ak $|\Sigma| = 2$; (b) ak $|\Sigma| = 3$; (c) ak $|\Sigma| = 4$; (d) ak $|\Sigma| = 5$;

Na obrázku 3.1 sú zobrazené grafy rozdelenia 2-stavových NKA pre rôzne mohutnosti vstupnej abecedy ($2 \leq |\Sigma| \leq 5$). Legenda na tomto obrázku

označuje s akými parametrami bol spustený program Výpočet (bližšie v kapitole 4).

Funkčná hodnota ($f(x)$) na grafe rozdelenia znamená, koľko minimálnych x-stavových DKA sa počas výpočtu vygenerovalo. Teda sa jedná o grafické vyobrazenie rozdelenia 2-stavových automatov. Na grafe je na Y-ovú os použitá logaritmická mierka, z dôvodu veľkej rozdielnosti výsledkov.

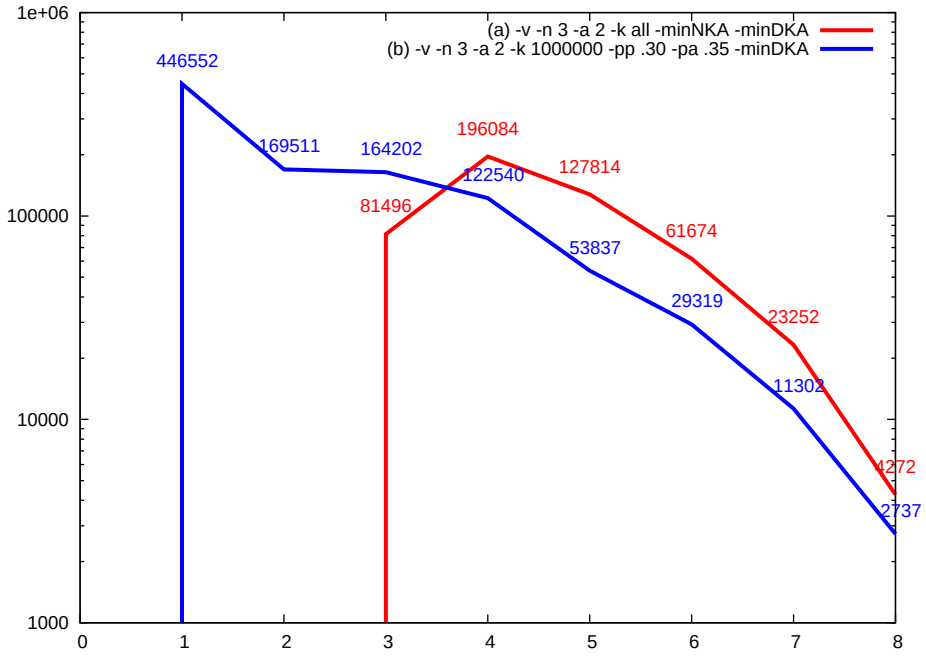
Z obrázku 3.1 vidno, že v rozdelení nad binárnou abecedou je najviac 3-stavových DKA. Tiež je zaujímavé, že 2-stavových DKA je viac ako 4-stavových. Avšak so zväčšujúcou sa množinou vstupnej abecedy sa zväčšuje aj zložitosť automatov DKA, preto v rozdelení s $|\Sigma| = 5$ je už najviac 4-stavových DKA.

3.1.2 3-stavové NKA

Podobne ako v predošlej časti, 3-stavových NKA je $2^{3^2 \cdot |\Sigma|+2}$ a ak $|\Sigma| = 2$, potom všetkých 3-stavových NKA nad binárnou abecedou je $2^{3^2 \cdot 2+2} = 2^{20}$. Tento počet je dostatočne malí, aby sa dali vygenerovať všetky. Rovnako ako pri 2-stavovom NKA, aj pre 3-stavový NKA vieme v realizovateľnom čase zistiť, či je minimálny.

Okrem predošlého presného výpočtu, som spravil aj nepresný pravdepodobnostný výpočet. Tento výpočet namiesto generovania všetkých 3-stavových NKA, vygeneruje zadaný počet krát náhodný NKA. Konštrukcia náhodného NKA závisí od dvoch hodnôt, pravdepodobnosť akceptácie (pa) a pravdepodobnosť prechodu (pp). Pravdepodobnosť akceptácie je hodnota, ktorá označuje pravdepodobnosť, že stav automatu je akceptačný. Pravdepodobnosť prechodu je hodnota, ktorá označuje pravdepodobnosť, že existuje prechod z nejakého stavu do nejakého stavu na nejaké písmeno.

Výsledky vyššie spomínaných výpočtov sú na obrázku 3.2, kde možno vidieť výsledky rozdelenia 3-stavových NKA. Ako vidno, nepresný pravdepodobnostný výpočet celkom dobre odhadol presné rozdelenie 3-stavových NKA na intervale $(2^2, 2^3]$. Podobne ako pri 2-stavovom NKA je veľmi málo 2^n-stavových DKA, konkrétne 8-stavových DKA. Z rozdelenia 3-stavových NKA je najviac 4-stavových DKA, teda ak vyberieme náhodný minimálny 3-stavový NKA s najväčšou pravdepodobnosťou k nemu ekvivalentný minimálny DKA bude mať 4 stavy.

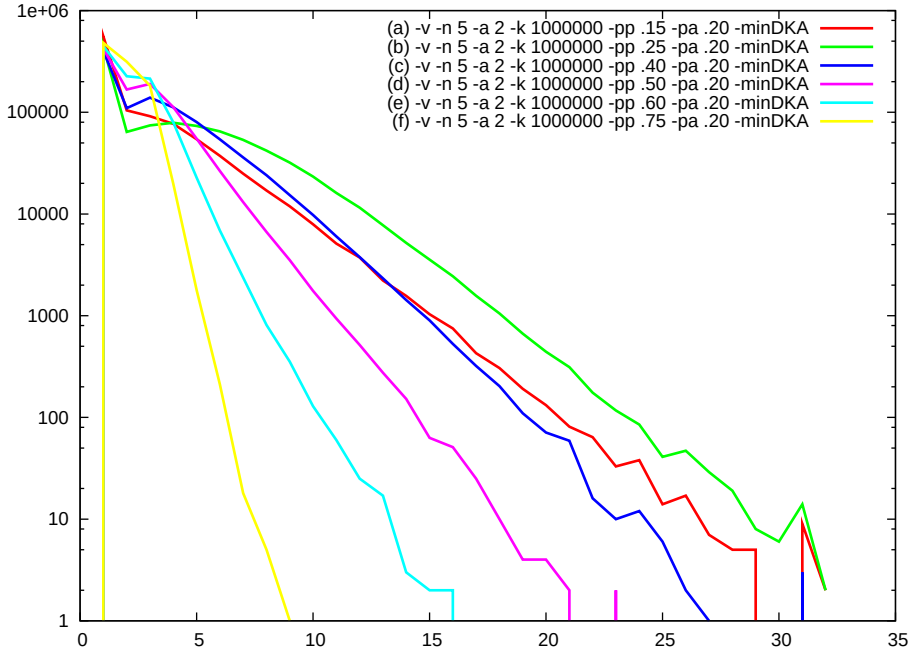


Obr. 3.2: Rozdelenie 3-stavových automatov nad binárnou abecedou s rôznymi metódami generovania NKA: (a) všetky 3-stavové minimálne NKA; (b) náhodných 10^6 NKA s pravdepodobnosťou akceptácie = 0.35 a pravdepodobnosťou prechodu = 0.3;

3.1.3 5-stavové NKA

Pri 5-stavových NKA je už situácia horšia. 5-stavových NKA nad binárnou abecedou je spolu až $2^{5^2 \cdot 2+2} = 2^{52}$, preto nie prakticky realizovateľné generovanie všetkých NKA. Z tohto dôvodu využijem pravdepodobnostné generovanie NKA, od ktorého očakávam, že poskytne približné rozdelenie 5-stavových NKA. Taktiež, pre 5-stavový NKA neviem efektívne zistiť, či je minimálny. Teda v tomto približnom rozdelení NKA sa môžu vyskytovať aj NKA, ktoré nie sú minimálne.

Výsledky výpočtov pri rôznych pravdepodobnostiach prechodu je možné vidieť na obrázku 3.3. Hodnota pravdepodobnosti akceptácie je zvolená tak, aby približne jeden stav automatu NKA bol akceptačný. Ako možno vidieť, pre väčšie hodnoty pravdepodobnosti prechodu (husté NKA) sa nepodarilo nájsť zložitejšie NKA, ktorých veľkosť ekvivalentného minimálneho DKA je z intervalu $(2^4, 2^5]$. Z toho možno usúdiť, že ak má byť NKA zložitý,

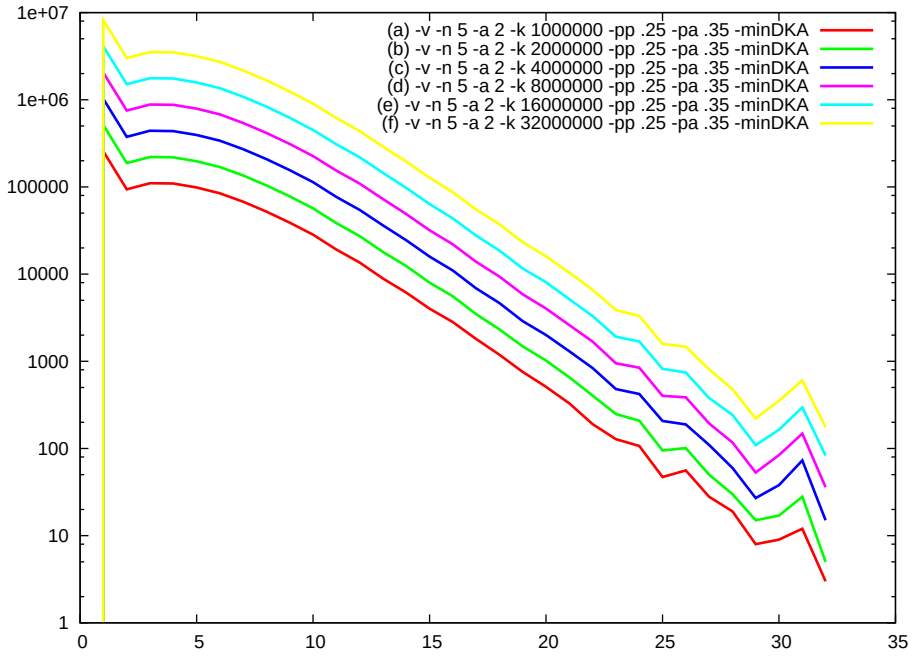


Obr. 3.3: Rozdelenie 5-stavových automatov nad binárnou abecedou pri približne jednom akceptačnom stave s rozličnými pravdepodobnosťami prechodu: (a) pp=0.15; (b) pp=0.25; (c) pp=0.40; (d) pp=0.50; (e) pp=0.60; (f) pp=0.75;

nesmie byť príliš hustý (z hľadiska počtu prechodov). Tiež možno usúdiť, že v rozdelení 5-stavových NKA, existuje len veľmi málo DKA s veľkosťou 2^5 v porovnaní s menej stavovými DKA z rozdelenia. Ako som už spomínal, na intervale $(0, 2^4]$ sú rozdelenia na obrázku čiastočne nekorektné, pretože v rozdelení sú započítané aj NKA, ktoré nie sú minimálne.

Prirodzene sa objaví otázka, či som v predchádzajúcom rozdelení vygeneroval dostatočne veľa NKA, aby výsledné rozdelenie bolo smerodajné. Preto som spravil výpočty pre rozličné počty vygenerovaných NKA a výsledné rozdelenia je možné vidieť na obrázku 3.4. Zjavne rozdielny počet NKA nemení krivosť krivky rozdelenia 5-stavových NKA (v logaritmickej mierke). Teda možno usúdiť, že 10^6 vygenerovaných NKA je dostatočne veľa.

Okrem predošlého pravdepodobnostného generovania NKA som vyskúšal aj iné pravdepodobnostné generovanie NKA. Nová konštrukcia náhodného NKA závisí taktiež od dvoch hodnôt, pravdepodobnosť akceptácie (pa) a obmedzenia prechodovej funkcie (pmax). Pravdepodobnosť akceptácie je



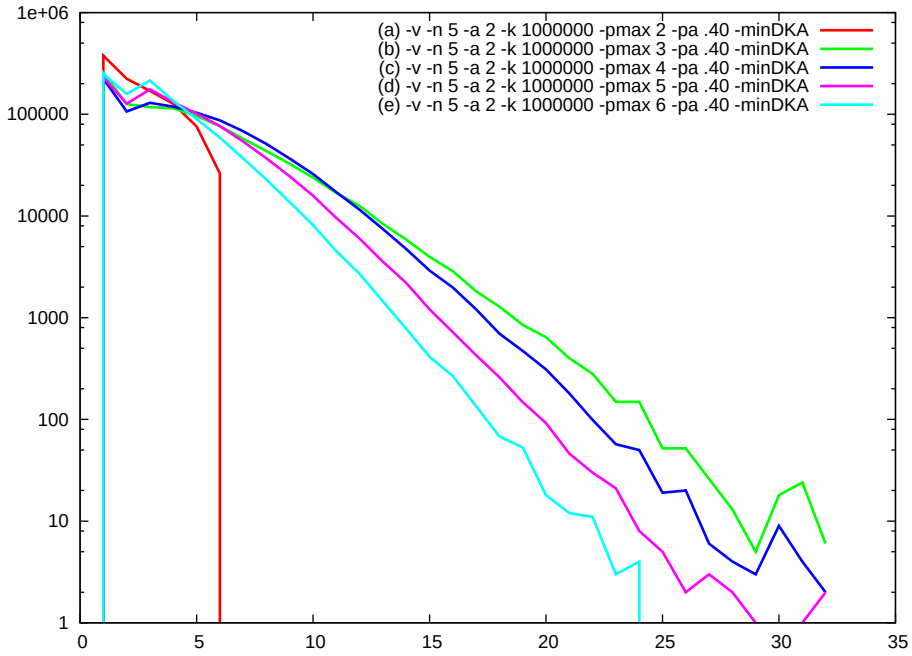
Obr. 3.4: Rozdelenie 5-stavových automatov nad binárnou abecedou pri pravdepodobnosti akceptácie=0.35 a pravdepodobnosťou prechodu=0.25 s rozličným počtom vygenerovaných NKA: (a) 10^6; (b) $2 \cdot 10^6$; (c) $4 \cdot 10^6$; (d) $8 \cdot 10^6$; (e) $16 \cdot 10^6$; (f) $32 \cdot 10^6$;

hodnota, ktorej význam je rovnaký ako v predošlom generovaní. Obmedzenie prechodovej funkcie (pmax) je hodnota, ktorá ohraničuje počet prechodov z nejakého stavu na nejaké písmeno, formálnejšie $\forall p \in K, \forall a \in \Sigma : |\delta(p, a)| <$ pmax.

Výsledky takéhoto pravdepodobnostného generovania pre 5-stavové NKA je možné vidieť na obrázku 3.5. Zjavne, pri obmedzení prechodovej funkcie pmax=2, sa vygenerovalo veľmi veľa nesúvislých alebo veľmi malo zložitých NKA. Opačne pri obmedzení prechodovej funkcie pmax=6, taktiež klesá počet zložitejších automatov. Z výsledkov možno usúdiť, že zložitejších 5-stavových NKA je oveľa menej ako jednoduchších NKA.

3.1.4 8-stavové NKA

8-stavových NKA nad binárnou abecedou je spolu až $2^{8^2 \cdot 2+2} = 2^{130}$. Preto je prakticky nerealizovateľné prejsť všetky 8-stavové NKA a taktiež zistiť či daný NKA je minimálny. Dokonca ani pravdepodobnostné prístupy, ktoré

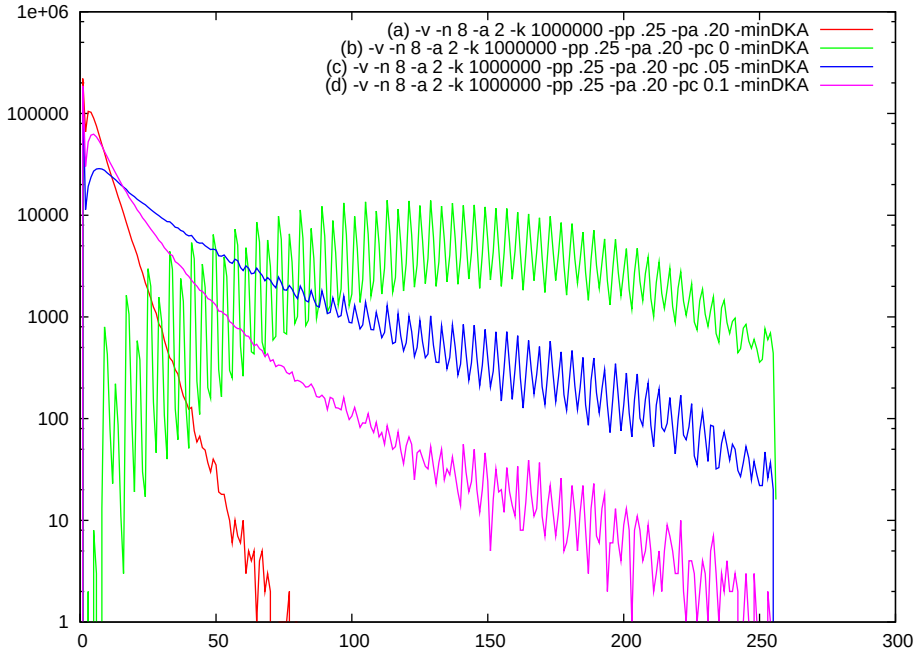


Obr. 3.5: Rozdelenie 5-stavových automatov nad binárnou abecedou pri pravdepodobnosti akceptácie=0.4 s rozličným obmedzením na prechodovú funkciu: (a) pmax=2; (b) pmax=3; (c) pmax=4; (d) pmax=5; (e) pmax=6;

som doteraz využil pri 5-stavových NKA nefungujú, ako vidno na krivke (a) z obrázku 3.6. Kedže tento výpočet nenájde žiaden zložitejší NKA na intervale $(2^7, 2^8]$ (čo nás najviac zaujíma), môžme sa domnievať, že počet zložitejších 8-stavových automatov je veľmi malý.

Pri hľadaní podstaty alebo podmienky, kedy je NKA zložitý som si všimol, že zložitý NKA obsahuje v prechodovej funkcii cyklus na jedno písmeno cez všetky stavy. Preto som naimplementoval a vyskúšal ďalšie pravdepodobnostné generovanie NKA. V tomto prípade konštrukcia NKA začína vytvorením cykla cez všetky stavy na písmeno "0". Ďalej konštrukcia závisí od troch hodnôt, pravdepodobnosť akceptácie (pa), pravdepodobnosť prechodu (pp) a pravdepodobnosť prechodu pre cyklus (pc). Pravdepodobnosť akceptácie je hodnota, určujúca pravdepodobnosť, že stav je akceptačný. Pravdepodobnosť prechodu je pravdepodobnosť, že existuje prechod z nejakého stavu do nejakého stavu na nejaké písmeno z množiny $\Sigma - \{0\}$. Pravdepodobnosť prechodu pre cyklus je znovu pravdepodobnosť, že existuje prechod z nejakého stavu do nejakého stavu na písmeno "0", okrem prechodov cyklu

z prechodovej funkcie, ktoré sú vytvorené na začiatku konštrukcie.



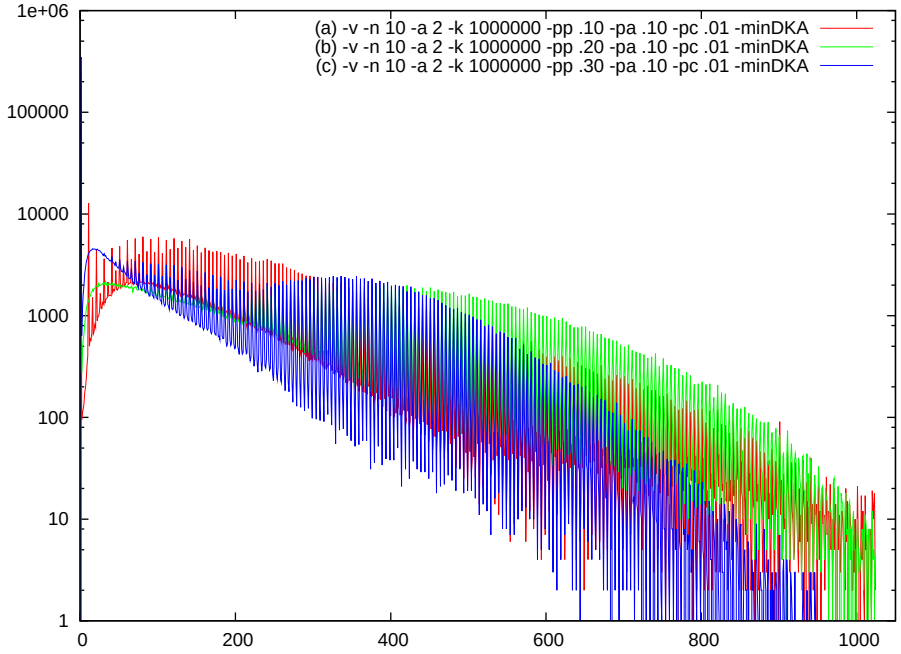
Obr. 3.6: Rozdelenie 8-stavových automatov nad binárnou abecedou pri pravdepodobnosti akceptácie=0.2 a s pravdepodobnosťou prechodu=0.25 : (a) bežné generovanie (pc=0.25); (b) pc=0; (c) pc=0.05; (d) pc=0.1;

Rozdelenie takýchto 8-stavových NKA s cyklom možno vidieť na obrázku 3.6. Ako vidieť, so zväčšujúcou sa pravdepodobnosťou prechodu na písmeno "0" sa zmenšuje počet zložitejších NKA. Taktiež je zaujímavé, že toto rozdelenie NKA s cyklom osciluje a líši sa od predchádzajúcich výsledkov. Podobne ako pri menej stavových NKA, aj v rozdelení 8-stavových NKA s cyklom je menej zložitých NKA oproti tým jednoduchším.

3.1.5 10-stavové NKA

10-stavových NKA nad binárnou abecedou je spolu až $2^{10^2 \cdot 2+2} = 2^{202}$. V takomto veľkom počte NKA je veľmi ťažké nájsť nejaký NKA, o ktorom by sme si boli istí, že je minimálny. Z tohto dôvodu sa zameriame na NKA obsahujúce cyklus cez všetky stavy. U takýchto NKA možno s väčšou pravdepodobnosťou očakávať, že budú minimálne a zložitejšie.

Na obrázku 3.7 možno vidieť rozličné rozdelenia 10-stavových automatov s cyklom. V každom rozdelení je zložitejších NKA menej ako tých jednoduch-



Obr. 3.7: Rozdelenie 10-stavových automatov s cyklom nad binárnou abecedou pri pravdepodobnosti akceptácie=0.1 a s pravdepodobnosťou prechodu na písmeno "0"=0.01 pre rôzne pravdepodobnosti prechodu: (a) pp=0.1; (b) pp=0.2; (c) pp=0.3;

ších. Tiež je zjavné, že v rozdelení redších NKA (NKA s pp=0.1) je oveľa viac zložitejších NKA. Naopak pre rozdelenie hustejších NKA (NKA s pp=0.3) platí, že zložitejších NKA ubúda. Pravdepodobnosť akceptácie som zvolil tak, aby približne jeden stav bol akceptačný. Z výsledkov rozdelenia možno zhodnotiť, že generovanie NKA s cyklom je dobrý spôsob ako dosiahnuť, aby NKA bol dostatočne zložitý.

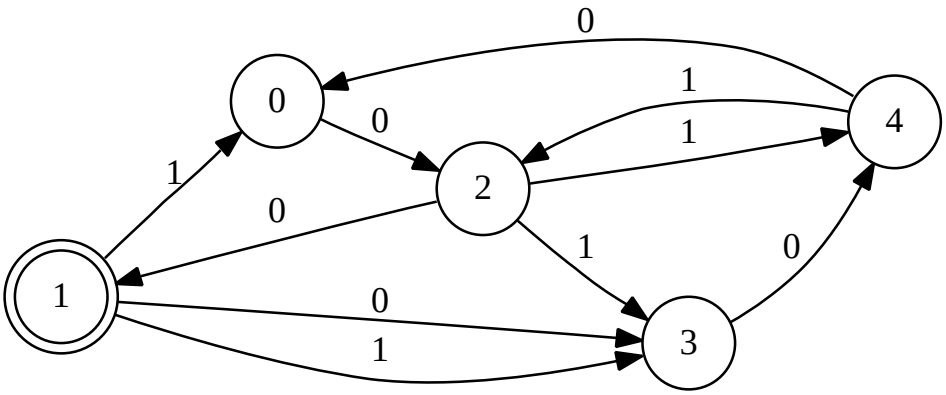
Pre viac-stavové NKA je možné dosiahnuť podobné výsledky. Hľadal som aj rozdelenie 11-stavových NKA s cyklom, avšak daný výpočet priniesol podobné výsledky ako pre 10-stavové NKA (na obrázku 3.7).

3.2 Hľadanie zložitých automatov

V predchádzajúcej časti pri hľadaní rozdelenia n-stavových NKA som dospel k hypotéze, že zložitejších NKA je značne menej ako jednoduchších NKA. Z toho je možné si položiť otázku, od čoho závisí zložitosť NKA. V nasledujúcej

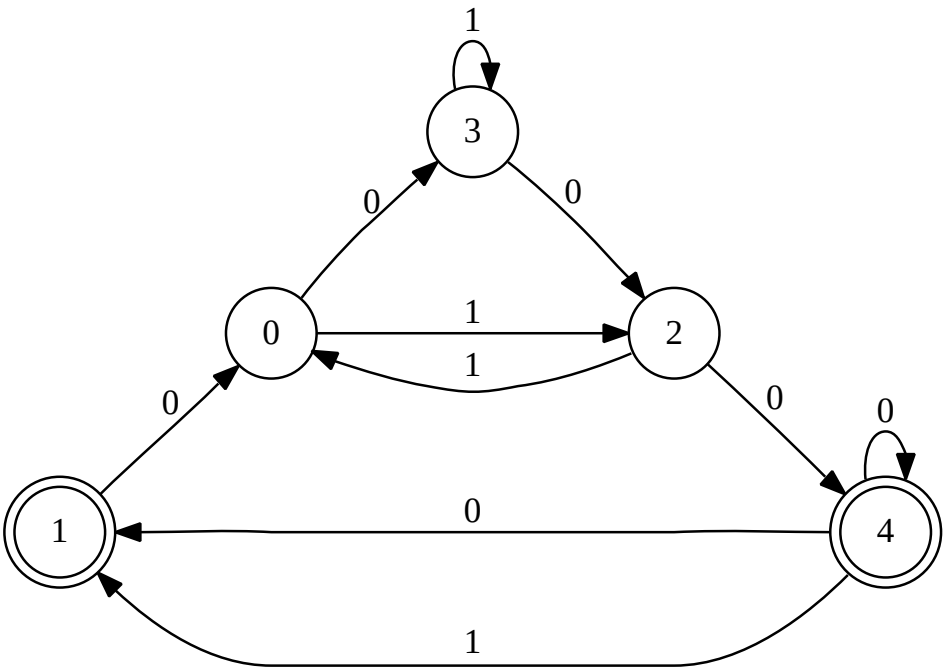
časti sa budem pokúšať hľadať odpoveď na túto otázku a uvediem aj príklady zložitých automatov.

Na začiatok som vygeneroval najzložitejšie 5-stavové NKA z rozdelenia na obrázku 3.3 zo strany 20. Možno usúdiť, že ak má NKA príliš veľkú množinu akceptačných stavov, potom s veľkou pravdepodobnosťou bude jednoduchý. Podobne, ak prechodová funkcia NKA je príliš veľká, potom z výsledkov rozdelení z predchádzajúcej časti možno povedať, že NKA s veľkou pravdepodobnosťou nie je minimálny alebo je jednoduchý.



Obr. 3.8: Príklad zložitého 5-stavového NKA, ktorého ekvivalentný minimálny DKA má $(2^5 - 1 =)$ 31 stavov.

Z vygenerovaných 5-stavových NKA uvediem dva príklady automatov, ktorých ekvivalentné minimálne DKA majú 31 a 32 stavov. Tieto príklady som vybral tak, aby mali čo najmenšie množiny akceptačných stavov a čo najmenšiu prechodovú funkciu. Na obrázku 3.8 možno vidieť prvý z nich, ktorého minimálny DKA má 31 stavov. Tento automat má iba jeden akceptačný stav a 9 prechodov medzi stavmi. Druhý automat je na obrázku 3.9, k nemu minimálny DKA má 32 stavov. Automat má dokonca až dva akceptačné stavy a 10 prechodov medzi stavmi. Na oboch automatoch si je možné všimnúť cyklus na písmeno "0" cez všetky stavy: na prvom automate 0-2-1-3-4 a na druhom automate 0-3-2-4-1. Dokonca druhý automat na obrázku 3.9 má okrem prechodov na písmeno "0", ktoré patria do cykla, aj prechod, ktorý je iba cyklus na stave. Je ťažké, len zo štruktúry oboch automatov určiť, prečo k nim ekvivalentné minimálne DKA sa líšia veľkosťou o jeden stav. Kedže oba automaty obsahujú cyklus cez všetky stavy a považujeme ich za zložité, zdá sa, že NKA s cyklom s veľkou pravdepodobnosťou budú



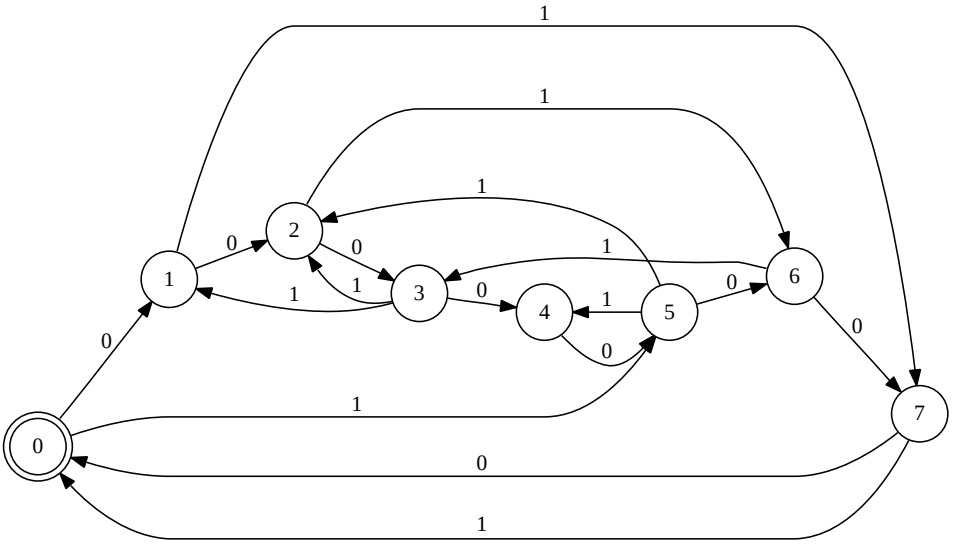
Obr. 3.9: Príklad zložitého 5-stavového NKA, ktorého ekvivalentný minimálny DKA má (2^5=) 32 stavov.

zložité.

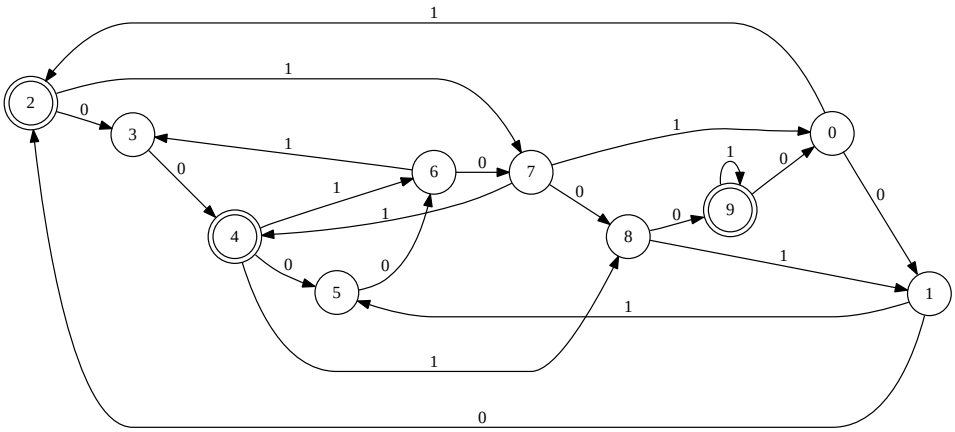
Z pozorovania, že zložité NKA majú cyklus, sa mi podarilo vygenerovať približne 180 zložitých 8-stavových NKA. Z nich som vybral jeden, ktorý mal najmenšiu prechodovú funkciu a iba jeden akceptačný stav (dokonca je to počiatočný). Tento automat je na obrázku 3.10, obsahuje 17 prechodov medzi stavmi a má iba jeden akceptačný stav. Taktiež obsahuje cyklus cez stavy 0-1-2-3-4-5-6-7. Okrem prechodov na písmeno "0", ktoré sú súčasťou cyklu, neobsahuje žiadne iné prechody na písmeno "0".

Podobne som našiel aj 10-stavový NKA, ktorého ekvivalentný minimálny DKA ma 1024 stavov. Automat je zobrazený na obrázku 3.11. NKA obsahuje dokonca až 3 akceptačné stavy a 20 prechodov medzi stavmi. Tiež obsahuje cyklus cez všetky stavy.

Z predošlých výsledkov vidno, že podmienka na NKA, aby automat obsahoval cyklus cez všetky stavy, je dobrá podmienka pre generovanie zložitejších automatov. V rozdelení 10-stavových NKA s cyklom na obrázku 3.7 (na strane 24) je aj veľa jednoduchších NKA, preto nemožno vysloviť



Obr. 3.10: Príklad zložitého 8-stavového NKA, ktorého ekvivalentný minimálny DKA má (2^8=) 256 stavov.

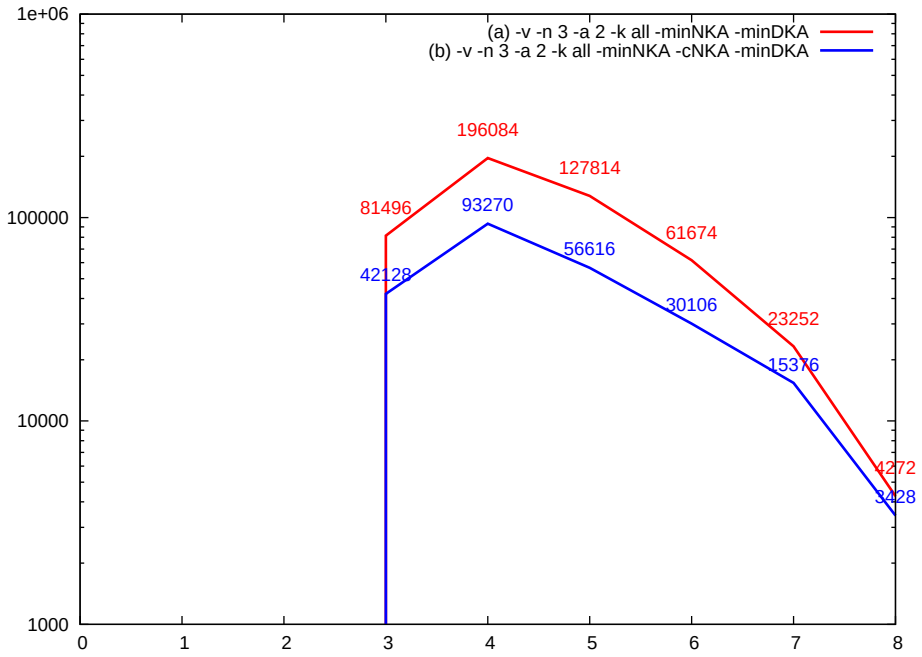


Obr. 3.11: Príklad zložitého 10-stavového NKA, ktorého ekvivalentný minimálny DKA má (2^{10}=) 1024 stavov.

tvrdenie, že väčšina NKA s cyklom (cez všetky stavy) sú zložitejšie.

Ďalšia zaujímavá otázka môže byť, či existuje zložitý NKA, ktorý neobsahuje cyklus cez všetky stavy. Teda, či tým, že obmedzíme výber NKA na automaty s cyklom cez všetky stavy, nevyradíme nejaké zaujímavé zložité automaty. Preto som vypočítal rozdelenie 3-stavových NKA s cyklom cez všetky stavy, ktoré je na obrázku 3.12. Ako vidno, existujú 3-stavové

NKA, ktoré sú zložité a neobsahujú cyklus cez všetky stavy. Taktiež možno usúdiť, že medzi jednoduchšími 3-stavovými NKA je viac tých NKA, ktoré neobsahujú cyklus cez všetky stavy.



Obr. 3.12: Rozdelenie 3-stavových automatov nad binárnou abecedou: (a) všetky 3-stavové minimálne NKA; (b) všetky 3-stavové minimálne NKA, ktoré obsahujú cyklus cez všetky stavy;

Z predchádzajúcich výsledkov je zrejmé, že ak na NKA dáme podmienku, aby obsahoval cyklus cez všetky stavy, zvýši sa pravdepodobnosť, že NKA bude zložitý. Avšak, ako som ukázal pri 3-stavových NKA, podmienka existencie cyklu v NKA nie je univerzálna podmienka pre generovanie zložitých NKA.

Kapitola 4

Implementačné detaily

V poslednej kapitole popíšem program, ktorý som použil na generovanie výsledkov, odprezentovaných v predchádzajúcich častiach. Taktiež spomeniem implementáciu použitých algoritmov a aj ich časovú zložitosť.

4.1 Popis programu Výpočet

Naprogramoval som program Výpočet, ktorého správanie a používanie v nasledujúcej časti popíšem.

Program Výpočet je konzolová aplikácia (console application), ktorej správanie sa modifikuje pomocou vstupných argumentov. Obyčajne, beh programu je nasledovný:

1. Vygeneruje sa NKA A, podľa vstupných parametrov.
2. NKA A sa transformuje na ekvivalentné DKA A_d podľa vstupných parametrov.
3. O DKA A_d sa zaznačí informácia, aká je jeho veľkosť.
4. Pokračuje sa krokom 1 podľa vstupných parametrov.
5. V poslednom kroku sa vypíšu dvojice čísel i a n_i, kde číslo n_i označuje počet, počas behu programu zostrojených deterministických konečných automatov, ktorých veľkosť je i.

4.1.1 Argumenty programu Výpočet

Beh programu Výpočet úzko súvisí od zadaných vstupných argumentov programu, ktorý daný beh modifikujú. Parametre programu Výpočet sú:

-h : namiesto bežného behu, vypíše návod (help) k programu.

-n int : int je celé kladné číslo. Pri tomto vstupnom argumente sa budú v kroku 1 behu programu generovať len NKA s veľkosťou int. Teda určuje veľkosť množiny stavov NKA.

-a int : int je opäť celé kladné číslo. Pri tomto vstupnom argumente sa budú v kroku 1 behu programu generovať len NKA nad množinou vstupnej abecedy, ktorej veľkosť je int. Teda určuje veľkosť množiny vstupnej abecedy.

-minNKA : Pri tomto vstupnom argumente sa budú v kroku 1 behu programu generovať NKA, ktoré sú minimálne. Tento argument je použiteľný iba pre malé veľkosti NKA z dôvodu veľkej časovej zložitosti problému overenia, či NKA je minimálny.

-cNKA : Pri tomto vstupnom argumente sa budú v kroku 1 behu programu generovať NKA, ktoré obsahujú cyklus cez všetky stavy. Tento argument je použiteľný iba pre malé veľkosti NKA z dôvodu veľkej časovej zložitosti problému overenia, či NKA obsahuje cyklus.

-minDKA : Pri tomto vstupnom argumente sa bude v kroku 2 behu programu transformovať NKA A na minimálny DKA A_d.

-pp float : float je reálne číslo z intervalu $(0, 1)$. Argument -pp vylučuje argument -pmax a vynucuje argument -pa. Tento vstupný argument modifikuje generovanie NKA $A = (K, \Sigma, \delta, g_0, F)$ v kroku 1 behu programu, konkrétne vytvorenie prechodovej funkcie. Prechodovú funkciu δ pre NKA A vytvorí nasledovne: pre každé stavy $p, q \in K$ a symbol $a \in \Sigma$, pravdepodobnosť udalosti $q \in \delta(p, a)$ je rovná float. Inými slovami, pravdepodobnosť, že zo stavu p do stavu q na písmeno a existuje prechod je rovná float.

-pc float : float je reálne číslo z intervalu $(0, 1)$. Argument -pc vylučuje argument -pmax a vynucuje argumenty -pa a -pp. Tento vstupný argument modifikuje generovanie NKA $A = (K, \Sigma, \delta, g_0, F)$ v kroku 1

behu programu, konkrétne vytvorenie prechodovej funkcie. Prechodovú funkciu δ pre NKA A vytvorí rovnako ako pri argumente -pp pre všetky symboly $a \in \Sigma - \{0\}$. Prechody na písmeno 0 sa vytvoria tak, aby vznikol cyklus cez všetky stavy K (prechody z i-teho stavu do $(i+1)$ mod $|K|$ stavu). Ostatné prechody na písmeno 0 existujú s pravdepodobnosťou float.

-pmax int : int je celé kladné číslo. Argument vynucuje argument -pa. Podobne ako predošly argument, modifikuje generovanie NKA A v kroku 1, konkrétne prechodovej funkcie. Prechodovú funkciu pre NKA A vygenerujem náhodne tak, že pre každý stav $p \in K$ a symbol $a \in \Sigma$ veľkosť množiny $\delta(p, a)$ je menšia ako int ($|\delta(p, a)| <$ int). Teda z každého stavu a pre každé písmeno existuje najviac int prechodov.

-pa float : float je reálne číslo z intervalu $(0, 1)$. Tento vstupný argument modifikuje generovanie NKA $A = (K, \Sigma, \delta, g_0, F)$ v kroku 1 behu programu, konkrétne množinu akceptačných stavov F. Množinu akceptačných stavov F pre NKA A vytvorí nasledovne: pre každý stav $p \in K$, pravdepodobnosť udalosti $p \in F$ je rovná float. Inými slovami, pravdepodobnosť, že stav p bude akceptačný je rovná float.

-k all/int : all je reťazec, int je cele kladne číslo. Pri tomto vstupnom argumente sa modifikuje beh programu v kroku 4. Ak je vstupný argument -k all, potom sa generujú všetky možné NKA A s ohľadom na ostatné argumenty. Ak je vstupný argument -k int, potom krok 4 sa vykoná int krát. Argument -k int vynucuje argumenty -pa a -pp alebo -pmax.

-r : Pri argumentoch -pa, -pp -pmax sa využíva nahodny generátor, argument -r znáhodní náhodný generátor (randomize).

-l int : int je celé kladné číslo. Pri tomto vstupnom argumente sa krok 5 nevykoná. Namiesto toho, ak v kroku 2 veľkosť DKA A_d je rovná int, potom sa vypíše NKA A vo formáte pre program Graphviz.

-v : Navyše sa vypíšu parametre s ktorými bol program spustený.

4.2 Použité dátové štruktúry a algoritmy

Väčšinu algoritmov, čo som implementoval, som už spomenul v kapitole 2 na strane 5. Preto spomeniem hlavne dátové štruktúry, ktoré som pri

implementácii použil.

Obidva automaty DKA aj NKA som implementoval ako triedy. Prechodovú funkciu konkrétnej realizácie DKA alebo NKA som uložil vo viacrozmernom dynamickom poli. Okrem toho je taktiež v dynamickom poli pre každý stav uložená informácia, či je stav akceptačný.

V triede pre nedeterministické konečné automaty som implementoval metódu pre nájdenie ekvivalentného DKA, bližší popis algoritmu možno nájsť v časti 2.2 na strane 7. Pri realizácii tohoto algoritmu som použil jednoduché dynamické polia a rad (FIFO), ktorý som využil pri zisťovaný, ktorý stav je dosiahnuteľný. Ďalšia metóda, ktorú som už spomínal v časti 2.4 na strane 12, je overenie minimálnosti NKA. Tiež som implementoval metódu pre overenie prítomnosti cykla cez všetky stavy. Pri implementácii tejto metódy som použil zásobník, ktorý som použil na ukladanie stavov pri prehľadávaní automatu. Algoritmus pre túto metódu je veľmi jednoduchý - pre každé písmeno zo vstupnej abecedy prehľadá pomocou zásobníka automat a ak zásobník obsahuje všetky stavy, tak našiel cyklus cez všetky stavy. Ešte som implementoval viacero metód pre modifikovanie nedeterministického konečného automatu, avšak ich implementácia je veľmi priamočiara a využil som len jednoduché dynamické polia. Tieto metódy som využil pri rozličnom generovaní nedeterministického konečného automatu.

V triede pre deterministické konečné automaty som implementoval metódu pre zistenie, či sú dva deterministické konečné automaty ekvivalentné. Algoritmus tejto metódy hľadá izomorfizmus na stavoch týchto dvoch deterministických automatov. Ak taký izomorfizmus existuje, je ho ľahké nájsť postupným prehľadávaním obidvoch konečných automatov zo začiatočného stavu. Tiež som pri prehľadávaní automatu využil rad (FIFO), do ktorého som vkladal stavy automatu, ktoré sa majú ešte spracovať. Metóda končí, keď sa overí, že zostrojený izomorfizmus je ozaj izomorfizmus na stavoch automatov. Túto metódu som využil pri zisťovaní, či je nejaký nedeterministický konečný automat minimálny. Okrem tejto metódy som implementoval aj algoritmus minimalizácie deterministického konečného automatu. Tento algoritmus som už bližšie popísal v časti 2.3.2 na strane 11, preto ho už nebudem opakovať. Akurát spomeniem, že pri implementácii som použil jednoduché dynamické polia a rad (FIFO) na zapamätanie si množiny N, ktorá obsahovala usporiadané dvojice (trieda, písmeno).

4.2.1 Časová zložitosť algoritmov

Na koniec ešte zhrniem časovú zložitosť algoritmov, ktoré som v predchádzajúcej časti spomínal.

Algoritmus, ktorý k zadanému NKA nájde ekvivalentný DKA, pracuje v čase $O(2^n)$. Ďalší algoritmus, ktorý overuje minimálnosť NKA, ma časovú zložitosť až $O(2^{n^k})$ (t.j. exponenciálnu). Rovnakú časovú zložitosť má aj metóda, ktorá overuje prítomnosť cykla cez všetky stavy. Rozličné metódy, ktoré som využil pri generovaní NKA, pracujú zväčša v čase $O(n^2)$, pretože počet prechodov medzi stavmi môže byt až n^2.

Pre deterministické konečné automaty implementovaný algoritmus, ktorý zisťuje ekvivalentnosť dvoch minimálnych DKA, má časovú zložitosť $O(n)$. Táto lineárna časová zložitosť je zrejmá z toho, že nájdenie izomorfizmu (na stavoch DKA) a jeho overenie je možné v lineárnom čase. Posledný spomenutý algoritmus na minimalizáciu deterministického konečného automatu ma zložitosť $O(n \log n)$.

Záver

V práci sme rozobrali rozličné algoritmy z oblasti konečných automatov, spolu s ich implementáciou. Spomenuli sme algoritmus ako k zadanému nedeterministickému konečnému automatu zostrojiť ekvivalentný deterministický konečný automat. Taktiež sme popísali doteraz najefektívnejší známy algoritmus (Hopcroft-ov) na minimalizáciu deterministického konečného automatu. Tiež sme prediskutovali problém nájdenia minimálneho nedeterministického konečného automatu.

V kapitole 3 sme skúmali vzájomné vzťahy medzi minimálnymi deterministickými a nedeterministickými konečnými automatmi z hľadiska počtu stavov. Oboznámili sme sa s experimentálnymi výsledkami pre nedeterministické konečné automaty s malým počtom stavov. Taktiež sme sa zaoberali hľadaním n-stavových minimálnych nedeterministických konečných automatov, ktorých ekvivalentný minimálny deterministický konečný automat má 2^n stavov.

Výsledky, ktoré boli v tejto práci spomínané, nastolili rad ďalších otvorených problémov a otázok. Zaujímavé by bolo hlbšie preskúmať štruktúru n-stavových minimálnych nedeterministických konečných automatov, ktorých minimálny deterministický konečný automat má 2^n stavov. Taktiež nájsť štruktúru takýchto n-stavových minimálnych nedeterministických konečných automatov, ktoré neobsahujú cyklus cez všetky stavy.

Literatúra

[Hop71] John E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Z. Kohavi, editor, *Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

[HU69] John E. Hopcroft and Jeffrey D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1969.

[Hud07] Ivana Hudáková. Zložitostné aspekty konečných automatov, 2007.

[Jir01] Galina Jirásková. Note on minimal finite automata. In *MFCS '01: Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science*, pages 421–431, London, UK, 2001. Springer-Verlag.

[JR93] Tao Jiang and B. Ravikumar. Minimal nfa problems are hard. *SIAM J. Comput.*, 22(6):1117–1141, 1993.

[Myh57] J. Myhill. Finite automata and the representation of events. Technical Report WADD TR-57-624, Wright Patterson Air Force Base, Ohio, 1957.

[Ner58] A. Nerode. Linear automaton transformations. *PAMS*, 9:541–544, 1958.