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

Zjednodušenie výpočtov prídavnou informáciou

(diplomová práca)

Pavel Labath

Vedúci: prof. RNDr. Branislav Rovan, PhD. Bratislava, 2010

Abstrakt

V práci skúmame vplyv prídavnej informácie na zložitosť automatov rozpoznávajúcich nejaké jazyky. „Prídavná informácia“ znamená to, že automatu dovolíme predpokladať, že vstup patrí do nejakého poradného jazyka. V práci skúmame jazyky rozpoznávané deterministickými zásobníkovými automatmi s regulárnym poradným jazykom. V prvej časti sa zaoberáme deterministickými bezkontextovými jazykmi z pohľadu zložitosti. Skúmame rôzne spôsoby definovania zložitosti zásobníkových automatov a jazykov ktoré oni rozpoznávajú, hľadáme tesné odhady zložitosti niektorých tried jazykov a ukazujeme konštrukciu normálneho tvaru „automat vždy dočíta vstup“, ktorá nevyžaduje pridanie exponenciálneho počtu stavov. V druhej časti využívame tieto poznatky na skúmanie vplyvu regulárnej prídavnej informácie na zložitosť. Dokazujeme existenciu nekonečnej podtriedy deterministických zásobníkových jazykov, pre ktoré vhodná regulárna prídavná informácia umožní zjednodušiť deterministický zásobníkový automat pre ich akceptovanie. Dokážeme aj existenciu nekonečnej triedy deterministických bezkontextových jazykov, obsahujúcej ľubovoľne zložité jazyky, pri ktorých žiadna regulárna prídavná informácia neumožní znížiť zložitosť deterministického zásobníkového automatu pre ich akceptovanie.

Kľúčové slová: deterministické zásobníkové automaty, popisná zložitosť, rozklad automatov, prídavná informácia

Abstract

We study the effect of providing supplementary information about the input word on the complexity of automata recognizing languages. "Supplementary information" means that the automaton can assume that its input belongs to a given advisory language. We study the languages recognized by deterministic push-down automata using regular supplementary information. In the first part of the thesis we present our results on deterministic context-free languages and their descriptional complexity. We prove tight complexity bounds for some classes of languages and exhibit a construction of the normal form "the automaton reads the whole input word" which does not require the addition of an exponential number of states. These results are then used in the second part, where we prove the existence of an infinite subclass of deterministic context-free languages with the property that certain type of regular supplementary information enables us to construct simpler automata accepting these languages. We also exhibit an infinite class of deterministic context-free languages (containing arbitrarily complex languages), whose minimal automata cannot by simplified by any regular supplementary information.

Keywords: deterministic push-down automata, descriptional complexity, decomposition of automata, supplementary information

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

. . . . . .

Touto cestou by som sa chcel poďakovať môjmu školiteľovi, prof. Branislavovi Rovanovi, PhD., za pomoc pri hľadaní zaujímavej témy, ako aj za neoceniteľné rady a pripomienky pri jej spracovaní.

Obsah

Kapitola 1

Úvod

V práci skúmame vplyv prídavnej informácie na zložitosť automatov rozpoznávajúcich nejaké jazyky. „Zjednodušovať“ tu znamená ku danému automatu zostrojiť iný, ktorý bude akceptovať rovnaký jazyk, ale bude v nejakom zmysle jednoduchší, menší.

Jednoduchší automat je možné zostrojiť tak, že mu dovolíme predpokladať niečo o jeho vstupe. „Prídavná informácia“ v našom prípade bude to, že automat bude môcť predpokladať, že vstup nie je ľubovoľné slovo zo Σ^*, ale že patrí do nejakého poradného (angl. *advisory*) jazyka. Potom definujeme jazyk akceptovaný automatom A s poradným jazykom L_A nasledovne:

Definícia 1.1. *Nech L_A je jazyk. Nech A je automat. Jazyk akceptovaný automatom A s poradným jazykom L_A je*

$$L(A, L_A) = \{w \mid A \text{ akceptuje } w \wedge w \in L_A\}$$

Ak sú jazyky $L(A, L_A)$ a L_A rozpoznávané nejakými automatmi A_L a A_A tak platí $L(A, L_A) = L(A_L) \cap L(A_A)$. Na tento problém sa potom dá pozrieť ako na problém rozkladu automatu na dva nové tak, aby sa z ich nezávislých výpočtov dal zistiť výsledok výpočtu pôvodného automatu. Rozklad je netriviálny, ak sú oba nové automaty jednoduchšie ako ten pôvodný.

Kompozície a dekompozície niektorých typov automatov už boli skúmané. Výsledky o sekvenčných strojoch sa môžu nájsť v [1] a [2], deterministické konečné automaty sú skúmané v [3] a [4]. V práci budeme skúmať zložitejší model: deterministické zásobníkové automaty s regulárnou prídavnou informáciou.

V kapitole 2 uvádzame formálne definície automatov a pojmov, ktoré budeme ďalej v práci používať.

Aby sme vedeli povedať, či sa nejaký automat dá rozložiť na jednoduchšie, musíme vedieť, ako porovnať zložitosť automatov. V kapitole 3 skúmame rôzne spôsoby definovania zložitosti automatov a hľadáme presné odhady zložitosti niektorých jazykov.

V kapitole 4 potom skúmame ako môže regulárna prídavná informácia pomôcť zjednodušiť automaty a jazyky. Ukazujeme, že existujú dobre rozložiteľné, ale aj nerozložiteľné jazyky, ktorým nepomôže žiadna regulárna informácia.

Kapitola 2

Základné definície a označenia

Ústredný pojem práce je deterministický zásobníkový automat. V nasledovnej časti preto uvádzame jeho definíciu.

2.1 Deterministické zásobníkové automaty

Definícia 2.1. *Deterministický zásobníkový automat (DPDA –* deterministic push-down automaton*) je 7-ica* $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$*, kde* K *je konečná množina stavov,* Σ *je (konečná) vstupná abeceda,* Γ *je (konečná) abeceda zásobníkových symbolov,* $q_0 \in K$ *je začiatočný stav,* $Z_0 \in \Gamma$ *je symbol, ktorý je v zásobníku na začiatku výpočtu,* $F \subseteq K$ *je množina akceptačných stavov a* $\delta : K \times (\Sigma \cup \varepsilon) \times \Gamma \to (K \times \Gamma^*) \cup \{\emptyset\}$ *je prechodová funkcia spĺňajúca*

$$\forall q \in K \quad \forall Z \in \Gamma \qquad \delta(q, \varepsilon, Z) \neq \emptyset \;\Rightarrow\; (\forall x \in \Sigma \quad \delta(q, x, Z) = \emptyset)$$

Definícia 2.2. *Konfigurácia deterministického zásobníkového automatu je trojica* $(q, w, s) \in K \times \Sigma^* \times \Gamma^*$*, kde* q *je stav automatu,* w *je nespracovaná časť vstupného slova a* s *je obsah zásobníka.*

Definícia 2.3. *Krok výpočtu DPDA* A *je relácia* $\vdash_A$ *na množine konfigurácií definovaná nasledovne:*

$$\forall p, q \in K \quad \forall a \in \Sigma \cup \{\varepsilon\} \quad \forall w \in \Sigma^* \quad \forall s, t \in \Gamma^* \quad \forall Z \in \Gamma$$
$$(q, aw, sZ) \vdash_A (p, w, st) \quad \overset{\text{def}}{\iff} \quad (p, t) = \delta(q, a, Z)$$

Relácia $\vdash_A^*$ *je reflexívno-tranzitívny uzáver relácie* $\vdash_A$*.*

Definícia 2.4. *Jazyk akceptovaný DPDA A* **akceptačným stavom** *je množina slov:* $L(A) = \{w \in \Sigma^* \mid \exists q \in F \; \exists s \in \Gamma^* \quad (q_0, w, Z_0) \vdash_A^* (q, \varepsilon, s)\}$. *Triedu jazykov, ktoré dokážu akceptovať deterministické zásobníkové automaty akceptačným stavom označujeme* $\mathscr{L}_{DPDA}$.

Definícia 2.5. *Jazyk akceptovaný DPDA A* **prázdnou pamäťou** *je množina slov:* $N(A) = \{w \in \Sigma^* \mid \exists q \in F \quad (q_0, w, Z_0) \vdash_A^* (q, \varepsilon, \varepsilon)\}$. *Triedu jazykov, ktoré dokážu akceptovať deterministické zásobníkové automaty prázdnou pamäťou označujeme* $\mathscr{L}_{eDPDA}$.

DPDA si môžeme predstaviť ako zariadenie pozostávajúce z konečnostavovej riadiacej jednotky, čítacej hlavy, vstupnej pásky a zásobníka. V každom kroku hlava číta jeden znak zo vstupu (pripadne aj ε) a na základe toho znaku a symbolu na vrchu zásobníka automat môže zmeniť stav a obsah zásobníka — vymazať symbol na vrchu a pridať (potenciálne prázdne) slovo na jeho miesto. Prechodová funkcia môže byť pre niektoré argumenty nedefinovaná (vracať $\emptyset$) — automat sa v tom prípade zasekne. Aby zariadenie bolo deterministické, musíme pridať jedno obmedzenie na δ: automat môže robiť krok na ε len ak pre danú dvojicu (stav, symbol z Γ) nemá definované prechody na žiadne písmeno.

Pre DPDA sú definované dva spôsoby akceptovania: akceptačným stavom a prázdnou pamäťou. Pri hore uvedených definíciách je akceptovanie stavom silnejšie — DPDA nemôže akceptovať prázdnou pamäťou jazyk obsahujúci w a nejaký prefix w. Situácia sa dá vylepšiť pridaním nového symbolu (\$) na koniec vstupnej pásky. Teraz už dva spôsoby akceptovania budú ekvivalentné. Automat akceptujúci stavom sa bude dať simulovať automatom akceptujúcim prázdnou pamäťou (za cenu pridania nových stavov a zásobníkových symbolov). Tu budeme používať akceptovanie prázdnou pamäťou tak ako je definované v 2.5 — bez špeciálneho symbolu na konci.

Kapitola 3

Zložitosť deterministických zásobníkových automatov

V tejto kapitole budeme hovoriť o zložitosti deterministických zásobníkových automatov rozpoznávajúcich niektoré jazyky. Uvidíme, že na rozdiel od konečných automatov[1], už len definovať zložitosť zásobníkových automatov nie je celkom jednoduché. Tu máme aspoň dva parametre: počet stavov a zásobníkových symbolov. Potom keď máme dva automaty, a jeden má viac stavov ale menej symbolov ako druhý, tak nie je jasné ktorý automat je zložitejší. Začneme s mierou ktorá vyrieši jednoduché prípady.

Definícia 3.1. *$\preceq$ a $\prec$ sú relácie (čiastočné usporiadania) na $\mathbb{N} \times \mathbb{N}$ definované nasledovne:*

$$(a,b) \preceq (a',b') \iff^{\text{def}} a \leq a' \wedge b \leq b'$$
$$(a,b) \prec (a',b') \iff^{\text{def}} (a,b) \preceq (a',b') \wedge (a,b) \neq (a',b')$$

Definícia 3.2. *Nech A a A' sú deterministické zásobníkové automaty. Hovoríme, že A je jednoduchší (resp. nie zložitejší) ako A' ak $(|K_A|, |\Gamma_A|) \prec (|K_{A'}|, |\Gamma_{A'}|)$ $\big($resp. $(|K_A|, |\Gamma_A|) \preceq (|K_{A'}|, |\Gamma_{A'}|)\big)$.*

Tu nebudeme porovnávať všetky automaty, ale sa obmedzíme na tie prípady kde je to jednoznačné — prvý automat je jednoduchší keď nemá viac stavov *a* zásobníkových symbolov ako druhý (a aspoň jedna nerovnosť je ostrá). Niektoré automaty budú neporovnateľné podľa tejto miery, ale s touto

[1] kde máme „prirodzenú“ mieru: „počet stavov“

mierou budeme porovnávať ostatné miery, ktoré definujeme — každá „rozumná“ miera by mala zachovávať usporiadanie definované v 3.2. Pre každú mieru môžeme definovať minimálny automat rozpoznávajúci nejaký jazyk nasledovne:

Definícia 3.3. *Nech $\prec$ je čiastočné usporiadanie na $\mathbb{N} \times \mathbb{N}$. Nech DPDA A má k stavov, z zásobníkových symbolov a rozpoznáva jazyk L. Hovoríme, že A je minimálny automat rozpoznávajúci jazyk L ak neexistuje DPDA A' (s k' stavmi a z' zas. symbolmi) rozpoznávajúci L taký, že:*

$$(k', z') \prec (k, z)$$

3.1 Meranie zložitosti automatov

Nech $L = \{a^n b^n \mid n \in \mathbb{N}^+\}$. Najprv zostrojíme niekoľko automatov, ktoré L akceptujú, demonštrujúc pri tom „prelievanie“ zložitosti medzi stavmi a zásobníkovými symbolmi. Potom dokážeme optimalitu automatov (podľa definície 3.2) a budeme skúmať iné miery zložitosti. Chceme definovať zložitosť deterministického bezkontextového *jazyka* ako zložitosť minimálneho automatu, ktorý ten jazyk rozpoznáva. Na to potrebujeme, aby zložitosť jazyka nezávisela od voľby (minimálneho) automatu.

$$\begin{aligned}
&A_1 = (\{q_0, q_1\}, \{a, b\}, \{Z_0, a\}, \delta_1, q_0, Z_0, \emptyset) \\
&\delta_1(q_0, a, Z_0) \;\; = (q_0, a) \\
&\delta_1(q_0, a, a) \;\;\;\; = (q_0, aa) \\
&\delta_1(q_0, b, a) \;\;\;\; = (q_1, \varepsilon) \\
&\delta_1(q_1, b, a) \;\;\;\; = (q_1, \varepsilon)
\end{aligned}$$

$$\begin{aligned}
&A_2 = (\{q_0, q_1, q_2\}, \{a, b\}, \{Z_0\}, \delta_2, q_0, Z_0, \emptyset) \\
&\delta_2(q_0, a, Z_0) \;\; = (q_1, Z_0) \\
&\delta_2(q_1, a, Z_0) \;\; = (q_1, Z_0 Z_0) \\
&\delta_2(q_1, b, Z_0) \;\; = (q_2, \varepsilon) \\
&\delta_2(q_2, b, Z_0) \;\; = (q_2, \varepsilon)
\end{aligned}$$

$$
\begin{aligned}
&A_3 = (\{q_0\}, \{a, b\}, \{Z_0, a, b\}, \delta_3, q_0, Z_0, \emptyset) \\
&\delta_3(q_0, a, Z_0) \quad = (q_0, a) \\
&\delta_3(q_0, a, a) \quad = (q_0, ba) \\
&\delta_3(q_0, b, a) \quad = (q_0, \varepsilon) \\
&\delta_3(q_0, b, b) \quad = (q_1, \varepsilon)
\end{aligned}
$$

Tvrdenie 3.1. $N(A_1) = N(A_2) = N(A_3) = L$

A_1 v stave q_0 používa zásobník na počítanie symbolov a. Keď príde na vstup prvý znak b tak prejde do druhého stavu v ktorom už len vyprázdňuje zásobník. Potrebuje dva stavy, aby sa nemohli na vstupe striedať znaky a a b. Dva zásobníkové symboly sú potrebné preto, aby v q_0 platil invariant: „počet prečítaných znakov a“ = „počet symbolov na zásobníku“.

A_2 ukazuje, že stačí aj jeden zásobníkový symbol — v tom prípade ale potrebujeme jeden dodatočný stav na zachovanie invariantu. A_3, naopak, má len jeden stav a striedanie znakov na vstupe znemožňuje použitím nového zásobníkového symbolu.

Lema 3.1. *Neexistuje DPDA A taký, že $N(A) = L$ a súčasne $|K| = 1$ a $|\Gamma| = 2$.*

Dôkaz. Bez ujmy na všeobecnosti, nech $K = \{q_0\}$ a $\Gamma = \{Z_0, Z_1\}$. Pozrime sa ako môže vyzerať výpočet A na $w \in L$. Výpočet môžeme rozdeliť na dve časti podľa toho, ktorý symbol A číta. Kým A číta symboly a, tak si musí niekde (na zásobník) ukladať informáciu o ich počte, aby mohol v druhej časti kontrolovať počet b.

1. Ak by A mal počas výpočtu (okrem na začiatku) symbol Z_0 na dne zásobníka tak by sa dali jeho výpočty „reťaziť“ a automat by akceptoval zlé slová. Teda, na dne zásobníka musí byť Z_1.

2. Teraz ukážeme, že výška zásobníka v prvej časti výpočtu neklesá (ani len dočasne). Rozoberme prípady ako by mohla výška klesnúť:

   - $\delta(q_0, a, Z_0) = (q_0, \varepsilon)$ alebo $\delta(q_0, \varepsilon, Z_0) = (q_0, \varepsilon) \quad \rightsquigarrow$ automat sa zasekne hneď na začiatku a nemôže akceptovať správne slová.

- $\delta(q_0, \varepsilon, Z_1) = (q_0, \varepsilon) \rightsquigarrow$ zásobníkový symbol Z_1 je vlastne zbytočný a DPDA s jedným stavom a jedným zásobníkovým symbolom očividne nemôže akceptovať L.
- $\delta(q_0, a, Z_1) = (q_0, \varepsilon)$: Vieme, že Z_1 je na dne zásobníka. Toto pravidlo umožní zmazať to dno pri čítaní $a \rightsquigarrow A$ bude akceptovať slová končiace symbolom a, ale L také slová neobsahuje.

3. Počas výpočtu A nemôže nastať situácia, v ktorej by sa Z_0 vyskytol vo vnútri[2] zásobníka. Ak sa to stane, tak vznikne problém v druhej časti výpočtu. Pri čítaní b sa zásobník bude vyprázdňovať a raz sa Z_0 dostane vrch. Na tom mieste môžeme vložiť výpočet A na nejakom slove z L. Tak by sme dostali akceptačný výpočet zlého slova.

4. Teraz vieme, že zásobník počas výpočtu bude tvaru ${Z_1}^* \cdot \{Z_0, \varepsilon\}$. Pozrime sa, ako môže A v druhej časti výpočtu vyprázdniť zásobník. Ak by mal prechod $\delta(q_0, \varepsilon, Z_1) = (q_0, \varepsilon)$ tak by hneď vyprázdnil zásobník a nemohol by kontrolovať počet $b \rightsquigarrow A$ bude mať prechod $\delta(q_0, b, Z_1) = (q_0, \varepsilon)$.

5. Zistili sme, že ak sa v prvej časti výpočtu Z_0 nachádza na zásobníku, tak je na vrchu. Teraz ukážeme, že *musí* byť na vrchu. Ak by nebol na vrchu, tak A musí mať definovaný prechod $\delta(q_0, a, Z_1)$. Z tohto a predchádzajúceho bodu potom vyplýva, že by sme mohli striedať symboly a a b vo vstupnom slove a dostať akceptačný výpočet zlého slova.

6. A teda musí mať prechod $\delta(q_0, a, Z_0) = (q_0, {Z_1}^i \cdot Z_0)$ pre nejaké $i \in \mathbb{N}^+$ $\rightsquigarrow$ po prečítaní a^n zásobník bude ${Z_1}^{i \cdot n} Z_0$. Na vyprázdnenie zásobníka A potrebuje aspoň $i \cdot n + 1$ krokov a v každom kroku musí prečítať symbol zo vstupu (ε-kroky nie sú možné, lebo pracujeme s deterministickým automatom a pre Z_0 a Z_1 už máme definované prechody, ktoré čítajú vstup) $\rightsquigarrow$ A nemôže akceptovať $a^n b^n$. □

Lema 3.2. *Neexistuje DPDA A taký, že $N(A) = L$ a súčasne $|K| = 2$ a $|\Gamma| = 1$.*

Dôkaz. Bez ujmy na všeobecnosti, nech $K = \{q_0, q_1\}$ a $\Gamma = \{Z_0\}$.

[2] vnútro ≡ ľubovoľné miesto okrem prvého a posledného

1. Predpokladajme, že si automat nejako pamätá informáciu o počte a v zásobníku a v druhej časti výpočtu kontroluje počet b a vyprázdni zásobník. Keďže Γ má len jeden symbol a teda v K musí byť taký stav, že ak sa A doň dostane tak dostatočne dlhá postupnosť b na vstupe vyprázdni zásobník. Ten stav nemôže byť q_0, lebo by automat akceptoval zlé slovo b^i pre nejaké $i \in \mathbb{N}_0$ $\rightsquigarrow$ ten stav bude q_1.

2. Pre dobré slová A potrebuje vyprázdniť zásobník, a teda musí mať nejaký prechod $q_0 \rightarrow q_1$. Nemôže to byť $\delta(q_0, \varepsilon, Z_0) = (q_1, {Z_0}^i)$ ani $\delta(q_0, b, Z_0) = (q_1, {Z_0}^i)$ lebo potom by aj z q_0 dosť dlhá postupnosť b vedela vyprázdniť zásobník $\rightsquigarrow$ bude to $\delta(q_0, a, Z_0) = (q_1, {Z_0}^i)$

3. Hneď po prvom kroku sa A dostane do q_1. Aby mohol akceptovať $a^n b^n$ pre $n > 1$ musí mať definovaný prechod z q_1 na a. q_1 už ma definovaný prechod na b $\rightsquigarrow$ ak dodefinujeme ľubovoľný prechod na a tak budeme môcť na vstupe striedať a a b a aj tak dostaneme akceptačný výpočet na zlom slove $\rightsquigarrow$ spor. Teda A nemôže akceptovať L s takýmto počtom stavov a zásobníkových symbolov. □

Veta 3.1. *Automaty A_1, A_2 a A_3 sú, podľa definície 3.2, minimálne deterministické zásobníkové automaty akceptujúce jazyk L*

Dôkaz. Menší automat by musel mať parametre $(1, 2)$, $(2, 1)$ alebo $(1, 1)$, ale také podľa predchádzajúcich dvoch lem neexistujú. □

Poznámka 3.1. *Aj keď sa L nedá akceptovať menšími automatmi, malou modifikáciou dostaneme jazyk, ktorý sa akceptovať dá. Tak napríklad jazyk $L' = \{a^{n-1}b^n \mid n \in \mathbb{N}^+\}$ akceptujú nasledovné automaty:*

$$
\begin{aligned}
&A_4 = (\{q_0\}, \{a, b\}, \{Z_0, b\}, \delta_4, q_0, Z_0, \emptyset)\\
&\delta_4(q_0, a, Z_0) \;\; = (q_0, bZ_0)\\
&\delta_4(q_0, b, Z_0) \;\; = (q_1, \varepsilon)\\
&\delta_4(q_0, b, b) \;\;\;\; = (q_1, \varepsilon)
\end{aligned}
$$

$$
\begin{aligned}
&A_5 = (\{q_0, q_1\}, \{a, b\}, \{Z_0\}, \delta_5, q_0, Z_0, \emptyset)\\
&\delta_5(q_0, a, Z_0) \;\; = (q_0, Z_0Z_0)\\
&\delta_5(q_0, b, Z_0) \;\; = (q_1, \varepsilon)\\
&\delta_5(q_1, b, Z_0) \;\; = (q_1, \varepsilon)
\end{aligned}
$$

Dôkaz z lemy 3.1 sa nemôže použiť na A_4, lebo sa nedá použiť argument o počte krokov v bode 6. Na A_5 sa nemôže aplikovať dôkaz podobný tomu v leme 3.2, lebo L' už obsahuje slovo b^i pre $i = 1$ (bod 1).

Keďže automaty A_1, A_2 a A_3 sú minimálne (a neporovnateľné, podľa čiastočného usporiadania z definície 3.2) automaty akceptujúce L, bolo by dobre keby mali rovnakú zložitosť. Pre tieto konkrétne automaty sa ponúka miera „súčet počtu stavov a zásobníkových symbolov“ ako vhodný kandidát. Avšak, ľahko sa dajú nájsť príklady kde by, napríklad, súčin vyhovoval lepšie.

Nastoľuje sa otázka, či existuje taká funkcia, ktorá by pre všetky dvojice minimálnych automatov vracala rovnakú hodnotu. Taka funkcia by potom umožňovala ľahko definovať zložitosť jazykov z $\mathscr{L}_{eDPDA}$ a v ďalších úvahách by nám umožňovala vyberať si, či chceme pracovať s automatom, ktorý má viac stavov alebo viac zásobníkových symbolov. Dokážeme teraz, že taká funkcia neexistuje.

Veta 3.2. *Neexistuje funkcia $f : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ spĺňajúca nasledovné podmienky:*

1. *Pre každé dva automaty A a A' rozpoznávajúce rovnaký jazyk L platí:*
$$(k, z) \prec (k', z') \Longrightarrow f(k, z) < f(k', z')$$
2. *Ak A a A' sú dva minimálne automaty pre jazyk L tak:*
$$f(k, z) = f(k', z')$$

Dôkaz. Sporom. Nech f je funkcia spĺňajúca podmienky 1 a 2. Budeme skúmať automaty akceptujúce dva jazyky a z toho odvodíme (navzájom si protirečiace) vlastnosti, ktoré by musela f spĺňať.

Pozrime sa na (regulárny) jazyk definovaný nasledovne:

$$L_{mod2} = \{wc \mid w \in \{a, b\}^* \wedge \#_a(w) \equiv \#_b(w) \equiv 0 \pmod 2\}$$

Ako môže vyzerať minimálny automat pre L_{mod2}? Ak zafixujeme počet zásobníkových symbolov na 1, tak musí mať 4 stavy, zodpovedajúce zvyškovým triedam počtov znakov a a b modulo 2. Zároveň 4 stavy stačia, lebo nasledovný automat rozpoznáva L_{mod2}:

$$\begin{aligned}
A_6 &= (\{q_{i,j} \mid i, j \in \{0, 1\}\}, \{a, b, c\}, \{Z_0\}, \delta_6, q_{0,0}, Z_0, \emptyset), \quad |A_6| = 4\\
\delta_6(q_{i,j}, a, Z_0) &= (q_{(i+1) \bmod 2,j}, Z_0) \quad i, j \in \{0, 1\}\\
\delta_6(q_{i,j}, b, Z_0) &= (q_{i,(j+1) \bmod 2}, Z_0) \quad i, j \in \{0, 1\}\\
\delta_6(q_{0,0}, c, Z_0) &= (q_{0,0}, \varepsilon)
\end{aligned}$$

Ak povolíme použiť dva zásobníkové symboly tak potom stačia 2 stavy, lebo A_7 tiež rozpoznáva L_{mod2}.

$$\begin{aligned} A_7 &= (\{q_0, q_1\}, \{a, b, c\}, \{Z_0, Z_1\}, \delta_7, q_0, Z_0, \emptyset), \quad |A_6| = 4 \\ \delta_7(q_i, a, Z_j) &= (q_{(i+1) \bmod 2}, Z_j) \quad i, j \in \{0, 1\} \\ \delta_7(q_i, b, Z_j) &= (q_i, Z_{(j+1) \bmod 2}) \quad i, j \in \{0, 1\} \\ \delta_7(q_0, c, Z_0) &= (q_0, \varepsilon) \end{aligned}$$

Teda musí platiť $f(4,1) \leq f(2,2)$. Z predchádzajúcej časti vieme, že A_1 a A_2 sú minimálne automaty pre L_1 a teda $f(2,2) = f(3,1)$. Ale podľa podmienky 1 platí $f(4,1) > f(3,1)$. Spor. □

Keďže neexistuje funkcia zachovávajúca zložitosť pri prechode z jedného minimálneho automatu na druhý, môžeme sa pokúsiť nájsť funkciu, ktorá zachováva zložitosť pri niektorých štandardných transformáciách automatov. Jedna taká transformácia je redukcia počtu zásobníkových symbolov na 2.

Lema 3.3. *Nech $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0)$ je DPDA rozpoznávajúci jazyk L. Potom existuje DPDA A' s $|K| \cdot \max(|\Gamma| - 1, 1)$ stavmi a dvomi zásobníkovými symbolmi taký, že $L(A') = L$.*

Dôkaz. Zaujímavý prípad je keď A má viac ako dva zásobníkové symboly. Nech teda $\Gamma = \{Z_0, Z_1, \ldots, Z_{z-1}\}$, $z > 2$. Potom potrebujeme nejako zakódovať z zásobníkových symbolov do slov nad dvojprvkovou abecedou. Tu použijeme unárne kódovanie[3] — počet za sebou idúcich znakov Z_1 bude kódovať číslo symbolu a Z_0 bude slúžiť ako oddeľovač. Formálnejšie, znaky budeme kódovať nasledovným homomorfizmom $h : \Gamma^* \to \{Z_0, Z_1\}^*$:

$h(Z_i) = Z_0 {Z_1}^i$ pre $i < z - 1$

$h(Z_{z-1}) = {Z_1}^{z-1}$

Pre posledný sme nepoužili oddeľovač, lebo výsledný kód je jednoznačný aj bez neho. Presne táto optimalizácia prinesie to „-1“ vo výslednej zložitosti. Teraz môžme zostrojiť A' nasledovne:

$K' = K \times \{0, 1, \ldots, z - 2\}$

[3] dalo by sa zostrojiť aj ekvivalentné „binárne“ kódovanie

$\Sigma' = \Sigma$

$\Gamma' = \{Z_0, Z_1\}$

δ':

$$\begin{array}{lll}
\delta'((q,i),x,Z_0) & =((p,0),h(\gamma)) & \text{ak } \delta(q,x,Z_i)=(p,\gamma), \quad x \in \Sigma \cup \{\varepsilon\} \\
\delta'((q,i),\varepsilon,Z_1) & =((q,i+1),\varepsilon) & \text{pre } i < z-2 \\
\delta'((q,z-2),x,Z_1) & =((p,0),h(\gamma)) & \text{ak } \delta(q,x,Z_i)=(p,\gamma), \quad x \in \Sigma \cup \{\varepsilon\}
\end{array}$$

$q'_0 = (q_0, 0)$

$Z'_0 = Z_0$

Indukciou vzhľadom na dĺžku výpočtu sa dá dokázať, že každému výpočtu A zodpovedá výpočet A' v ktorom je obsah zásobníka zobrazený homomorfizmom h a pridané sú kroky na dekódovanie symbolov. A obrátene, každému výpočtu A' zodpovedá výpočet A a teda automaty rozpoznávajú rovnaký jazyk (L). A' očividne spĺňa požiadavky na počet stavov a zásobníkových symbolov a teda je tvrdenie vety dokázané. □

Dôsledok. *Ak zložitosť deterministického zásobníkového automatu A definujeme ako funkciu počtu stavov a zásobníkových symbolov nasledovne*

$$|A| = f(|K|, |\Gamma|) = |K| \cdot (|\Gamma| - 1)$$

tak predošlá konštrukcia zachováva zložitosť automatu pre $|\Gamma| \geq 2$.

Dôkaz. Zložitosť pôvodného automatu je $f(|K|, |\Gamma|) = |K| \cdot (|\Gamma| - 1)$.

Zložitosť nového automatu je

$$f(|K| \cdot \max(|\Gamma| - 1, 1), 2) = |K| \cdot \max(|\Gamma| - 1, 1) \cdot (2 - 1) = |K| \cdot (|\Gamma| - 1)$$

□

Táto konštrukcia nám dáva horný odhad na zložitosť automatu rozpoznávajúceho daný jazyk s dvomi zásobníkovými symbolmi. Vieme povedať, že najmenší taký automat nebude mať viac ako $|K| \cdot \max(|\Gamma| - 1, 1)$ stavov.

Počet stavov a zásobníkových symbolov nie sú jediné parametre automatu, ktoré môžu vstupovať do zložitostnej funkcie. V našom modeli môže

automat v jednom kroku uložiť ľubovoľne veľké slovo na zásobník. To umožňuje aj relatívne zložité jazyky akceptovať s malými automatmi. Tak napríklad jazyky $L_1^{(k)} = \{a^n b^{nk} \mid n \in \mathbb{N}^+\}$ pre $k \in \mathbb{N}^+$ akceptujú automaty

$$\begin{aligned} A_1^{(k)} &= (\{q_0, q_1\}, \{a, b\}, \{Z_0, a\}, \delta_1^{(k)}, q_0, Z_0, \emptyset) \\ \delta_1^{(k)}(q_0, a, Z_0) &= (q_0, a^k) \\ \delta_1^{(k)}(q_0, a, a) &= (q_0, aa^k) \\ \delta_1^{(k)}(q_0, b, a) &= (q_1, \varepsilon) \\ \delta_1^{(k)}(q_1, b, a) &= (q_1, \varepsilon) \end{aligned}$$

s parametrami $(2, 2)$, kým by pre nejaký normalizovaný automat (ktorý môže uložiť najviac jeden symbol na zásobník) počet stavov a zásobníkových symbolov závisel od k. Tento problém by sa dal riešiť buď obmedzením sa na automaty v normálnom tvare, alebo pridaním ešte jedného parametra, ktorý bude nejako závisieť od dĺžky pravých strán δ-funkcie.

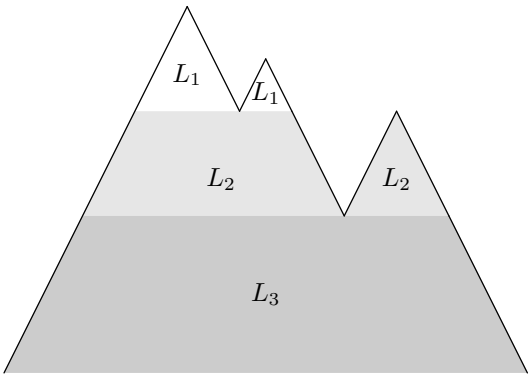
3.2 Stavová zložitosť

Iný prístup je zobrať hrubšiu mieru zložitosti. Videli sme, že už pre „dvojrozmernú“ funkciu sa ťažko definuje totálne usporiadanie automatov. Pre *deterministické* zásobníkové automaty má zmysel definovať mieru „počet stavov“, keďže sa tu nedá použiť tá transformácia, ktorá na nedeterministických automatoch zredukuje počet stavov na 1. Preto má zmysel uvažovať o minimálnom počte stavov potrebnom na akceptovanie nejakého jazyka, a v ďalšom texte sa budeme práve s touto mierou zaoberať.

Definícia 3.4. *Nech A sú deterministické zásobníkové automaty. Zložitosť automatu A sa rovná počtu stavov. A je jednoduchší (resp. nie zložitejší) ako DPDA A' ak $|K_A| < |K_{A'}|$ (resp. $|K_A| \leq |K_{A'}|$).*

Definícia 3.5. *Nech $L \in \mathscr{L}_{eDPDA}$. Zložitosť jazyka L je zložitosť najjednoduchšieho automatu, ktorý rozpoznáva L.*

Všimnime si, že toto usporiadanie zachováva vzťahy vyplývajúce z definície 3.2: Ak má automat A menej stavov ako automat A' (tj. je jednoduchší podľa 3.4) tak buď je A jednoduchší aj podľa 3.2, alebo sú automaty neporovnateľné. Zložitosť jazyka je dobre definovaná, lebo zložitosť automatu je prirodzené číslo a množina prirodzených čísel je dobre usporiadaná.



Obr. 3.1: Ukážka grafického znázorňovania slov z L_i

V nasledovnej kapitole skúmame ako môže prídavná informácia ovplyvniť zložitosť automatov a jazykov. Aby sme to mohli robiť, potrebujeme presne určiť zložitosť nejakých jazykov. Preto sa teraz pozrieme na nasledovnú postupnosť jazykov:

- $L_1 = \{a^n b^n \mid n \in \mathbb{N}^+\}$
- $L_{i+1} = \{a^n w b^n \mid n \in \mathbb{N}^+ \wedge w \in {L_i}^*\}$

Slová z L_i obsahujú rovnaký počet „správne uzátvorkovaných“ symbolov a a b. Rozdiel je v počte vnorených hniezd zátvoriek ktoré slová môžu obsahovať. Tak napríklad slovo $aaaaaaabbabbbaabbbbb$ patrí do $L_3 \setminus L_2$, lebo

$$aaaaaaabbabbbaabbbbb = a^3 \underbrace{a^2 \overbrace{a^2b^2}^{\in L_1} \overbrace{ab}^{\in L_1} b^2}_{\in L_2 \setminus L_1} \underbrace{a^2b^2}_{\in L_2} b^3 \in L_3 \setminus L_2$$

Slová z L_i môžeme aj graficky znázorňovať. Ako príklad uvádzame graficku reprezentáciu slova $a^7b^2abb^2a^3b^2b^3$ na obrázku 3.1.

Pre každé $n \geq 2$ zostrojíme automat[4], ktorý bude mať n stavov a $n + 2$ zásobníkových symbolov a bude akceptovať L_n. Následne ukážeme, že neexistuje automat, ktorý by mal menší počet stavov.

[4]jazyk L_1 sme už skúmali v časti 3.1

$$A_n = (\{q_1, q_2, \ldots, q_n\}, \{a, b\}, \{Z_0, Z_1, \ldots, Z_{n-1}, a, b\}, \delta_n, q_1, Z_0, \emptyset)$$

$$\begin{array}{lcll}
\delta_n(q_1, a, Z_0) & = & (q_1, a) & \\
\delta_n(q_1, a, a) & = & (q_1, ba) & \\
\delta_n(q_1, b, a) & = & (q_1, \varepsilon) & \\
\delta_n(q_i, b, b) & = & (q_i, \varepsilon) & i = 1 \ldots n \\
\delta_n(q_i, b, Z_j) & = & (q_{max(i,j)+1}, \varepsilon) & i, j = 1 \ldots n-1 \\
\delta_n(q_i, a, Z_j) & = & (q_1, Z_{max(i,j)}a) & i, j = 1 \ldots n-1 \\
\delta_n(q_i, a, b) & = & (q_1, Z_{i+1}a) & i = 1 \ldots n
\end{array}$$

Tvrdenie 3.2. $\forall n \geq 2 \quad N(A_n) = L_n$

Automat pracuje podobne ako A_2 z časti 3.1 — pri práci vždy zachováva invariant „počet otvorených zátvoriek“ = „počet symbolov na zásobníku“. Navyše však musí kontrolovať úroveň vnorenia hniezd zátvoriek (symbolov a a b). To robí na dvoch miestach. Na zásobník si ukladá symboly Z_i, ktoré označujú miesta, kde sa končí nejaké hniezdo. Číslo i hovorí koľko vnorených hniezd tam bolo. V stave si A_n pamätá koľko vnorení videl v hniezde, ktoré práve spracuje.

Pre každý jazyk z $\mathscr{L}_{CF}$ existuje nedeterministický zásobníkový automat, ktorý ma len jeden stav. Pre DPDA to neplatí a tu ukážeme, že sú v tomto zmysle automaty A_n optimálne.

Veta 3.3. *Nech $n \in \mathbb{N}^+$. Neexistuje DPDA A taký, že $N(A) = L_n$ a súčasne $|K_A| < n$.*

Dôkaz. Sporom, nech taký A existuje. Bez ujmy na všeobecnosti, nech $|K_A| = n - 1$ a $|\Gamma_A| = z$. Pozrime sa na výpočet A na slove $c_1 c_2 \cdots c_l$ ($c_i \in \{a, \varepsilon\}$).

$$(p_0, c_1 c_2 \cdots c_l, z_0) \vdash (p_1, c_2 \cdots c_l, s_1 z_1) \vdash \cdots \vdash (p_l, \varepsilon, s_l z_l)$$

Pre každé i je a^i prefix nejakého slova z L_n a teda A sa nesmie zaseknúť pri jeho čítaní. Pre $l > z \cdot n$ sa, podľa Dirichletovho princípu, musí vo výpočte vyskytnúť nejaká dvojica (stav, zásobníkový symbol na vrchu zásobníka) aspoň dvakrát. Ak na vstup budú aj ďalej prichádzať symboly a, A bude pracovať v „cykloch“. Konfigurácie sa nemôžu opakovať, lebo A by potom nevedel,

koľko a prečítal a nemohol by akceptovať správne slovo. Zásobník sa preto bude postupne zväčšovať, ale bude obsahovať blok, ktorý sa periodicky opakuje. Formálnejšie, zvoľme i a j také, že $(q_i, z_i) = (q_j, z_j)$ a zásobník A v krokoch $i \ldots j$ nikdy neklesol pod úroveň, na ktorej bol v i-tom kroku. Nech γ je obsah zásobníka, ktorý pribudol v tých krokoch. Potom pre všetky dosť veľké i zásobník A po prečítaní a^i bude $s_p\gamma^k s_s^{(i)}$ (pre nejaké pevné s_p a k, $s_s^{(i)}$ závislé od i). Zvoľme nejaké pevné s_s a pre $k = 1, 2, \ldots$ nech n_k sú také prirodzené čísla, že zásobník A po prečítaní a^{n_k} bude $s_p\gamma^k s_s$.

Nech m je dostatočne veľké prirodzené číslo. Definujme postupnosť slov w_i takto:

- $w_1 = ab$
- $w_{i+1} = aw_ibab$

Ako sa bude A správať na $u_i = a^{n_k}w_ib^{n_k}$ pre $i = 1 \ldots n$ (vtedy $u_i \in L_n$)? Pri spracovaní strednej časti slova zásobník určite nemôže hlboko pod úroveň, na ktorej bol po prečítaní a^{n_k}, lebo „zabudne“ koľko znakov a prečítal. Pri čítaní b^{n_k} sa zásobník musí vyprázdniť, a teda raz sa automat dostane do konfigurácie $(q_{x_i}, b^{l_i}, s_p\gamma^{k-1})$ pre nejaké l_i, x_i. Ale A má len $n-1$ stavov $\rightsquigarrow$ $q_{x_i} = q_{x_j}$ pre nejaké $i, j \in \{1, 2, \ldots, n\}; i < j$.

Nech teraz

$$v_i = a^{n_k}w_ib^{n_k-l_i}(abb)^{n-i}b^{l_i-(n-i)} \in L_n$$

$$v_j = a^{n_k}w_jb^{n_k-l_j}(abb)^{n-j}b^{l_j-(n-j)} \in L_n$$

Tieto slová majú rovnaké prefixy ako u_i a u_j a teda pri ich spracovaní sa automat dostane do konfigurácie

$$(q_{x_i}, (abb)^{n-i}b^{l_i-(n-i)}, s_p\gamma^{k-1})$$

$$(q_{x_j}, (abb)^{n-j}b^{l_j-(n-j)}, s_p\gamma^{k-1})$$

Ale $q_{x_i} = q_{x_j}$ $\rightsquigarrow$ A nemôže tieto dve konfigurácie rozlíšiť a môžeme vymeniť sufixy slov a prislúchajúce časti výpočtu a dostaneme akceptačný výpočet na slove $a^{n_k}w_jb^{n_k-l_j}(abb)^{n-i}b^{l_i-(n-i)}$, ktoré nepatrí do L_n, lebo obsahuje viac ako n vnorených hniezd (j hniezd vo w_j a $n-i > n-j$ kvôli časti $(abb)^{n-i}$) $\rightsquigarrow$ spor s $N(A) = L_n$. □

Automat A_n potrebuje n stavov, lebo si musí pamätať informáciu o úrovni vnorenia aj keď zásobník klesá. Tu ukážeme, že to je vlastne jediný prípad, keď si (deterministický) zásobníkový automat musí pamätať informáciu v stave.

Veta 3.4. *Nech $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0)$ je DPDA rozpoznávajúci jazyk L. Nech K_ε je množina všetkých stavov p takých, že sa A do nich dostane v kroku v ktorom klesol zásobník. Presnejšie,*

$$K_\varepsilon = \{p \in K \mid \exists q \in K \;\; \exists x \in \Sigma \cup \{\varepsilon\} \;\; \exists Z \in \Gamma \quad \delta(q, x, Z) = (p, \varepsilon)\}$$

Potom existuje DPDA A' rozpoznávajúci jazyk L ktorý má len $|K_\varepsilon|$ stavov.

Dôkaz. Ak $K_\varepsilon = K$ tak $A' = A$. Nech teda $K \setminus K_\varepsilon \neq \emptyset$. Nech p^* je ľubovoľný stav z K_ε. Automat A' zostrojíme nasledovne:

$K' = K_\varepsilon$, $\Sigma' = \Sigma$

$\Gamma' = \Gamma \cup (\Gamma \times (K \setminus K_\varepsilon))$

δ':

$$\begin{array}{lll}
\delta'(q, x, Z) = (p^*, \gamma \cdot (Z', q')) & \text{ak } \delta(q, x, Z) = (q', \gamma Z'), q \in K_\varepsilon, q' \notin K_\varepsilon \\
\delta'(q, x, Z) = (q', \gamma Z') & \text{ak } \delta(q, x, Z) = (q', \gamma Z'), q \in K_\varepsilon, q' \in K_\varepsilon \\
\delta'(p^*, x, (Z, q)) = (p^*, \gamma \cdot (Z', q')) & \text{ak } \delta(q, x, Z) = (q', \gamma Z'), q \notin K_\varepsilon, q' \notin K_\varepsilon \\
\delta'(p^*, x, (Z, q)) = (q', \gamma Z') & \text{ak } \delta(q, x, Z) = (q', \gamma Z'), q \notin K_\varepsilon, q' \in K_\varepsilon \\
\delta'(q, x, Z) = (q', \varepsilon) & \text{ak } \delta(q, x, Z) = (q', \varepsilon), \quad q \in K_\varepsilon, q' \in K_\varepsilon \\
\delta'(p^*, x, (Z, q)) = (q', \varepsilon) & \text{ak } \delta(q, x, Z) = (q', \varepsilon),
\end{array}$$

$q'_0 = q_0$, $Z'_0 = Z_0$ ak $q_0 \in K_\varepsilon$

$q'_0 = p^*$, $Z'_0 = (Z_0, q_0)$ ak $q_0 \notin K_\varepsilon$

Jediný rozdiel medzi A a A' je v tom, že si A', keď sa to dá, ukladá aj informáciu o stave na zásobník. To môže robiť vtedy keď výška zásobníku neklesá. Ľahko možno vidieť (a indukciou dokázať), že konfigurácie A a A' budú rovnaké až na znak na vrchu zásobníka a stav automatu.

Teda, bude platiť nasledovný invariant: Nech q je stav a Z je zásobníkový symbol A, $i \in \mathbb{N}$, $w \in \Sigma^*$. Ak $q \notin K_\varepsilon$ tak i-ta konfigurácia A pri práci na w bude mať stav q a vrchný zásobníkový symbol Z práve vtedy keď i-ta konfigurácia A' na w bude mať stav p^* a vrchný zásobníkový symbol (Z, q). Ak $q \in K_\varepsilon$ tak i-té konfigurácie budú rovnaké. □

Z tejto vety vyplýva aj to, že niektoré operácie na jazyku zachovávajú zložitosť. Môžeme do slov vložiť nejakú regulárnu časť, prípadne vykonať regulárnu substitúciu a výsledný jazyk sa bude dať akceptovať s rovnakým počtom stavov ako ten pôvodný. Avšak, zmena musí byť taká, aby sa dalo jednoznačne počas čítania povedať, že sme v regulárnej časti (napr. substituovaný jazyk môže mať disjunktnú abecedu). V tom prípade si nemusíme stav konečného automatu pamätať v stave DPDA, ale môžeme ho držať vo vrchnom symbole zásobníka.

Dôležité je všimnúť si, že táto konštrukcia nemusí vyrobiť minimálny automat. Môže sa stať, že existuje iný automat, ktorý akceptuje rovnaký jazyk, ale na úplne iný spôsob, a potrebuje na to menej stavov.

3.3 Stavová zložitosť doplnku jazyka

Je známe, že trieda jazykov akceptovaných *deterministickými* zásobníkovými automatmi je uzavretá na komplement. Formálny dôkaz môžeme nájsť napríklad v [5]. Kľúčovou časťou toho dôkazu je lema o existencii normálneho tvaru, v ktorom automat vždy dočíta vstup.

Lema 3.4. *Ku každému deterministickému zásobníkovému automatu A existuje DPDA A' taký, že $L(A) = L(A')$ a A' vždy dočíta vstup a zastane.*

Dôkaz lemy v [5] je konštruktívny a zostrojí automat, ktorý má exponenciálne veľa stavov. V tých stavoch si A' pamätá koľko po sebe idúcich krokov na ε automat urobil. Tak môže overiť či A nerobí kroky na ε do nekonečna. To sa stane ak A urobí viac ako $s(t+1)^{rst}$ ε-krokov, pre $t = |\Gamma|$, $s = |K|$ a r — dĺžku najdlhšieho slova, ktoré vie A v jednom kroku vložiť na zásobník. V tom prípade A' prejde do nového stavu, v ktorom už len dočíta vstup (a neakceptuje).

Tu ukážeme, že sa to isté dá urobiť s výrazne menším počtom stavov. Myšlienka spočíva v predvýpočte — keďže A nečíta žiadne symboly zo vstupu, jeho správanie závisí len od vlastností δ-funkcie. My si teda môžeme pre každý stav a symbol na vrchu zásobníka vypočítať do akej konfigurácie sa A dostane ε-krokmi. A' potom už len skočí do tej konfigurácie.

Lema 3.5. *Ku každému deterministickému zásobníkovému automatu A existuje DPDA A' taký, že $L(A) = L(A')$ a A' vždy dočíta vstup a zastane. Navyše, ak má A aspoň jeden neakceptačný stav, tak platí $|K'| = |K|$ a $|\Gamma'| = |\Gamma| + 2$. Ak $F = K$ tak $|K'| = |K| + 1$ a $|\Gamma'| = |\Gamma| + 2$.*

Dôkaz. Stavy A budú aj stavmi A'. Označme s q_R nejaký neakceptačný stav A. Ak A taký stav nemá, tak pridáme nový stav q_R do K'.

A sa môže zaseknúť tak, že si vyprázdni zásobník. Tento prípad ošetríme tak, že pridáme dva nové zásobníkové symboly: Z_0' a Z_s. Z_0' bude nový začiatočný zásobníkový symbol a jeho jediný účel je, aby ho automat v prvom kroku nahradil s $Z_s Z_0$ $\big(\delta(q_0, Z_0', \varepsilon) = (q_0, Z_s Z_0)\big)$. Ďalej bude výpočet pokračovať rovnako ako v pôvodnom automate, ale na spodku zásobníka bude zarážka Z_s. Ak sa to Z_s dostane na vrch zásobníka, tak sa A zasekol. A' potom prejde do stavu q_R, v ktorom už len dočíta vstup (a neakceptuje). Presnejšie, A' bude mať nasledovné prechody:

$$\delta'(q, a, Z_s) = (q_R, Z_s) \quad q \in K,\ a \in \Sigma$$

Môže sa stať, že sa A zasekne preto, že nebude mať definovaný prechod pre nejakú dvojicu (stav, písmeno vstupu, zásobníkový symbol). Aby sa A' nezasekol, pre každú takú dvojicu dodefinujeme prechod:

$$\delta'(q, a, Z) = (q_R, Z_s) \quad a \in \Sigma,\ \delta(q, a, Z) = \emptyset \wedge \delta(q, \varepsilon, Z) = \emptyset$$

Takto sa Z_s dostane na vrch zásobníka, a A' potom už bude „vedieť“, že má len dočítať vstup.

Jediný spôsob ako automat teraz môže nedočítať vstup je ak bude robiť ε-kroky do nekonečna. Preto potrebujeme upraviť prechodovú funkciu. Nech má A definovaný ε-prechod pre kombináciu stavu q a zásobníkového symbolu Z. Pozrime sa ako bude vyzerať výpočet A z konfigurácie (q, ε, Z). Zaujíma nás, do akých konfigurácií sa automat dostane bez čítania vstupu keď na zásobníku bude mať len Z. Môžu nastať nasledovné možnosti:

1. A si vyprázdni zásobník, tj. dostane sa do konfigurácie $(p, \varepsilon, \varepsilon)$. Potom môžeme do A' pridať skratku $\delta'(q, \varepsilon, Z) = (p, \varepsilon)$.

2. A bude robiť ε-kroky do nekonečna. Vieme, že sa toto stane, ak A urobí aspoň $s(t+1)^{rst}$ krokov. V tom prípade A neakceptuje vstup a my predefinujeme prechod na $\delta'(q, \varepsilon, Z) = (q_R, Z_s)$.

3. A sa zasekne v stave p a zásobníkom $\gamma \neq \varepsilon$. Podľa [5] vieme, že dĺžka γ je najviac rst. V A' definujeme $\delta'(q, \varepsilon, Z) = (p, \gamma)$.

A' sa nezasekne, lebo má pre každú kombináciu stavu a zásobníkového symbolu definovaný buď ε-prechod, alebo prechody na všetky písmená vstupnej abecedy. ε-kroky nemôže robiť donekonečna, lebo po každom ε-kroku typu 2 alebo 3 už nasleduje krok na písmeno zo vstupu, a počet krokov typu 1 je obmedzený veľkosťou zásobníka. Na slovách na ktorých sa A nezasekne (a necyklí) sa automaty správajú rovnako, až na to že A' používa skratky pri ε-krokoch. Ak sa A zasekne, alebo by mal robiť ε-kroky donekonečna tak A' prejde do stavu q_R v ktorom už len dočíta vstup a neakceptuje. Teda automaty akceptujú rovnaký jazyk a keďže sme pridali najviac jeden stav a dva zásobníkové symboly tak je lema dokázaná. □

Veta 3.5. *Ku každému deterministickému zásobníkovému automatu A existuje DPDA A' taký, že $L(A)^C = L(A')$. Navyše, ak má A aspoň jeden neakceptačný stav, tak platí $|K'| = 3|K|$ a $|\Gamma'| = |\Gamma| + 2$. Ak $F = K$ tak $|K'| = 3|K| + 3$ a $|\Gamma'| = |\Gamma| + 2$.*

Dôkaz. Dôkaz prvej časti tvrdenia môžeme nájsť v [5]. Konštrukcia používa tri kópie množiny stavov automatu v normálnom tvare. Keď použijeme odhad z lemy 3.5 dostaneme aj druhú časť. □

Kapitola 4

Využitie prídavnej informácie

V tejto kapitole budeme skúmať ako (a či vôbec) môže regulárna prídavná informácia pomôcť deterministickému zásobníkovému automatu. V prvej časti ukážeme príklady, kedy nám tá informácia umožní zostrojiť výrazne jednoduchší automat. Kapitolu uzavrieme sériou tvrdení, ktoré ukazujú, že (minimálnym) automatom pre postupnosť jazykov L_n nepomôže žiadna regulárna informácia (t.j. automaty sú nerozložiteľné).

4.1 Dobre rozložiteľné jazyky

Skúmanie vplyvu prídavnej informácie na zložitosť môžeme robiť na dvoch úrovniach. Prvá možnosť je, že si vezmeme konkrétny automat A. Potom sa snažíme nájsť taký poradný jazyk, aby nám informácia, že slovo patrí do toho jazyka pomohla zostrojiť jednoduchší automat A'. Nový automat musí, samozrejme, rozpoznávať rovnaký jazyk, ale bude mať pomoc v tvare regulárneho poradného jazyka.

Napríklad, skúmajme takúto postupnosť automatov $A_1, A_2, \ldots$

$$
\begin{aligned}
&A_n = (\{q_0, \ldots, q_{n-1}\}, \{a, b\}, \{Z_0, Z_1, a\}, \delta_n, q_0, Z_0, \emptyset) \\
&\delta_n(q_0, a, Z_0) = (q_1, Z_1) \\
&\delta_n(q_i, a, x) \;\; = (q_{(i+1) \bmod n}, xa) \qquad i \in \{0, \ldots, n-1\} \quad x \in \{Z_1, a\} \\
&\delta_n(q_i, b, a) \;\; = (q_i, \varepsilon) \qquad\qquad i \in \{0, \ldots, n-1\} \\
&\delta_n(q_0, b, Z_1) = (q_0, \varepsilon)
\end{aligned}
$$

Automat A_n rozpoznáva jazyk

$$D_{(\text{mod n})} = \{awb \mid w \in D_1 \wedge \#_a(awb) \equiv 0 \pmod{n}\}$$

kde D_1 je *Dyck*ov jazyk pozostávajúci zo správne uzátvorkovaných symbolov a a b. Automat má zložitosť n (keď uvažujeme mieru „počet stavov“). Avšak, ľahko vidno že ak za (regulárny) poradný jazyk zvolíme

$$L_{(\text{mod n})} = \{w \in \{a,b\}^+ \mid \#_a(w) \equiv 0 \pmod{n}\}$$

tak $D_{(\text{mod n})}$ môžeme akceptovať s len jedným stavom: keď už vieme, že slovo má správny počet symbolov a tak stačí skontrolovať či sú symboly dobre uzátvorkované. To vie urobiť automat $A' = A_1$ s jedným stavom a teda platí

$$D_{(\text{mod n})} = L(A_1, L_{(\text{mod n})}) = L_{A_1} \cap L_{(\text{mod n})}$$

Problém s týmto prístupom je, že my, vo všeobecnosti, nevieme povedať či je zjednodušenie automatu spôsobené prítomnosťou poradného jazyka. Môže sa stať, že existuje automat A_m, jednoduchší od nami zvoleného automatu A,[1] ktorý rozpoznáva rovnaký jazyk. Preto je lepšie skúmať efekty prídavnej informácie na úrovni jazykov. Jazyk môžeme stotožniť s (nejakým) minimálnym automatom, ktorý ho rozpoznáva. Teraz keď dokážeme, že s regulárnou pomocou dokážme zostrojiť menší automat, ako je minimálny automat pre daný jazyk, tak to znamená, že tá regulárna informácia je *pre tento konkrétny problém*[2] dôležitá. Najzložitejšia časť je potom dôkaz, že nejaký automat *je* minimálny automat pre daný jazyk, čo môžeme aj vidieť z nasledujúcej vety.

Veta 4.1. *Pre každé $n \in \mathbb{N}^+$ platí, že automat A_n je minimálnym automatom pre jazyk $L_{(mod\ n)}$. Tj. neexistuje deterministický zásobníkový automat A'', ktorý rozpoznáva $L_{(mod\ n)}$ a platí $|K''| < n$.*

Pred tým, ako pristúpime k dôkazu vety, uvedieme pomocné definície, ktoré budeme používať v tejto kapitole.

Definícia 4.1. *Nech A je deterministický zásobníkový automat. Nech $w \in \Sigma^*$.* Situácia *po prečítaní slova w automatom A je dvojica $(s,q) \in \Gamma^* \times K$, kde s je obsah zásobníka a q je stav automatu a platí:*

$$(q_0, w, Z_0) \vdash_A^* (q, \varepsilon, s)$$

[1] dokonca, môže byť aj jednoduchší ako A'

[2] inému jazyku/automatu tá informácia nemusí vôbec pomôcť

Definícia 4.2. *Nech A je DPDA. Čiastočné funkcie $c_A : \Gamma^* \times K \to \mathbb{N}_0$, $B_A : \Gamma^* \times K \to \{b\}^*$ a $S_A : \Gamma^* \times K \to K$ definujeme nasledovne:*

Nech $s \in \Gamma^*$ a $q \in K$. Nech existujú $m \in \mathbb{N}_0$ a $p \in K$ také, že

$$(q, b^m, s) \vdash_A^* (p, \varepsilon, \varepsilon)$$

potom $c_A(s, q) = m$, $B_A(s, q) = b^m$, $S_A(s, q) = p$.

Keď je jasné, o ktorý automat ide, tak nepíšeme index A.

Tieto pojmy budeme používať v nasledujúcich dôkazoch na postupné zostrojovanie vstupu do automatu: Automatu dáme časť vstupu, pozrieme sa aká je „situácia“ po jeho dočítaní a na základe toho pridáme ďalšiu časť. Ak vieme, že sa už prečítané slovo dá doplniť symbolmi b na slovo z akceptačného jazyka skúmaného automatu tak funkcia c_A udáva počet symbolov. Napríklad, ak slovo w' má tú vlastnosť, a situácia po jeho prečítaní je (s', q') tak potom situácia po prečítaní $w'b^{c(s',q')} = w'B(s', q')$ je $(\varepsilon, S(s', q'))$.[3] Navyše, funkcie majú nasledovné vlastnosti:

$$c(s_2 s_1, q) = c(s_1, q) + c(s_2, S(s_1, q)) \tag{4.1}$$

$$B(s_2 s_1, q) = B(s_1, q) \cdot B(s_2, S(s_1, q)) \tag{4.2}$$

Po tejto odbočke už môžeme pokračovať v dôkaze.

Dôkaz (vety 4.1)**.** Zvoľme pevné $n \geq 2$ (pre $n = 1$ nie je čo dokazovať). Postupovať budeme sporom. Nech taký A'' existuje. Nech má $n - 1$ stavov a nech $K'' = \{q_0, \ldots, q_{n-2}\}$. Automat musí kontrolovať počet symbolov a a b. Preto môžeme, podobne ako vo vete 3.3, dokázať že na vstupe a^i bude zásobník automatu neobmedzene rásť.

Keď A'' prečíta všetky dostatočne dlhé slová a^i, situácia bude tvaru $(s_p \gamma^k s_s^{(i)}, q^{(i)})$, pričom $\gamma \neq \varepsilon$ a hodnota k, $s_s^{(i)}$ a $q^{(i)}$ závisí od i. Pri čoraz väčšom i sa hodnota k postupne zväčšuje a hodnoty $s_s^{(i)}$ a $q^{(i)}$ sa periodicky opakujú. Preto ďalšie úvahy nezávisia od konkrétnej hodnoty i. My teda zvolíme nejaké pevné s_s, q a i budeme meniť tak, aby sme dosiahli potrebnú hodnotu k (zachovávajúc pritom invarianty $s_s^{(i)} = s_s$ a $q^{(i)} = q$).[4]

Zvoľme i také, aby k bolo väčšie ako n. Nech situácia po prečítaní a^i je $(s_p \gamma^k s_s, q)$. Keď pridáme na vstup ešte $n - 1$ symbolov a, zásobník nemôže

[3] Uvažujeme akceptovanie prázdnou pamäťou

[4] Takýto spôsob uvažovania budeme používať aj v ďalších dôkazoch

klesnúť príliš hlboko — určite neklesne až po s_p. Preto pre $j = 0, \ldots, n-1$ môžeme situáciu po prečítaní a^{i+j} zapísať ako $(s_p\gamma_j, p_j)$. Definujme slová $w_j = a^{i+j}B(\gamma_j, p_j)$. Potom po prečítaní w_j situácia bude (s_p, p'_j), kde $p'_j = S(\gamma_j, p_j)$. Keďže máme n slov a automat má len $n-1$ stavov, musí platiť, že niektoré dve situácie sú rovnaké. Nech sú to situácie j_1 a j_2 ($j_1 \neq j_2$). Slovo w_j obsahuje $i+j$ symbolov a. Čísla $i+j$ majú rôzne zvyšky po delení n. Nech číslo x označuje počet symbolov a potrebných na to, aby $i+j_1$ bolo deliteľné s n ($x = n-(i+j_1) \bmod n$). To znamená že A'' by mal akceptovať slovo

$$w_{j_1}\ a^x\ b^x\ B(s_p, p'_{j_1})$$

a zamietnuť

$$w_{j_2}\ a^x\ b^x\ B(s_p, p'_{j_2})$$

To ale nie je možné, lebo A'' nevie rozlíšiť slová w_{j_1} a w_{j_2} (situácie po ich prečítaní sú rovnaké). Spor. □

Z tohto dôkazu vyplýva, že informácia že slovo patrí do jazyka $L_{(\text{mod n})}$ *je* dôležitá keď rozhodujeme príslušnosť do jazyka $D_{(\text{mod n})}$ a umožňuje nám zmenšiť počet stavov zásobníkového automatu z n na 1. Teda, môžeme povedať, že rozklad jazyka $D_{(\text{mod n})}$ na jazyky $L_{(\text{mod n})}$ a D_1 je netriviálny, podľa nasledovnej definície.

Definícia 4.3. *Nech $L, L_1 \in \mathscr{L}_{eDPDA}$. Nech $L_2 \in \mathscr{R}$ a nech platí $L = L_1 \cap L_2$. Nech A resp. A_1 sú automaty rozpoznávajúce jazyky L resp. L_1 a nech A je minimálny automat rozpoznávajúci L (podľa miery „počet stavov“). Hovoríme, že deterministický bezkontextový jazyk L je* netriviálne *rozložiteľný na jazyky L_1 a L_2 ak automat A_1 má (ostro) menej stavov ako A.*

Táto definícia neobmedzuje zložitosť konečného automatu rozpoznávajúceho L_2 a teda môže sa stať, že automat bude mať výrazne viac stavov ako pôvodný automat A. Aj napriek tomu, my taký rozklad budeme považovať za netriviálny, keďže konečný automat (aj keď má viacej stavov) je v istom zmysle jednoduchší ako ľubovoľný zásobníkový automat. V ďalších častiach ukážeme príklady jazykov, ktoré nebudú netriviálne rozložiteľné, aj keď konečnému automatu dovolíme mať ľubovoľný počet stavov.

4.2 Nerozložiteľné jazyky

V kapitole 3 sme definovali postupnosť jazykov $\{L_n\}_{n=1}^{\infty}$ (strana 14) a dokázali sme, že zložitosť n-tého jazyka postupnosti je presne n. Teraz nás zaujíma, či nejaká regulárna informácia nám umožní zjednodušiť automat rozpoznávajúci L_n. Na prvý pohľad by sa mohlo zdať, že to je možné, lebo slová tých jazykov majú nejaké obmedzenia na úroveň zahniezdenia — pre pevné n je to konštanta a teda by to možno konečný automat mohol kontrolovať. Potom by zásobníkový automat kontroloval len počet jednotlivých symbolov, a na to mu stačí jeden stav.

Po krátkom zamyslení však zistíme, že to nie je pravda. Totiž, úroveň zahniezdenia závisí od presného počtu symbolov v slove. Napríklad, slovo

$$a^{3m}b^m a^m b^m a^m b^{3m} = a^m a^m \overbrace{a^m b^m}^{\in L_1} \overbrace{a^m b^m}^{\in L_1} \overbrace{a^m b^m}^{\in L_1} b^m b^m$$

patrí do L_2, ale slovo

$$a^{3m}b^m a^m b^{2m} a^m b^{2m} = a^m \underbrace{a^m \overbrace{a^m b^m}^{\in L_1} \overbrace{a^m b^m}^{\in L_1} b^m}_{\in L_2 \setminus L_1} \underbrace{a^m b^m}_{\in L_2} b^m$$

už patrí do $L_3 \setminus L_2$. V tejto časti aj formálne dokážeme, že sa minimálne automaty jazykov L_n nedajú rozložiť na dva jednoduchšie automaty: deterministický zásobníkový automat s menším počtom stavov a konečný automat (s ľubovoľným počtom stavov). Formálny dôkaz bude pozostávať z dvoch častí:

1. nájdeme nekonečnú množinu slov (označme ju L_{zle}), ktoré musí akceptovať DPDA ktorý akceptuje nadmnožinu L_n. Tento dôkaz je podobný tomu z vety 3.3, ktorý vlastne nájde jednoprvkovú množinu L_{zle}. Tu to však musíme spraviť podrobnejšie, lebo potrebujeme nekonečnú (a neregulárnu) množinu, aby sme ju mohli použiť v druhom kroku.

2. ukážeme, že sa nedá zostrojiť DKA, ktorý bude akceptovať (všetky) slová z L_n a súčasne *nebude* akceptovať slová z L_{zle}. Tu využijeme neregularitu L_{zle} a ukážeme, že automat nemôže odlíšiť „dobré“ slovo z L_n a „zlé“ slovo z L_{zle}.

Prvý krok z didaktických dôvodov ešte rozčleníme na viac lem. Začneme s najjednoduchším prípadom — jazykom L_2.[5]

Lema 4.1. *Nech A je jednostavový DPDA taký, že $L_2 \subseteq N(A)$. Potom existuje $m \in \mathbb{N}^+$ také, že pre všetky $j, l \in \mathbb{N}^+$ existujú $i, k, t \in \mathbb{N}_0 : k > 2j$ také, že A akceptuje*

$$a^i b^t \; a^l b^l \; b^{jm} \; ab \; b^{(k-j)m} b^{i-t-km}$$

Dôkaz. Keďže A má len jeden stav (označme ho q_0), tak druhý parameter funkcií c_A a B_A nebudeme písať. Špeciálne, vzťahy 4.1 a 4.2 môžeme zapísať ako:

$$c(s_2 s_1) = c(s_1) + c(s_2) \tag{4.3}$$

$$B(s_2 s_1) = B(s_1) \cdot B(s_2) \tag{4.4}$$

a platí, že ak w je prefix slova z L_2 a situácia po jeho prečítaní je (s, q_0) tak potom $c(s) = \#_a(w) - \#_b(w)$.

Zásobník A po prečítaní dosť dlhých a^i bude tvaru $s_p \gamma^k s_s^{(i)}$, pričom $\gamma \neq \varepsilon$ a hodnota k a $s_s^{(i)}$ závisí od i. Keď postupne zväčšujeme i tak hodnota k stúpa, a hodnota $s_s^{(i)}$ sa periodicky opakuje. Preto ďalšie úvahy nezávisia od konkrétnej hodnoty i. My teda zvolíme nejaké pevné s_s a i budeme meniť tak, aby sme dosiahli potrebnú hodnotu k (zachovávajúc pritom invariant $s_s^{(i)} = s_s$).

Teda, po prečítaní $u = a^i B(\gamma s_s)$ zásobník bude $s_p \gamma^{k-1}$ (obsah zásobníka možno sledovať aj na obrázku 4.1). Aký bude obsah zásobníka po prečítaní $u\, a^l b^l$, $l \in \mathbb{N}^+$? Počas spracovania $a^l b^l$ zásobník nikdy neklesne na úroveň $s_p \gamma^{k-2}$. Ukážeme to sporom. Nech zásobník klesne na $s_p \gamma^{k-2}$ po prečítaní uu_0, kde u_0 je nejaký prefix $a^l b^l$. Potom platí $\#_a(u_0) \geq \#_b(u_0)$ a A bude akceptovať slovo $uu_0 B(s_p \gamma^{k-2})$ ale nie aj $uu_0 b^{\#_a(uu_0) - \#_b(uu_0)} \in L_2$, lebo

$$\#_a(uu_0) - \#_b(uu_0) \geq i - \#_b(u) = c(s_p \gamma^k s_s) - c(\gamma s_s) > c(s_p \gamma^{k-2})$$

Situáciu po prečítaní $u\, a^l b^l$ môžeme teda zapísať ako $(s_p \gamma^{k-2} \gamma', q_0)$. Hodnota γ' môže závisieť od l, ale pre každé l musí platiť

$$c(\gamma') = c(\gamma)$$

[5] Jazyk L_1 nemá zmysel uvažovať, keďže jeho minimálny automat má len jeden stav a teda zrejme neexistuje jednoduchší automat

Zásobník						
	s_s					
	γ					
	γ	γ	γ'			
	$\vdots$	$\vdots$	$\vdots$			
	γ	γ	γ	γ	γ''	
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
	γ	γ	γ	γ	γ	
	s_p	s_p	s_p	s_p	s_p	
Slovo 1	a^i	$B(\gamma s_s)$	$a^l b^l$	$B(\gamma^j)$	a	$B(s_p\gamma^{k-j-1})\ b$
Slovo 2	a^i	$B(\gamma s_s)$		$B(\gamma^j)$	a	$B(s_p\gamma^{k-j-1})\ b$

Tabuľka 4.1: Obsah zásobníka A pri spracovaní dvoch slov

lebo $u\,a^l b^l$ má rovnaký počet „otvorených zátvoriek“[6] ako u.

Pre $j = 2 \dots k-1$ sa automat A po prečítaní

$$v_j = u\ a^l b^l\ B(\gamma^{j-2}\gamma') = a^i B(\gamma s_s)\ a^l b^l\ B(\gamma^{j-2}\gamma') = a^i B(\gamma s_s)\ a^l b^l\ B(\gamma^j - 1)$$

dostane do situácie $(s_p\gamma^{k-j}, q_0)$. Do rovnakej situácie sa dostane aj po prečítaní $w_j = a^i B(\gamma^j s_s)$. $v_j a$ nie je prefix slova z L_2, ale $w_j a$ je. Keďže A nevie tie dve slová rozlíšiť, môžme povedať, že sa A po prečítaní $v_j a$ a $w_j a$ dostane do $(s_p\gamma^{k-1-j}\gamma'', q_0)$, pričom

$$c(\gamma'') = c(\gamma) + 1 \tag{4.5}$$

Podľa predpokladu, A akceptuje $w_j a B(s_p\gamma^{k-1-j}\gamma'') \in L_2$ a teda aj

$$v_j a B(s_p\gamma^{k-1-j}\gamma'') = a^i B(\gamma s_s)\ a^l b^l\ B(\gamma^j) a B(s_p\gamma^{k-1-j}\gamma'')$$

Využijúc vzťah 4.5 a vlastnosti funkcie B (4.4) upravíme zápis slova a dostaneme

$$a^i B(\gamma s_s)\ a^l b^l\ B(\gamma)^j\ ab\ B(\gamma)^{k-j} B(s_p)$$

Na začiatku sme zvolili ľubovoľné i a teda pre každé $j \in \mathbb{N}^+$ vieme nájsť dosť veľké i (a teda aj k) aby platilo $k > 2j$. Keď potom nahradíme $c(\gamma s_s)$ s t a $c(\gamma)$ s m dostaneme[7]

$$a^i b^t\ a^l b^l\ b^{jm}\ ab\ b^{(k-j)m} B(s_p)$$

[6] $\#_a(u\,a^l b^l) - \#_b(u\,a^l b^l) = \#_a(u) - \#_b(u)$

[7] resp. nahradíme $B(\gamma s_s)$ s b^t a $B(\gamma)$ s b^m

Navyše, z definície c vyplýva, že

$$i = c(s_p\gamma^k s_s) = c(s_p) + kc(\gamma) + c(s_s) = c(s_p) + km + t$$

Čiže, $c(s_p) = i - t - km$ a teda uvedené slovo spĺňa požiadavky z tvrdenia. □

Ďalšia lema sa bude zaoberať zložitejším jazykom — L_3. Minimálny automat pre tento jazyk ma 3 stavy a teda jednoduchší automat už môže mať viac ako jeden stav. To znamená, že niektoré kroky z predošlého dôkazu už nebudeme môcť použiť. Najväčší rozdiel bude v tom, že pri používaní vzťahu 4.1 musíme overiť, či stavy nadväzujú na seba.

Lema 4.2. *Nech A je DPDA taký, že $K_A = \{q_0, q_1\}$ a $L_3 \subseteq N(A)$. Potom existuje $m \in \mathbb{N}^+$ také, že pre všetky dostatočne veľké $j, l \in \mathbb{N}^+$ existujú $i, k, t, x \in \mathbb{N}_0 : k > 2j$ také, že A akceptuje*

$$a^i b^t\ a^l b^l\ b^{jm}\ abb^x\ ab\ b^{(k-j)m} b^{i-t-km-x}$$

Dôkaz. Po prečítaní všetkých a^i pre dostatočne veľké i, situácia bude tvaru $(s_p\gamma^k s_s^{(i)}, q^{(i)})$, pričom $\gamma \neq \varepsilon$ a hodnota k, $s_s^{(i)}$ a $q^{(i)}$ závisí od i. Pri čoraz väčšom i sa hodnota k postupne zväčšuje a hodnoty $s_s^{(i)}$ a $q^{(i)}$ sa periodicky opakujú. Preto, podobne ako v predošlom dôkaze, zvolíme nejaké pevné s_s, q a index (i) už nebudeme písať.

Teda, po prečítaní $u_0 = a^i B(\gamma s_s, q)$ situácia bude $(s_p\gamma^{k-1}, p_0)$. Bez ujmy na všeobecnosti môžeme predpokladať, že $S(\gamma, p_0) = p_0$. Ak by automat pri vyberaní γ^k náhodou skákal medzi stavmi tak môžeme zvoliť $\gamma := \gamma^2$. Keďže sme už jedno γ vybrali zo zásobníka, tak nemôžeme byť v chvoste žiadneho cyklu a teda bude platiť $S(\gamma, p_0) = p_0$. Proces vybratia jedného γ zo zásobníka bude, keď ho rozložíme na menšie kroky, vyzerať takto:

$$(p_0, b^{c(\gamma,p_0)}, \gamma) \vdash^+ (p_{0,1}, b^{c(\gamma,p_0)-1}, \gamma_1) \vdash^+ (p_{0,2}, b^{c(\gamma,p_0)-2}, \gamma_2) \vdash^+ \cdots \vdash^+ (p_0, \varepsilon, \varepsilon)$$

Teraz sa pozrime aká bude dvojica po prečítaní $u_0\, a^l b^l$, pre $l \in \mathbb{N}^+$? Počas spracovania $a^l b^l$ zásobník nikdy neklesne veľmi hlboko. Určite existuje konštanta h taká, že automat sa nedostane do konfigurácie so zásobníkom $s_p\gamma^{k-1-h}$. Ukážeme to sporom.

Nech pre každé h' existuje l' také, že (pre dosť veľké i) platí

$$(p_0, a^{l'}b^{l'}, s_p\gamma^{k-1}) \vdash_A (p', u', s_p\gamma^{k-1-h'})$$

Zvoľme teraz i_1 také, že $s_s^{(i_1)} = s_s$. Podľa predpokladu, existuje l_1 také, že sa na nejakom prefixe $a^{l_1}b^{l_1}$ automat dostane do konfigurácie (p_1, u_1, s_p). Zvoľme i_2 také, že $s_s^{(i_2)} = s_s$ a $i_2 > i_1 + l_1$. Rovnako ako predtým, existuje l_2 také, že:

$$(q_0, a^{i_2}\, a^{l_2}b^{l_2}, Z_0) \vdash_A (p_2, u_2, s_p)$$

Analogicky, zvolíme $i_3 > i_2 + l_2$, ku ktorému nájdeme l_3. Slová, ktoré automat prečítal kým sa dostal do konfigurácie (p_x, u_x, s_p) sú prefixy slov z L_3 a majú rôzny počet „otvorených zátvoriek“. To znamená, že hodnoty $c(s_p, p_1)$, $c(s_p, p_2)$, $c(s_p, p_3)$ musia byť rôzne. To ale nie je možné, lebo A má len dva stavy $\rightsquigarrow$ spor.

Keďže teda zásobník neklesne viac ako o γ^h, situáciu po prečítaní $u_0\, a^l b^l$ môžeme zapísať ako $(s_p\gamma^{k-1-h}\gamma', p')$. To znamená, že po prečítaní $u_1 = u_0\, a^l b^l\, B(\gamma\gamma', p')$ situácia bude $(s_p\gamma^{k-2-h}, p_1)$.

Teraz môžu nastať dva prípady:

$p_0 = p_1$: Automat je v rovnakej situácii, do akej by sa dostal po prečítaní $u_0\, B(\gamma^{h+1}, p_0)$. To bude platiť aj keď na konce oboch slov pridáme $B(\gamma^j, p_0)$ pre $j = 0, \ldots, k-3-h$. Zvoľme g' z množiny $\{0, \ldots, c(\gamma, p_0)\}$ tak, aby $c(\gamma\gamma', p') + g'$ bol (celočíselný) násobok $c(\gamma, p_0)$ a pridajme ešte $b^{g'}$. Zatiaľ sú obe slová prefixy slov z L_3 a teda musí platiť, že $c(s_p\gamma^{k-3-h-j}\gamma_{g'}, p_{0,g'})$ sa rovná rozdielu počtov a a b v slovách. Keď na koniec druhého slova pridáme $abb\, aB(s_p\gamma^{k-3-h-j}\gamma_{g'}, p_{0,g'})$ dostaneme slovo z L_3. Keďže A nevie tie dve slová rozlíšiť, môžeme ten istý sufix pridať aj na prvé slovo. To znamená, že A bude akceptovať slovo ktoré patrí do $L_4 \setminus L_3$. To slovo je tvaru:

$$a^i B(\gamma s_s, q)\, a^l b^l B(\gamma^{j+1}\gamma', p') b^{g'}\, abb\, aB(s_p\gamma^{k-3-h-j}\gamma_{g'}, p_{0,g'})$$

$p_0 \neq p_1$: Na koniec slova pridáme $B(\gamma^j, p_1) b^{g''} ab$ pričom $j = 0, \ldots, k-3-h$ a g'' je znovu vhodné číslo ktoré to zaokrúhli na násobok $c(\gamma, p_0)$. Keďže počet symbolov b potrebných na odstránenie γ zo zásobníku nezávisí od toho z ktorého stavu automat začne, tak ani hodnota g'' nebude

závisieť od voľby j. Situácia po prečítaní slova bude $(s_p\gamma^{k-3-h-j}\gamma'', p'')$. Pridajme teraz ešte $B(\gamma'', p'')$. Potom situácia bude $(s_p\gamma^{k-3-h-j}, p_2)$. Znovu potrebujeme analyzovať dve možnosti:

$p_0 = p_2$**:** Automat sa nachádza v takej istej situácii, v akej by bol po prečítaní $u_0 B(\gamma^{h+j+2}, p_0)$. Toto slovo označíme s u''.

$p_1 = p_2$**:** Automat sa nachádza v takej istej situácii, v akej by bol po prečítaní $u_1 B(\gamma^{j+1}, p_1)$. Toto slovo označíme s u''.

V každom prípade je u'' prefix slova z L_3 a teda môžeme ho zreťaziť s $ab\,B(s_p\gamma^{k-3-h-j}, p_2)$. Ale A nevie rozlíšiť u'' od pôvodného slova. To znamená, že ten istý sufix môžeme pridať aj na koniec toho slova a A ho bude akceptovať.

$$a^i B(\gamma s_s, q)\, a^l b^l\, B(\gamma^{j+1}\gamma', p') b^{g''}\, ab\, B(\gamma'', p'')\, ab\, B(s_p\gamma^{k-3-h-j}, p_2)$$

Slovo patrí do L_4, lebo z konštrukcie vyplýva, že má rovnaký počet a a b.

V oboch prípadoch sme dostali slovo, ktoré A musí akceptovať, aj keď nepatrí do L_3. Teraz, podobne ako v dôkaze lemy 4.1, potrebujeme ukázať že to slovo je požadovaného tvaru.

Keďže sme g' a g'' volili tak aby počet b v príslušnej časti slova bol násobok $c(\gamma, p_0)$, tak tú časť môžeme zapísať ako b^{jm}. Symbolom t označíme číslo $c(\gamma s_s, q)$. x bude 1 alebo $c(\gamma'', p'')$, podľa toho ktorý prípad nastal. Keďže slová patria do L_4, počet symbolov b v poslednej časti musí byť taký, aby vyvážil počet a a b. i môže byť ľubovoľne veľké, a teda vždy vieme vybrať také aby ten počet bol kladný. Keď teraz napíšeme to slovo, zistíme že je v požadovanom tvare.

$$a^i b^t\, a^l b^l\, b^{jm}\, ab\, b^x\, ab\, b^{i-t-jm-x} = a^i b^t\, a^l b^l\, b^{jm}\, ab\, b^x\, ab\, b^{(k-j)m} b^{i-t-km-x}$$

□

Nasledovná lema sa zaoberá všeobecným prípadom, keď n môže byť ľubovoľné. V dôkaze tejto lemy sa už nebudeme zaoberať technickými detailami, ktoré sa riešia rovnako ako v predošlej leme, ale sa len odvoláme na dôkaz lemy 4.2. Tu sa sústredíme na pridanie indukcie, ktorá nám umožní vytvoriť slovo z $L_{n+1} \setminus L_n$ požadovaného tvaru.

Lema 4.3. *Nech $n \in \mathbb{N}^+$. Nech A je DPDA taký, že $|K_A| = n-1$ a $L_n \subseteq N(A)$. Potom existuje $m \in \mathbb{N}^+$ také, že pre všetky dostatočne veľké $j, l \in \mathbb{N}^+$ existujú $i, k, t, x_1, \ldots x_{n-1} \in \mathbb{N}_0 : k > 2j$ také, že A akceptuje*

$$a^i b^t\ a^l b^l\ b^{jm}\ abb^{x_1}\ \cdots\ abb^{x_{n-1}}\ b^{(k-j)m} b^{i-t-km-\sum_{s=1}^{n-1} x_s}$$

Dôkaz. Nech $K_A = \{q_0, \ldots, q_{n-2}\}$. Nech situácia po prečítaní (dosť dlhých) a^i je $(s_p \gamma^k s_s, q)$. Potom, po prečítaní $u_1 = a^i B(s_s, q)$ situácia bude $(s_p \gamma^k, p_1)$. Bez ujmy na všeobecnosti, môžeme predpokladať, že $S(\gamma, p_1) = p_1$. Ak by sa automat pri vyberaní γ^k cyklil, tak môžeme položiť $\gamma := \gamma^{(n-1)!}$. Potom už určite bude platiť $S(\gamma, p_1) = p_1$, lebo cyklus môže byť dĺžky najviac $n-1$ a teda $(n-1)!$ je určite násobok dĺžky cyklu. Chvost cyklu, ak existuje, môžeme schovať do s_s.

Teraz k u_1 pridáme $a^l b^l$. Podobne ako v dôkaze lemy 4.2, môžeme dokázať, že automatu neklesne zásobník „príliš hlboko“. Dôkaz postupuje analogicky, len potrebujeme zvoliť n hodnôt $i_1, \ldots i_n$ namiesto troch.

Označme teda situáciu po prečítaní $u_1\, a^l b^l$ ako $(s_p \gamma^{k-h} \gamma', p')$. Potom, keď k slovu pridáme $B(\gamma', p')$ situácia bude $(s_p \gamma^{k-h}, p_2)$. Keď ešte pridáme $B(\gamma^j, p_2)$ pre $j = 1, \ldots k-n-h$ tak situácia bude $(s_p \gamma^{k-h-j}, p_2)$.

Teraz chceme nájsť dvojicu slov takých, že po ich prečítaní bude automat bude v rovnakom stave a zásobník bude $s_p \gamma^x$. Ak platí $p_1 = p_2$, tak sme ich našli. Ak $p_1 \neq p_2$ tak musíme pokračovať ďalej.

Pridajme k slovu b^{g_1} tak, aby sme počet symbolov b v tejto skupine, zaokrúhlili na násobok $c(\gamma, p_1)$. Potom pridajme ešte abb. Nech je výsledná situácia $(s_p \gamma k - h - j - 1, \gamma'', p'')$. Pridajme $c(\gamma'', p'')$ symbolov b a označme výsledný stav p_3. Ak $p_3 = p_1$ alebo $p_3 = p_2$ tak sme našli dvojicu slov.

Ak nie, tak postupujeme induktívne: pridáme abb a vhodný počet symbolov b tak, aby sme zásobník „zaokrúhlili“ na $s_p \gamma^{k-h-j-x}$ (pre $x = 3, 4, \ldots$). Postupne pre každé x takto dostaneme stav p_x. Toto stačí robiť po $x = n$ lebo A má len $n-1$ stavov a teda po n krokoch už určite nájdeme dva rovnaké stavy. Takto zostrojené slová uvádzame aj v tabuľke 4.2.

Nech teraz x a y $(x < y)$ sú najmenšie čísla také, že $p_x = p_y$. Ak $y = 1$ tak položíme $\xi_y = u_y b^{g_1}$. Označme situáciu po prečítaní ξ_y ako $(s_p, \gamma^{k-h-j-1} \gamma_\xi, p_\xi)$. Ak $y \neq 1$ tak položíme $\xi_y = u_y$, $\gamma_\xi = \gamma$ a $p_\xi = p_y$.

Nech ξ_x označuje slovo, ktoré vznikne zreťazením u_x a určitého počtu symbolov b. Počet symbolov zvolíme tak, aby automat mal rovnaký zásobník

Slovo	Stav po prečítaní slova	Zásobník po prečítaní slova
$u_1 = a^i b^{c(s_s,q)}$	p_1	$s_p \gamma^k$
$u_2 = u_1\, a^l b^l\, B(\gamma^j \gamma', p')$	p_2	$s_p \gamma^{k-h-j}$
$u_3 = u_2\, b^{g_1}\, abb\, b^{g_2}$	p_3	$s_p \gamma^{k-h-j-1}$
$u_4 = u_3\, abb\, b^{g_3}$	p_4	$s_p \gamma^{k-h-j-2}$
$\vdots$		
$u_n = u_{n-1}\, abb\, b^{g_{n-1}}$	p_n	$s_p \gamma^{k-h-j-n+2}$

Tabuľka 4.2: Zoznam dôležitých slov použitých v dôkaze

a stav po prečítaní ξ_x a ξ_y. To sa dá, lebo treba len pridať vhodný počet b aby sme zmazali niekoľko γ a potom ešte b^{g_1} ak $y = 1$. Keďže A bol na začiatku v rovnakom stave ($p_x = p_y$) tak aj po prečítaní ξ_x, ξ_y bude v rovnakom stave (a s rovnakým obsahom zásobníka).

ξ_x má úroveň zahniezdenia x. To znamená, že slovo

$$\xi_x (abb)^{n-y+1} b^{c(s_p \gamma^{k-h-j-y+1} \gamma_\xi, p_\xi) - (n-y+1)}$$

má úroveň zahniezdenia $x+n-y+1 \leq n$, patrí do L_n a teda ho A musí akceptovať. Keď to isté slovo pridáme na koniec ξ_y, dostaneme slovo z $L_{n+1} \setminus L_n$, ale A ho tiež bude akceptovať. Teraz už jednoduchými úpravami, podobnými tým v dôkaze lemy 4.2, môžeme dokázať, že to slovo je v požadovanom tvare. □

Teraz dokážeme druhú časť nášho tvrdenia. Pre každé n máme množinu slov z $L_{n+1} \setminus L_n$, ktoré musí akceptovať „jednoduchý“ automat akceptujúci všetky slová z L_n. My chceme ukázať, že neexistuje konečný automat, ktorý by zamietal všetky tieto zlé slova a pritom akceptoval všetky dobré slová z L_n. Z toho potom vyplynie, že prienik jazykov akceptovaných touto dvojicou automatov nemôže byť L_n a teda že L_n (a jeho minimálny automat) je nerozložiteľný.

Veta 4.2. *Nech $n \in \mathbb{N}^+$. Neexistujú $n-1$ stavový deterministický zásobníkový automat A a deterministický konečný automat A' (s ľubovoľným počtom stavov) také, že $N(A) \cap L(A') = L_n$.*

Dôkaz. Sporom. Nech existujú DPDA A a DKA A' také, že $N(A) \cap L(A') = L_n$. Potom platí $L_n \subseteq N(A)$ a podľa predchádzajúcich lem existuje $m \in \mathbb{N}^+$ také, že pre všetky (dosť veľké) $j, l \in \mathbb{N}^+$ existujú $i, k, t, x_1, \ldots x_{n-1} \in \mathbb{N}_0$: $k > 2j$ také, že A akceptuje

$$w_{j,l} = a^i b^t \; a^l b^l \; b^{jm} \; abb^{x_1} \; \cdots \; abb^{x_{n-1}} \; b^{(k-j)m} b^{i-t-km-\sum_{s=1}^{n-1} x_s}$$

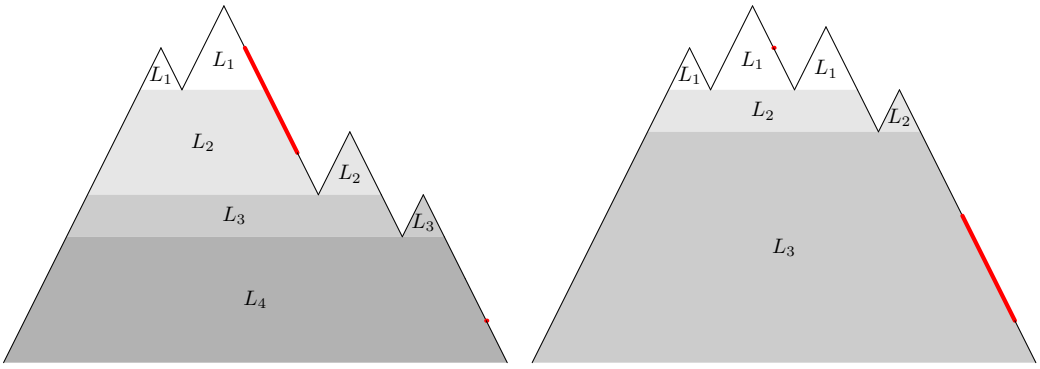
$w_{j,l}$ (pre $n = 3$ je slovo znázornené na obrázku 4.1 vľavo) patrí do $N(A)$ ale nepatrí do L_n a teda A' nesmie akceptovať $w_{j,l}$ pre žiadne j a l. Nech $l = j = |K_{A'}|!$. Pozrime sa, ako A' bude pracovať na $w_{j,l}$. Časť $b^l b^{jm}$ má dĺžku $|K_{A'}|! \cdot (m+1)$. To je viac ako počet stavov A' a teda sa automat bude cykliť. Dĺžka cyklu určite delí číslo $|K_{A'}|! \cdot m$ a teda môžeme toľko znakov b vynechať a A' si to „nevšimne“ — nebude akceptovať ani slovo:

$$a^i b^t \; a^{|K_{A'}|!} b^{|K_{A'}|!} \; abb^{x_1} \; \cdots \; abb^{x_{n-1}} \; b^{k-|K_{A'}|!} b^{i-t-km-\sum_{s=1}^{n-1} x_s}$$

Teraz skúmajme výpočet A' na časti $b^{(k-|K_{A'}|!)m}$. Jej dĺžka je aspoň $|K_{A'}|!$ (lebo $k > 2j$) a teda sa aj tu automat bude cykliť a tu môžeme pridať $|K_{A'}|! \cdot m$ symbolov b. To znamená, že A' nebude akceptovať ani slovo:

$$\begin{aligned} w' = & a^i b^t \; a^{|K_{A'}|!} b^{|K_{A'}|!} \; abb^{x_1} \cdots abb^{x_{n-1}} \; b^{km} b^{i-t-km-\sum_{s=1}^{n-1} x_s} \\ = & \underbrace{a^{i-t-\sum_{s=1}^{n-1} x_s} \; a^{x_{n-1}} \cdots \underbrace{a^{x_1} \overbrace{a^t b^t}^{\in L_1} \overbrace{a^{|K_{A'}|!} b^{|K_{A'}|!}}^{\in L_1} abb^{x_1}}_{\in L_2} \cdots \overbrace{ab}^{\in L_1} b^{x_{n-1}} b^{i-t-\sum_{s=1}^{n-1} x_s}}_{\in L_n} \end{aligned}$$

w' (obrázok 4.1 vpravo) nepatrí do $L(A')$ a preto nemôže ani patriť do prieniku $L(A')$ a $N(A)$. To je spor s $N(A) \cap L(A') = L_n$. □



Obr. 4.1: Slovo $w_{j,t}$ pred a po „pumpovaní“

Kapitola 5

Záver

V práci sme skúmali vplyv regulárnej prídavnej informácie na zložitosť deterministických zásobníkových automatov a jazykov, ktoré ony rozpoznávajú. Ukázali sme prípady, v ktorých tá informácia pomôže výrazne zjednodušiť automat, ako aj automaty, ktoré sa nedajú zjednodušiť *regulárnou* prídavnou informáciou. Preto, že zložitosť zásobníkových automatov nebola veľmi skúmaná, museli sme časť práce venovať aj tejto problematike. Preskúmali sme rôzne spôsoby definovania zložitosti automatov a vylepšili sme konštrukciu normálneho tvaru „automat vždy dočíta vstup“.

Táto práca by sa dala rozšíriť v mnohých smeroch. Dalo by sa skúmať (uzáverové) vlastnosti triedy dobre rozložiteľných a nerozložiteľných jazykov. Taktiež, mohli by sme skúsiť iné definície zložitosti automatov a jazykov: (ne)rozložiteľné jazyky podľa miery „počet stavov“ by sa mohli stať rozložiteľnými keby sme zobrali jemnejšiu mieru. Ďalšia možnosť môže byť skúsiť nejaké obmeny modelu automatu s ktorým pracujeme (napr. akceptovanie stavom, alebo zásobník s pevným dnom), prípadne nahradiť zásobníkové automaty silnejším modelom.

Literatúra

[1] F. Gécseg and I. Peák, *Algebraic Theory of Automata.* Akadémiai Kiadó, Budapest, 1972.

[2] F. Gécseg, *Products of Automata.* Springer-Verlag, Berlin, 1986.

[3] P. Gaži, "Parallel decompositions of finite automata," Master's thesis, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, 2006.

[4] P. Gaži and B. Rovan, "Assisted problem solving and decompositions of finite automata," in *SOFSEM* (V. Geffert, J. Karhumäki, A. Bertoni, B. Preneel, P. Návrat, and M. Bieliková, eds.), vol. 4910 of *Lecture Notes in Computer Science*, pp. 292–303, Springer, 2008.

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