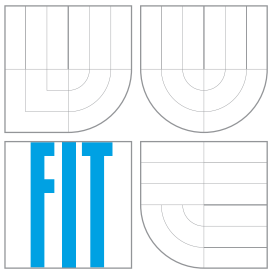
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SKÁKAJÍCÍ KONEČNÉ AUTOMATY A PŘEVODNÍKY

JUMPING FINITE AUTOMATA AND TRANSDUCERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JURAJ HRUBÝ
AUTHOR

VEDOUCÍ PRÁCE — prof. RNDr. ALEXANDER MEDUNA, CSc.
SUPERVISOR

BRNO 2013

Abstrakt

Tato bakalářská práce navazuje na studium skákajících konečných automatů a zavádí skákající konečné převodníky. Skákající konečné automaty jsou modifikované konečné automaty tak, že symboly ze vstupní pásky nejsou čteny spojitě zleva-doprava, ale čtecí hlava se může pohybovat po vstupní pásce pomocí skoků. Skákající konečné převodníky jsou podobně modifikované konečné převodníky. Aby bylo možné skákající konečné automaty a převodníky implementovat, byly zavedeny jejich striktně deterministické verze omezením konečné stavové kontroly a modifikací binární skokové relace. Práce se dále zabývá možným využitím skákajících konečných automatů a převodníků a popisem implementace striktně deterministického skákajícího konečného automatu.

Abstract

The bachelor's thesis builds upon the study of jumping finite automata and introduces jumping finite transducers. Jumping finite automata are finite automata modified in such a way that symbols from the input tape are not read continuously from left to right but that the reading head can make moves on the input tape by jumps. Jumping finite transducers are finite transducers modified in a very similar way. In order to implement jumping finite automata and transducers, strictly deterministic versions of them were introduced by restricting the finite state control and by modification of the binary jumping relation. The thesis furthermore focuses on possible usage of jumping finite automata and transducers and on the description of the implementation of strictly deterministic jumping finite automata.

Klíčová slova

skákající konečné automaty, skákající konečné převodníky, striktní determinismus, využití, implementace

Keywords

jumping finite automata, jumping finite transducers, strict determinism, usage, implementation

Citace

Juraj Hrubý: Skákající konečné automaty a převodníky, bakalářská práce, Brno, FIT VUT v Brně, 2013

Skákající konečné automaty a převodníky

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. RNDr. Alexandra Meduny CSc.. Další informace mi poskytli Ing. Ivana Burgetová Ph.D. a Ing. Petr Zemek. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Juraj Hrubý
14. května 2013

Poděkování

Touto cestou bych chtěl poděkovat vedoucímu prof. RNDr. Alexandrovi Medunovi CSc. za odbornou pomoc a cenné rady pri řešení mé bakalářské práce. Také bych chtěl poděkovat za konzultace Ing. Ivaně Burgetová Ph.D. a Ing. Petrovi Zemkovi.

Obsah

Kapitola 1

Úvod

Formálny jazyk môžeme popísať gramatikou, ktorou je generovaný alebo automatom, ktorý ho dokáže prijať. Táto práca sa zaoberá skákajúcimi konečnými automatmi a prevodníkmi, ktoré sú veľmi podobné konečným automatom a prevodníkom. Rozdiel spočíva v tom, že symboly zo vstupnej pásky nečítajú zľava-doprava, ale nespojito, čo znamená, že čítacia hlava sa po prečítaní nejakého symbolu môže presunúť kamkoľvek na vstupnej páske. Zaujímavou vlastnosťou skákajúcich konečných automatov je to, že dokážu prijať aj niektoré kontextové jazyky, pričom nemajú zásobník.

Táto práca je rozdelená do nasledúcich logických celkov:

- Prvou kapitolou je úvod, ktorý má čitateľovi predstaviť základnú myšlienku skákajúcich konečných automatov a prevodníkov.
- V druhej kapitole rekapitulujeme základné definície a pojmy z oblasti matematiky a teórie formálnych jazykov, ktoré sú potrebné pre pochopenie nasledujúcich kapitôl.
- V tretej kapitole je čitateľ oboznámený so skákajúcimi konečnými automatmi zavedenými v článku [6]. Podobne modifikujeme konečné prevodníky a zavádzame skákajúce konečné prevodníky.
- V štvrtej kapitole sa zaoberáme determinizmom skákajúcich konečných automatov a prevodníkov. Na príkladoch je ukázané, že klasický determinizmus založený na obmedzení konečnej stavovej kontroly je pre skákajúce konečné automaty a prevodníky nedostatočný. Preto zavádzame ich striktne deterministické verzie.
- V piatej kapitole sa zaoberáme možnosťami využitia skákajúcich konečných automatov a prevodníkov. Čitateľ je oboznámený s rôznymi jazykmi, ktoré je možné prijať a aj s ukážkou prekladu. V tejto kapitole sú spomenuté aj jednoduché modifikácie skákajúcich konečných automatov, vďaka ktorým môžeme prijímať nové typy jazykov a taktiež kombinácia konečného prevodníka so skákajúcim konečným automatom, ktorá bola aj implementovaná.

Kapitola 2

Základné definície a pojmy

Táto kapitola opakuje potrebnú terminológiu, značenie a matematické operácie z oblasti formálnych modelov a jazykov, ktoré su nimi popísané. Pojmy, definície a tvrdenia v tejto kapitole boli prevzaté z viacerých publikácii (pozri [5] ,[7], [1], [4], [3]).

2.1 Množiny a relácie

Definícia 2.1.1. Množina
Množina je súhrn nejakých rozlíšiteľných objektov, ktoré nazývame *prvky* množiny. O každom prvku môžeme povedať, či do množiny patrí alebo nie.

Množina, ktorá neobsahuje žiadne prvky, sa nazýva *prázdna množina* a značíme ju $\emptyset$. Zápis $M = \{a, b\}$ znamená, že množina M obsahuje prvky a a b. Skutočnosť, že a je prvkom množiny M, značíme $a \in M$ a naopak, $c \notin M$ vyjadruje, že prvok c do množiny M nepatrí. Každý prvok sa môže v množine vyskytovať iba raz.

Definícia 2.1.2. Mohutnosť množiny
Mohutnosť množiny je definovaná nasledovne:

- $|M| = 0$ ak $M = \emptyset$,
- $|M| = n$ ak $M = \{a_1, a_2, a_3, \ldots, a_n\}$, kde $n \in \mathbb{N}$,
- $|M| = \infty$ ak M je nekonečná množina.

Množina M je konečná, ak $|M| \in \mathbb{N}_0$, inak sa jedná o nekonečnú množinu.
Niektoré dôležité množiny z matematiky:

- $\mathbb{N} = \{1, 2, 3, \ldots\}$ je množina všetkých prirodzených čísel.
- $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$ je množina všetkých prirodzených čísel s nulou.
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ je množina všetkých celých čísel.
- $\mathbb{Q}$ – racionálne čísla.
- $\mathbb{I}$ – iracionálne čísla.
- $\mathbb{R}$ – reálne čísla.
- $\mathbb{C}$ – komplexné čísla.

Prvky množiny môžu byť určené vymenovaním, ale spoľahlivejší a prehľadnejší je zápis pomocou podmienok, ktoré musia prvky množiny spĺňať. Napr. $B = \{x|\ 4 \leq x \leq 21$ a zároveň $x \in \mathbb{N}\}$ je množina všetkých prirodzených čísel od 4 do 21.

Pre dve množiny A, B vravíme, ze A je *podmnožinou* množiny B, $A \subseteq B$, ak každý prvok množiny A je súčasne prvkom množiny B. Množiny A, B považujeme za *sebe rovné*, $A = B$, ak obashujú práve tie isté prvky, t.j. každý prvok množiny A je prvkom množiny B a každý prvok množiny B je prvkom množiny A.

Je zrejmé, že platí $A \subseteq A$ a taktiež $\emptyset \subseteq A$ pre ľubovolnú množinu A. Ak A je podmnožinou B, ale $A \neq B$, píšeme stručne $A \subset B$ a množinu A nazývame *vlastnou podmnožinou* množiny B.

Definícia 2.1.3. Potenčná množina
Nech M je množina. *Potenčná množina* množiny M, značená 2^M, je množina všetkých podmnožín množiny M. Formálne zapísané $2^M = \{X|\ X \subseteq M\}$.

Definícia 2.1.4. Zjednotenie, prienik a rozdiel dvoch množín
Nech A a B sú množiny.

- *Zjednotenie* množín A a B značíme $A \cup B$, pričom $A \cup B = \{a \mid a \in A$ alebo $a \in B\}$.
- *Prienik* množín A a B značíme $A \cap B$, pričom $A \cap B = \{a \mid a \in A$ a zároveň $a \in B\}$.
- *Rozdiel* množín A a B značíme $A - B$, pričom $A - B = \{a \mid a \in A$ a zároveň $a \notin B\}$.

Definícia 2.1.5. Usporiadaná dvojica
Nech a a b sú dva objekty. Potom (a, b) značí *usporiadanú dvojicu* pozostávajúcu z a a b v tomto poradí. Vravíme, že $(a, b) = (c, d)$ iba keď $a = c$ a $b = d$. Na rozdiel od množín, kde $\{a, b\} = \{b, a\}$.

Definícia 2.1.6. Karteziánsky súčin
Nech A a B sú dve množiny. *Karteziánsky súčin* A a B, značený $A \times B$, je definovaný ako $A \times B = \{(a, b)|\ a \in A$ a $b \in B\}$.

Príklad 2.1.7. Karteziánsky súčin
Nech $A = \{1, 2\}$ a $B = \{7, 8, 9\}$. Potom karteziánsky súčin je:

$$A \times B = \{(1, 7), (1, 8), (1, 9), (2, 7), (2, 8), (2, 9)\}.$$

Definícia 2.1.8. Binárna relácia
Nech A a B sú dve množiny. *Binárna relácia*, zjednodušene *relácia*, z A do B je ľubovoľná podmnožina $A \times B$. Ak $A = B$, vravíme, že relácia je na A. Ak R je relácia z A do B, píšeme aRb, keď $(a, b) \in R$.

Príklad 2.1.9. Relácia
Relácia porovnania dvoch celých čísel:

$$<= \{(a, b)|\ a \text{ je menšie ako } b, \text{ pričom } a, b \in \mathbb{Z}\}$$

Definícia 2.1.10. Definičný obor a obor hodnôt relácie
Nech $R \subseteq A \times B$ je binárna relácia z A do B. Potom:

- $\mathrm{Dom}\, R = \{a|$ existuje $b \in B$, že $(a, b) \in R\}$.
- $\mathrm{Im}\, R = \{b|$ existuje $a \in A$, že $(a, b) \in R\}$.

Množinu $\mathrm{Dom}\, R$ nazývame *definičný obor* alebo *domain* relácie R. Množinu $\mathrm{Im}\, R$ nazývame *obor hodnôt, ko-obor* alebo *image* relácie R.

Definícia 2.1.11. Zobrazenie (funkcia)
Nech A a B sú dve množiny a M je binárna relácia z A do B. M je *zobrazením* z A do B, iba ak pre každé $a \in A$ existuje maximálne jedno $b \in B$, pričom $(a, b) \in R$. Ak $(a, b) \in M$ a aj $(a, c) \in M$, potom $b = c$.

2.2 Jazyky a reťazce

Definícia 2.2.1. Abeceda
Abeceda je konečná neprázdna množina prvkov, ktoré sa nazývajú symboly.

Definícia 2.2.2. Reťazec
Nech Σ je abeceda. Potom platí:

- ε je *reťazec* nad abecedou Σ.
- Ak x je *reťazcom* nad abecedou Σ a $a \in \Sigma$, potom $y = xa$ je tiež reťazcom nad abecedou Σ.

ε je *prázdny reťazec*, ktorý neobsahuje žiadne symboly. Množina všetkých reťazcov nad abecedou Σ sa značí Σ^*.

Príklad 2.2.3. Reťazec
Nech $\Sigma = a, b$ je abeceda. Potom $aab, aa, a, \varepsilon, ba, \ldots$ sú reťazce nad abecedou Σ.

Definícia 2.2.4. Dĺžka reťazca
Nech x je reťazec nad abecedou Σ. Potom *dĺžka reťazca* x, značená $|x|$, je definovaná nasledovne:

- Ak $x = \varepsilon$, potom $|x| = 0$.
- Ak $x = a_1, a_2, a_3 \ldots a_n$, pričom $a_i \in \Sigma$ pre všetky $i = 1, 2, 3, \ldots, n$, potom $|x| = n$.

Nech w je reťazec nad abecedou Σ a symbol $a \in \Sigma$. Potom počet výskytov symbolu a v reťazci w sa značí $|w|_a$. Množina symbolov, ktoré obsahuje reťazec w, je značená $alph(w)$.

Definícia 2.2.5. Konkatenácia dvoch reťazcov
Nech x a y sú dva reťazce nad abecedou Σ. *Konkatenáciou* reťazcov x a y vznikne reťazec xy.

Príklad 2.2.6. Konkatenácia
Nech $\Sigma = \{a, b\}$ je abeceda. Potom konkatenáciou reťazcov ab a aa nad abecedou Σ je reťazec $abaa$.

Definícia 2.2.7. Prefix reťazca
Nech x a y sú dva reťazce nad abecedou Σ. Reťazec x je *prefixom* y, ak existuje nad abecedou Σ reťazec z, pričom platí $xz = y$.

Definícia 2.2.8. Sufix reťazca
Nech x a y sú dva reťazce nad abecedou Σ. Reťazec x je *sufixom* y, ak existuje nad abecedou Σ reťazec z, pričom platí $zx = y$.

Definícia 2.2.9. Podreťazec reťazca
Nech x a y sú dva reťazce nad abecedou Σ. Reťazec x je *podreťazcom* y, ak existujú nad abecedou Σ dva reťazce z a z', pričom platí $zxz' = y$.

Definícia 2.2.10. Jazyk
Nech Σ je abeceda a Σ^* množina všetkých reťazcov nad Σ. Každá podmnožina $L \subseteq \Sigma^*$ je *jazykom* nad abecedou Σ.

Definícia 2.2.11. Preklad
Nech Σ je vstupná abeceda a Δ výstupná abeceda. Definujeme *preklad* z jazyka $L_1 \subseteq \Sigma^*$ do jazyka $L_2 \subseteq \Delta^*$ ako ľubovoľnú reláciu z Σ^* do Δ^* takú, že definičným oborom T je L_1 a oborom hodnôt T je L_2.

Definícia 2.2.12. Homomorfný preklad
Nech τ je preklad z jazyka L_1 nad abecedou Σ do jazyka L_2 nad abecedou Δ. Preklad τ je *homomorfným*, ak pre každý $x \in L_1$ existuje najviac jeden $y \in L_2$ taký, že $(x, y) \in \tau$.

2.3 Gramatiky, Automaty a Prevodníky

V tejto časti stručne zopakujeme formálne gramatiky, automaty a prevodníky. Začneme gramatikami, ktoré sú pravdepodobne najdôležitejšou triedou generátorov jazykov. Gramatika je matematický systém na definovanie jazyka, ako aj nástroj, ktorý určuje štruktúru slov jazyka.

Gramatika pre jazyk L používa dve disjunktné množiny symbolov: neterminálne symboly N a terminálne symboly Σ. Slová, ktoré patria do jazyka L, obsahujú iba terminálne symboly. Neterminály sa používajú pri generovaní slov. Základom gramatiky je množina pravidiel P, ktoré popisujú ako sú slová generované.

Definícia 2.3.0.1. Frázová gramatika
Frázová gramatika G je štvorica (N, Σ, P, S), kde:

- N je konečná množina neterminálnych symbolov.
- Σ je konečná množina terminálnych symbolov, pričom $N \cap \Sigma = \emptyset$.
- P je podmnožina $(N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*$.
  Prvok $(\alpha, \beta) \in P$ sa zapisuje $\alpha \to \beta$.
- S je počiatočný symbol, pričom $S \in N$.

Jazyk definovaný gramatikou G značíme $L(G)$.

Gramatiky sú generátormi jazyka a automaty akceptormi. Frázovou gramatikou je možné popísať aj veľmi zložité jazyky, čo môže byť pre praktické využitie niekedy zbytočné. Preto existujú aj obmedzené gramatiky, ktoré generujú triedu jazykov, ktorá je podmnožinou triedy generovanej frázovou gramatikou. Obmedzenie gramatík spočíva iba v úprave tvaru prepisových pravidiel.

2.3.1 Obmedzené gramatiky

Definícia 2.3.1.1. Kontextová gramatika
Kontextová gramatika G je štvorica (N, Σ, P, S), kde:

- N je konečná množina neterminálnych symbolov.
- Σ je konečná množina terminálnych symbolov, pričom $N \cap \Sigma = \emptyset$.
- P je podmnožina $(N \cup \Sigma)^* N (N \cup \Sigma) \times (N \cup \Sigma)^*$.
  Prvok $(\alpha, \beta) \in P$ sa zapisuje $\alpha \to \beta$ a platí, že $|\alpha| \leq |\beta|$.
- S je počiatočný symbol, pričom $S \in N$.

Definícia 2.3.1.2. Bezkontextová gramatika
Bezkontextová gramatika G je štvorica (N, Σ, P, S), kde:

- N je konečná množina neterminálnych symbolov.
- Σ je konečná množina terminálnych symbolov, pričom $N \cap \Sigma = \emptyset$.
- P je podmnožina $N \times (N \cup \Sigma)^*$.
  Prvok $(\alpha, \beta) \in P$ sa zapisuje $\alpha \to \beta$.
- S je počiatočný symbol, pričom $S \in N$.

Definícia 2.3.1.3. Regulárna gramatika
Regulárna gramatika G je štvorica (N, Σ, P, S), kde:

- N je konečná množina neterminálnych symbolov.
- Σ je konečná množina terminálnych symbolov, pričom $N \cap \Sigma = \emptyset$.
- P je podmnožina $N \times \Sigma^* (N \cup \{\varepsilon\})^*$.
  Prvok $(\alpha, \beta) \in P$ sa zapisuje $\alpha \to \beta$.
- S je počiatočný symbol, pričom $S \in N$.

Triedu jazykov, ktoré generujú frázové gramatiky, **RE**, nazývame triedou reukrzívne vyčísliteľných jazykov; triedu generovaných kontextovými gramatikami, **CS**, nazývame triedou kontextových jazykov; triedu generovanú bezkontextovými gramatikami, **CF**, nazývame triedou bezkontextových jazykov a triedu generovanú regulárnymi gramatikami, **REG**, nazývame triedou regulárnych jazykov.

Teorém 2.3.1.4. Chomského hierarchia jazykov
Pre tieto triedy jazykov platí:
REG $\subset$ **CF** $\subset$ **CF** $\subset$ **RE**

2.3.2 Automaty

Podobne ako pri gramatikách, tak aj pri automatoch existuje viacero typov. Automaty sa môžu líšiť tvarom pravidiel, schopnosťou vkladať symboly spať na vstupnú pásku, smerom čítania vstupnej pásky, atď.

Trieda jazykov, ktoré su generované frázovými gramatikami, je totožná s triedou, ktoré dokáže akceptovať *Turingov stroj*. Turingove stroje boli predstavené Alanom Turingom. Páska v Turingovom stroji je rozdelená na bunky, kde v každej bunke sa vyskytuje vždy len jeden symbol.

Definícia 2.3.2.1. Turingov stroj
Turingov stroj (ďalej len TS) M je sedmica $(Q, \Sigma, \Gamma, R, s, B, F)$, kde:

- Q je konečná množina stavov,
- Σ je vstupná abeceda,
- Γ je pásková abeceda, pričom $\Gamma \cap Q = \emptyset$ a $\Sigma \subset \Gamma$,
- $s \in Q$ je počiatočný stav,
- $B \in \Gamma - \Sigma$ je prázdny symbol,
- $F \subseteq Q$ je množina koncových stavov,
- R je konečná množina pravidiel tvaru $qa \to r\gamma t$, kde $q, r \in Q$, $\gamma \in \Gamma^*$ a $t \in \{\mathbf{R},\mathbf{L}\}$.
  - **L** značí posun na páske doľava,
  - **R** značí posun na páske doprava.

Definícia 2.3.2.2. Konfigurácia TS
Nech $M = (Q, \Sigma, \Gamma, R, s, B, F)$ je TS. *Konfigurácia TS* M je trojica $(\alpha, q, X\beta)$, kde:

- $X \in \Gamma$ je symbol v bunke, ktorá je pod hlavou TS,
- $q \in Q$ je aktuálny stav, v ktorom sa TS nachádza,
- $\alpha \in \Gamma^*$ je aktuálny obsah pásky, ktorý je naľavo od hlavy TS,
- $\beta \in \Gamma^*$ je aktuálny obsah pásky, ktorý je napravo od hlavy TS.

Definícia 2.3.2.3. Binárna prechodová relácia TS
Nech $M = (Q, \Sigma, \Gamma, R, s, B, F)$ je TS. *Binárna prechodová relácia TS* nad konfiguráciami $\Gamma^* \times Q \times \Gamma^*$, symbolicky značená $\vdash$, je definovaná nasledovne:

$$\begin{aligned}
(\alpha, q, X\beta) &\vdash (\alpha Y, p, \beta) \text{ ak } qX \to pY\mathbf{R} \in R \\
(\alpha, q, \omega) &\vdash (\alpha Y, p, \omega) \text{ ak } qB \to pY\mathbf{R} \in R \\
(\alpha Z, q, X\beta) &\vdash (\alpha, p, ZY) \text{ ak } qX \to pY\mathbf{L} \in R \\
(\alpha Z, q, \omega) &\vdash (\alpha, p, ZY) \text{ ak } qB \to pY\mathbf{L} \in R
\end{aligned}$$

kde $\alpha, \beta \in \Gamma^*$, $X, Y, Z \in \Gamma$, $q, p \in Q$ a ω značí reťazec prázdnych symbolov B.

Môžeme rozšíriť $\vdash$ *na* $\vdash^m$, kde $m \geq 0$; $\vdash^+$ je tranzitívny uzáver a $\vdash^*$ je tranzitívno-reflexívny uzáver.

Vstupný reťazec $x \in \Sigma^*$ je prijatý TS M, len ak $(\varepsilon, s, x) \vdash^* (\alpha, f, \beta)$, kde s je počiatočný stav, $\alpha, \beta \in \Gamma^*$ a $f \in F$.

Definícia 2.3.2.4. Jazyk prijímaný TS
Nech $M = (Q, \Sigma, \Gamma, R, s, B, F)$ je TS. Jazyk $L(M)$, prijímaný TS M, je množina reťazcov $\{x|\ (\varepsilon, s, x) \vdash^* (\alpha, f, \beta)$ pre nejaký $f \in F$ a $\alpha, \beta \in \Gamma^*\}$.

Teorém 2.3.2.5. Jazyky prijímané TS
Trieda jazykov, ktoré dokáže Turingov stroj prijať, je totožná s triedou jazykov generovaných frázovými gramatikami.

Turingov stroj dokáže prijímať rekurzívne vyčísliteľné jazyky, ale výpočet Turingovho stroja môže vyžadovať neobmedzené množstvo pamäte a nemusí vždy skončiť. Ak obmedzíme Turingov stroj tak, že páska bude vľavo aj vpravo ohraničená, získame *lineárne ohraničený automat*.

Definícia 2.3.2.6. Lineárne ohraničený automat
Lineárne ohraničný automat (ďalej len LOA) je Turingov stroj, ktorý nemôže rozširovať svoju pásku žiadnym smerom.

Teorém 2.3.2.7. Jazyky prijímané LOA
Trieda jazykov, ktoré dokáže lineárne ohraničený automat prijať, je totožná s triedou jazykov generovaných kontextovými gramatikami.

Ďalšími typom automatov sú *zásobníkový* a *konečný automat*. Ich hlavným rozdielom je to, že zásobníkový automat sa rozhoduje pre zmenu stavu nielen na základe stavu, v ktorom sa nachádza, ale aj podľa obsahu zásobníka, na ktorý môže vkladať, ale aj odoberať terminálne a neterminálne symboly.

Definícia 2.3.2.8. Zásobníkový automat
Zásobníkový automat (ďalej len ZA) M je sedmica $(Q, \Sigma, \Gamma, R, s, z, F)$, kde:

- Q je konečná množina stavov,
- Σ je vstupná abeceda,
- Γ je zásobníková abeceda,
- R je konečná množina prechodových pravidiel tvaru $ypa \rightarrow \gamma q$, kde $p, q \in Q$ a $a \in \Sigma \cup \{\varepsilon\}$, $y \in \Gamma$ a $\gamma \in \Gamma^*$,
- $s \in Q$ je počiatočný stav,
- $z \in \Gamma$ je počiatočný symbol na zásobníku,
- $F \subseteq Q$ je množina koncových stavov.

Definícia 2.3.2.9. Konfigurácia ZA
Nech $M = (Q, \Sigma, \Gamma, R, s, z, F)$ je ZA. *Konfigurácia ZA* M je trojica (γ, q, y), kde:

- $y \in \Sigma^*$ je reťazec na vstupnej páske s najľavejším symbolom pod čítacou hlavou,
- $q \in Q$ je stav, v ktorom sa automat nachádza,
- $\gamma \in \Gamma$ je reťazec, ktorý reprezentuje obsah zásobníka s najľavejším symbolom predstavujúcim vrchol zásobníka.

Definícia 2.3.2.10. Binárna prechodová relácia ZA
Nech $M = (Q, \Sigma, \Gamma, R, s, z, F)$ je ZA. *Binárna prechodová relácia ZA* nad konfiguráciami $\Gamma^* \times Q \times \Sigma^*$, symbolicky značená $\vdash$, je definovaná nasledovne:

$$(Z\alpha, q, ax) \vdash (\gamma\alpha, r, x)$$

pre nejaké $q, r \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $x \in \Sigma^*$, $Z \in \Gamma$, $\gamma, \alpha \in \Gamma^*$, pre ktoré $Zqa \rightarrow \gamma r \in R$.

Môžeme rozšíriť $\vdash$ *na* $\vdash^m$, kde $m \geq 0$; $\vdash^+$ je tranzitívny uzáver a $\vdash^*$ je tranzitívno-reflexívny uzáver.

Vstupný reťazec $x \in \Sigma^*$ je prijatý ZA M, len ak $(z, s, x) \vdash^* (\gamma, f, \varepsilon)$, kde s je počiatočný stav, z je počiatočný symbol na zásobníku, $\gamma \in \Gamma^*$ a $f \in F$.

Definícia 2.3.2.11. Jazyk prijímaný ZA
Nech $M = (Q, \Sigma, \Gamma, R, s, z, F)$ je ZA. Jazyk $L(M)$, prijímaný ZA M, je množina reťazcov $\{x |\ (z, s, x) \vdash^* (\gamma, f, \varepsilon)$ pre nejaký $f \in F$ a $\gamma \in \Gamma^*\}$.

Teorém 2.3.2.12. Jazyky prijímané ZA
Trieda jazykov, ktoré dokáže nedeterministický zásobníkový automat prijať, je totožná s triedou jazykov generovaných bezkotextovými gramatikami.

Poznámka:
Trieda jazykov, ktoré prijíma deterministický zásobníkový automat je podmnožinou triedy jazykov, ktoré dokáže prijať nedeterministický.

Definícia 2.3.2.13. Konečný automat
Konečný automat (ďalej len KA) M je pätica (Q, Σ, R, s, F), kde:

- Q je konečná množina stavov,
- Σ je vstupná abeceda,
- R je konečná množina prechodových pravidiel tvaru $pa \to q$, kde $p, q \in Q$ a $a \in \Sigma \cup \{\varepsilon\}$,
- $s \in Q$ je počiatočný stav,
- $F \subseteq Q$ je množina koncových stavov.

Definícia 2.3.2.14. Konfigurácia KA
Nech $M = (Q, \Sigma, R, s, F)$ je KA. *Konfigurácia KA* M je dvojica (q, y), kde:

- $y \in \Sigma^*$ je reťazec na vstupnej páske s najľavejším symbolom pod čítacou hlavou,
- $q \in Q$ je stav, v ktorom sa automat nachádza.

Definícia 2.3.2.15. *Binárna prechodová relácia KA*
Nech $M = (Q, \Sigma, R, s, F)$ je KA. *Binárna prechodová relácia KA* nad konfiguráciami $Q \times \Sigma^*$, symbolicky značená $\vdash$, je definovaná nasledovne:

$$(q, ax) \vdash (r, x)$$

pre nejaké $q, r \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $x \in \Sigma^*$, pre ktoré $qa \to r \in R$.

Môžeme rozšíriť $\vdash$ *na* $\vdash^m$, kde $m \geq 0$; $\vdash^+$ je tranzitívny uzáver a $\vdash^*$ je tranzitívno-reflexívny uzáver.

Vstupný reťazec $x \in \Sigma^*$ je prijatý KA M, len ak $(s, x) \vdash^* (f, \varepsilon)$, kde s je počiatočný stav a $f \in F$.

Definícia 2.3.2.16. Jazyk prijímaný KA
Nech $M = (Q, \Sigma, R, s, F)$ je KA. Jazyk $L(M)$, prijímaný KA M, je množina reťazcov $\{x |\ (s, x) \vdash^* (f, \varepsilon)$ pre nejaký $f \in F\}$.

Teorém 2.3.2.17. Jazyky prijímané KA
Trieda jazykov, ktoré dokáže konečný automat prijať, je totožná s triedou jazykov generovaných regulárnymi gramatikami.

2.3.3 Prevodníky

Prekladom je dvojica reťazcov, v ktorej jeden je vstupom a druhý výstupom. Napr. prekladač definuje preklad, v ktorom jeden reťazec je zdrojový kód vo vyššom programovacom jazyku a druhý reťazec je strojový kód.

Prevodníky sa podobajú automatom, avšak okrem iného majú aj výstupnú abecedu a pásku, na ktorú môžu v každom kroku počas spracovávania reťazca zapisovať symboly.

Definícia 2.3.3.1. Konečný prevodník
Konečný prevodník (ďalej len KP) M je šestica $(Q, \Sigma, \Delta, R, s, F)$, kde:

- Q je konečná množina stavov,
- Σ je vstupná abeceda,
- Δ je výstupná abeceda,
- R je konečná množina prechodových pravidiel tvaru $pa \to qz$, kde $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$ a $z \in \Delta^*$,
- $s \in Q$ je počiatočný stav,
- $F \subseteq Q$ je množina koncových stavov.

Definícia 2.3.3.2. Konfigurácia KP
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je KP. *Konfigurácia KP* M je trojica (q, y, z), kde:

- $q \in Q$ je stav, v ktorom sa prevodník nachádza,
- $y \in \Sigma^*$ je reťazec na vstupnej páske s najľavejším symbolom pod čítacou hlavou,
- $z \in \Delta^*$ je reťazec na výstupnej páske.

Definícia 2.3.3.3. Binárna prechodová relácia KP
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je KP. *Binárna prechodová relácia KP* nad konfiguráciami $Q \times \Sigma^* \times \Delta^*$, symbolicky značená $\vdash$, je definovaná nasledovne:

$$(q, ax, z) \vdash (r, x, zb)$$

pre nejaké $q, r \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $x \in \Sigma^*$, $z, b \in \Delta^*$ pre ktoré $qa \to rb \in R$.

Môžeme rozšíriť $\vdash$ na $\vdash^m$, kde $m \geq 0$; $\vdash^+$ je tranzitívny uzáver a $\vdash^*$ je tranzitívno-reflexívny uzáver.

Vravíme, že y je prekladom vstupného reťazca x, ak $(s, x, \varepsilon) \vdash^* (f, \varepsilon, y)$ pre nejaké $f \in F$.

Definícia 2.3.3.4. Preklad definovaný konečným prevodníkom
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je KP. Preklad definovaný KP M, $\tau(M)$, je množina usporiadaných dvojíc reťazcov $\{(x, y) |\ (s, x, \varepsilon) \vdash^* (f, \varepsilon, y)$ pre nejaké $f \in F\}$. Preklad definovaný konečným prevodníkom sa môže nazývať aj regulárny preklad.

Kapitola 3

Skákajúce konečné automaty a prevodníky

Diskontuálne spracovanie informácii v modernej výpočtovej technike je popisované formálnymi modelmi, ako sú napr. konečné automaty, zásobníkové automaty (pozri 2.3.2 na str. 7), ktoré spracovávajú vstupnú pásku zaujžívane zľava-doprava. Tento mierny rozpor viedol k myšlienke profesora Medunu zaviesť formálny model, ktorý by nepracoval zaužívane zľava-doprava, ale naopak, ktorý by mohol skákať po vstupnej páske. So svojím študentom, inžinierom Zemkom, definovali skákajúce konečné automaty a popísali niektoré vlastnosti, čím otvorili novú oblasť teoretickej informatiky. Na záver aj poukázali na viacero otvorených problémov, ktoré je potrebné riešiť.

V nasledujúcich odstavcoch sa budeme venovať modifikácii konečných automatov, následne aj definícii skákajúcich konečných automatov a popisom ich vlastností zavedených v článku [6].

3.1 Modifikácia konečného automatu

Najprv predpokladajme klasický konečný automat $M = (Q, \Sigma, R, s, F)$ (pozri def. 2.3.2.13 na str. 10) pozostávajúci zo vstupnej pásky, čítacej hlavy a konečnej stavovej kontroly. Vstupná páska je rozdelená na bunky, kde každá bunka môže obsahovať vždy len jeden symbol vstupnej abecedy Σ. Pod čítacou hlavou sa nachádza jeden symbol a. Stavová kontrola pozostáva z množiny stavov Q a množiny prechodových pravidiel R. M robí výpočet pomocou prechodov medzi stavmi. Každý prechod je vykonaný na základe príslušného pravidla, ktoré popisuje do akého stavu automat prejde a či bude symbol a zo vstupnej pásky prečítaný. Po prečítaní symbolu z pásky je čítacia hlava posunutá o jednu bunku doprava. Reťazec w je automatom M prijatý ak prechodmi prejde počas spracovávania celého reťazca z počiatočného do niektorého z koncových stavov. Inak je w odmietnutý.

Skákajúci konečný automat pracuje veľmi podobne až na jeden rozdiel, a to ten, že nečíta symboly zo vstupnej pásky jeden-po-druhom zľava-doprava. To znamená, že po prečítaní symbolu môže M preskočiť istú časť pásky, buď doľava alebo doprava, a pokračovať vo výpočte odtiaľ. Symbol je po prečítaní na danej pozícii zo vstupnej pásky odstránený.

3.2 Skákajúce konečné automaty

Rozdiel oproti konečnému automatu spočíva v definícii konfigurácie a binárnej prechodovej relácie nad konfiguráciami, ktorú pri skákajúcom konečnom automate nazývame *binárna skoková relácia.*

Definícia 3.2.1. Skákajúci konečný automat
Skákajúci konečný automat (ďalej len SKA) M je pätica (Q, Σ, R, s, F), kde:

- Q je konečná množina stavov,
- Σ je vstupná abeceda,
- R je konečná množina prechodových pravidiel tvaru $pa \to q$, kde $p, q \in Q$ a $a \in \Sigma \cup \{\varepsilon\}$,
- $s \in Q$ je počiatočný stav,
- $F \subseteq Q$ je množina koncových stavov.

Definícia 3.2.2. Konfigurácia SKA
Nech $M = (Q, \Sigma, R, s, F)$ je SKA. *Konfigurácia SKA* M je trojica (x, q, y), kde:

- $xy \in \Sigma^*$ je vstupný reťazec na vstupnej páske s najľavejším symbolom reťazca y pod čítacou hlavou,
- $q \in Q$ je stav, v ktorom sa automat nachádza.

Definícia 3.2.3. Binárna skoková relácia SKA
Nech $M = (Q, \Sigma, R, s, F)$ je SKA. *Binárna skoková relácia SKA* nad konfiguráciami $\Sigma^* \times Q \times \Sigma^*$, symbolicky značená $\curvearrowright$, je definovaná nasledovne:

$$(x, q, ay) \curvearrowright (x', r, y')$$

pre nejaké $q, r \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $x, y, x', y' \in \Sigma^*$, pre ktoré $qa \to r \in R$ a $xy = x'y'$.

Môžeme rozšíriť $\curvearrowright$ na $\curvearrowright^m$, kde $m \geq 0$, $\curvearrowright^+$ je tranzitívny uzáver a $\curvearrowright^*$ je tranzitívno-reflexívny uzáver.

Vstupný reťazec xy je prijatý SKA M len ak $(x, s, y) \curvearrowright^* (\varepsilon, f, \varepsilon)$, kde s je počiatočný stav a $f \in F$.

Definícia 3.2.4. Jazyk prijímaný SKA
Nech $M = (Q, \Sigma, R, s, F)$ je SKA. Jazyk $L(M)$, prijímaný SKA M, je množina reťazcov $\{xy |\ (x, s, y) \curvearrowright^* (\varepsilon, f, \varepsilon)$ pre nejaký $f \in F\}$.

Príklad 3.2.5. SKA
Predpokladajme SKA $M = (\{s, r, t\}, \Sigma, R, s, \{s\})$, kde $\Sigma = \{a, b, c\}$ a R obsahuje pravidlá $sa \to r$,$rb \to t$ a $tc \to s$. Začínajúc v stave s, M číta postupne symboly a, b a c, potom sa vracia späť do počiatočného stavu, ktorý je zároveň aj koncovým. Jednotlivé symboly sa môžu vyskytovať kdekoľvek na vstupnej páske.
Jazyk $L(M) = \{w \in \Sigma^* |\ |w|_a = |w|_b = |w|_c\}$, ktorý patrí do triedy kontextových jazykov.

SKA dokážu prijať napr. unárne regulárne jazyky a niektoré kontextové jazyky. Z kontextových jazykov sú to jazyky, ktorých reťazce sú definované pomerom výskytov jednotlivých symbolov, ale nie ich poradím.

3.3 Modifikácia konečného prevodníka

Pri modifikácii konečného prevodníka na *skákajúci konečný prevodník* budeme postupovať podobne ako pri modifikovaní konečného automatu.

Predpokladajme, že $M = (Q, \Sigma, \Delta, R, s, F)$ je konečný prevodník (pozri def. 2.3.3.1 na str. 11), ktorý pozostáva zo vstupnej pásky s čítacou hlavou, výstupnej pásky so zapisovacou hlavou a konečnej stavovej konroly. Vstupná aj výstupná páska sú rozdelené na bunky. Bunka vstupnej pásky obsahuje jeden symbol vstupnej abecedy Σ a bunka výstupnej obsahuje symbol výstupnej abecedy Δ. Stavová kontrola pozostáva z množiny stavov Q a množiny prechodových pravidiel R. Preklad reťazca prevodníkom M spočíva v tom, že pri každom prechode medzi stavmi, ktorý sa deje na základe pravidla, môžu byť na výstupnú pásku zapísané nejaké symboly. Po prečítaní symbolu zo vstupnej pásky sa čítacia hlava, rovnako ako pri konečnom automate, presúva presne o jeden symbol doprava. Reťazec w môžeme nazvať výstupom pre daný vstupný reťazec, pre ktorý prevodník M počas spracovávania prejde z počiatočného stavu s do niektorého z koncových stavov. Inak sa nejedná o validný výstup.

Našou úlohou je aplikovať podobnú modifikáciu (pozri modif. KA 3.1 na str. 12) na KP M.

Skákajúci konečný prevodník pracuje z časti podobne ako SKA a z časti ako KP. Počas spracovávania reťazca prechádza medzi stavmi, pričom čítacia hlava sa nemusí presúvať zaužívane zľava-doprava, ale môže, ako pri SKA, časť pásky preskočiť a pokračovať spracovanie odtiaľ. Avšak narozdiel od SKA, môže skákajúci konečný prevodník v každom stave vložiť nejaké symboly na výstupnú pásku.

3.4 Skákajúce konečné prevodníky

Definícia 3.4.1. Skákajúci konečný prevodník
Skákajúci konečný prevodník (ďalej len SKP) M je šestica $(Q, \Sigma, \Delta, R, s, F)$, kde:

- Q je konečná množina stavov,
- Σ je vstupná abeceda,
- Δ je výstupná abeceda,
- R je konečná množina prechodových pravidiel tvaru $pa \to qz$, kde $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$ a $z \in \Delta^*$,
- $s \in Q$ je počiatočný stav,
- $F \subseteq Q$ je množina koncových stavov.

Definícia 3.4.2. Konfigurácia SKP
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je SKP. *Konfigurácia SKP* M je štvorica (x, q, y, z), kde:

- $q \in Q$ je stav, v ktorom sa prevodník nachádza,
- $xy \in \Sigma^*$ je reťazec na vstupnej páske s najľavejším symbolom reťazca y pod čítacou hlavou,
- $z \in \Delta^*$ je reťazec na výstupnej páske.

Definícia 3.4.3. Binárna skoková relácia SKP
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je SKP. *Binárna skoková relácia SKP* nad konfiguráciami $\Sigma^* \times Q \times \Sigma^* \times \Delta^*$, symbolicky značená $\curvearrowright$, je definovaná nasledovne:

$$(x, q, ay, z) \curvearrowright (x', r, y', zb)$$

pre nejaké $q, r \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $x, y, x', y' \in \Sigma^*$, $z, b \in \Delta^*$, pre ktoré $qa \to rb \in R$ a $xy = x'y'$.

Môžeme rozšíriť $\curvearrowright$ *na* $\curvearrowright^m$, kde $m \geq 0$; $\curvearrowright^+$ je tranzitívny uzáver a $\curvearrowright^*$ je tranzitívno-reflexívny uzáver.

Vravíme, že w je prekladom vstupného reťazca xy, ak $(x, s, y, \varepsilon) \curvearrowright^* (\varepsilon, f, \varepsilon, w)$ pre nejaké $f \in F$.

Definícia 3.4.4. Preklad definovaný SKP
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je SKP. Preklad definovaný SKP M, $\tau(M)$, je množina usporiadaných dvojíc reťazcov $\{(xy, w) |\ (x, s, y, \varepsilon) \curvearrowright^* (\varepsilon, f, \varepsilon, w)$ pre nejaké $f \in F\}$.

Kapitola 4

Determinizmus

Doteraz sme sa vždy zaoberali nedeterministickými verziami formálnych modelov. Pri nedeterministických automatoch môže nastať zmena stavu aj bez toho, aby bol nejaký symbol zo vstupnej pásky prečítaný alebo pre danú konfiguráciu existuje viacero kongigurácií, do ktorých môže na základe súčasného stavu prejsť.

Príklad 4.0.1. Nedeterministický KA
Predpokladajme nedeterministický konečný automat $M = (\{s,t\}, \{a,b\}, R, s, \{t\})$, kde množina pravidiel R obsahuje pravidlá $sa \to s$, $sa \to t$ a $tb \to t$. Vstupným reťazcom bude ab. M začne spracovávať reťazec ab v konfigurácii (s, ab), v ktorej má k dispozícii dve pravidlá, podľa ktorých môže vykonať prechod. Možeme použiť pravidlo pravidlo $sa \to s$, pomocou ktorého sa zmení konfigurácia na (s, b) alebo pravidlo $sa \to t$, ktorým M prejde do konfigurácie (t, b).

Ako môžeme vidieť, v prvej z možností sa M zasekne, keďže pre konfiguráciu (s, b) neexistuje žiadne pravidlo. Pre konfiguráciu (t, b) môže výpočet pokračovať prechodom pomocou pravidla $tb \to t$, čím sa M dostane do koncovej konfigurácie (t, ε) a reťazec ab je prijatý.

Ak chceme nejaký formálny model implementovať, musíme byť schopný v každom kroku výpočtu určiť maximálne jeden nasledujúci stav. Nemôže nastať situácia, že máme na výber viac ako jednu možnosť.

4.1 Tradičný determinizmus SKA

Vyššie uvedená definícia 3.2.1 popisuje nedeterministický SKA. Pre zavedenie deterministického konečného skákajúceho automatu je najprv potrebné zabezpečiť, aby zmena stavu automatu nemohla nastať bez prečítania nejakého symbolu zo vstupnej pásky. Docielime to eliminovaním ε-pravidiel tvaru $q \to t$, kde q, t sú stavy automatu.

V nasledovných riadkoch sa budeme zaoberať deterministickými skákajúcimi konečnými automatmi zavedenými v článku [6] a poukážeme na problém, že tradičný determinizmus pre SKA nestačí, čím sa otvára nová úloha – zavedenie striktného determinizmu.

Prevod SKA na SKA bez ε-pravidiel je možný pomocou klasického prevodového algoritmu KA na KA bez ε-pravidiel [6].

Definícia 4.1.1. SKA bez ε-pravidiel
Nech $M = (Q, \Sigma, R, s, F)$ je SKA. M je *SKA bez ε-pravidiel* vtedy, ak pre všetky pravidlá $qa \to r \in R$ platí $|a| = 1$.

Podobne ako pri eliminácii ε-pravidiel, aj pri prevode SKA bez ε-pravidiel na *deterministický SKA* môžeme použiť klasický prevodový algoritmus KA bez ε-pravidiel na deterministický KA [6].

Pomocou SKA bez ε-pravidiel už môžeme definovať deterministický SKA.

Definícia 4.1.2. Deterministický SKA
Nech $M = (Q, \Sigma, R, s, F)$ je SKA bez ε-pravidiel. M je deterministický SKA (ďalej len DSKA) vtedy, ak pre všetky $q \in Q$ a $a \in \Sigma$ existuje najviac jedno $r \in Q$ také, že $qa \to r \in R$.

Definícia 4.1.3. DSKA (podrobná definícia)
DSKA M je pätica (Q, Σ, R, s, F), kde:

- Q je konečná množina stavov,
- Σ je vstupná abeceda,
- R je konečná množina prechodových pravidiel tvaru $qa \to r$, kde $q, r \in Q$ a $a \in \Sigma$, pričom pre všetky dvojice $q \in Q$ a $a \in \Sigma$ existuje najviac jedno $r \in Q$ také, že $qa \to r \in R$,
- $s \in Q$ je počiatočný stav,
- $F \subseteq Q$ je množina koncových stavov.

Definícia 4.1.4. Konfigurácia DSKA
Nech $M = (Q, \Sigma, R, s, F)$ je DSKA. *Konfigurácia DSKA* M je trojica (x, q, y), kde:

- $xy \in \Sigma^*$ je vstupný reťazec na vstupnej páske s najľavejším symbolom reťazca y pod čítacou hlavou,
- $q \in Q$ je stav, v ktorom sa automat nachádza.

Definícia 4.1.5. Binárna skoková relácia DSKA
Nech $M = (Q, \Sigma, R, s, F)$ je DSKA. *Binárna skoková relácia DSKA* nad konfiguráciami $\Sigma^* \times Q \times \Sigma^*$, symbolicky značená $\curvearrowright$, je definovaná nasledovne:

$$(x, q, ay) \curvearrowright (x', r, y')$$

pre nejaké $q, r \in Q$, $a \in \Sigma$, $x, y, x', y' \in \Sigma^*$, pre ktoré $qa \to r \in R$ a $xy = x'y'$.

Môžeme rozšíriť $\curvearrowright$ *na* $\curvearrowright^m$, kde $m \geq 0$, $\curvearrowright^+$ je tranzitívny uzáver a $\curvearrowright^*$ je tranzitívno-reflexívny uzáver.

Vstupný reťazec xy je prijatý DSKA M len ak $(x, s, y) \curvearrowright^* (\varepsilon, f, \varepsilon)$, kde s je počiatočný stav a $f \in F$.

Definícia 4.1.6. Jazyk prijímaný DSKA
Nech $M = (Q, \Sigma, R, s, F)$ je DSKA. Jazyk $L(M)$, prijímaný DSKA M, je množina reťazcov $\{xy |\ (x, s, y) \curvearrowright^* (\varepsilon, f, \varepsilon)$ pre nejaký $f \in F\}$.

Avšak pre DSKA M tu vzniká problém. V binárnej skokovej relácii $(x, q, ay) \curvearrowright (x', r, y')$ nie je explicitne uvedené, ako sa zmení konfigurácia. Vieme len, že $xy = x'y'$ a $a \in \Sigma$. Nemáme žiadnu informáciu o najľavejšom symbole reťazca y', nad ktorým bude čítacia hlava v nasledujúcom stave. To znamená, že pre M existuje $n+1$ možných konfigurácií, do ktorých môže skočiť, pričom $n = |x'y'|$.

Príklad 4.1.7. Nedostatočný determinizmus DSKA
Predpokladajme DSKA $M = (\{s, t\}, \{a, b\}, R, s, \{s\ \})$, kde množina R obsahuje pravidlá:

1. $sa \to t$
2. $tb \to s$

Vstupný reťazec je $abba$. M začne v konfigurácii $(\varepsilon, s, abba)$, pre ktorú existuje pravidlo 1. Na základe tohto pravidla vieme, že M prejde do stavu t. Kedže pre DSKA nie je definované kam sa má presunúť čítacia hlava, existujú štyri možné konfigurácie:

1. (ε, t, bba)
2. (b, t, ba)
3. (bb, t, a)
4. (bba, t, ε)

pričom pre posledné dve z nich sa M zasekne.

4.2 Striktný determinizmus SKA

V predošlej časti sme poukázali na fakt, že tradičný determinizmus založený na obmedzení stavovej kontroly nie je pre skákajúce konečné automaty dostatočný, kedže z neho nijako nevyplýva kam sa má čítacia hlava presunúť. Obmedzenie stavovej kontroly DSKA spočíva v tom, že pre každú kombináciu stavu a nejakého vstupného symbolu existuje vždy maximálne jeden stav, do ktorého sa može automat presunúť. Toto obmedzenie budeme naďalej uvažovať.

Našou úlohou je teraz vyriešiť problém nedostatočného determinizmu DSKA.
Idea: Určiť symbol, nad ktorý sa má čítacia hlava v budúcom kroku výpočtu presunúť. Ktorý symbol to bude?

Nech $M = (Q, \Sigma, R, s, F)$ je DSKA. Ak (x, q, ay) je konfigurácia M a $qa \to r \in R$, potom pre dvojicu (q, a) existuje množina *nezasekujúcich symbolov* $A \subseteq \Sigma$, pre ktoré platí, že ak M v budúcom stave r nejaký symbol $b \in A$ prečíta, tak sa nezasekne. Množina A môže byť aj prázdna. V tom prípade sa DSKA M zasekne.

Príklad 4.2.1. Množina symbolov, pre ktoré sa DSKA nezasekne
Predpokladajme DSKA $M_A = (\{s, t\}, \{a, b, c\}, R, s, \{s\ \})$, kde množina R obsahuje pravidlá:

1. $sa \to t$
2. $tb \to s$
3. $tc \to s$

Vstupný reťazec je $caacca$. M_A začne v konfigurácii $(ca, s, acca)$, pre ktorú existuje pravidlo 1. Na základe tohto pravidla vieme, že M_A prejde do stavu t. Pre M_A existuje viacero možných konfigurácií, do ktorých môže prejsť:

1. $(\varepsilon, t, cacca)$
2. $(c, t, acca)$

3. (ca, t, cca)
4. (cac, t, ca)
5. $(cacc, t, a)$
6. $(cacca, t, \varepsilon)$

Pri konfiguráciach 1, 3 a 4 sa M_A nezasekne a pri konfiguráciach 2, 5 a 6 sa zasekne. Nech A je množina symbolov, pre ktoré sa M_A nezasekne. Všimnime si, že ak $tb \to s \in R$ a $tc \to s \in R$, potom $b \in A$ a $c \in A$, aj keď $b \notin alph(cacca)$.

Z príkladu vidíme, že pre každú dvojicu (q, a), kde $q \in Q$, $a \in \Sigma$ a $qa \to r \in R$ existuje nejaká množina *nezasekujúcich symbolov* $A_{(q,a)}$, ktoré môže automat M v stave r prijať bez toho, aby sa zasekol.

Definícia 4.2.2. Relácia nezasekujúcich symbolov pre DSKA
Nech $M = (Q, \Sigma, R, s, F)$ je DSKA. *Relácia nezasekujúcich symbolov* α z $Q \times \Sigma$ do Σ pre DSKA M je definovaná nasledovne:

- ak $qa \to r \in R$, potom pre všetky symboly $b, c \in \Sigma$, pričom $rb \to c \in R$, platí $((q, a), b) \in \alpha$

Skutočnosť, že $((q, a), b) \in \alpha$ môžeme zapisovať aj $b \in \alpha(q, a)$.

Pre každý DSKA môžeme pomocou nasledovného algoritmu zostaviť reláciu α.

Algoritmus 1: Relácia α pre DSKA

Input: DSKA $M = (Q, \Sigma, R, s, F)$
Output: Relácia α pre M

```
1: α = ∅
2: for all qa → r ∈ R do
3:     for all rb → t ∈ R do
4:         add ((q, a), b) to α
5:     end for
6: end for
7: return α
```

Pre dosiahnutie striktného determinizmu skákajúcich konečných automatov budeme modifikovať binárnu skokovú reláciu.

4.2.1 Modifikácia binárnej skokovej relácie

Pomocou relácie nezasekujúcich symbolov α budeme môcť v binárnej skokovej relácii DSKA určiť, ako sa zmení konfigurácia. V prípade, že sa niektoré symboly vysktujú v reťazci viac krát, nebudeme vedieť nad ktorý sa má čítacia hlava presunúť. Preto pre dosiahnutie striktného determinizmu je potrebné ešte určiť presne jeden výskyt symbolu na vstupnej páske.

Predpokladajme DSKA $M = (Q, \Sigma, R, s, F)$ a α je relácia nezasekujúcich symbolov pre M. M je v konfigurácii (x, q, ay), kde $q \in Q$, $a \in \Sigma$ a $x, y \in \Sigma^*$.
Idea: Čítacia hlava sa presunie nad prvý výskyt symbolu $b \in \alpha(q, a)$ v reťazci xy zľava. V prípade, že reťazec xy neobsahuje žiadny symbol $b \in \alpha(q, a)$, M sa zasekne.

4.2.2 Zavedenie striktne deterministických SKA

Ak aplikujeme modifikáciu binárnej skokovej relácie, načrtnutú v minulom odstavci, sme schopní definovať striktne deterministický skákajúci konečný automat.

Definícia 4.2.2.1. Striktne deterministický SKA
Striktne deterministický SKA (ďalej len SDSKA) M je pätica (Q, Σ, R, s, F), kde:

- Q je konečná množina stavov,
- Σ je vstupná abeceda,
- R je konečná množina prechodových pravidiel tvaru $qa \to r$, kde $q, r \in Q$ a $a \in \Sigma$, pričom pre všetky dvojice $q \in Q$ a $a \in \Sigma$ existuje najviac jedno $r \in Q$ také, že $qa \to r \in R$.
- $s \in Q$ je počiatočný stav,
- $F \subseteq Q$ je množina koncových stavov.

Definícia 4.2.2.2. Konfigurácia SDSKA
Nech $M = (Q, \Sigma, R, s, F)$ je SDSKA. *Konfigurácia SDSKA* M je trojica (x, q, y), kde:

- $xy \in \Sigma^*$ je vstupný reťazec na vstupnej páske s najľavejším symbolom reťazca y pod čítacou hlavou,
- $q \in Q$ je stav, v ktorom sa automat nachádza.

Definícia 4.2.2.3. Počiatočná konfigurácia SDSKA
Nech $M = (Q, \Sigma, R, s, F)$ je SDSKA a α je relácia nezasekujúcich symbolov pre M. *Počiatočná konfigurácia SDSKA* M pre vstupný reťazec xy je trojica (x, s, y), kde platí:

- pre každé $a \in \Sigma$ ak $sa \to t \in R$ a $a \in alph(xy)$, potom a je najľavejším symbolom reťazca y, pričom $a \notin alph(x)$
- ak pre každé pravidlo $sa \to t \in R$ platí, že $a \notin alph(xy)$, potom M sa v tejto konfigurácii zasekne

Definícia 4.2.2.4. Deterministická binárna skoková relácia SDSKA
Nech $M = (Q, \Sigma, R, s, F)$ je SDSKA a α je relácia nezasekujúcich symbolov pre M. *Deterministická binárna skoková relácia SDSKA* nad konfiguráciami $\Sigma^* \times Q \times \Sigma^*$, symbolicky značená $\curvearrowright_d$, je definovaná nasledovne:

$$(x, q, ay) \curvearrowright_d (x', r, y')$$

pre nejaké $q, r \in Q$, $a \in \Sigma$, $x, y, x', y' \in \Sigma^*$, pre ktoré $qa \to r \in R$, $xy = x'y'$, pričom najľavejší symbol reťazca y' je nejaký symbol $b \in \alpha(q, a)$ a pre všetky $c \in alph(x')$ platí $c \notin \alpha(q, a)$. Ak pre každý symbol $d \in alph(x'y')$ platí, že $d \notin \alpha(q, a)$, prípadne ak $x'y' = \varepsilon$, potom sa M v tejto konfigurácii zastaví.

Môžeme rozšíriť $\curvearrowright_d$ *na* $\curvearrowright_d^m$, kde $m \geq 0$, $\curvearrowright_d^+$ je tranzitívny uzáver a $\curvearrowright_d^*$ je tranzitívno-reflexívny uzáver.

Vstupný reťazec xy je prijatý SDSKA M len ak $(x, s, y) \curvearrowright_d^* (\varepsilon, f, \varepsilon)$, kde s je počiatočný stav a $f \in F$.

Definícia 4.2.2.5. Jazyk prijímaný SDSKA
Nech $M = (Q, \Sigma, R, s, F)$ je SDSKA. Jazyk $L(M)$, prijímaný SDSKA M, je množina reťazcov $\{xy|\ (x, s, y) \curvearrowright_d^* (\varepsilon, f, \varepsilon)$ pre nejaký $f \in F\}$.

Príklad 4.2.2.6. Príklad SDSKA
Nech $M = (\{s, t\}, \{a, b\}, R, s, \{s\})$ je SDSKA. Množina R obsahuje pravidlá:

1. $sa \to t$
2. $tb \to s$

$L(M) = \{w \in \{a, b\}^*|\ |w|_a = |w|_b\}$. Uvažujme vstupný reťazec $baba$. M začína výpočet v počiatočnej konfigurácii (b, s, aba). Výpočet SDSKA M bude nasledovný:

$$(b, s, aba) \curvearrowright_d (\varepsilon, t, bba) \curvearrowright_d (b, s, a) \curvearrowright_d (\varepsilon, t, b) \curvearrowright_d (\varepsilon, s, \varepsilon)$$

Reťazec $baba$ je prijatý.

Uvažujme reťazec baa. M začína výpočet v počiatočnej konfigurácii (b, s, aa). Výpočet SDSKA M bude nasledovný:

$$(b, s, aa) \curvearrowright_d (\varepsilon, t, ba) \curvearrowright_d (\varepsilon, s, a) \curvearrowright_d (\varepsilon, t, \varepsilon)$$

Reťazec baa je zamietnutý, kedže $t \notin F$.

4.3 Tradičný determinizmus SKP

Pri definovaní deterministických skákajúcich konečných prevodníkoch budeme postupovať podobne ako pri automatoch. Avšak nie pre každý nedeterministický skákajúci konečný prevodník existuje jeho deterministický ekvivalent. Dva prevodníky M_1 a M_2 považujeme za ekvivalentné, práve ak $\tau(M_1) = \tau(M_2)$. Ako príklad uvediem ε-pravidlá.

Príklad 4.3.1. Príklad prekladu nedeterministického SKP
Predpokladajme SKP $M = (\{s\}, \{a\}, \{b, c\}, R, s, \{s\})$, kde R obsahuje:

1. $s \to sbb$
2. $sa \to sc$

Prekladmi vstupného reťazca aa sú nasledovné reťazce:

- $cbbbbbc$
- cc
- $bbcc$
- $ccbb$
- ...

Všetky sú správne a je ich nekonečne veľa. Pre daný prevodník M neexistuje ekvivalentný SKP bez ε-pravidiel.

Definícia 4.3.2. Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je SKP. M je *deterministickým skákajúcim konečným prevodníkom* (ďalej len DSKP), ak pre každé pravidlo $pa \to qb \in R$ platí, že $p, q \in Q$, $a \in \Sigma, b \in \Delta^*$ a pre všetky $r \in Q$, že ak $r \neq q$, potom $pa \to rc \notin R$.

Rovnako, ako aj pre DSKA (pozri 4.1.7 na str. 18), vzniká pre DSKP problém nedostatočného determinizmu, že pre nejakú konfiguráciu existuje viacero možných konfigurácii, do ktorých skočiť. Pre niektoré sa DSKP zasekene a pre niektoré nie. Z nejednoznačnosti výberu nasledujúcej konfigurácie vyplýva aj ďalší problém. Preklad DSKP nie je homomorfným.

Príklad 4.3.3. Príklad nehomomorfného prekladu DSKP
Predpokladajme DSKP $M = (\{s, t\}, \{a, b, c\}, \{a', b', c'\}, R, s, \{s\ \})$, kde R obsahuje:

1. $sa \to ta'$
2. $tb \to sb'$
3. $tc \to sb'$

Pre vstupný reťazec *caba* existuje viacero prekladov:

$$(c, s, aba, \varepsilon) \curvearrowright (c, t, ba, a') \curvearrowright (c, s, a, a'b') \curvearrowright (\varepsilon, t, c, a'b'a') \curvearrowright (\varepsilon, s, \varepsilon, a'b'a'c')$$

$$(c, s, aba, \varepsilon) \curvearrowright (\varepsilon, t, cba, a') \curvearrowright (b, s, a, a'c') \curvearrowright (\varepsilon, t, b, a'c'a') \curvearrowright (\varepsilon, s, \varepsilon, a'c'a'b')$$

Aby sme mohli definovať striktne deterministické konečné prevodníky, potrebujeme definovať *reláciu nezasekujúcich symbolov* α pre DSKP.

Definícia 4.3.4. Relácia nezasekujúcich symbolov pre DSKP
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je DSKP. *Relácia nezasekujúcich symbolov* α z $Q \times \Sigma$ do Σ pre DSKP M je definovaná nasledovne:

- ak $qa \to rx \in R$, potom pre všetky symboly $b, c \in \Sigma$, pričom $rb \to cy \in R$, platí $((q, a), b) \in \alpha$

Skutočnosť, že $((q, a), b) \in \alpha$ môžeme zapisovať aj $b \in \alpha(q, a)$.

4.4 Striktný determinizmus SKP

Definícia 4.4.1. Striktne deterministický SKP
Striktne deterministický SKP (ďalej len SDSKP) M je šestica $(Q, \Sigma, \Delta, R, s, F)$, kde:

- Q je konečná množina stavov,
- Σ je vstupná abeceda,
- Δ je výstupná abeceda,
- R je konečná množina prechodových pravidiel tvaru $qa \to ry$, kde $q, r \in Q$, $a \in \Sigma$ a $y \in \Delta^*$, pričom pre všetky dvojice $q \in Q$ a $a \in \Sigma$ existuje najviac jedno $r \in Q$ také, že $qa \to ry \in R$.
- $s \in Q$ je počiatočný stav,
- $F \subseteq Q$ je množina koncových stavov.

Definícia 4.4.2. Konfigurácia SDSKP
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je SDSKP. *Konfigurácia SDSKP* M je štvorica (x, q, y, z), kde:

- $xy \in \Sigma^*$ je reťazec na vstupnej páske s najľavejším symbolom reťazca y pod čítacou hlavou,
- $q \in Q$ je stav, v ktorom sa automat nachádza,
- $z \in \Delta^*$ je reťazec na výstupnej páske.

Definícia 4.4.3. Počiatočná konfigurácia SDSKP
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je SDSKP a α je relácia nezasekujúcich symbolov pre M. *Počiatočná konfigurácia SDSKP* M pre vstupný reťazec xy je štvorica (x, s, y, ε), kde platí:

- pre každé $a \in \Sigma$ ak $sa \to t \in R$ a $a \in alph(xy)$, potom a je najľavejším symbolom reťazca y, pričom $a \notin alph(x)$
- ak pre každé pravidlo $sa \to t \in R$ platí, že $a \notin alph(xy)$, potom $x = \varepsilon$ a M sa v tejto konfigurácii zasekne

Definícia 4.4.4. Deterministická binárna skoková relácia SDSKP
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je SDSKP a α je relácia nezasekujúcich symbolov pre M. *Deterministická binárna skoková relácia SDSKP* nad konfiguráciami $\Sigma^* \times Q \times \Sigma^* \times \Delta^*$, symbolicky značená $\curvearrowright_d$, je definovaná nasledovne:

$$(x, q, ay, z) \curvearrowright_d (x', r, y', zp)$$

pre nejaké $q, r \in Q$, $a \in \Sigma$, $p, z \in \Delta^*$, $x, y, x', y' \in \Sigma^*$, pre ktoré $qa \to rp \in R$, $xy = x'y'$, pričom najľavejší symbol reťazca y' je nejaký symbol $b \in \alpha(q, a)$ a pre všetky $c \in alph(x')$ platí $c \notin \alpha(q, a)$. Ak pre každý symbol $d \in alph(x'y')$ platí, že $d \notin \alpha(q, a)$, prípadne ak $x'y' = \varepsilon$, potom sa M v tejto konfigurácii zastaví.

Môžeme rozšíriť $\curvearrowright_d$ *na* $\curvearrowright_d^m$, kde $m \geq 0$, $\curvearrowright_d^+$ je tranzitívny uzáver a $\curvearrowright_d^*$ je tranzitívno-reflexívny uzáver.

Vravíme, že w je prekladom vstupného reťazca xy, ak $(x, s, y, \varepsilon) \curvearrowright_d^* (\varepsilon, f, \varepsilon, w)$ pre nejaké $f \in F$.

Definícia 4.4.5. Preklad definovaný SDSKP
Nech $M = (Q, \Sigma, \Delta, R, s, F)$ je SKP. Preklad definovaný SDSKP M, $\tau(M)$, je množina usporiadaných dvojíc reťazcov $\{(xy, w) |\ (x, s, y, \varepsilon) \curvearrowright_d^* (\varepsilon, f, \varepsilon, w)$ pre nejaké $f \in F\}$.

Vďaka tomu, že v deterministickej skokovej relácii SDSKP je presne definovaný tvar nasledovnej konfigurácie, je preklad SDSKP homomorfným. Kedže vždy pri prekladaní reťazca skáčeme nad najľavejší výskyt symbolu, pre ktorý sa SDSKP nezasekne, existuje len jedna postupnosť pravidiel, pomocou ktorých sa reťazec preloží.

Kapitola 5

Využitie

Do množiny jazykov, ktoré je možné akceptovať skákajúcimi konečnými automatmi patria niektoré regulárne jazyky (unárne), bezkontextové a kontextové jazyky, ktoré obashujú také reťazce, že poradie symbolov nie je dôležité. Informácia, ktorú môžu reťazce takýchto jazykov prenášať je určená pomerom počtu výskytov jednej skupiny symbolov voči inej skupine.

5.1 Jazyky

Príklad 5.1.1. Príklad SDSKA
Predpokladajme SDSKA $M_1 = (\{s,t,r\}, \Sigma, R, s, \{s\ \})$, kde $\Sigma = \{a,b,c\}$ a množina R obsahuje pravidlá:

1. $sa \to t$
2. $tb \to r$
3. $rc \to s$

Vstupným reťazcom je $ccbaab$. M_1 začína v konfigurácii (ccb, s, aab). Spracovanie reťazca bude nasledovné:

$$(ccb,s,aab) \curvearrowright_d (cc,t,bab) \curvearrowright_d (\varepsilon,t,ccab) \curvearrowright_d (c,s,ab) \curvearrowright_d (c,t,b) \curvearrowright_d (\varepsilon,r,c) \curvearrowright_d (\varepsilon,s,\varepsilon)$$

Reťazec $ccbaab$ je SDSKA M_1 prijatý. Ako vidíme, jazyk, ktorý M_1 prijíma, je $L(M_1) = \{w \in \Sigma^* |\ |w|_a = |w|_b = |w|_c\}$, ktorý patrí do triedy kontextových jazykov.

Skákajúce konečné automaty dokážu prijať reťazce definované pomermi počtov výskytov jednotlivých symbolov. Tieto pomery nemusia byť nutne iba 1:1. V prípade, že by sme chceli napríklad prijímať reťazce, kde je počet výskytov symbolu a trikrát vyšší ako počet výskytov symbolov b, bude vyzerať SDSKA nasledovne.

Príklad 5.1.2. Príklad pomeru 3:1
Predpokladajme SDSKA $M_2 = (\{s,t,r,p\}, \Sigma, R, s, \{s\ \})$, kde $\Sigma = \{a,b\}$ a množina R obsahuje pravidlá:

1. $sa \to t$
2. $ta \to p$

3. $pa \to r$

4. $rb \to s$

Vstupným reťazcom je $abaa$. M_2 začína v konfigurácii $(\varepsilon, s, abaa)$. Spracovanie reťazca bude nasledovné:

$$(\varepsilon, s, abaa) \curvearrowright_d (b, t, aa) \curvearrowright_d (b, p, a) \curvearrowright_d (\varepsilon, r, b) \curvearrowright_d (\varepsilon, s, \varepsilon)$$

Reťazec $abaa$ je SDSKA M_2 prijatý a jazyk, ktorý M_2 prijíma, je $L(M_2) = \{w \in \Sigma^* |\ |w|_a = 3 * |w|_b\}$.

Jazyky nemusia byť definované iba pomermi počtov výskytov jednotlivých symbolov, ale môžu to byť aj pomery počtov symbolov z nejakých množín symbolov.

Príklad 5.1.3. Príklad pomeru množín symbolov
Predpokladajme SDSKA $M_3 = (\{s, t\}, \Sigma, R, s, \{s\ \})$, kde $\Sigma = \{a, b, c\}$ a množina R obsahuje pravidlá:

1. $sa \to t$

2. $tb \to s$

3. $tc \to s$

Vstupným reťazcom je $abca$. M_3 začína v konfigurácii $(\varepsilon, s, abca)$. Spracovanie reťazca bude nasledovné:

$$(\varepsilon, s, abca) \curvearrowright_d (\varepsilon, t, bca) \curvearrowright_d (c, s, a) \curvearrowright_d (\varepsilon, t, c) \curvearrowright_d (\varepsilon, s, \varepsilon)$$

Reťazec $abca$ je SDSKA M_3 prijatý a jazyk, ktorý M_3 prijíma, je $L(M_3) = \{w \in \Sigma^* |\ |w|_a = |w|_b + |w|_c\}$.

Ako sme videli na príkladoch, jazyky definované akýmkoľvek celočíselným pomerom počtov výskytov symbolov je možné akceptovať skákajúcimi konečnými automatmi. Na poradí symbolov reťazcov, ktoré SKA dokážu prijať, nezáleží.

Príklad 5.1.4. Príklad prekladu SDSKP
Predpokladajme SDSKP $M = (\{s, t\}, \Sigma, \{a, b\}, R, s, \{s\ \})$, kde $\Sigma = \{a, b\}$ a množina R obsahuje pravidlá:

1. $sa \to ta$

2. $tb \to sb$

Vstupným reťazcom je $bbaaab$. M začína v konfigurácii $(bb, s, aaab, \varepsilon)$. Prekladanie reťazca bude nasledovné:

$$(bb, s, aaab, \varepsilon) \curvearrowright_d (\varepsilon, t, bbaab, a) \curvearrowright_d (b, s, aab, ab) \curvearrowright_d (\varepsilon, t, bab, aba) \curvearrowright_d (\varepsilon, s, ab, abab) \curvearrowright_d$$

$$(\varepsilon, t, b, ababa) \curvearrowright_d (\varepsilon, s, \varepsilon, ababab)$$

Ako vidíme prekladom reťazca $bbaaab$ je reťazec $ababab$.

SDSKP M prekladá reťazce z jazyka $L_1 = \{w \in \Sigma^* |\ |w|_a = |w|_b\}$ do jazyka $L_2 = \{\varepsilon, ab, abab, ababab, \ldots\}$. $L_2 \subset L_1$, pričom L_2 je jazyk regulárny.

5.2 Regulárne podreťazce a ich eliminácia

V predchádzajúcej časti sme predstavili jazyky, ktoré je možné prijať pomocou SDSKA a jednu ukážku prekladu SDSKP z neregulárneho jazyka na jazyk regulárny. Všetky tieto jazyky boli definované nejakým pomerom počtov výskytov symbolov. V tejto časti si ukážeme, ako spracovávať jazyky, v ktorých je dôležitý pomer počtov výskytov *regulárnych podreťazcov* voči iným symbolom, prípadne podreťazcom. *Regulárny podreťazec* je časť reťazca, ktorú je možné akceptovať konečným automatom.

Príklad 5.2.1. Príklad regulárnych podreťazcov
Predpokladajme reťazec $x = aaaaBRAVOabABBAaag$. Za regulárny podreťazec môžeme považovať ktorýkoľvek podreťazec, ale nás budú zaujímať podreťazce $BRAVO$ a $ABBA$. Našou úlohou je zisiť, či reťazec x patrí do jazyka, ktorého reťazce musia obsahovať rovnaký počet podreťazcov $BRAVO$ a $ABBA$.

Aby sme mohli aplikovať skákajúce konečné automaty musíme predtým reťazec preložiť konečným prevodníkom tak, aby sa každý regulárny podreťazec, ktorého počet výskytov nás zaujíma, prepísal buď na jeden alebo na viacero symbolov.

5.3 Modifikácia podmienky prijímaného reťazca

Doteraz sme vždy považovali prijatý reťazec za taký, ktorý je automatom celý spracovaný a zároveň automat skončí v jednom z koncových stavov. Táto podmienka prijatia reťazca však nedovoľuje iné pomery počtov výskytov symbolov ako tie, ktoré sú vyjadrené presne, rovnosťou. Napr. podmienka, že v reťazci w musí byť rovnaký počet symbolov a ako je symbolov b, teda $|w|_a = |w|_b$. V prípade, že by sme chceli pomer $|w|_a > |w|_b$, skákajúce konečné automaty na to zatiaľ nestačia.

Všimnime si, že tieto pomery závisia od podmienky prijatia reťazca. Ak oslabíme túto podmienku tak, že nie je nutné, aby bol prečítaný celý reťazec, ale na prijatie reťazca bude postačovať zaseknutie v koncovom stave, získame pomery vyjadrené nerovnosťou. Napr. $|w|_a > |w|_b$ alebo $|w|_a < |w|_b$.

SDSKA s oslabenou podmienkou prijímania (ďalej len SDSKA s OPP) je definovaný rovnako ako SDSKA (pozri SDSKA na str. 20), líši sa len v reťazci, ktorý prijíma a následne v definícii prijímaného jazyka.

Nech $M = (Q, \Sigma, R, s, F)$ je SDSKA s OPP. Vstupný reťazec xy je prijatý, ak $(x, s, y) \curvearrowright_d^* (m, f, n)$ pre nejaký $f \in F$ a $mn \in \Sigma^*$, pričom z konfigurácie (m, f, n) už nie je možný skok do inej konfigurácie.

Definícia 5.3.1. Jazyk prijímaný SDSKA s OPP
Nech $M = (Q, \Sigma, R, s, F)$ je SDSKA s OPP. Jazyk $L(M)$, prijímaný M, je množina reťazcov $\{xy |\ (x, s, y) \curvearrowright_d^* (m, f, n)$ pre nejaký $f \in F$, $mn \in \Sigma^*$ a pre všetky $c \in alph(mn)$ platí $fc \to q \notin R\}$.

5.4 CpG ostrovčeky

Našou úlohou bolo aplikovať skákajúce konečné prevodníky na lexikálne štruktúry, ktoré nie sú regulárne. Skúmali sme využitie SKA a SKP v analýze reťazcov DNA nad abecedou 4 nukleotidov $\{C, G, A, T\}$.

CpG ostrovček je všeobecne definovaný ako oblasť s viac ako 200 bázovými dvojicami, kde výskyt nukleotidov C a G je vyšší ako 50 % a pomer pozorované/očakávané (obs/exp) vyšší ako 60 % [2]. Pomer pozorované/očakávané sa počíta podľa vzorca:

$$\frac{N_{CpG}}{N_C * N_G} * N$$

kde N_{CpG} je počet výskytov dvojíc CG a GC, N_C je počet výskytov nukleotidu C, N_G je počet výskytov nukleotidu G a N je počet všetkých nukleotidov v danom úseku reťazca.

Dvojice CG a GC môžeme eliminovať konečným prevodníkom, ale aj tak nie je možné overiť pomer pozorované/očakávané pomocou SDSKA, kedže sa v ňom nachádza násobenie počtov výskytov jednotlivých symbolov. Na overenie pomerov, v ktorých sa vyskytuje násobenie, by sme potrebovali počítadlá, prípadne zásobníky symbolov. Naším cieľom je však demonštrácia prijímacej sily SDSKA, ktorý je riadený iba konečnou stavovou kontrolou.

Uvažujme jazyk L_{CG}, ktorého reťazce spĺňajú podmienku, že výskyt dvojíc CG a GC v danom segmente reťazca je vyšší ako 50 %. Tieto reťazce nespĺňajú podmienky pre CpG ostrovčeky, ale môžeme na nich demonštrovať využitie SDSKA s OPP. Túto podmienku už je možné overiť pomocou konečného prevodníka, ktorý eliminuje dvojice CG a GC a následne pomer pomocou SDSKA s OPP.

Príklad 5.4.1. Reťazce jazyka L_{CG}
Príklady reťazcov spĺňajúce podmienku:

- CGC
- $AGCCGACGCG$
- $GCACGT$

Príklady reťazcov nespĺňajúce podmienku:

- $CCGA$
- $AGTTC$
- $CCCGAAT$

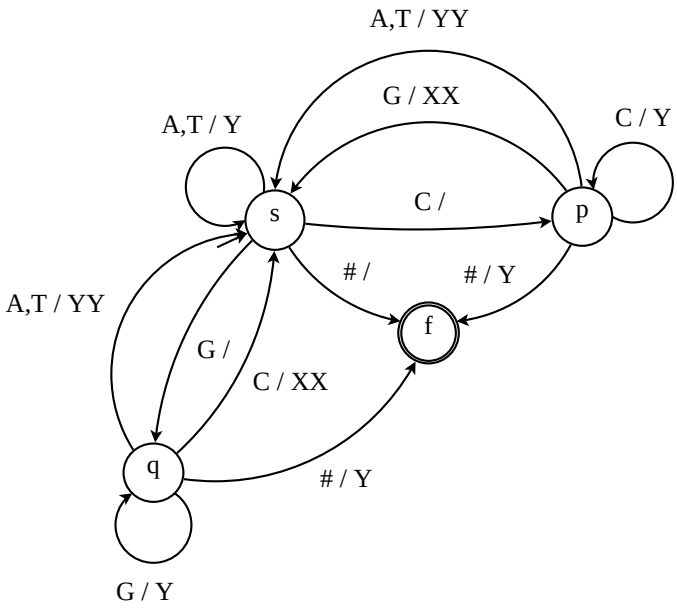
Reťazce, v ktorých je výskyt dvojíc CG a GC presne 50 % už do jazyka L_{CG} nezaraďujeme.

Dvojice CG a GC budeme musieť najprv eliminovať konečným prevodníkom prekladom na dvojicu symbolov XX a všetky ostatné symboly budeme prekladať na syboly Y. Preklad budeme realizovať konečným prevodníkom $M_1 = (Q_1, \Sigma_1, \Delta_1, R_1, s, F_1)$, kde:

- $Q_1 = \{s, p, q, f\}$
- $\Sigma_1 = \{A, C, G, T, \#\}$, # značí koniec vstupného reťazca
- $\Delta_1 = \{X, Y\}$
- R_1 obsahuje pravidlá:
  1. $sA \to sY$
  2. $sT \to sY$
  3. $sG \to q\varepsilon$

4. $sC \to p\varepsilon$
5. $s\# \to f\varepsilon$
6. $pA \to sYY$
7. $pT \to sYY$
8. $pC \to pY$
9. $pG \to sXX$
10. $p\# \to fY$
11. $qA \to sYY$
12. $qT \to sYY$
13. $qG \to qY$
14. $qC \to sXX$
15. $q\# \to fY$

- $F_1 = \{f\}$



Obr. 5.1: KP na elimináciu dvojíc CG a GC

Preklad definovaný KP M_1 značíme τ_{CG}. Ukážeme si preklad reťazca $CCGC\#$:

$$(s, CCGC\#, \varepsilon) \vdash (p, CGC\#, \varepsilon) \vdash (p, GC\#, Y) \vdash (s, C\#, YXX) \vdash$$
$$(p, \#, YXX) \vdash (f, \varepsilon, YXXY)$$

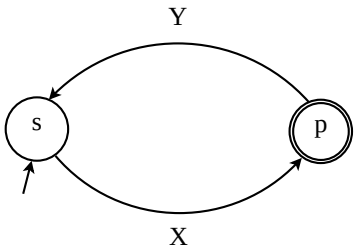
a ešte $ACT\#$:

$$(s, ACT\#, \varepsilon) \vdash (s, CT\#, Y) \vdash (p, T\#, Y) \vdash (s, \#, YYY) \vdash (f, \varepsilon, YYY)$$

teda $(CCGC\#, YXXY) \in \tau_{CG}$ a $(ACT\#, YYY) \in \tau_{CG}$.

Po eliminácii regulárnych podreťazcov prekladom do abecedy $\{X, Y\}$ musíme ešte overiť pomer výskytov symbolov X a Y. Tento pomer budeme overovať pomocou SDSKA s OPP $M_2 = (Q_2, \Sigma_2, R_2, s, F_2)$, kde:

- $Q_2 = \{s, p\}$
- $\Sigma_2 = \{X, Y\}$
- R_2 obsahuje pravidlá:
  1. $sX \to p$
  2. $pY \to s$
- $F_2 = \{p\}$



Obr. 5.2: SDSKA s OPP na overenie pomeru symbolov X a Y

SDSKA s OPP M_2 prijíma jazyk $L_{XY} = \{w \in \Sigma_2^* \mid |w|_X > |w|_Y\}$. Aby sme o nejakom reťazci z mohli prehlásiť, že patrí do jazyka L_{CG}, teda výskyt dvojíc CG a GC je vyšší ako 50 %, musí platiť $(z, z') \in \tau_{CG}$ a $z' \in L_{XY}$, inak $z \notin L_{CG}$.

Vstupný reťazec DNA je predspracovaný pomocou KP a následne analyzovaný SDSKA s OPP. Výstupom tohto systému je informácia, či daný reťazec patrí alebo nepatrí do jazyka L_{CG}. Pre popis takéhoto prekladu nebolo nutné priamo využívať SDSKP, kedže výstup SDSKA s OPP spočíva v rozhodnutí, či reťazec je alebo nieje prvkom jazyka L_{XY}.

5.5 Implementácia

Zložitosť algoritmu, ktorým je SDSKA, prípadne SDSKA s OPP popísaný, závisí od relácie nezasekuúcich symbolov α. Ak pre nejakú dvojicu stav a vstupný symbol existuje viacero symbolov, pre ktoré sa automat nezasekne, potom sa postupne porovnávajú symboly zo vstupnej pásky (začínajúc prvým symbolom v reťazci zľava) s nezasekujúcimi symbolmi pre danú konfiguráciu. To znamená, že ak je n nezasekujúcich symbolov, potom každý symbol z pásky je n-krát porovnávaný, až kým nenájdeme na páske najľavejší nezasekujúci symbol.

Ďalším faktorom, ktorý ovplyvňuje efektivitu algoritmu, je spôsob implementácie vstupnej pásky. Pásku je možné implementovať ako pole alebo zoznam. Výhodou implementácie pomocou poľa je menšia pamäťová náročnosť, ale pri každom odstránení symbolu z pásky je potrebné v cykle presúvať mnoho symbolov. Naopak, pri implementácii pomocou zoznamu

odpadá zložité mazanie symbolu z pásky, ale zvyšuje sa pamäťová náročnosť, kedže každý prvok zoznamu obsahuje odkaz na nasledovníka.

Operáciu odstraňovania symbolu z pásky môžeme z algoritmu odstrániť tým, že zabezpečíme, že symbol nebude prečítaný viac krát ako raz. SDSKA s OPP M_2, ktorý implementujeme, striedavo číta symbol X v stave s a Y v stave p (po prečítaní symbolu X prečíta Y, potom znova X atď.). Pred každým prečítaním symbolu je potrebné daný symbol vyhľadať na páske. Vždy hľadáme najľavejší symbol v reťazci, pre ktorý sa M_2 v budúcom stave nezasekne. V stave s vyľadávame najľavejší nezasekujúci symbol Y a v stave p symbol X. Ak zabezpečíme vyľadávanie nezasekujúceho symbolu vždy od pozície symbolu, ktorý je pravým susedom naposledy prečítaného nezasekujúceho symbolu, nie je potrebné odstraňovať symboly z pásky.

Napr. symbol X bol prečítaný v reťazci na pozícii 10, symbol Y na pozícii 4 a sme v stave p. Nový nezasekujúci symbol X začneme vyhľadávať až od pozície 11. Ak najdeme, automat prejde do stavu s. V stave s je symbol X prečítaný a vyhľadáva sa nezasekujúci symbol Y od pozície 5. Takto pokračujeme až kým sa automat nezasekne. Celý reťazec je spracovaný striedavo iba dvomi prechodmi (v jednom sa vyhľadávajú symboly X a v druhom symboly Y).

SDSKA s OPP M_2 prijímajúci jazyk L_{XY} je popísaný algoritmom 2 na strane 31.

Aplikáciu som sa kvôli prenositeľnosti rozhodol implementovať pomocou knižnice *Qt* v jazyku *C++*. V predošlej časti sme formálne popísali spôsob, akým je možné o nejakom vstupnom reťazci rozhodnúť, či do jazyka L_{CG} patrí alebo nie. Cieľom aplikácie **CpgAnalyser** je demonštrovať prijímaciu silu SDSKA v kombinácii s KP.

Systém, ktorého vstupom je reťazec a výstupom informácia, či daný reťazec patrí alebo nepatrí do jazyka L_{CG}, pozostáva z dvoch podsystémov. Prvým je KP a druhým je SDSKA s OPP. Tento systém je implementovaný statickou triedou `CpgAnalyser`.

KP M_1 z predošlej časti je implementovaný metódou `CpGtoXY(const QString &str, QString &outstr)` typu `bool`, ktorá iba pri úspešnom preklade vracia hodnotu `true`. Vstupný reťazec predávame parametrom `str` a po dokončení prekladu je výstupný reťazec prekladu v parametri `outstr`.

SDSKA s OPP M_2 je implementovaný metódou `AnalyseXY(QString &str)` typu `bool`, ktorá vracia hodnotu `true`, iba ak reťazec `str` patrí do jazyka L_{XY}.

Výpočet systému pozostávajúceho z KP M_1 a SDSKA s OPP M_2 implementuje nasledovná metóda:

```
bool CpgAnalyser::AnalyseString(const QString &str)
{
   QString outstr;
   if (CpGtoXY(str, outstr) && AnalyseXY(outstr))
   {
      return true;
   }
   return false;
}
```

Metóda `AnalyseString` vracia `true` iba vtedy, ak vstupný reťazec `str` patrí do jazyka L_{CG}.

Algoritmus 2: SDSKA AKCEPTUUJÚCI REŤAZCE JAZYKA L_{XY}

Input: inputString
Output: TRUE if inputString $\in L_{XY}$ otherwise FALSE

```
 1: finished = FALSE
 2: currentState = s
 3: currentSymbolIndex = inputString.firstX_fromIndex(0)
 4: if inputString.isEmpty() then
 5:     return FALSE
 6: end if
 7: beginX = currentSymbolIndex + 1
 8: beginY = 0
 9: repeat
10:     switch (currentState)
11:     case s:
12:         currentState = p
13:         currentSymbolIndex = inputString.firstY_fromIndex(beginY)
14:         beginY = currentSymbolIndex + 1
15:         if Y ∉ inputString then
16:             finished = TRUE
17:         end if
18:     case p:
19:         currentState = s
20:         currentSymbolIndex = inputString.firstX_fromIndex(beginX)
21:         beginX = currentSymbolIndex + 1
22:         if X ∉ inputString then
23:             finished = TRUE
24:         end if
25:     end switch
26: until not finished
27: if currentState == p then
28:     return TRUE
29: end if
30: return FALSE
```

Kapitola 6

Záver

Cieľom tejto bakalárskej práce bolo skúmanie skákajúcich konečných automatov, zavedenie skákajúcich konečných prevodníkov a ich aplikácia pri spracovaní neregulárnych lexikálnych štruktúr. Na to, aby sme mohli implementovať skákajúce konečné automaty a prevodníky, bolo potrebné zavedenie ich deterministických verzii.

Ako sa však ukázalo, klasický determinizmus, ktorý bez straty na prijímacej sile u konečných automatov funguje, pri SKA a SKP nepostačoval. Problém spočíval v tom, že v binárnej skokovej relácii SKA a SKP nie je uvedené, kam sa má presunúť čítacia hlava. Kvôli tomu by bolo po prečítaní nejakého symbolu možné skočiť kamkoľvek na vstupnej páske. Preto bolo potrebné zaviesť striktne deterministické verzie SKA a SKP. Striktný determinizmus spočíval v obmedzení binárnej skokovej relácie tak, aby sa čítacia hlava presunula vždy nad najľavejší symbol na vstupnej páske, pre ktorý by sa automat v nasledujúcom stave nezasekol.

V piatej kapitole sme skúmali, aké jazyky je možné prijímať pomocou SDSKA. Ukázali sme, že keď chceme prijímať reťazce, v ktorých sú dôležité pomery počtov výskytov regulárnych podreťazcov, môžeme použiť konečný prevodník na elimináciu týchto podreťazcov a následne pomery overiť pomocou SDSKA, prípadne SDSKP. Túto kombináciu sme použili pri overovaní podmienky pre reťazec DNA, či je výskyt dvojíc CG a GC vyšší ako 50%.

Implementácia kombinácie KP a SDSKA s OPP ukázala, že je možné niektoré bezkontextové a kontextové jazyky prijímať bez použitia počítadiel a zásobníkov. Toto zistenie poukazuje na skutočnosť, že spôsob, akým pracujeme s informáciami v pamäti môže zohrávať významnú úlohu.

Nevýhodou SDSKA môže byť ich implementácia, kedže pred každým skokom je potrebné nájsť na páske najľavejší nezasekujúci symbol. Preto je v budúcnosti potrebné hľadať aj iné riešenia striktného determinizmu, aby sme mohli porovnať, ktoré je efektívnejšie. Nová oblasť výskumu, ktorá sa týka efektivity implementácie skákajúcich konečných automatov je aj zavádzanie viacerých skákajúcich konečných automatov, ktoré budú môcť paralelne spracovávať vstupnú pásku.

Literatura

[1] Aho, A. V.; Ullman, J. D. *The Theory of Parsing, Translation and Compiling: Parsing.* New Jersey: Prentice-Hall, 1972. ISBN 0-13-914556-7.

[2] Gardiner Garden, M.; Frommer, M. CpG Islands in vertebrate genomes. *Journal of Molecular Biology.* 1987, roč. 196, s. 261–282.

[3] Kolář, J., Štěpánková, O.; Chytil, M. *Logika, algebry a grafy.* Praha: SNTL, 1989, s. 7–21.

[4] Martin, J. C. *Introduction to Languages and the Theory of Computiation.* New York: McGraw-Hill, 4th edition, 2011. ISBN 978-0-07-319146-1.

[5] Meduna, A. *Automata and Languages: Theory and Applications.* London: Springer, 2000. ISBN 81-8128-333-3.

[6] Meduna, A.; Zemek, P. Jumping Finite Automata. *International Journal of Foundations of Computer Science.* 2012, roč. 23, č. 7, s. 1555–1578. ISSN 0129-0541.

[7] Rozenberg, G.; Salomaa, A. *Handbook of Formal Languages: Word, Language, Grammar.* Berlin: Springer, 1997. ISBN 3-540-60420-0.

Seznam zkratek

TS Turingov stroj

LOA lineárne ohraničený automat

ZA zásobníkový automat

KA konečný automat

KP konečný prevodník

SKA skákajúci konečný automat

SKP skákajúci konečný prevodník

DSKA deterministický skákajúci konečný automat

DSKP deterministický skákajúci konečný prevodník

SDSKA striktne deterministický skákajúci konečný automat

SDSKP striktne deterministický skákajúci konečný prevodník

SDSKA s OPP striktne deterministický skákajúci konečný automat s oslabenou podmienkou prijímania

Příloha A

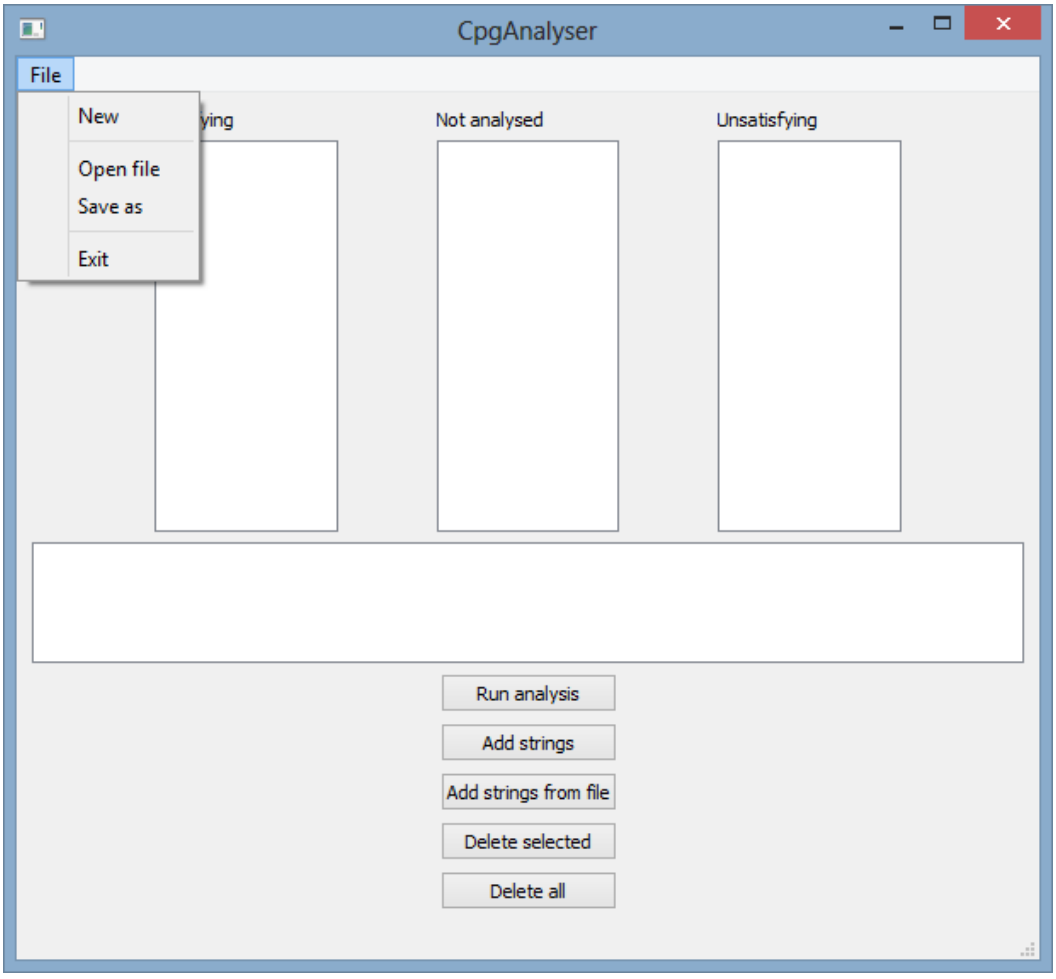
Obsah CD

- **App_CentOS_CVT/CpgAnalyser** spustiteľný binárny súbor aplikácie **CpgAnalyser** pre operačný systém CentOS v CVT
- **App_CentOS_CVT/sources** adresár so zdrojovými kódmi aplikácie **CpgAnalyser**
- **App_CentOS_CVT/examples** adresár s príkladmi reťazcov pre analýzu aplikáciou **CpgAnalyser**
- **bp_pdf** adresár s textom bakalárskej práce vo formáte PDF
- **bp_latex** adresár so zdrojovými kódmi textu bakalárskej práce
- **readme.TXT** textový súbor popisjúci obsah priloženého disku, návod k prekladu a spusteniu aplikácie **CpgAnalyser**

Příloha B

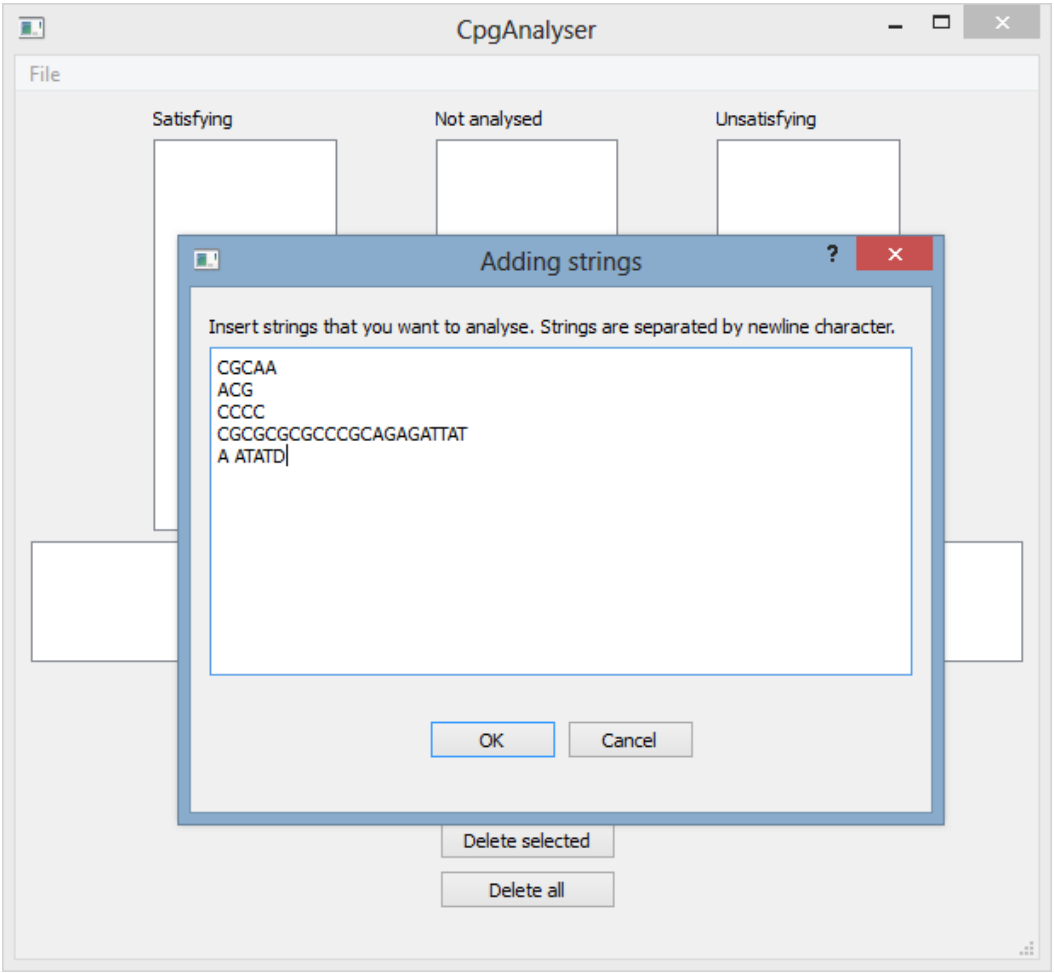
Používateľská príručka

Aplikácia **CpgAnalyser** je implementovaná v jazyku *C++* v knižnici *Qt*. Aplikácia je testovaná pre *Qt 4.7* a vyššie.



Obr. B.1: Menu

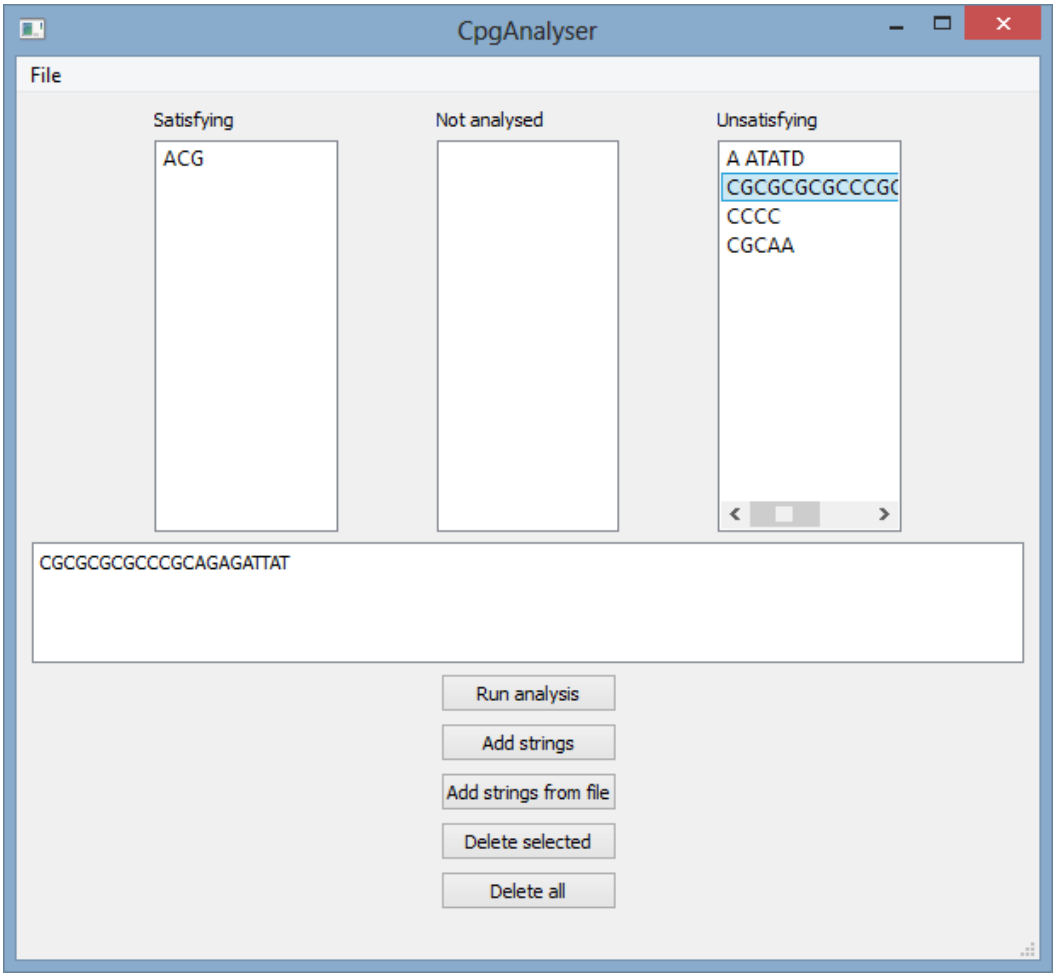
Po spustení aplikácie môžeme pridať nové reťazce do zoznamu neanalyzovaných reťazcov. Kliknutím na tlačítko *Add strings* sa otvorí dialógové okno pre pridávanie nových reťazcov (pozri Obr. B.2). Každý reťazec musí byť na samostatnom riadku. Prázdny riadok sa bude analyzovať ako prázdny reťazec. Reťazce môžeme načítať aj zo súboru typu *CPG* (**.cpg*) dialógovým oknom, ktoré otvoríme tlačítkom *Add strings from file*.



Obr. B.2: Pridávanie reťazcov

Následne môžeme spustiť analýzu tlačítkom *Run analysis*. Výsledkom analýzy je roztriedenie reťazcov do dvoch skupín – spĺňajúce (*Satisfying*) a nespĺňajúce (*Unsatisfying*) podmienku, že výskyt dvojíc *CG* a *GC* v reťazci je vyšší ako 50% (pozri Obr. B.3). Po kliknutí na nejaký reťazec v zoznamoch, sa vybraný reťazec zobrazí v textovom poli. Vybraný reťazec môžeme odstrániť z analýzy tlačítkom *Delete selected*. Všetky reťazce sa odstraňujú tlačítkom *Delete all*.

Výsledky analýzy môžeme uložiť pomocou voľby *Save as* v menu *File* (pozri Obr. B.1). Súbor s výsledkami analýzy je vo formáte *RCPG* (**.rcpg*). Ak chceme načítať súbor s výsledkami nejakej uloženej analýzy alebo súbor s ešte neanalyzovanými reťazcami, použijeme voľbu *Open file* v menu *File*.



Obr. B.3: Výsledok analýzy