ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA PEDAGOGICKÁ
KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

PŘÍPRAVA KOMPONENT PRO E-KURZ KONEČNÉ AUTOMATY A FORMÁLNÍ JAZYKY

BAKALÁŘSKÁ PRÁCE

Luděk Hroch
Informatika se zaměřením na vzdělávání

Vedoucí práce: Mgr. Tomáš Přibáň, Ph.D.

Plzeň 2015

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni 26. června 2015

.....

vlastnoruční podpis

Poděkování

Tímto bych rád poděkoval vedoucímu práce, panu Mgr. Tomášovi Přibáňovi, Ph.D, za odborné vedení a velkou trpělivost.

Originální zadání

Rozhodnutí o náhradním termínu odevzdání

Obsah

ÚVOD

Tématem této bakalářské práce je příprava komponent a vytvoření praktických příkladů pro e-kurz předmětu Konečné automaty a formální jazyky (KAFJB). Jak vyplývá ze zásad pro vypracování, práce je rozdělena na dvě části. V první – teoretické – jsou vysvětleny základní pojmy a definice dle odborné literatury. Druhá, praktická část sestává z vytvoření praktického příkladu k dané látce, jeho popisu a převedení do grafické interaktivní podoby. Jednotlivé kapitoly jsou řazeny podle sylabu předmětu KAFJB a obsahují i slovní popis praktického příkladu. Vypracované příklady a vytvořené komponenty pro e-kurz lze nalézt v příloze bakalářské práce.

S konečnými automaty se běžně setkáváme, aniž si to uvědomujeme. Jejich pomocí se řídí stroje od automatu na kávu přes automatické otevírání dveří po městské křižovatky. Jde o abstraktní matematický model, schopný rozpoznávat určitý jazyk, vhodný pro zpracování počítačovým procesorem. Z tohoto důvodu vnímám jako velice pozitivní, že je výuka konečných automatů zařazena do osnov studia.

Pro vytvoření komponent jsem použil programovací jazyk Scratch. Jedná se o projekt, který byl navržen a je spravován Lifelong Kindergarten Group v Massachusettském technologickém institutu. Software umožňuje programovat v podstatě bez znalostí syntaxe kteréhokoliv z ostatních programovacích jazyků díky vytvořeným blokům, jejichž grafické reprezentace skládáte k jednotlivým objektům podobně, jako při sestavování kostek stavebnice. Použití je proto velice jednoduché. Jednotlivé vlastní výtvory je možné sestavovat online přímo na stránkách scratch.mit.edu nebo s použitím stažitelného offline editoru. Použití i následná prezentace jsou zcela zdarma. Výsledek práce je ukládán pouze s příponou .sb2, rozpoznatelnou softwarem Scratch. Existují však programy třetích stran umožňující převod projektu do běžnějších formátů, jako je .app nebo .swf. Komponenty lze vložit i na internetové stránky pomocí tagu <embed>.

Obrázky grafů, stromů a stavových diagramů byly vytvořeny v Open Source editoru Dia Diagram. Jedná se o program pro kreslení strukturovaných diagramů. V něm jsem využil převážně již vytvořené prvky pro grafickou reprezentaci automatů a možnost exportu souboru jako vektorovou grafiku.

1 ZÁKLADNÍ POJMY

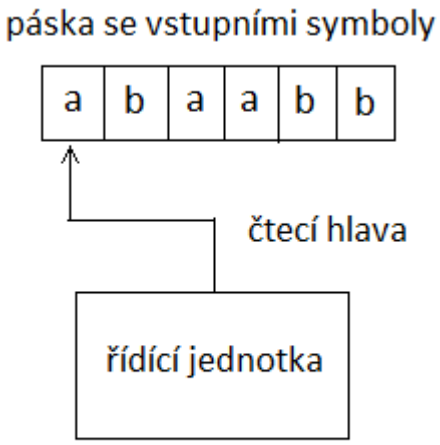
Před samotným studiem předmětu Konečné automaty a formální jazyky je předpokládána znalost několika základních pojmů, značení a názvosloví z ostatních matematických a informatických oborů. Znalost a pochopení těchto pojmů je pro správné porozumění definicím následujících kapitol velice důležité. V případě, že si student není jistý významem definice či pojmů, které obsahuje, může dojít k chybnému pochopení a následně i k chybné interpretaci. V kapitole nejsou základní pojmy vysvětleny tak podrobně, jak by se vysvětlit dali a zasluhovali. Takový popis s příklady a použitím by vydal na vlastní téma bakalářské práce, proto zde lze najít pouze postačující informace pro pochopení kapitol následujících.

1.1 Co je konečný automat

Konečným automatem (zkráceně KA) obecně rozumíme systém, který může nabývat konečně mnoho stavů. Aktuální stav je měněn na základě vnějšího vstupního podnětu právě tehdy, když je jednoznačně určen stav následující. Počet stavů a podnětů je omezen výpočetní silou při zpracovávání tak, aby výsledný stav byl dokončen v daném a námi použitelném čase. (Jančar, 2010, s. 54)

Aby KA mohl pracovat a poskytl nám výslednou informaci, musí obsahovat řídící jednotku s pamětí, ve které je uložen jeden z množiny stavů, a čtecí hlavu, která se pohybuje po pásce, ze které načítá vstupní symboly. Po načtení symbolu dojde k přechodu stavů a posunutí čtecí hlavy. V případě, že dojde k „zaseknutí“ automatu z důvodu, že z daného stavu nemůže do stavu dalšího na základě načteného symbolu nebo při načtení neplatného symbolu, činnost automatu končí. Pokud se po přečtení celé pásky dostal do stavu, který je zároveň konečný, můžeme říci, že slovo bylo automatem přijato. (Černá, Křetínský, Kučera, 2002, s. 9)

Obrázek 1 - Schéma automatu



Zdroj:vlastní

1.2 Množiny

Pojem množina je používán v matematických oborech již dlouhou dobu. Jedná se o kolekci vzájemně odlišitelných objektů. Tyto objekty se nazývají prvky. V souvislosti s konečnými automaty se s množinami setkáme prakticky všude. Ať už se jedná o samotný zápis automatu nebo jeho účel – akceptování určité množiny slov. Množiny obvykle značíme velkým písmenem, například *M*. Prvky pak písmenem malým. Pokud je některý prvek *x*, prvkem množiny *M*, zápis vypadá následovně

$$x \in M$$

V opačném případě prvek množině nenáleží

$$x \notin M$$

Každý z prvků náležící množině je v ní obsažen právě jednou. Prvky nejsou nijak řazeny. Jejich pořadí je proto zaměnitelné, nicméně pro snadnější orientaci je jednodušší prvky seřadit. Množinu, která obsahuje prvky *a*, *b* a *c* lze proto zapsat několika způsoby, které si však budou rovny

$$M = \{a, b, c\} = \{c, a, b\}$$

(Xavier, 2005, p. 1 - 2)

Ze zápisu je patrné, že prvky množiny jsou zapisovány do složených závorek. Pokud množina neobsahuje žádný prvek, jedná se o prázdnou množinu se symbolem $\emptyset$. Uvedená množina M obsahuje tři prvky. Tento počet je její velikost, též nazývaný kardinalita. Zapisuje se s použitím svislítek (pipe)

$$|M|=3$$

Podmnožinou N množiny M pak rozumíme takovou množinu N, jejíž všechny prvky jsou obsaženy v množině M. Zároveň musí množina M obsahovat některé další prvky, jinak by se jednalo o rovnost množin. Zápis pak vypadá následovně

$$N \subseteq M, \text{ ale existuje } x \in M, x \notin N$$

Pomocí operací s množinami můžeme vytvářet množiny nové. Základními operacemi jsou průnik, sjednocení, rozdíl a kartézský součin.

Průnik množin M a N je taková množina, která bude obsahovat veškeré prvky patřících do množin M a N současně

$$M \cap N = \{x : x \in M \text{ a } x \in N\}$$

Sjednocením množin vznikne množina obsahující prvky z množiny M nebo N nebo obou.

$$M \cup N = \{x: x \in M \text{ nebo } x \in N\}$$

Rozdíl množin M a N dá vzniknout množině prvků, které má množina M rozdílné od množiny N

$$M - N = \{x: x \in M \text{ a } x \notin N\}$$

Poslední důležitou operací, kterou zde uvedu, je kartézský součin. Množina vzniklá kartézským součinem obsahuje všechny uspořádané dvojice z množin M a N, kde první prvek patří do množiny M a druhý do N. U uspořádané dvojice prvků je důležité poznamenat, že záleží na jejich pořadí. Dvojice (m, n) je tedy jiná než (n, m), na rozdíl od prvků v množinách M a N, kde na pořadí nezáleželo. Zápis kartézského součinu vypadá následovně

$$M \times N = \{(m, n) : m \in M, n \in N\}$$

(Xavier, 2005, p. 1 - 4)

1.3 Logické spojky a kvantifikátory

Jak je vidět u množin výše, jejich zápis byl proveden slovně s použitím například *a* a *nebo*. V případě zápisu komplikovanějšího výroku je však dobré omezit se na používané logické spojky a kvantifikátory. Nejen, že to zpřehlední zápis, ale bude také všeobecně čitelný i v jiném jazyce. Odpadá také přemýšlení, zda *a*, které jste použil je nějaká neznámá, daný prvek nebo pouze použitá slovní spojka. (Chytil, 1984, s. 10)

Uvedu zde několik logických spojek, se kterými se můžeme v oblasti automatů a formálních jazyků setkat.

Konjunkci výroků zapisujeme znakem & a znamená, že současně platí oba výroky. Při použití definice použité výše, rozdíl množin by byl zapsán takto

$$M - N = \{x : x \in M \,\&\, x \notin N\}$$

Disjunkce zapsaná znakem V symbolizuje výrok *nebo*. Například sjednocení množin zapíšeme

$$M \cup N = \{x : x \in M \vee x \in N\}$$

Implikaci značíme symbolem ⇒ a slovně znamená „z toho vyplývá“. Jedná se o logický důsledek výroku, kde když platí první výrok, platí i výrok druhý.

$$A \Rightarrow B$$

(Chytil, 1984, s. 10)

Ekvivalenci pak můžeme vyjádřit jako „právě tehdy, když“ a má symbol ⇔. Pro názornost můžeme použít následující příklad:

Výrok A: Bude pršet
Výrok B: Vlaštovky létají nízko

Zápis: $(A \Leftrightarrow B) \Leftrightarrow [(A \Rightarrow B) \,\&\, (B \Rightarrow A)]$

Výroky jsou ekvivalentní právě tehdy, když z prvního vyplývá druhý a z druhého první.

Kvantifikátory budeme používat dva – obecný a existenční. Pro obecný kvantifikátor je používán symbol ∀. Slovně bychom ho vyjádřili jako „platí pro vše“. Výrok „Pro všechny prvky *p* z množiny *M*“ bychom pak zapsali jako

$$\forall p \in M$$

Existenční kvantifikátor, jak už název napovídá, lze slovně vyjádřit jako „existuje“. Označujeme ho znakem ∃. Pokud je ke znaku připojen ještě vykřičník, znamená to, že „existuje právě jeden“. Například výrok

$$\exists! p \, (p \in M \,\&\, p \in N)$$

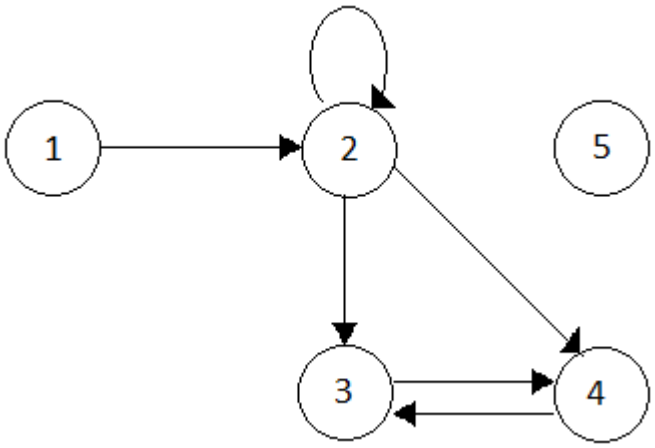
lze interpretovat tak, že existuje právě jeden prvek, který je společný množinám *M* i *N*. (Chytil, 1984, s. 11)

1.4 Grafy

V následujících kapitolách zabývajících se automaty bude řeč také o grafech jako o způsobu jejich zápisu. Používat budeme hlavně tyto dva základní typy grafů – orientované a neorientované.

Orientovaný graf G = (V, H) je takový graf, kde množina *V* je konečná množina vrcholů grafu a množina *H* je množinou hran. Množina hran H ⊆ V x V, jednotlivé hrany zapíšeme jako (u, v) ∈ H a představují uspořádané dvojice, kde *u* je vrchol, ze kterého hrana vychází, a *v* je vrchol, do kterého hrana vstupuje. Ke grafickému znázornění grafů používáme kolečka pro vrcholy a v případě orientovaného grafu, přímky se šipkou pro hrany. Z logiky věci vyplývá, že šipka se zakresluje u vstupujícího vrcholu. Pokud jedna hrana vychází a vstupuje do stejného vrcholu, vzniká smyčka – *u* = *v*. Na obrázku 1 je příklad grafu G = (V, H), kde V = {1, 2, 3, 4, 5} a H = {(1, 2), (2, 2), (2, 3), (2, 4), (3, 4), (4, 3)}. (Pavlásek, 2010, s. 9)

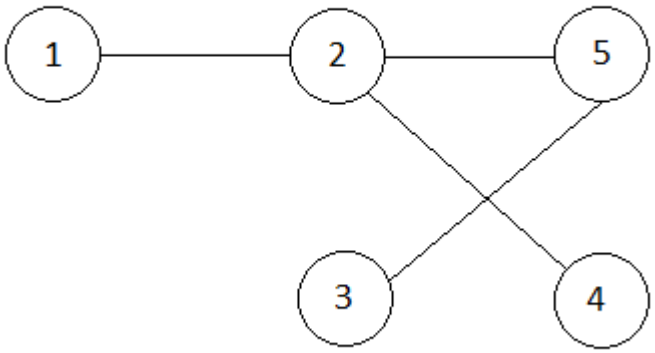
Obrázek 2 - Orientovaný graf



Zdroj: vlastní

Neorientovaný graf se od orientovaného liší v množině hran, což jsou neuspořádané dvojice – (u, v) a (v, u) se tak považují za stejnou hranu. Grafické znázornění je pak obdobné, na konci hran pouze nezakreslujeme šipku. Příklad neorientovaného grafu můžeme vidět na obrázku 3. G = (V, H), kde V = {1, 2, 3, 4, 5} a H = {(2, 1), (2, 4), (2, 5), (5, 3)}. (Pavlásek, 2010, s. 11)

Obrázek 3 - Neorientovaný graf



Zdroj:vlastní

Dále se můžeme setkat s pojmem stupeň vrcholu. U orientovaného grafu se navíc dělí na vstupní a výstupní. Jedná se o počet hran, které do daného vrcholu míří nebo naopak z něj vycházejí. U neorientovaného grafu se jedná o celkový počet hran s daným vrcholem spojeným. (Pavlásek, 2010, s. 12)

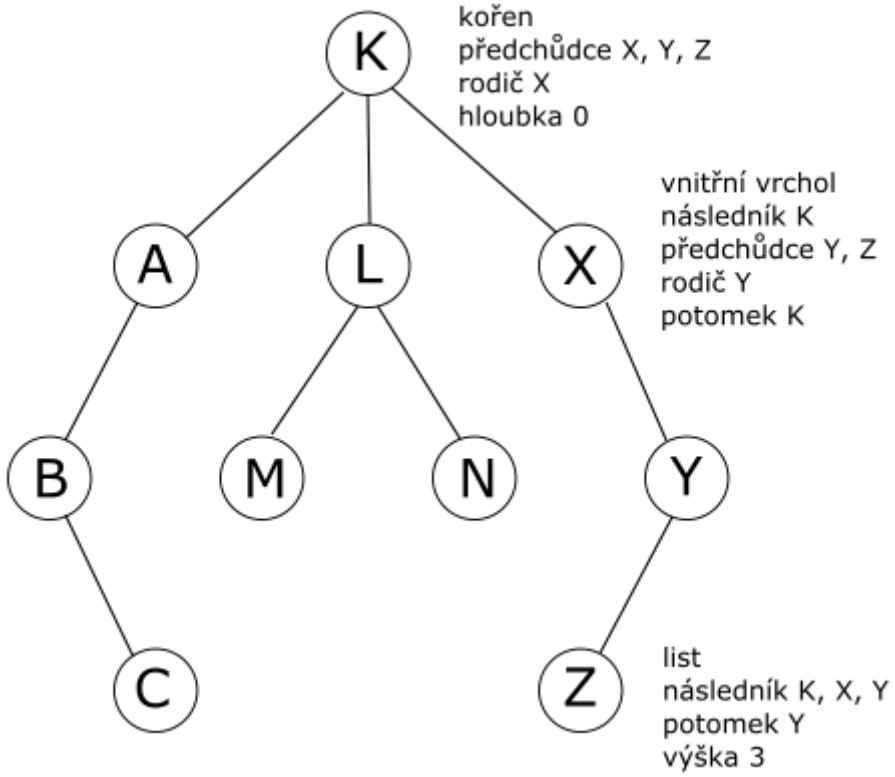
Speciálním příkladem grafu je strom, který musí splňovat určité podmínky:

1) Je souvislý – mezi každými jeho vrcholy ∃! cesta z hran.
2) Je acyklický – neobsahuje cyklus
3) Je neorientovaný
4) ∃! vrchol V, který nazveme kořenem. Je to jediný vrchol, který nemá rodiče

Z podmínek vyplývá, že z kořene *K* vede do vrcholu *Z* právě jedna cesta přes vrcholy *X* a *Y*. Libovolný vrchol před vrcholem *Z* na této cestě je jeho předchůdce. Rodičem je však pouze poslední předchůdce na cestě, v tomto případě vrchol *Y*. Stejně tak jsou vrcholy *X, Y, Z* následníky vrcholu *K*. Vrchol *Z* je zároveň potomkem vrcholu *Y*. Vrchol, který nemá žádné potomky, se nazývá list, vrcholům mezi kořenem a listy říkáme vnitřní vrcholy. (Chytil, 1984, s. 13)

U stromu můžeme určit několik vlastností, jako je hloubka nebo výška. Hloubkou vrcholu je délka cesty od kořene, kořen tak má hloubku rovnou nule. Výška je pak nejdelší cestou z vrcholu do jednoho z jeho následníků. Maximální výška stromu, tedy výška od kořene k nejvzdálenějšímu listu, je rovna největší hloubce ve stromu. (Chytil, 1984, s. 20)

Obrázek 4 - Příklad stromu



Zdroj: vlastní

1.5 Reprezentace konečných automatů

Z uvedených informací bychom už měli být schopni zapsat a porozumět všem možnostem reprezentace konečného automatu. Reprezentaci KA je možno provést výčtem prvků, stavovým diagramem, stromem nebo tabulkou. Pro jednotlivé ukázky zápisu použiji hodnoty ze zadání praktického příkladu z kapitoly 2.2.

Výčet prvků automatu vycházejícího z definice automatu $A = (Q, \Sigma, \delta, q_0, F)$ je jeden z nejjednodušších způsobů reprezentace. Každý člen z každé množiny je vypsán a jsou uvedena pravidla přechodových funkcí. Pro příklad křižovatky bude výčet prvků následující:

$A = (\{q_0, q_{11} .. q_{17}, q_{21} .. q_{27}\}, \{0, 1, 2\}, \delta, q_0, q_0)$ kde pro δ platí:

$\delta(q_0, 0) = q_{11}$ $\delta(q_{21}, 0) = q_{22}$ $\delta(q_{11}, 1) = q_0$ $\delta(q_{11}, 2) = q_{22}$

$\delta(q_{11}, 0) = q_{12}$ $\delta(q_{22}, 0) = q_{23}$ $\delta(q_{17}, 1) = q_{11}$ $\delta(q_{14}, 2) = q_{25}$

$\delta(q_{12}, 0) = q_{13}$ $\delta(q_{25}, 0) = q_{26}$ $\delta(q_{21}, 1) = q_0$ $\delta(q_{15}, 2) = q_{26}$

$\delta(q_{13}, 0) = q_{14}$ $\delta(q_{26}, 0) = q_{27}$ $\delta(q_{27}, 1) = q_{21}$ $\delta(q_{16}, 2) = q_{27}$

$\delta(q_{14}, 0) = q_{15}$ $\delta(q_{17}, 2) = q_{21}$

$\delta(q_{15}, 0) = q_{16}$ $\delta(q_{23}, 2) = q_{14}$

$\delta(q_{16}, 0) = q_{17}$

Každá přechodová funkce δ udává aktuální stav a znak z abecedy, po jehož načtení se automat přesune do nového stavu (hodnota za rovnítkem). (Kozel, 2008)

Výše vypsané přechodové funkce můžeme přehledně prezentovat v tabulce přechodů. Jedná se o další způsob zápisu konečného automatu. V záhlaví jednotlivých sloupců se nacházejí symboly vstupní abecedy, v záhlaví řádků pak jednotlivé stavy automatu. Průnik řádku a sloupce obsahuje výsledek přechodové funkce. Není-li funkce pro některý ze stavů definována, použijeme znak -. Počáteční stav označujeme symbolem → u příslušného řádku tabulky, koncový stav pak symbolem ←. Pokud automat má počáteční a koncový stav shodný, jako v našem případě, použijeme znak ↔. (Kozel, 2008)

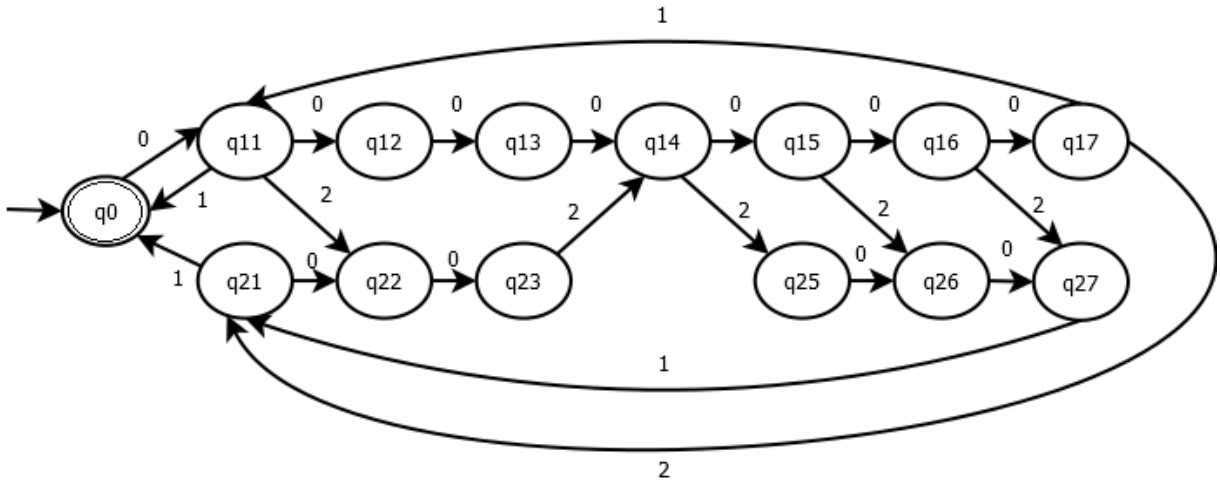
Tabulka 1 - Tabulka přechodů

	0	**1**	**2**
↔ q_0	q_{11}	-	-
q_{11}	q_{12}	q_0	q_{22}
q_{12}	q_{13}	-	-
q_{13}	q_{14}	-	-
q_{14}	q_{15}	-	q_{25}
q_{15}	q_{16}	-	q_{26}
q_{16}	q_{17}	-	q_{27}
q_{17}	-	q_{11}	q_{21}
q_{21}	q_{22}	q_0	-
q_{22}	q_{23}	-	-
q_{23}	-	-	q_{14}
q_{25}	q_{26}	-	-
q_{26}	q_{27}	-	-
q_{27}	-	q_{21}	-

Zdroj: vlastní

Graficky nejnázornější reprezentací KA je stavový diagram. Jedná se o orientovaný ohodnocený graf, ve kterém jsou vrcholy jednotlivé stavy automatu a hrany vyobrazují přechody. Hrany jsou kresleny pouze mezi vrcholy, pro které existuje přechodová funkce. Ohodnocení hrany je pak symbol ze vstupní abecedy konečného automatu. Počáteční stav je stejně jako v tabulkové reprezentaci značen šipkou → vstupující do daného vrcholu. Konečný stav je vyobrazen pomocí druhé soustředné kružnice vrcholu. (Kozel, 2008)

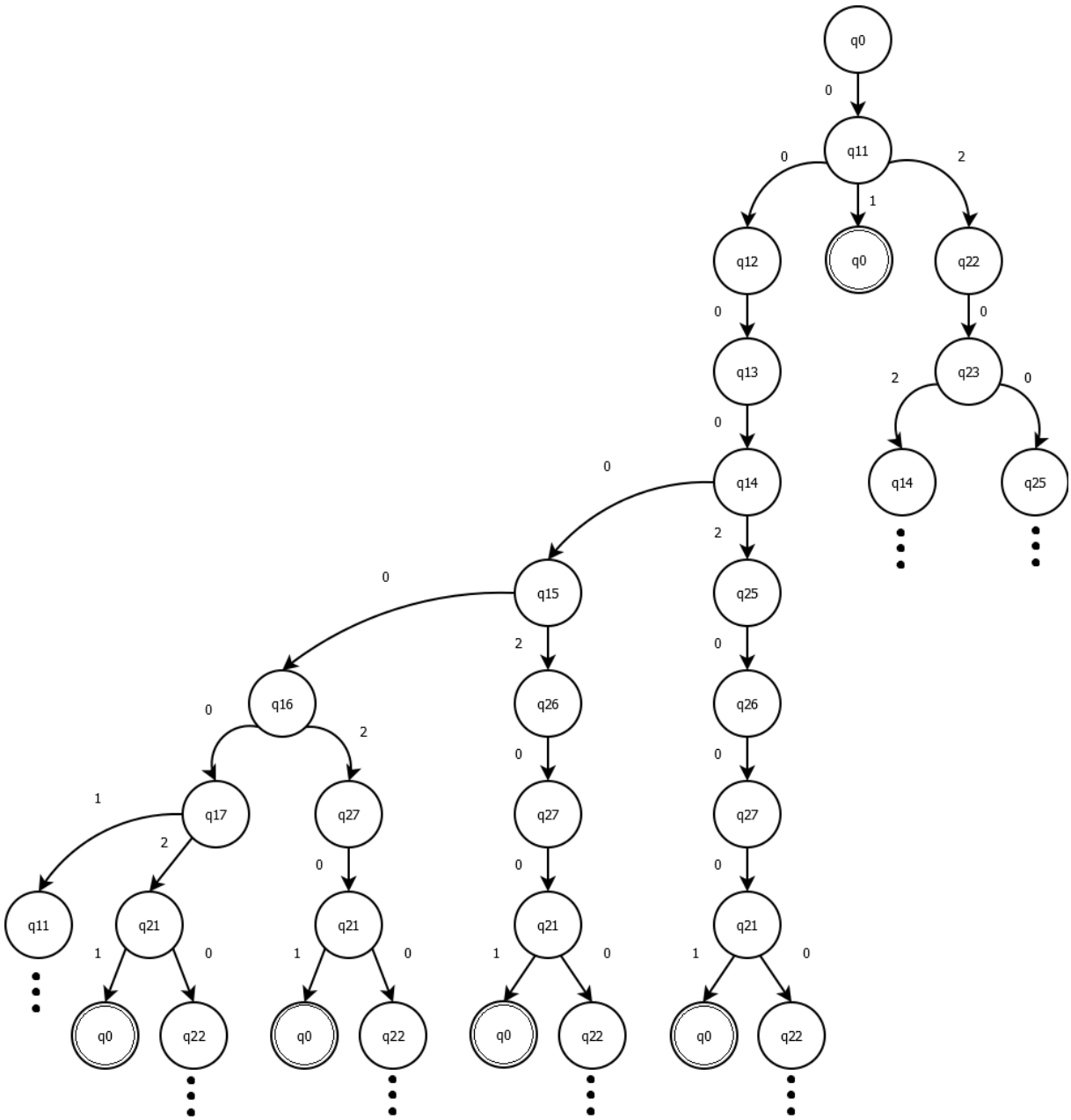
Obrázek 5 - Stavový diagram



Zdroj: vlastní

Poslední z možností reprezentace konečného automatu je reprezentace stavovým stromem. Kořen stromu je počáteční stav, listy mohou být pouze stavy koncové. Ty jsou opět značeny dvěma soustřednými kružnicemi. Slovo přijímané automatem je pak celá větev symbolů od kořene po list. Nevýhodou stavového stromu je, že může často být nekonečný. V následujícím zpracování křižovatkového automatu je opakování znázorněno symbolem tří teček, kdy je další postup ve vrcholech shodný s již existujícím postupem ve stromu, a proto je jeho zakreslování nadbytečné. (Kozel, 2008)

Obrázek 6 - Stavový strom



Zdroj: vlastní

2 DETERMINISTICKÝ KONEČNÝ AUTOMAT BEZ VÝSTUPU

2.1 Definice

Konečný automat jako formální výpočetní prostředek se dělí na dva druhy – deterministický a nedeterministický. V této kapitole se budeme zabývat konečným automatem deterministickým (DKA), a to konkrétně automatem bez výstupu.

Konečný automat *A* je uspořádaná pětice prvků $A = (Q, \Sigma, \delta, q_0, F)$ kde:

- Q je konečná neprázdná množina stavů
- Σ je konečná neprázdná množina vstupních symbolů = vstupní abeceda
- δ je přechodová funkce $\delta: Q \times \Sigma \rightarrow Q$
- q_0 je počáteční stav, $q0 \in Q$
- F je množina koncových stavů $F \subseteq Q$

U množiny koncových stavů je možné, aby byla prázdná ($F = \emptyset$), na rozdíl od množiny stavů a vstupních symbolů. Z toho vyplývá, že přijímané symboly nikdy neskončí v koncovém stavu automatu, tudíž žádné slovo nebude automatem přijato. (Hopcroft, Motwany, Ullman, 2001, p. 46)

2.2 Praktický příklad

Jako příklad komplexního deterministického konečného automatu jsem zvolil simulaci jednoproudé křižovatky ve tvaru T s přechodem pro chodce. Křižovatka obsahuje dva semafory pro automobily a jeden pro chodce. Pro zvýšení reálnosti křižovatky je implementováno i tlačítko pro chodce. Chodec, který stiskne tlačítko, se tak rychleji dočká možnosti bezpečného přechodu.

Změna semaforů na křižovatce je řízena dokola se opakujícími signály, které nabývají logických hodnot *0, 1, 2* a určují změnu světel všech tří semaforů zároveň, čímž je jednoduchým způsobem vyřešena synchronizace.

V této simulaci se nachází celkem 14 stavů. Každý stav obsahuje informaci o tom, jaké světlo svítí na všech semaforech a časový interval *t*, po který je tento stav aktivní. Po stisku tlačítka pro chodce dojde ke zkrácení základních časových intervalů t_1 na kratší intervaly t_2, a tím se zajistí chodci kratší čekací doba na semaforech. Časový interval v simulaci je pro t_1 pět vteřin ve stavech, kdy svítí červená nebo zelená, a dvě vteřiny pro oranžovou. V rychlejší části po stisknutí tlačítka pro chodce jsou tyto intervaly zkráceny na tři a jednu vteřinu s označením t_2. Přehledně to shrnuje tabulka 2.

Tabulka 2 - Vlastnosti semaforů

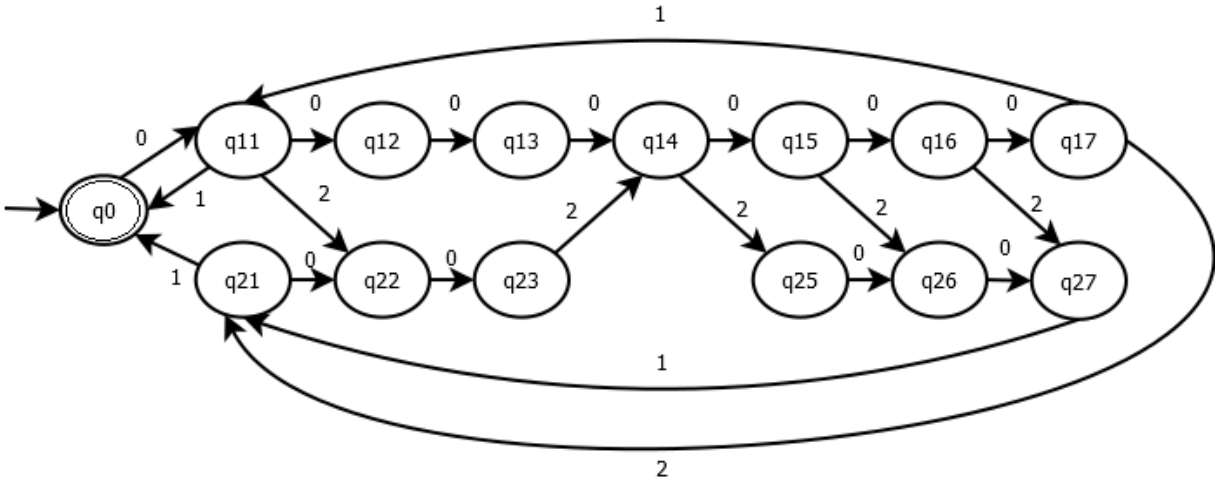
	Semafor A	**Semafor B**	**Semafor C**	**Čas t**
$\mathbf{q_0}$	oranžová	oranžová	vypnuto	t_0
$\mathbf{q_{11}}$	červená	červená	červená	t_1
$\mathbf{q_{12}}$	oranžová	červená	zelená	t_1
$\mathbf{q_{13}}$	zelená	červená	zelená	t_1
$\mathbf{q_{14}}$	oranžová	červená	červená	t_1
$\mathbf{q_{15}}$	červená	oranžová	červená	t_1
$\mathbf{q_{16}}$	červená	zelená	červená	t_1
$\mathbf{q_{17}}$	červená	oranžová	červená	t_1
$\mathbf{q_{21}}$	červená	červená	červená	t_2
$\mathbf{q_{22}}$	oranžová	červená	zelená	t_2
$\mathbf{q_{23}}$	zelená	červená	zelená	t_2
$\mathbf{q_{25}}$	červená	oranžová	červená	t_2
$\mathbf{q_{26}}$	červená	zelená	červená	t_2
$\mathbf{q_{27}}$	červená	oranžová	červená	t_2

Zdroj: vlastní

Relativně velký počet stavů je dán zaprvé zajištěním bezpečnosti provozu křižovatky, a to konkrétně tak, aby na semaforech pro automobily nebyly v obou případech oranžová nebo zelená světla naráz, a také, aby na semaforu pro chodce nebyla zelená v době projíždění křižovatkou ze strany semaforu *b*. Druhým důvodem je zakomponování části spuštěné po stisknutí tlačítka na přechodu pro chodce, kde jsou sice stavy semaforů

shodné, ale liší se v době trvání. Řešení příkladu sestává ze dvou konečných automatů, vzájemně propojených, se společným vstupním a výstupním stavem. Ten je značen jako q_0 a je aktivní několik vteřin na začátku simulace (časová proměnná t_0), na semaforech bliká oranžové světlo a semafor pro chodce je vypnutý. Grafická reprezentace stavů křižovatky je na obrázku 7.

Obrázek 7 - Stavový diagram

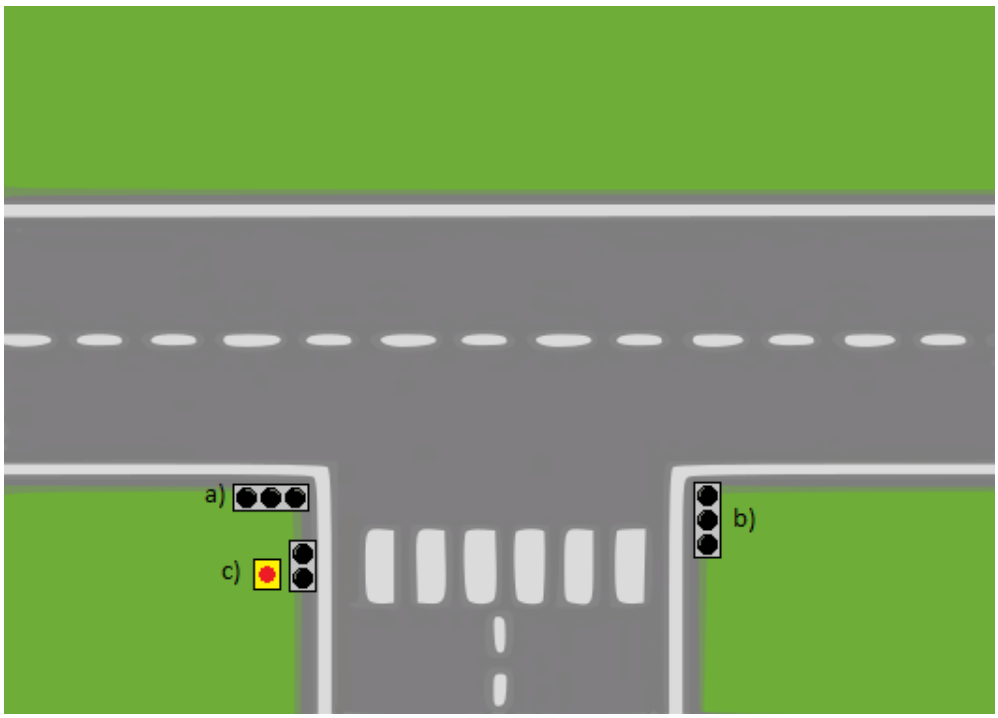


Zdroj: vlastní

Z tabulky a grafu je vidět, že při stisknutí tlačítka pro chodce dojde k přechodu do „spodní“ části automatu, která má kratší časové intervaly pro přechod mezi jednotlivými stavy. Tím je zajištěna kratší čekací doba chodce na červené. Dále vidíme, že přechod není možný ze stavu q_{12} a q_{13}. Z tabulky lze vyčíst, že v těchto stavech svítí na přechodu pro chodce zelená, není proto nutné urychlovat cyklus. Logické také je, že opakovaný stisk tlačítka nemá na další urychlení cyklu žádný vliv.

Na obrázku 8 se nachází výstup zpracování komponenty s dodatečně označenými semafory – jeden pro průjezd rovně (a), druhý odbočovací doleva a doprava (b) a třetí pro chodce s tlačítkem pro urychlení cyklu (c). Podle konkrétního stavu semaforů projíždí křižovatku automobil či přechází člověk na přechodu.

Obrázek 8 - Křižovatka



Zdroj: vlastní

3 KLASIFIKAČNÍ AUTOMAT

3.1 Definice

Deterministický klasifikační automat se od klasického konečného automatu liší v množině koncových stavů, konkrétně se jedná o její rozklad na podmnožiny - třídy. Několik koncových stavů, které jsou součástí jedné podmnožiny, tak automat zařadí do stejné třídy. (Kocur, 2000)

Klasifikační automat je tedy uspořádaná pětice prvků $A = (Q, \Sigma, \delta, q_0, \{Q_i\})$, kde

- Q je konečná neprázdná množina stavů
- Σ je konečná neprázdná množina vstupních symbolů = vstupní abeceda
- δ je přechodová funkce $\delta: Q \times \Sigma \rightarrow Q$
- q_0 je počáteční stav, $q0 \in Q$
- $\{Q_i\}$ je rozklad množiny koncových stavů do tříd $Q_1, Q_2, .. , Q_n$

Pro rozklad $\{Q_i\}$ dále platí, že:

- $Q_1 \cup Q_2 \cup .. \cup Q_n = \{Q_i\}$
- $Q_i \cap Q_j = \emptyset, \forall\, i \neq j \quad i,j = 1, 2, .., n$

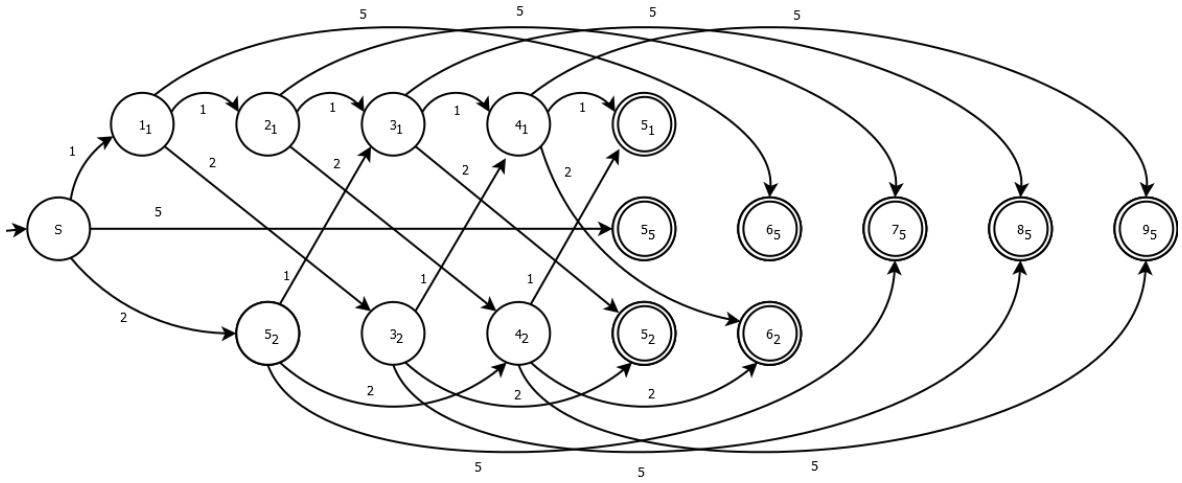
Sjednocením všech tříd Q_n dostaneme množinu koncových stavů Q jako u konečného automatu. Zároveň však pro každé dvě různé třídy musí být jejich průnik roven prázdné množině, nemůže tedy být jeden koncový stav ve více třídách. (Kocur, 2000)

3.2 Praktický příklad

Při ukázce praktického příkladu deterministického konečného automatu se často setkáváme s automatem na kávu. Po menší modifikaci můžeme takový automat použít pro ukázku rozdělení koncových stavů do tříd podle toho, kolik peněz bude takový automat

vracet při objednávce. Automat bude přijímat mince v hodnotě *1*, *2* a *5* korun. Cena kávy je pět korun. K výdeji kávy tedy dojde, pokud vhozené mince budou v minimální částce 5 Kč. To se může stát třemi způsoby – v automatu budou již 4 koruny a vhodíme libovolnou minci, automat bude obsahovat tři koruny a my vhodíme minimálně dvoukorunu nebo kdykoliv vhodíme pětikorunu. Všechny následné stavy jsou koncové, dojde tedy k výdeji kávy a zároveň může dojít k vrácení příslušné částky. Stavy vracející stejnou částku pak můžeme zařadit do stejné třídy. Automat můžeme reprezentovat následujícím stavovým diagramem.

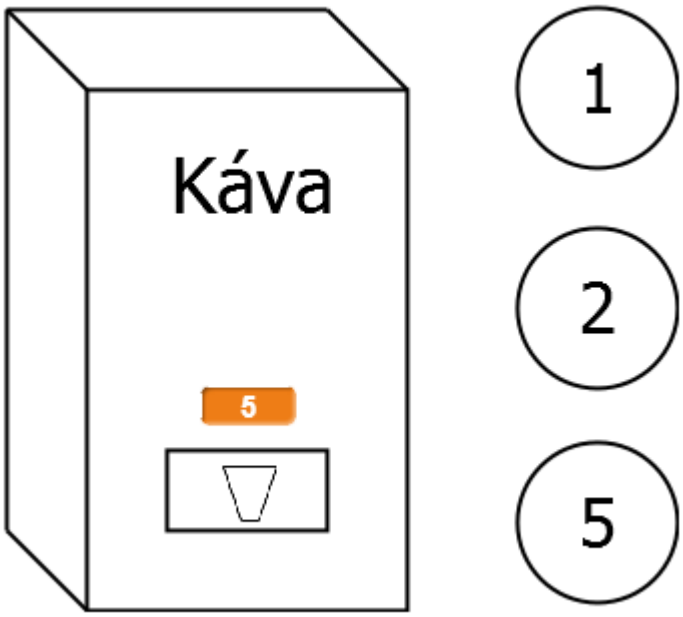
Obrázek 9 - Stavový diagram



Zdroj: vlastní

Třídy se shodnými vlastnosti jsou $Q_1 = \{5_1, 5_2, 5_5\}$, $Q_2 = \{6_2, 6_5\}$, $Q_3 = \{7_5\}$, $Q_4 = \{8_5\}$, $Q_5 = \{9_5\}$. Žádný ze stavů není ve dvou třídách a všechny stavy dávají dohromady množinu koncových stavů – podmínka rozkladu je splněna. Stavy ze stejné třídy v praktickém příkladu na obrázku 10 vracejí stejnou částku.

Obrázek 10 - Automat na kávu



Zdroj: vlastní

4 NEDETERMINISTICKÝ AUTOMAT

4.1 Definice

Každý zdánlivě jednoduchý deterministický konečný automat splňující podmínku v přijímaném slovu, ať už se jedná o prefix, podslovo nebo sufix, se může stát velice komplikovaným a časově náročným jak na kompozici, tak na ověření funkčnosti. Do této chvíle měl KA jasně daný postup po načtení vstupního znaku – choval se jednoznačně. Pokud to však přímo nevyžaduje zadání problému, můžeme si práci ulehčit zavedením nedeterminismu. Nedeterministický konečný automat (zkráceně NKA) může ve svých stavech volit svůj další krok. Jedná se v podstatě o zobecnění pojmu konečného automatu. Stav, do kterého DKA přejde, je u NKA nahrazen množinou stavů. To znamená, že například ve stavovém diagramu bude z jednoho stavu vycházet stejně ohodnocená hrana do dvou či více stavů. Druhým rozdílným rysem je obecně více prvková množina počátečních stavů. Tyto automaty lze analogicky reprezentovat výčtem, tabulkou, stavovým diagramem a stromem. Přijímané slovo nedeterministickým automatem je takové, které odpovídá nějaké cestě z některého počátečního do některého koncového stavu. Na přijetí slova nic nemění skutečnost, že některé z cest v koncovém stavu nekončí. (Chytil, 1984, s. 70 - 71)

Na každý konečný automat lze pohlížet jako na speciální případ nedeterministického konečného automatu. Proto můžeme ke každému NKA vytvořit systém, který je realizací jistého konečného automatu, rozpoznávající stejný jazyk – existuje ekvivalentní DKA (převod a praktický příklad převodu je v následující kapitole). (Chytil, 1984, s. 71)

Nedeterministický konečný automat je tedy podle definice uspořádaná pětice prvků $A = (Q, \Sigma, \delta, I, F)$, kde

- Q je konečná neprázdná množina stavů
- Σ je konečná neprázdná množina vstupních symbolů = vstupní abeceda
- δ je přechodová funkce $\delta: Q \times \Sigma \rightarrow 2^Q$
- I je množina počátečních stavů, $I \subseteq Q$
- F je množina koncových stavů, $F \subseteq Q$

Matematicky můžeme přechodovou funkci zapsat jako $\delta(q_i, a) = \{q_1, q_2, .. , q_n\}$, kde automat nacházející se ve stavu q_i přečte vstupní symbol a, díky kterému se dostane do jednoho ze stavů za rovnítkem. (Chytil, 1984, s. 71)

4.2 Praktický příklad

Jako praktický příklad nedeterministického konečného automatu jsem si vybral zpracování matematického hlavolamu Hanojské věže. Tento hlavolam vymyslel francouzský matematik Edouard Lucas v roce 1883. Skládá se ze tří věží. Na začátku je na jedné z nich několik kotoučů s různými poloměry seřazených odspodu od největšího po nejmenší. Úkolem je přemístit kotouče z jedné věže na jinou tak, že je možné vždy přemisťovat pouze jeden kotouč, a to vždy ten vrchní. Není dovoleno umístit větší kotouč na menší. Matematicky je dokázáno, že pro přesun n kotoučů je nutné provést v ideálním případě 2^n-1 tahů. (hlavolamy.info, 2012)

V mém příkladu jsem se rozhodl pro 3 kotouče. To se na první pohled může zdát jako nízké číslo, ale minimální počet tahů je podle vzorečku 2^3-1 = 7. Celkový počet stavů nedeterministického automatu, kterým tento hlavolam budu zpracovávat, však musí umět reagovat na všechny tahy, ne jen na vítězné, a těch už je 27, včetně jednoho vstupního a dvou koncových, čili téměř dvakrát tolik co u příkladu deterministického konečného automatu.

Pro reprezentaci NKA Hanojských věží použiji tabulku přechodů a stavový diagram. Hodnoty v záhlaví sloupců jsou znaky vstupní abecedy, symbolizující velikost kotoučů, kde jednička je pro první, nejmenší kotouč, dvojka pro střední kotouč a poslední, největší kotouč je označen číslem tři. Počáteční a koncový stav je stejně jako u DKA označen šipkami. Jak je z tabulky patrné, nejmenší kotouč lze přemístit vždy na jednu ze dvou zbývajících věží nehledě na jejich momentální stav – prázdná, se středním, velkým nebo oběma kotouči. Nikdy se také nenaskytne možnost přesunu druhého a třetího kotouče zároveň. Tím se přímo nabízí možnost úpravy a minimalizace automatu na pouhé dva znaky vstupní abecedy. Pro lepší přehlednost a ilustraci jsem se však rozhodl ponechat automat v tomto stavu. U výstupních stavů je hlavolam vyřešen – kotouče jsou přesunuty na druhou či třetí věž.

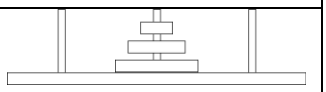
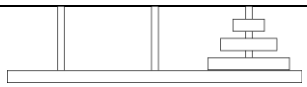
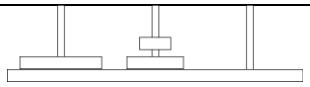
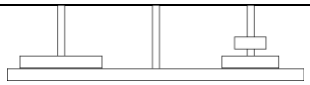
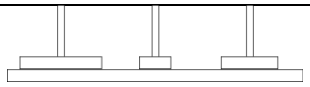
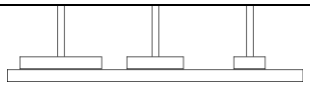
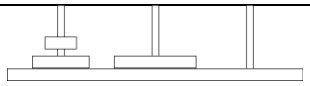
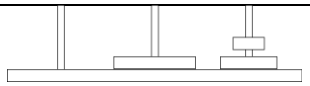
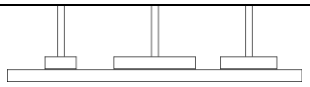
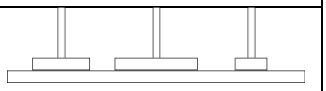
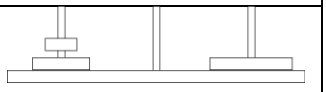
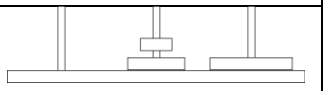
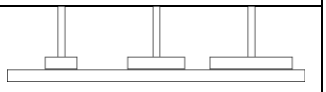
Tabulka 3 - Přechodové funkce

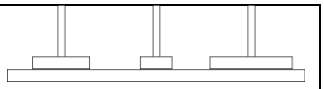
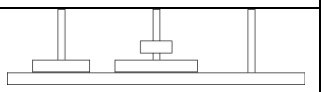
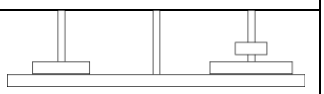
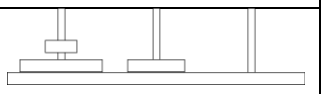
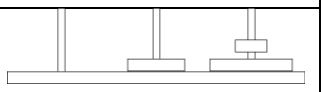
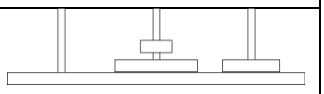
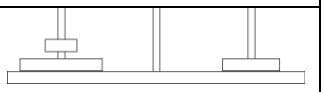
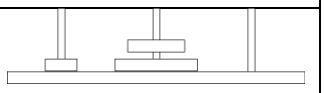
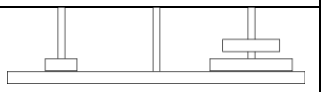
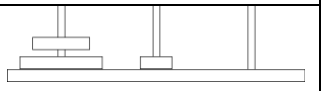
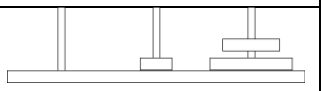
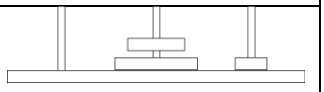
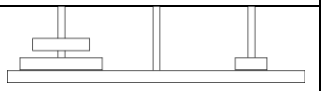
	1	**2**	**3**
→ **q_0**	q_{23} , q_{26}	-	-
← **q_1**		-	-
← **q_2**		-	-
q_3	q_6 , q_{17}	-	q_{12}
q_4	q_5 , q_{20}	-	q_8
q_5	q_4 , q_{20}	q_{23}	-
q_6	q_3 , q_{17}	q_{26}	-
q_7	q_{10} , q_{15}	-	q_{11}
q_8	q_9 , q_{19}	-	q_4
q_9	q_8 , q_{19}	q_{21}	-
q_{10}	q_7 , q_{15}	q_{25}	-
q_{11}	q_{14} , q_{16}	-	q_7
q_{12}	q_{13} , q_{18}	-	q_3
q_{13}	q_{12} , q_{18}	q_{22}	-
q_{14}	q_{11} , q_{16}	q_{24}	-
q_{15}	q_7 , q_{10}	q_{19}	-
q_{16}	q_{11} , q_{14}	q_{18}	-
q_{17}	q_3 , q_6	q_{20}	-
q_{18}	q_{12} , q_{13}	q_{16}	-
q_{19}	q_8 , q_9	q_{15}	-
q_{20}	q_4 , q_5	q_{17}	-
q_{21}	q_1 , q_{25}	q_9	-
q_{22}	q_2 , q_{24}	q_{13}	-
q_{23}	q_0 , q_{26}	q_5	-
q_{24}	q_2 , q_{22}	q_{14}	-
q_{25}	q_1 , q_{21}	q_{10}	-
q_{26}	q_0 , q_{23}	q_6	-

Zdroj: vlastní

V následující tabulce je u každého stavu zobrazena jeho grafická reprezentace v animaci. Tyto statické obrázky poslouží jako základ pro práci v programu Scratch, kde se budou měnit v závislosti na tahu hráče.

Tabulka 4 - Vyobrazení stavů

q_0	
q_1	
q_2	
q_3	
q_4	
q_5	
q_6	
q_7	
q_8	
q_9	
q_{10}	
q_{11}	
q_{12}	
q_{13}	

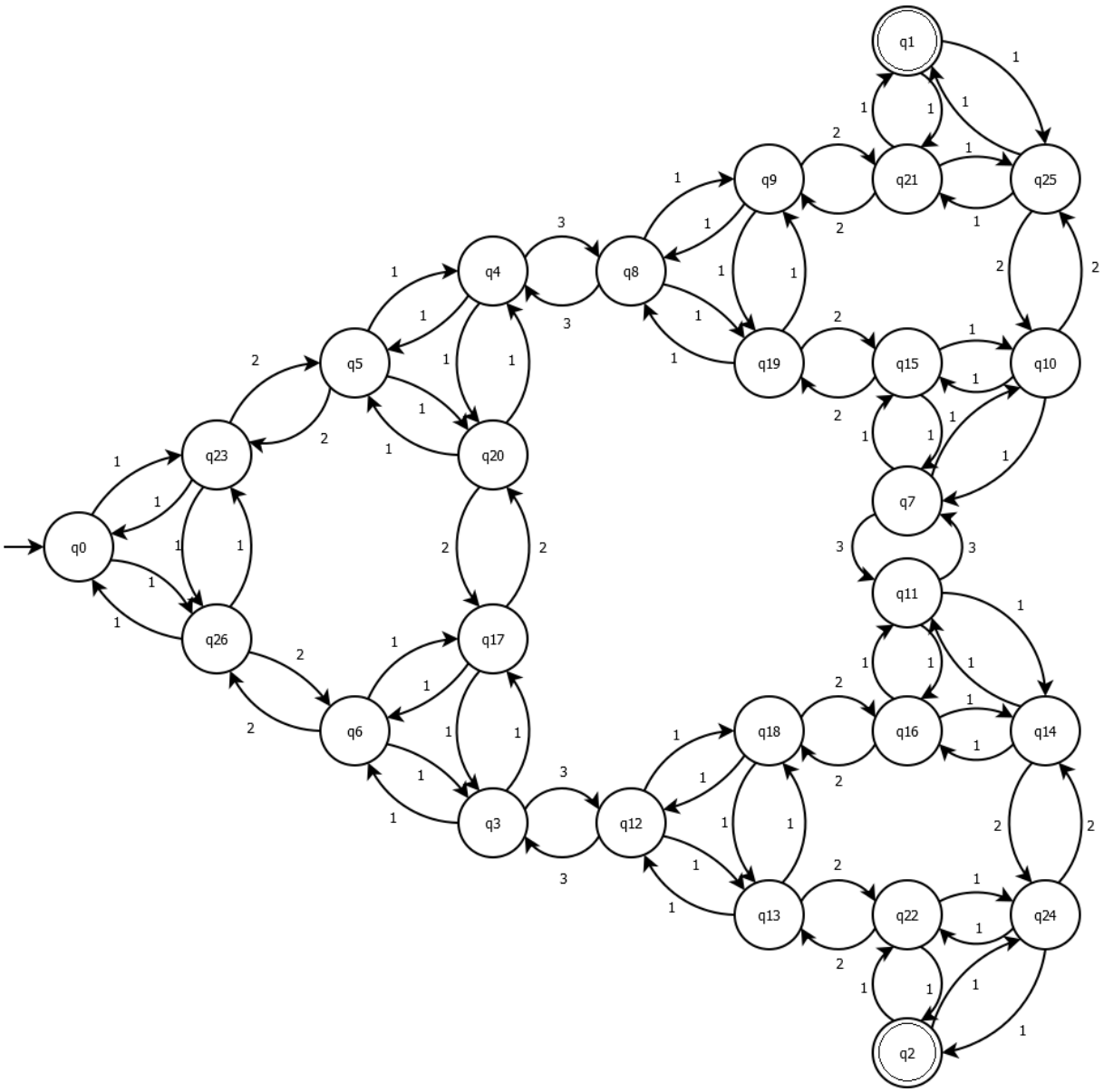
q_{14}	
q_{15}	
q_{16}	
q_{17}	
q_{18}	
q_{19}	
q_{20}	
q_{21}	
q_{22}	
q_{23}	
q_{24}	
q_{25}	
q_{26}	

Zdroj: vlastní

Ve stavovém diagramu vidíme počáteční a koncový stav odlišný od ostatních stavů. Do počátečního vstupuje neohodnocená hrana, koncové stavy jsou ohraničeny dvěma soustřednými kružnicemi. Na první pohled je patrné, že se jedná o nedeterministický

konečný automat, protože v podstatě ze všech stavů vychází hrany ohodnocené stejným symbolem vstupní abecedy, konkrétně jedničkou.

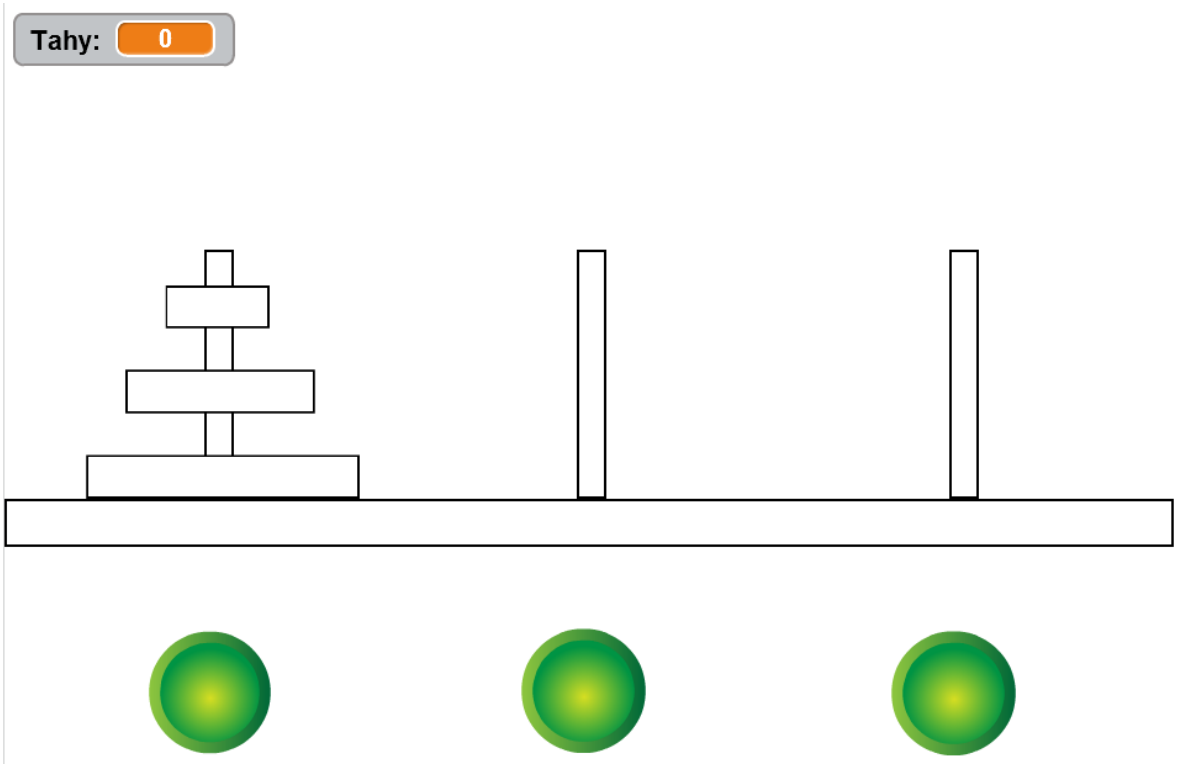
Obrázek 11- Stavový diagram NKA



Zdroj: vlastní

Změna kotoučů ve zpracovaném hlavolamu je prováděna pomocí tlačítek pod jednotlivými věžemi. Po stisknutí tlačítka příslušné věže se vždy nejvrchnější kotouč přesune mimo věž, stisknutím dalšího tlačítka se kotouč na věž umístí, pokud toto umístění umožňují pravidla. V levém horním rohu je umístěna praktická proměnná ukazující počet tahů v aktuálním řešení hlavolamu (obrázek 12)

Obrázek 12 - Hanojské věže



Zdroj: vlastní

5 KONFIGURACE A PŘEVOD

5.1 Konfigurace automatu

Konfigurace automatu je jeden z možných způsobů, jak definovat jazyk akceptovatelný konečným automatem. Konfigurace automatu *A* přesně popisuje momentální stav výpočtu, který automat na daném slově provádí. Obsahuje úplnou informaci, která je potřebná pro jeho další pokračování. „Programem“ pro tento výpočet je samozřejmě přechodová funkce. (Černá, Křetínský, Kučera, 2002, s. 14)

Odborná literatura definuje konfiguraci konečného automatu jako každou uspořádanou dvojici (q, w) ∈ Q x Σ* automatu $A = (Q, \Sigma, \delta, q_0, F)$, kde *q* je aktuální stav automatu a *w* je zatím nepřečtená část řetězce. Na množině všech konfigurací automatu *A* zavedeme binární relaci krok výpočtu, označovanou ⊢, pomocí předpisu (q, aw) ⊢ (p, w), který je ekvivalentní k zápisu přechodové funkce δ(q, a) = p. (Kocur, 2000)

Konfigurace konečného automatu můžeme obecně rozdělit na počáteční a koncovou. Počáteční konfigurace je uspořádaná dvojice (q_0, w_0), přičemž stav q_0 je počáteční a w_0 je celé vstupní slovo. Koncová konfigurace konečného automatu je uspořádaná dvojice (q_n, ε), kde q_n je jeden z koncových stavů (platí $q_n \in F$) a ε je prázdný řetězec. (Kocur, 2000)

5.2 Převod nedeterministického konečného automatu na deterministický

Jak už bylo řečeno, ke každému nedeterministickému konečnému automatu existuje ekvivalentní deterministický automat. Návrh DKA rozpoznávající zadáním omezený jazyk může být velice komplikované. U NKA nejsme omezováni a automat lze často sestavit velmi snadno a intuitivně. Pokud tedy máme sestavený NKA, potřebujeme znát mechanismus, jak tento druh automatu převést na deterministický.

K NKA $A = (Q, \Sigma, \delta, I, F)$ sestrojíme DKA $A' = (Q', \Sigma, \delta', I, F')$, kde

- Q‘ = *P*(Q) je množina všech podmnožin stavů Q
- I ∈ Q‘ je počáteční stav

- $F' = \{K \in Q' \mid K \cap F \neq \emptyset\}$
- $\delta' : Q' \times \Sigma \rightarrow Q'$

Sestrojený DKA tak obsahuje více stavů, než NKA, protože jeho množina stavů *Q'* je množinou všech podmnožin původního automatu. Matematickým důkazem nám vychází, že těchto stavů je 2^n, kde *n* je původní počet stavů. Z původních například deseti stavů se tam dostáváme k číslu 1024. V mém praktickém příkladu na NKA je stavů dvacet sedm, čímž bychom se u převodu dostali k milionům, což už může být časově i výpočetně náročné. Naštěstí při převodu dojde k sestrojení automatu pouze s dosažitelnými stavy, kterých je ve většině případů podstatně menší počet. Nicméně existují i nedeterministické automaty, u nichž má i ten nejmenší ekvivalentní DKA maximální počet stavů. Z definice je dále zřejmé, že množina *F'* deterministického konečného automatu je množinou koncových stavů jak ji známe, protože obsahuje všechny podmnožiny z množiny *Q*, které obsahují některý koncový stav z množiny *F*. Přechodová funkce δ' každé podmnožině původních stavů $K \subseteq Q$ a znaku abecedy *a* přiřadí podmnožinu $\{q \in Q \mid \exists q' \in K : q' \xrightarrow{a} q\}$. (Jančar, 2010, s. 95 - 98)

Algoritmický postup převodu je názorně vysvětlen u praktického příkladu.

5.3 Praktický příklad

5.3.1 Příklad konfigurace automatu

Je dán automat $A = (\{q_0, q_1, q_2, q_3\}, \{e, t, x\}, \delta, q_0, q_0)$ s přechodovými funkcemi rozpoznávající například řetězec *text* v zadaném vstupním slově.

$\delta(q_0, e) = q_2$ $\delta(q_0, t) = q_1$ $\delta(q_0, x) = q_3$

$\delta(q_1, e) = q_2$ $\delta(q_3, t) = q_0$ $\delta(q_2, x) = q_3$

Vstupní slovo je: *text*

Počáteční konfigurace je uspořádaná dvojice (q_0, text). Konfigurace automatu relací je $(q_0, \text{text}) \vdash (q_1, \text{ext}) \vdash (q_2, \text{xt}) \vdash (q_3, \text{x}) \vdash (q_0, \varepsilon)$ Z toho vyplývá koncová konfigurace (q_3, ε)

5.3.2 Příklad převodu NKA na DKA

Je dán NKA $A = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, q_3)$

$\delta(q_0, 0) = q_1$ $\delta(q_0, 1) = q_2$
$\delta(q_1, 0) = q_2$ $\delta(q_1, 1) = q_0, q_2$
$\delta(q_2, 0) = q_2, q_3$ $\delta(q_2, 1) = q_1$

Z definice víme, že při převodu nedeterministického konečného automatu na deterministický bude ve tvaru $A' = (Q', \Sigma, \delta', I, F')$. Vstupní abeceda Σ zůstává stejná, vstupní stav také. Množinu Q' získáme rozšířením tabulky přechodů původního automatu o nové podmnožiny stavů, které jsou v tabulce u vstupní abecedy, ale nemají svůj vlastní ohodnocený řádek přechodů. V tomto konkrétním případě se jedná o stavy q_2, q_3 a q_0, q_2. Těmto množinám stavů přiřadíme vlastní řádek a doplníme podle přechodů v tabulce pro každý stav z množiny. Pokud výsledná množina není v tabulce obsažena, přidáme ji do tabulky a postup opakujeme. Protože mluvíme o množinách, jednotlivé stavy v buňkách tabulky uzavřeme do složených závorek.

Tabulka 5 - Převod na DKA

	0	1
→ $\{q_0\}$	$\{q_1\}$	$\{q_2\}$
$\{q_1\}$	$\{q_2\}$	$\{q_0, q_2\}$
$\{q_2\}$	$\{q_2, q_3\}$	$\{q_1\}$
← $\{q_3\}$	Ø	Ø
$\{q_0, q_2\}$	$\{q_1, q_2, q_3\}$	$\{q_1, q_2\}$
← $\{q_2, q_3\}$	$\{q_2, q_3\}$	$\{q_1\}$
← $\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$	$\{q_0, q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_2, q_3\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_1, q_2, q_3\}$	$\{q_0, q_1, q_2\}$

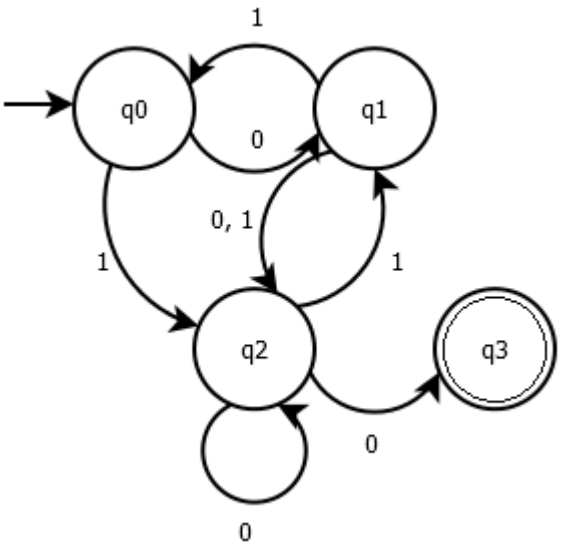
Zdroj: vlastní

Vidíme, že převodem na DKA má automat devět stavů, což je o pět více, než původně. Pořád je to ale méně, než maximální očekávaný počet šestnáct ze vzorečku 2^n.

Koncové stavy jsou veškeré podmnožiny, které obsahují alespoň jeden z koncových stavů původního NKA.

Původní NKA reprezentovaný stavovým diagramem vypadá následovně

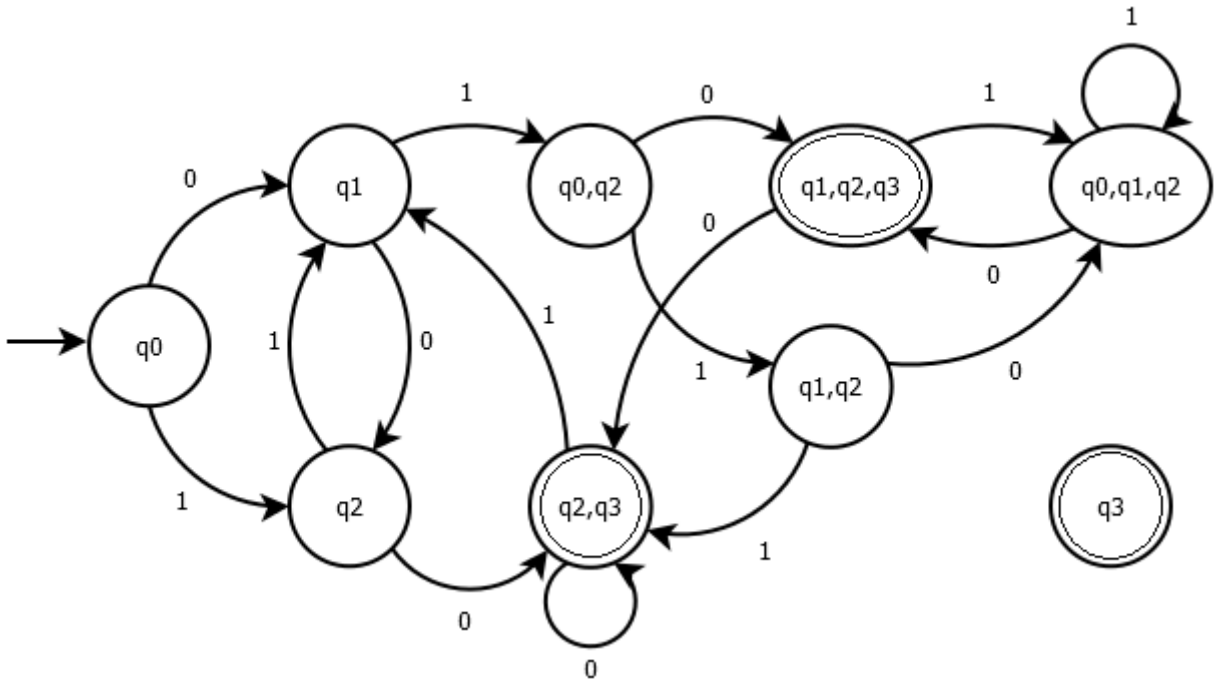
Obrázek 13 – Původní NKA



Zdroj: vlastní

Sestavený DKA podle tabulky je na první pohled složitější a vidíme dokonce jeden nedosažitelný stav, který v rámci minimalizace automatu můžeme vyloučit. I když se jedná o koncový stav, automat bude stále funkční a bude rozpoznávat stejný jazyk, jako automat nedeterministický.

Obrázek 14 - Převedený DKA



Zdroj: vlastní

6 AUTOMATY S VÝSTUPEM

6.1 Deterministický automat s výstupem

Automaty využívané v praxi ke zpracování dat nám poskytují všechny možné druhy výstupů. Námi definované DKA ani NKA tuto vlastnost zatím neobsahovaly. Je tedy nutné k nim přidat jednu neprázdnou množinu navíc – výstupní abecedu.

6.1.1 Moorův automat

Jedním typem konečného deterministického automatu s výstupem je Moorův automat, v literatuře někdy též označovaný jako Moorův stroj. Automaty s výstupem se od již uvedených automatů liší především v tom, že na posloupnost symbolů vstupní abecedy postupně reaguje posloupností symbolů abecedy výstupní. Takový výstup pochopitelně umožňuje suplovat roli koncových stavů. Definován je jako uspořádaná pětice $M = (Q, \Sigma, \Delta, \delta, \mu)$, kde

- Q je konečná neprázdná množina stavů
- Σ je konečná neprázdná množina vstupních symbolů
- Δ je konečná neprázdná množina výstupních symbolů
- δ je přechodová funkce $Q \times \Sigma \rightarrow Q$
- μ je značkovací funkce $Q \rightarrow \Delta$

Výstupní symbol tak podle definice závisí na aktuálním stavu stroje a ne na právě čteném prvku vstupního slova. (Chytil, 1984, s. 45)

6.1.2 Mealyho automat

Konečným Mealyho automatem je uspořádaná pětice $M = (Q, \Sigma, \Delta, \delta, \lambda)$, kde

- Q je konečná neprázdná množina stavů
- Σ je konečná neprázdná množina vstupních symbolů

- Δ je konečná neprázdná množina výstupních symbolů
- δ je přechodová funkce Q x Σ → Q
- λ je výstupní funkce Q x Σ → Δ

Výstup automatu je tak na rozdíl od Moorova typu dán jak stavem, tak i právě čteným vstupním symbolem. Přechodová funkce tedy určuje nejen nový výstupní stav, jak tomu bylo doposud, ale i symbol výstupní abecedy.

Mealyho a Moorův automat jsou výpočetně stejně silné, tj. ke každému Mealyho automatu lze nalézt ekvivalentní Moorův automat a naopak. (Vaníček, Papík, Pergl, Vaníček, 2007, s. 187- 191)

6.2 Iniciální deterministický automat s výstupem

Jak jste si mohli všimnout, Moorův i Mealyho automat se v definici od klasického deterministického automatu liší ještě v jedné věci. Jedná se o počáteční (tzv. iniciální) stav, který nemají definovaný. To řeší definice iniciálního DKA s výstupem, což je uspořádaná šestice $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$, kde právě onen přidaný stav q_0 je iniciální (počáteční). (Přibáň, 2014)

6.3 Praktický příklad

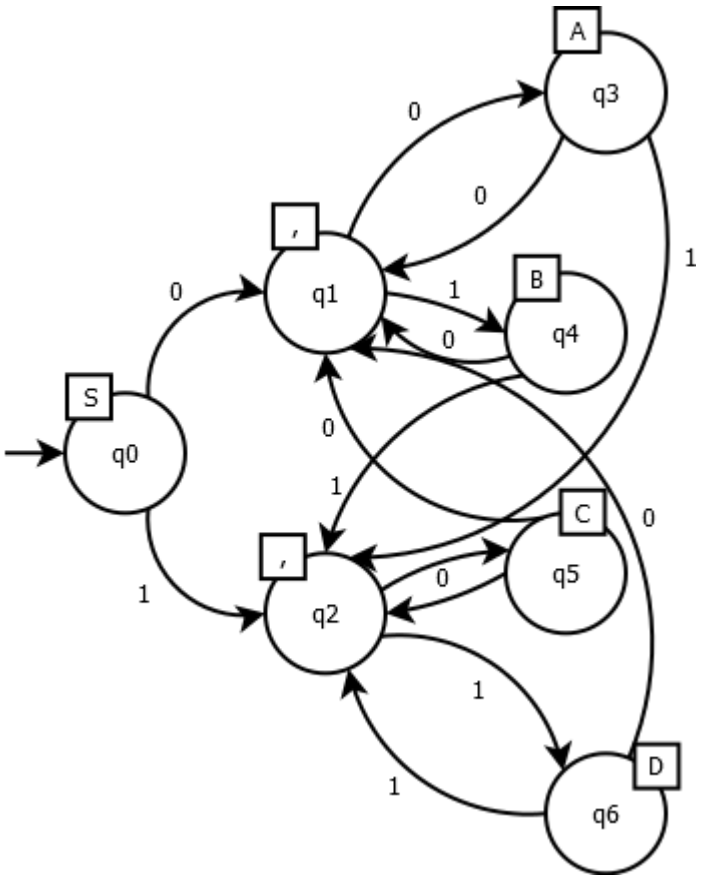
U praktických příkladů Moorova i Mealyho automatu využijeme definici s iniciálním stavem. Výstup těchto automatů nám bude reprezentovat dekódování přijatého slova v binární podobě do latinské abecedy, která je pro nás lépe srozumitelná.

6.3.1 Příklad Moorova automatu s výstupem

V tomto příkladu vytvoříme automat, který na výstupu vypisuje řetězec ze znaků *A, B, C, D*, jednotlivé znaky jsou odděleny čárkou a začátek řetězce je označen písmenem *Z*, symbolizující začátek převodu. Vstupní abeceda je tvořena znaky *0, 1*. Automat

znázorníme stavovým diagramem a tabulkou přechodů, ve které oproti klasické tabulce přechodů bude ve sloupečku se stavy navíc výstupní znak.

Obrázek 15 - Mooreův automat



Zdroj: vlastní

Tabulka 6 - Mooreův automat

Stav; výstup	0	1
q0; S	q_1	q_2
q1; ,	q_3	q_4
q2; ,	q_5	q_6
q3; A	q_1	q_2
q4; B	q_1	q_2
q5; C	q_2	q_1
q6; D	q_2	q_1

Zdroj: vlastní

Vidíme, že pouhými dvěma znaky vstupní abecedy můžeme dostat na výstupu větší počet výstupních symbolů, záleží pouze na stavech automatu. Následující řetězec nul a jedniček převedeme na výstup automatu.

w = *00110110011*

Výstup automatu: S, A, D, B, C, B,

Automat nemá množinu koncových stavů, jako klasický DKA nebo NKA, vidíme pouze výstup po zpracování slova z množiny vstupní abecedy. Rozšířením automatu o více stavů bychom byly schopni kódovat znaky celé české abecedy pomocí několika nul a jedniček. Ve vytvořeném praktickém příkladu je možné dekódovat libovolnou binární posloupnost znaků.

Obrázek 16 - Dekódování slov

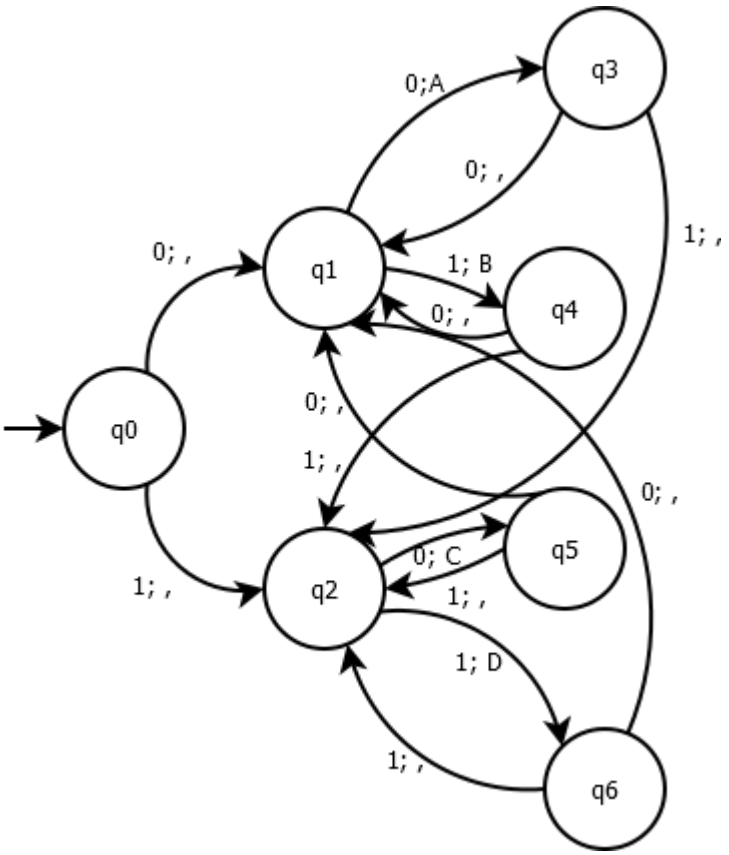
Zdroj: vlastní

6.3.2 Příklad Mealyho automatu s výstupem

Jak už je uvedeno v definici, Mealyho automat se liší ve výstupní funkci $\lambda: Q \times \Sigma \rightarrow \Delta$. V kombinované tabulce přechodů tak nebude písmeno výstupní abecedy ve sloupci stavů, ale ve sloupcích vstupních znaků. Už z toho vyplývá, že výstup bude záležet na aktuálním stavu a přečteném symbolu, tedy orientované hraně, kterou se bude stav měnit. Pro názornost použijeme shodný automat, jako v příkladu 6.3.1 Moorova automatu,

pozměníme podle definice stavový diagram a upravíme tabulku přechodů. Vstupní slovo necháme stejné a nakonec porovnáme, jak se oba výstupy liší.

Obrázek 17 - Mealyho automat



Zdroj: vlastní

Tabulka 7 - Mealyho automat

Stav, výstup	0	1
q0	q_1; ,	q_2; ,
q1	q_3; A	q_4; B
q2	q_5; C	q_6; D
q3	q_1; ,	q_2; ,
q4	q_1; ,	q_2; ,
q5	q_2; ,	q_1; ,
q6	q_2; ,	q_1; ,

Zdroj: vlastní

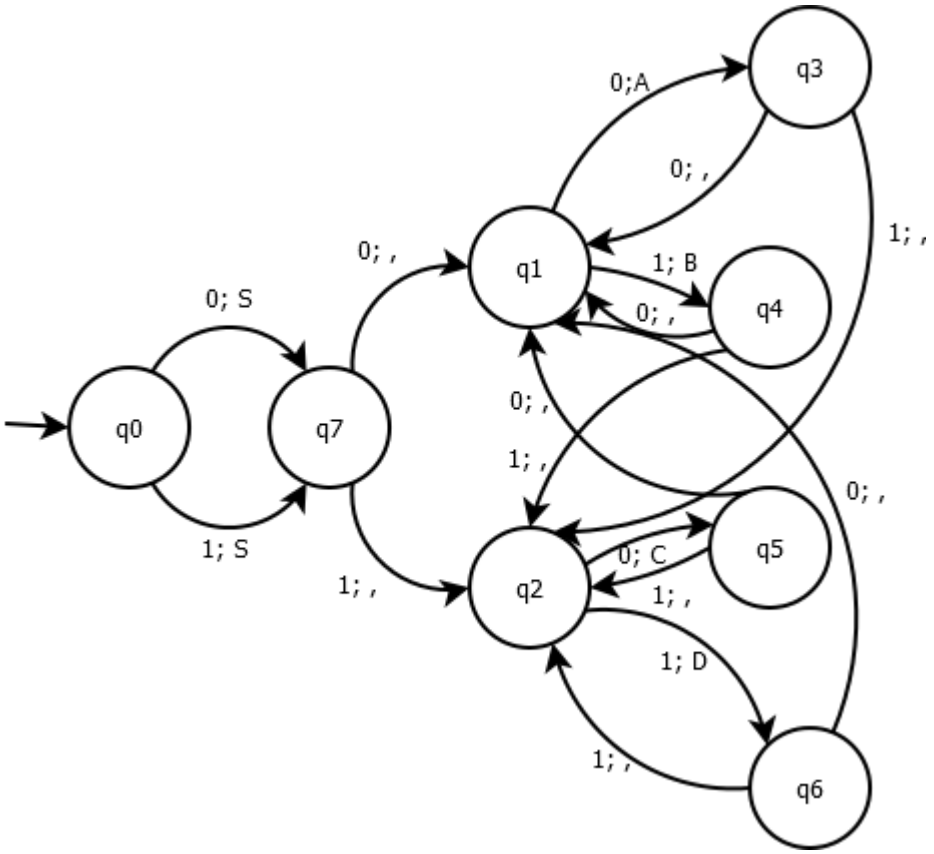
w = *00110110011*

Výstup automatu: , A, D, B, C, B,

Jak je vidět, při změně automatu z Moorova na Mealyho jsme při stejném načteném řetězci přišli o první znak a zároveň jeden symbol výstupní abecedy. Daný automat tak

už nesplňuje počáteční podmínku, aby výstupní řetězec začínal písmenem *S*, které automat ani neobsahuje. Pro splnění podmínky bychom museli automat mírně přepracovat přidáním dalšího stavu. Do něj se musíme dostat z počátečního stavu hranou ohodnocenou symbolem *S*. Do vstupního řetězce je navíc nutné na první místo přidat znak *0, 1*, aby došlo k načtení prvního symbolu. Upravený stavový diagram je znázorněn na obrázku 18.

Obrázek 18 - Opravený Mealyho automat



Zdroj: vlastní

Upravený vstupní řetězec pro shodný výstup s Moorovo automatem:

w = *000110110011*

7 REDUKCE A EKVIVALENCE

7.1 Redukce konečného automatu

Podle definice konečných automatů a praktických příkladů výše, existují stavy automatu, do kterých není možné se z počátečního stavu libovolnou posloupností znaků ze vstupní abecedy dostat. Takové stavy se nazývají nedosažitelné a na přijímaný jazyk automatem nemají žádný vliv. Z toho důvodu jsou zbytečné a můžeme je odstranit, čímž v podstatě dostaneme automat ekvivalentní. (Jančar, 2010, s. 74 - 75)

7.2 Ekvivalence automatů

Ekvivalence automatů je definována jako rovnost mezi přijímanými jazyky jednotlivých automatů. Matematický zápis pro automaty A_1 a A_2:

$$L(A_1) = L(A_2)$$

Způsob, jakým se určuje jazyk přijímaný automatem, je podrobněji popsán v následující kapitole. Nejtriviálnější způsob vytvoření ekvivalentního automatu je přidání nedosažitelného stavu. (Jančar, 2010, s. 82 - 83)

7.3 Praktický příklad

7.3.1 Redukce

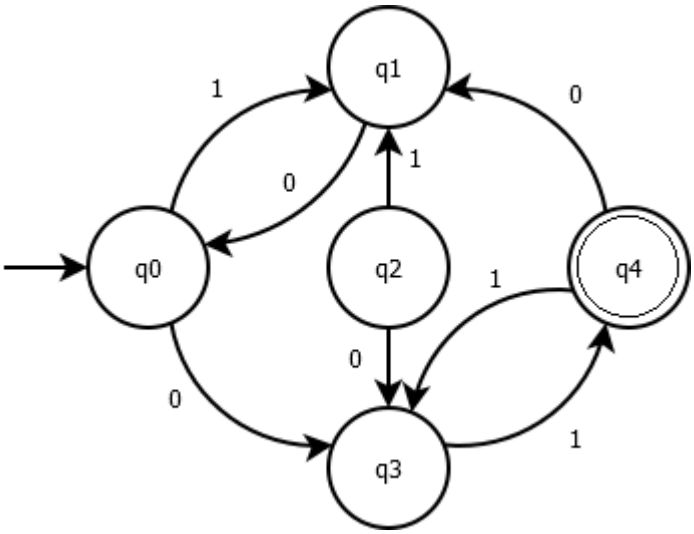
Abychom redukovali automat, musí všechny stavy q automatu $M = (Q, \Sigma, \delta, q_0, F)$ splňovat podmínku přechodové funkce $\delta(q0, w) = q$ pro slovo $w \in \Sigma$. Takové stavy budou dosažitelné. Ostatní – nedosažitelné - můžeme odstranit.

Zjištění dosažitelnosti je velice snadné. Pro názornost můžeme použít například stavový diagram, ve kterém budeme dosažitelné stavy označovat. První označíme

počáteční stav. Orientovanou hranou z tohoto stavu se dostaneme a zároveň označíme stavy následující. Toto opakujeme pro každý již označený stav, dokud již žádný další označit nejde.

Na obrázku stavového diagramu máme automat s pěti stavy, na který použijeme výše popsaný postup redukce.

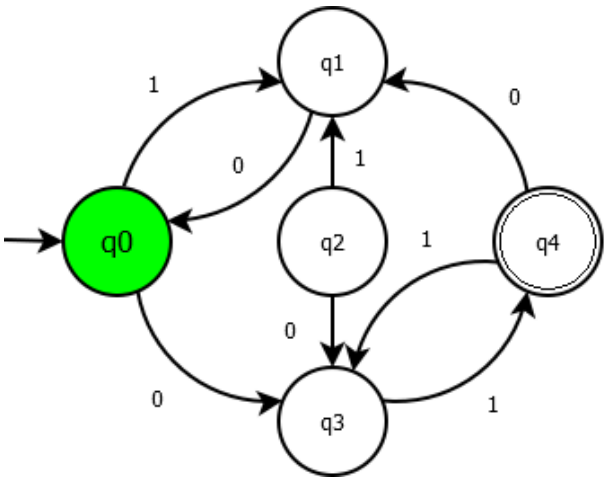
Obrázek 19 - Start redukce



Zdroj: vlastní

Nejprve si označíme počáteční stav q_0.

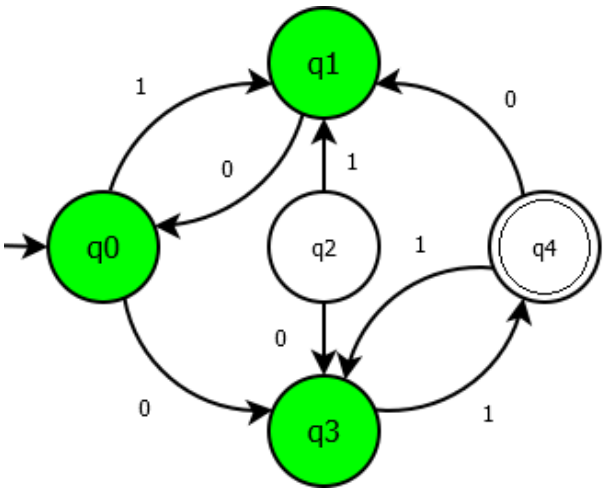
Obrázek 20 - Postup redukce 1



Zdroj: vlastní

Dále označíme stavy, do kterých z označeného počátečního stavu vedou ohodnocené hrany, tedy stavy q_1 a q_3.

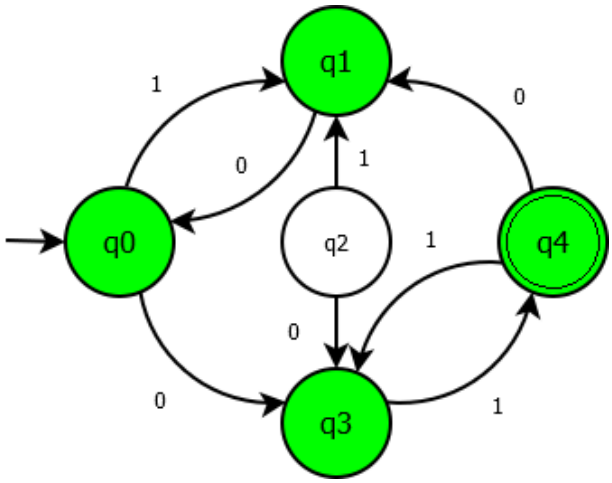
Obrázek 21 - Postup redukce 2



Zdroj: vlastní

Ze stavu q_1 vede hrana pouze do již označeného stavu, přejdeme proto na stav q_3, ze kterého můžeme označit q_4.

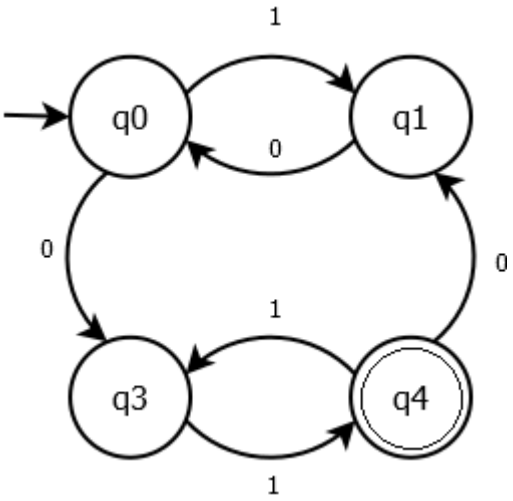
Obrázek 22 - Postup redukce 3



Zdroj: vlastní

Z q_4 opět vedou hrany do již označených stavů. Neoznačený stav q_2 je tedy nedosažitelný a můžeme o něj automat redukovat. Výsledný stavový diagram automatu bude vypadat takto.

Obrázek 23 - Ukončení redukce

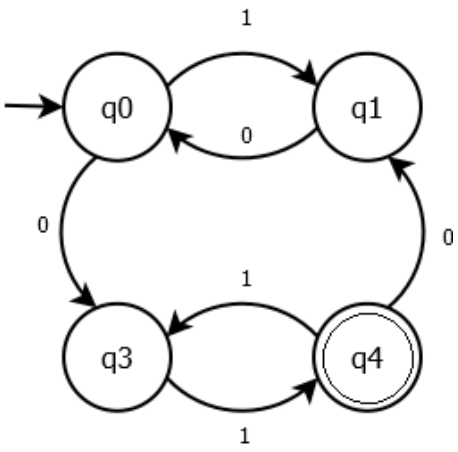


Zdroj: vlastní

7.3.2 Ekvivalence

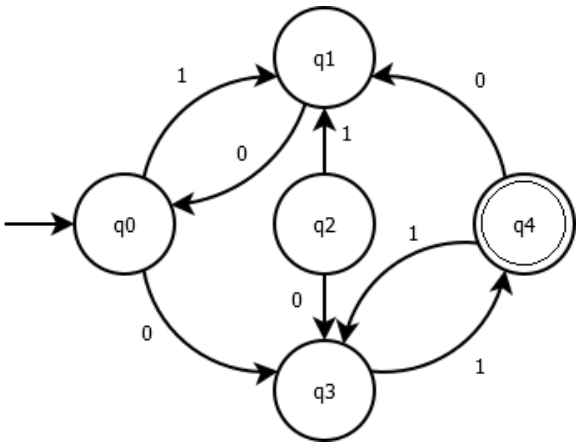
Automat A_1 je ekvivalentní automatu A_2. Ten byl vytvořen přidáním nedosažitelného stavu q_2.

Obrázek 24 - Původní automat



Zdroj: vlastní

Obrázek 25 - Ekvivalentní automat



Zdroj: vlastní

8 FORMÁLNÍ JAZYKY

8.1 Základní pojmy

Pro práci s jazyky a gramatikami je zapotřebí znát několik základních pojmů, které v této kapitole nadefinujeme. Jedná se o abecedu, slovo a samotný formální jazyk.

Abeceda je libovolná neprázdná konečná množina Σ. Jednotlivé její prvky nazýváme znaky, písmena nebo též symboly. Abeceda $\Sigma = \{x, y\}$ obsahuje dvě písmena x a y. Jako Σ^+ označujeme množinu všech konečných neprázdných posloupností utvořených z prvků množiny Σ. Jedná se tedy o množinu všech neprázdných řetězců. Prázdny řetězec značíme e. Sloučením množiny Σ^+ s prázdným řetězcem e získáme množinu všech slov nad abecedou Σ, kterou značíme Σ^*.

Slovo nebo též řetězec nad abecedou Σ je libovolná konečná posloupnost písmen z dané množiny. Značí se symbolem w. V abecedě $\Sigma = \{x, y\}$ tak může být slovem například

$$w = xxyyyx$$

Posloupnost písmen množiny zapisujeme bez čárek mezi jednotlivými prvky. Délku slova neboli počet písmen v řetězci, značíme $|w|$ a z uvedeného příkladu je $|w| = 6$. Prázdný řetězec e má tak velikost nula. Setkat se můžeme také s výrazem $|w|_a$, čímž označujeme počet výskytů symbolu a ve slově w. Z našeho příkladu tak vidíme, že $|w|_x = 3$.

Se slovy lze provádět asociativní operaci zřetězení, definovanou jako $\Sigma^* \times \Sigma^* \rightarrow \Sigma^*$. Pro dvě slova z abecedy Σ platí, že

$$w_1 = a_1 .. a_n$$
$$w_2 = b_1 .. b_n$$
$$w_1 * w_2 = a_1 .. a_n b_1 .. b_n$$

(Křetínský, 1991, s. 1)

Operace je asociativní, neboť platí u*v*w = (u*v)*w = u* (v*w). Zřetězení může být také n-násobné. Třínásobné zřetězení slova w pak vypadá takto: $w^3 = www$.

Je dohodnuto, že exponent má vždy větší prioritu, než zřetězení, proto u^3vw^2 je výsledné slovo *uuuvww*.

Úsek znaků, kterým nějaké slovo začíná, budeme nazývat předponou, neboli odborně prefixem. Obdobně se úsek znaků, kterým slovo končí, nazývá přípona neboli sufix. Jakoukoliv jinou část slova budeme nazývat podslovem (nebo podřetězcem). (Jančar, 2010, s. 29 - 33)

Formální jazyk se od jazyka přirozeného, například češtiny, liší. Formální jazyk (dále už pouze jazyk) nad abecedou Σ je libovolná podmnožina Σ^*. Jazyk značíme jako L a platí tedy, že $L \subseteq \Sigma^*$. Z jednotlivých slov jazyka netvoříme věty jako u jazyka přirozeného. Také si ve většině případů vystačíme s mnohem méně prvkovou abecedou pro reprezentaci automatů. Jazyky mohou být konečné i nekonečné. Konečné jsou řešeny například výčtem slov, nekonečné zápisem podmínek jazyka.

8.2 Jazyky rozpoznatelné konečnými automaty

Existuje-li konečný automat A takový, že jazyk $L = L(A)$, říkáme, že jazyk L je rozpoznatelný konečným automatem. (Chytil, 1984, s. 22)

V zadání praktického příkladu tedy dostaneme jazyk, podle kterého se pokusíme sestrojit konečný automat. Ne každý jazyk, i když na první pohled jednoduchý, je KA rozpoznatelný. Důkazem rozpoznatelnosti se zabývá **Nerodova věta**, která říká:

Nechť L je jazyk nad konečnou abecedou Σ. Pak L je rozpoznatelný konečným automatem, právě když existuje pravá kongruence konečného indexu taková, že L je sjednocením jistých tříd rozkladu $\Sigma^*|{\sim}$. Pro úplné pochopení musíme ještě definovat pojmy pravá kongruence a konečný index. (Chytil, 1984, s. 23)

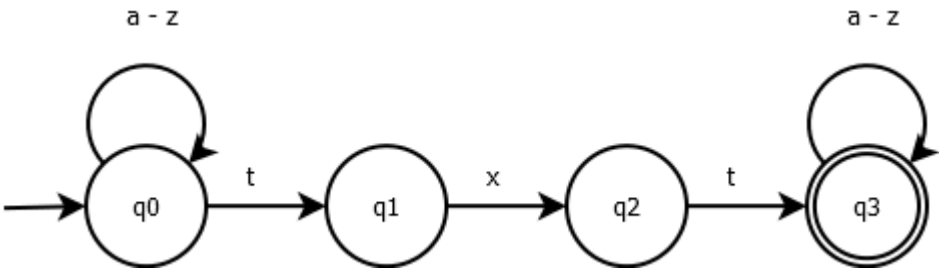
Pravá kongruence je relace ekvivalence ~ nad konečnou abecedou Σ^*, jestliže pro všechna $u, v, w \in \Sigma^*$ ze vztahu $u \sim v$ plyne, že $uw \sim vw$.

O relaci ekvivalence ~ na množině M říkáme, že je konečného indexu, jestliže rozklad $M|{\sim}$ má konečný počet tříd. (Chytil, 1984, s. 23)

8.3 Praktický příklad

Práce s konečnými automaty rozpoznávající daný jazyk přímo vybízí k využití pro vyhledávání řetězce v textu. Na následujícím příkladu si tedy nejdříve ověříme definicí Nerodovi věty, jaký jazyk je automatem rozpoznáván.

Obrázek 26 - Automat rozpoznávající řetězec



Zdroj: vlastní

Určíme si třídy rozkladem množiny Σ^*, čili určíme pravou kongruenci. Ta musí být konečného indexu, to znamená, že počet tříd nesmí být nekonečný.

$T_0 = \{w; \delta (q_0, w) = q_0\}$

Toto je třída slov, které se libovolným slovem *w* dostanou z počátečního stavu q_0 do stavu q_0. Je to tedy třída obsahující libovolnou posloupnost znaků od *a* do *z*.

$T_1 = \{w; \delta (q_0, w) = q_1\}$

Třída slov T_1 obsahuje libovolný počet znaků *a – z*, ale posledním znakem je písmeno *t*.

$T_2 = \{w; \delta (q_0, w) = q_2\}$

Třída slov obsahující libovolnou posloupnost znaků od *a* do *z* končící *tx*.

$T_3 = \{w; \delta (q_0, w) = q_3\}$

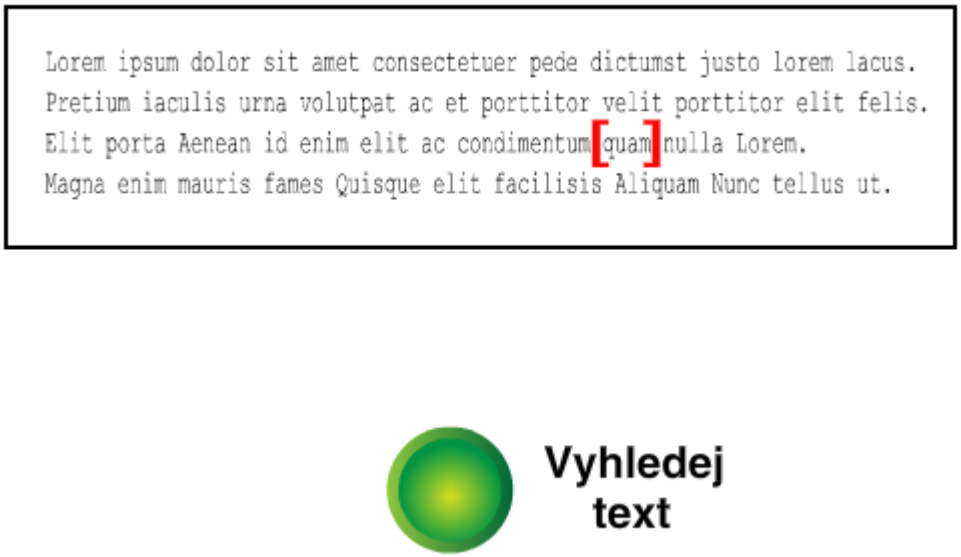
Třída slov z celé abecedy obsahující řetězec *txt*.

Toto jsou všechny třídy automatu, podmínka konečného indexu je tedy splněna. Jazyk rozpoznávaný naším automatem je sjednocením všech tříd koncových stavů, tedy T_3. Automatem rozpoznávaný jazyk je tedy takový jazyk, který obsahuje řetězec *txt*. Takový jazyk bychom mohli zapsat jako

$L = (w \in \{a - z\}^* \mid w$ obsahuje *txt*)

Pokud tedy automat nějaký námi zadaný text přijme, víme, že obsahuje řetězec *txt*. Záleží pak už na nás, jak s touto informací naložíme, případně zobrazíme. Vytvořená komponenta proto obsahuje několik vět a možnost vyhledávat libovolný textový řetězec. Ten je v případě nalezení označen červenými hranatými závorkami.

Obrázek 27 - Komponenta na vyhledání textu



Zdroj: vlastní

9 GRAMATIKY

9.1 Definice

Gramatika je čtveřice $G = (N, T, P, S)$ s následujícím významem:

- N je množina neterminálů, které figurují jako tzv. metasymboly
- T je množina terminálních symbolů (terminálů), u které platí $N \cap T = \emptyset$. Sjednocením *N* a *T* obdržíme množinu všech symbolů gramatiky, kterou označujeme symbolem *V*, $V = N \cup T$
- P je množina pravidel, $P \subseteq V * NV^* \times V^*$; jedná se tedy o dvojice slov, kde první obsahuje alespoň jeden neterminální symbol
- S je počáteční symbol (též kořen) gramatiky *G*, jedná se o neterminál

(Křetínský, 1991, s. 2)

Jedná se tedy o systém, kterým pomocí přepisovacích pravidel můžeme zkonstruovat všechna slova z daného jazyka. Pravidla z množiny *P* zapisujeme ve tvaru $\alpha \rightarrow \beta$. Z definice vidíme, že levá strana (α) musí obsahovat alespoň jeden neterminál. Na pravou stranu pravidla obecně nejsou kladeny žádné požadavky. Může se jednat i o prázdné slovo *e*. Pořadí při výběru přepisovacího pravidla není stanoveno. Je tak možné získat různá slova při použití stejných pravidel v jiném pořadí. Obdobně tak lze často jedno slovo získat více pravidly. Dále si zavedeme domluvené konvence týkající se značení:

- Kořen gramatiky značíme symbolem *S*
- Neterminální symboly značíme velkými písmeny (obvykle ze začátku) latinské abecedy (A, B, C, ..)
- Terminální symboly značíme malými písmeny ze začátku latinské abecedy (a, b, c)
- Řetězce složené výhradně z terminálních symbolů (tzv. terminální řetězce) značíme malými písmeny z konce latinské abecedy (x, y, z)
- Řetězce složené z terminálních i neterminálních symbolů značíme malými řeckými písmeny ze začátku abecedy (α, β, γ)
- Pravidla se stejnou levou stranou zapisujeme stručněji jako $\alpha \rightarrow \beta \mid \gamma$

(Černá, Křetínský, Kučera, 2002, s. 4 - 5)

Dále každá gramatika G určuje binární relaci $\Rightarrow_g$ přímého odvození na množině V^* definovanou jako $x \Rightarrow_g y$ právě tehdy, když existuje $\alpha \rightarrow \beta \in P$ a slova $u, v \in V^*$ taková, že $x = u\alpha v$ a $y = u\beta v$. (Černá, Křetínský, Kučera, 2002, s. 4)

9.2 Klasifikace gramatik

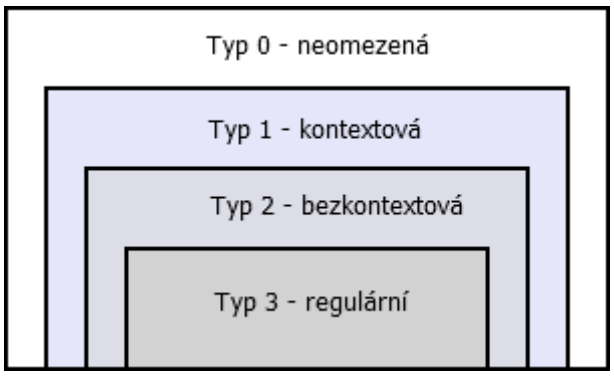
Na základě tvaru přepisovacích pravidel rozdělil Noam Chomsky gramatiky do čtyř skupin. V dnešní době se jedná o nejznámější klasifikaci, která má navíc význam jako výhodné rozdělení gramatik podle jejich popisné síly. Jedná se o tyto čtyři typy:

- Typ 0 – libovolná gramatika je gramatikou typu nula, na tvar pravidel se nekladou žádné omezující požadavky. Někdy též se takové gramatiky označují jako gramatiky bez omezení či frázové gramatiky.
- Typ 1 – nebo též kontextová, je gramatika, pro jejíž každé pravidlo $\alpha \rightarrow \beta$ platí $|\alpha| \leq |\beta|$, s výjimkou u pravidla $S \rightarrow e$, pokud se kořen S nevyskytuje na pravé straně žádného pravidla
- Typ 2 – nebo také bezkontextová gramatika splňuje podmínku, kde každé její pravidlo je ve tvaru $A \rightarrow \alpha$ a $|\alpha| \geq 1$, opět s výjimkou pravidla $S \rightarrow e$, pokud se S nevyskytuje na pravé straně žádného pravidla.
- Typ 3 – neboli gramatika regulární, je gramatika, u které každé její pravidlo má tvar $A \rightarrow aB$ nebo $A \rightarrow a$ s eventuelní výjimkou pravidla $S \rightarrow e$, pokud se kořen S nevyskytuje na pravé straně žádného pravidla.

(Černá, Křetínský, Kučera, 2002, s. 5 - 6)

Chomského rozdělení gramatik je hierarchické, jak ukazuje obrázek 28. Každá regulární gramatika typu 3 je bezkontextová, každá bezkontextová je i kontextová a každá kontextová gramatika je zároveň i gramatikou typu 0. V následující kapitole jsou regulární gramatiky blíže popsány spolu s regulárními jazyky, výrazy a spojením s konečnými automaty.

Obrázek 28 - Chomského hierarhie



Zdroj: vlastní

9.3 Praktický příklad

V příkladu si ukážeme využití přepisovacích pravidel gramatiky pro kontrolu základní větné stavby jednoduché věty. Ta se skládá z konkrétních podstatných a přídavných jmen a slovesa, což jsou pro nás terminální symboly. Neterminálními symboly jsou reprezentovány větnými členy, jako je například podmět a přísudek. Kořenem gramatiky je pak celá věta.

Je dána gramatika G = ({Věta, Podmět, Přísudek, Přívlastek, Předmět, Podstatné jméno, Přídavné jméno, Sloveso}, {Karel, člověk, řidič, problém, oheň, hlavolam, vyřešil, vymyslel, udělal, velký, líný, chytrý}, P, Věta). Přepisovací pravidla P jsou následující:

Věta → Podmět Přísudek

Podmět → Přívlastek Podmět | Podstatné jméno

Přívlastek → Přídavné jméno

Přísudek → Sloveso Předmět

Předmět → Přívlastek Předmět | Podstatné jméno

Podstatné jméno → Karel | člověk | řidič | problém | oheň | hlavolam

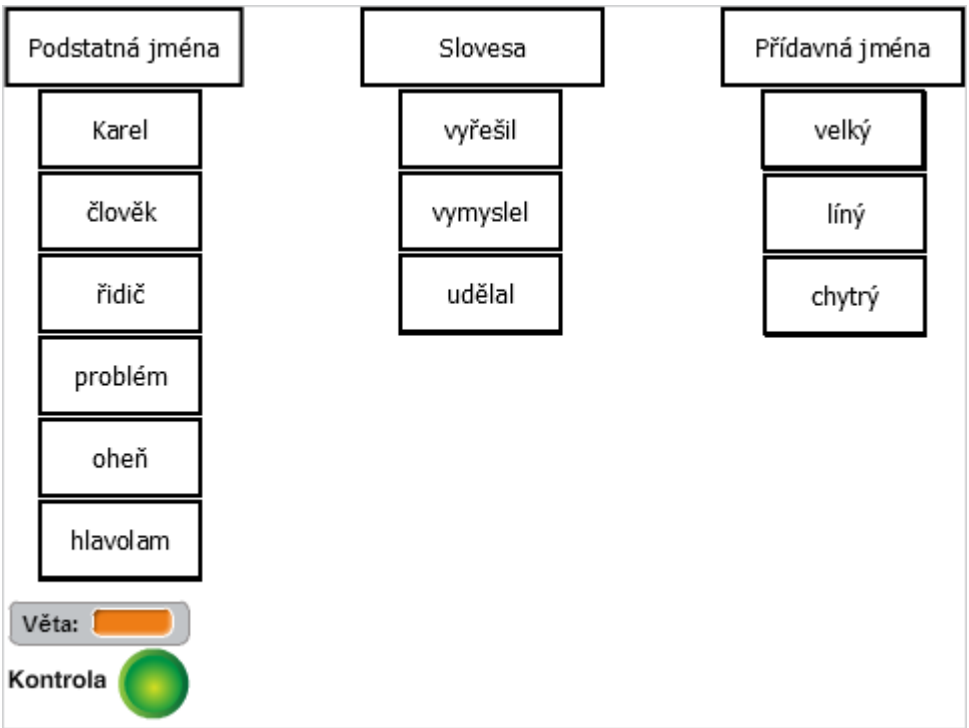
Sloveso → vyřešil | vymyslel | udělal

Přídavné jméno → velký | líný | chytrý

Z přepisovacích pravidel vidíme, že se jedná o bezkontextovou gramatiku typu 2.

V připraveném příkladu si může uživatel vyzkoušet sestavit větu z vybraných slov a zkontrolovat si její větnou stavbu, která je dána právě přepisovacími pravidly. Každá správně složená věta proto musí obsahovat podmět a přísudek. Podmět je složen z libovolného počtu přídavných jmen a jednoho podstatného jména. Přísudek je složen ze slovesa a podstatného jména, před kterým se stejně jako u podmětu může nalézat libovolný počet přídavných jmen. Dále v přepisovacích pravidlech najdeme výpis jednotlivých podstatných jmen, sloves a přídavných jmen. Po vybrání slova z nabídky je toto slovo přepsáno do proměnné *Věta*, kterou si po dokončení můžeme stiskem zeleného tlačítka nechat zkontrolovat (obrázek 29). Nejkratším příkladem správně sestavené věty je například „Člověk vymyslel oheň“, rozvinutější věta pak může vypadat následovně „Chytrý Karel vyřešil velký hlavolam“. Neakceptované věty jsou pak takové, které neodpovídají přepisovacím pravidlům – obsahují například dvě podstatná jména za sebou nebo větný člen na nesprávném místě, jako je slovesem začínající věta.

Obrázek 29 - Kontrola větné stavby



Zdroj: vlastní

10 REGULÁRNÍ VÝRAZY V KONEČNÝCH AUTOMATECH A FORMÁLNÍCH JAZYCÍCH

10.1 Regulární výraz a množina

V jednom z předešlých příkladů jsme viděli, že konečným automatem lze vyhledávat konkrétní slova nebo jejich části v textu. Pokud bychom ale měli vyhledávat obecněji s použitím podmínek, musíme použít regulární výrazy a jimi reprezentované jazyky. Definujme tedy množinu regulárních výrazů *RV (Σ)* nad abecedou *Σ* jako nejmenší množinu slov v abecedě $\Sigma \cup \{ Ø, e, +, -, \cdot, *, (,)\}$ splňující tyto podmínky:

- $Ø \in RV(\Sigma)$, $e \in RV(\Sigma)$, pro každé $a \in \Sigma$ je $a \in RV(\Sigma)$
- Jestliže $\alpha, \beta \in RV(\Sigma)$, pak také $(\alpha + \beta) \in RV(\Sigma)$, $(\alpha \cdot \beta) \in RV(\Sigma)$, $(\alpha^*) \in RV(\Sigma)$

za předpokladu, že $Ø, e, +, -, \cdot, *, (,) \notin \Sigma$. (Jančar, 2003, s. 35 - 36)

Množina je tedy regulární právě tehdy, když ji lze popsat regulárním výrazem. Symboly α, β z definice mohou být symboly i řetězce. Zápisem (α + β) u regulárních výrazů myslíme sjednocení, tedy se na daném místě výrazu musí vyskytovat symbol α nebo β. Zápis (α · β) vyjadřuje uspořádané zřetězení, záleží tedy na pořadí. Iterace (a^*) umožňuje řetězci *n*-násobné opakování. Regulárním výrazem *Ø* označujeme regulární množinu *Ø*. Obdobně také RV *e* označuje množinu {*e*}. Abychom zpřehlednili zápis jednotlivých regulárních výrazů, je možné vypustit některé ze závorek. Zpravidla se jedná o vnější pár, uzavírající celý výraz. Další závorky můžeme vypustit díky asociativitě součinu.

10.2 Regulární jazyk a gramatika

O regulární gramatice *G* už víme, že je to čtveřice složená z neterminálních a terminálních symbolů, přepisovacích pravidel a kořene gramatiky. Abychom o gramatice mohli prohlásit, že je regulární, čili jedná se o gramatiku třetího typu podle Chomského hierarchie, musí jej přepisovací pravidla být ve tvaru $A \rightarrow aB$ nebo $A \rightarrow a$. Výjimku tvoří

pravidlo pro kořen S → *e* v případě, že se *S* nevyskytuje na pravé straně žádného pravidla. A právě tento typ gramatik generuje regulární jazyky. Jedná se tedy o generativní systém. Jak bylo názorně ukázáno v praktickém příkladu deváté kapitoly, z počátečního symbolu se za použití přepisovacích pravidel vygenerovala celá věta.

Abychom o formálním jazyku *L* mohli prohlásit, že je regulární, musí splňovat některou z následujících podmínek:

- Může být generován regulární gramatikou, to znamená, že existuje gramatika *G* taková, že L(G) = L
- Je akceptovaný nějakým deterministickým konečným automatem (existuje automat *A* takový, že L(A) = L)
- Je akceptovaný nějakým nedeterministickým konečným automatem (existuje automat *B* takový, že L(B) = L)
- Může být popsán regulárním výrazem *RV*, L(RV) = L

(Sipser, 2013, p. 66)

Tyto podmínky byly v předešlých kapitolách naznačeny a v odborné literatuře jsou podrobně popsané i jejich důkazy. Znamená to tedy, že pokud se nám například nepodaří k danému jazyku sestrojit příslušný konečný automat, jazyk není regulární. K tomu se váže **Kleeneho věta**, která říká, že libovolný jazyk je popsatelný regulárním výrazem právě tehdy, když je rozpoznatelný konečným automatem. A jelikož víme, že jazyky určené regulárními výrazy jsou také regulární, musí být i ony rozpoznatelné KA. (Habiballa, 2013, s. 70 - 71)

Pro zjištění neregularity jazyka se využívá nutná podmínka, tzv Lemma o vkládání (pumping lemma), která říká, že máme-li regulární jazyk *L*, pak existuje takové číslo *n* z řádu přirozených čísel **N** takové, že slovo $w \in L$, jehož délka je alespoň *n*, lze psát ve tvaru $w = xyz$, kde $|xy| \leq n$, $y \neq e$ a $xy^i z \in L$ pro každé *i* z řádu nezáporných kladných čísel $\mathbf{N}^0$. Jedná se o nutnou podmínku, nikoliv postačující, proto i když bude splněna, neznamená to, že jazyk je regulární. Při nesplnění podmínky naopak jistě víme, že jazyk regulární není. (Jančar, 2003, s. 48 - 51)

Druhou možností pro zjištění regularity je Myhill-Nerodova věta. Jedná se o rozšíření původní věty z kapitoly **8.2** o prefixovou ekvivalenci. Tato věta je nutnou a postačující podmínkou pro regularitu jazyka. Definujme tedy prefixovou ekvivalenci - je

dán libovolný jazyk L nad abecedou Σ. Na množině Σ^* definujeme relaci $\sim_L$ zvanou prefixová ekvivalence pro L jako:

$u \sim_L v \Leftrightarrow$ pro všechna $w \in \Sigma^* : uw \in L \Leftrightarrow vw \in L$

(Černá, Křetínský, Kučera, 2002, s. 25)

Slovně bychom mohli vyjádřit definici tak, že prefixová ekvivalence je taková vlastnost, kdy po přidání libovolného slova w ke slovům u a v budou nově vzniklá slova uv a vw patřit do jazyka L buď obě, nebo žádné. Nyní už jsme schopni pochopit vyjádření Myhill-Nerodovi věty, kde je dán jazyk L nad abecedou Σ s následujícími ekvivalentními tvrzeními:

- L je rozpoznatelný konečným automatem
- L je sjednocením některých tříd rozkladu určeného pravou kongruencí na Σ^* s konečným indexem
- Relace $\sim_L$ má konečný index

(Černá, Křetínský, Kučera, 2002, s. 25)

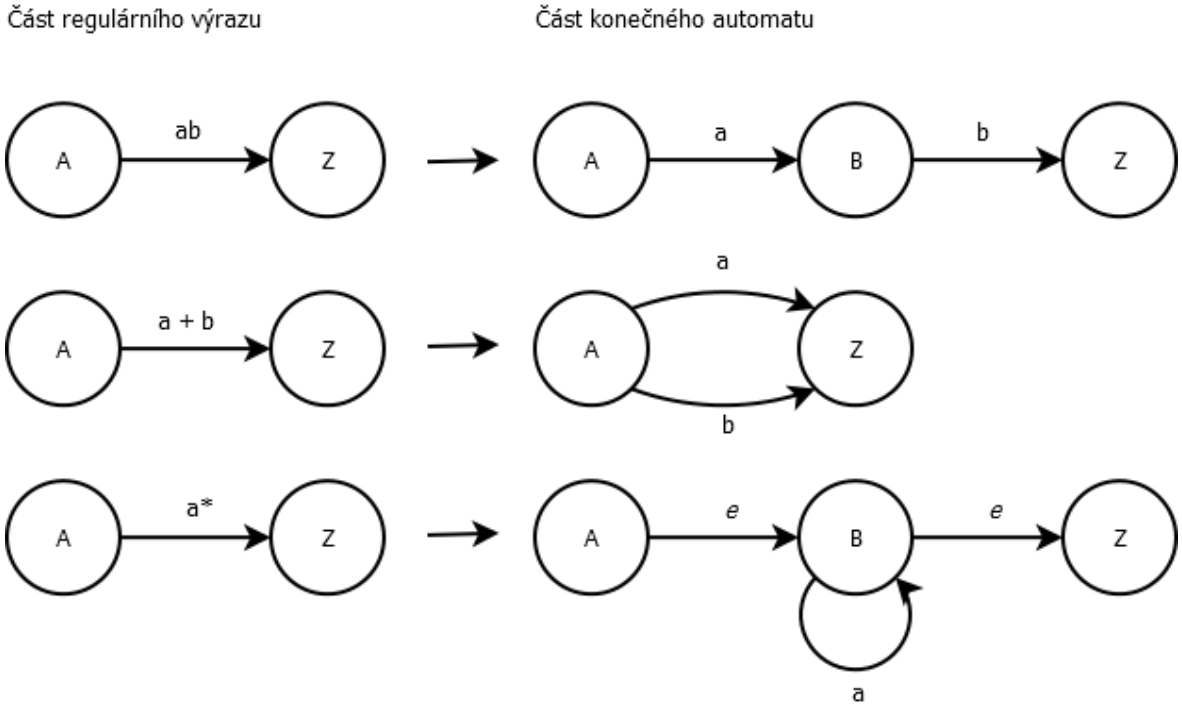
10.3 Převod regulárního výrazu na konečný automat a zpět

Jak už víme, regulární výrazy popisují regulární jazyky a ty jsou rozpoznávány konečnými automaty. Existuje proto způsob, jak převést regulární výraz na KA a zpět. Pro převod se využívají přechodové grafy. Přechodový graf T nad abecedou Σ je orientovaný multigraf, jehož každá hrana je označena návěštím, které je slovem nad Σ. Pro slovo $w \in \Sigma^*$ nazveme w-cestou z uzlu i do uzlu j v grafu T cestu z i do j takovou, že zřetězením všech návěští na hranách po cestě získáme slovo w. Slovo je akceptováno, jestliže existuje w-cesta z počátečního do některého z koncových stavů grafu. Slovo e je navíc akceptováno v případě, že počáteční uzel je zároveň koncovým. (Křetínský, 1991, s. 16)

Na začátku převodu tedy máme dva uzly s ohodnocenou hranou regulárním výrazem. Tyto dva uzly jsou zároveň počátečním a koncovým stavem výsledného automatu. Během převodu graf postupně upravujeme, jak rozkládáme a zjednodušujeme

regulární výraz. Nové uzly a ohodnocené hrany přidáváme tak dlouho, dokud všechny stavy nejsou spojeny ohodnocenou hranou slovem délky jedna nebo *e*. Pro úpravu používáme pravidla z obrázku 30.

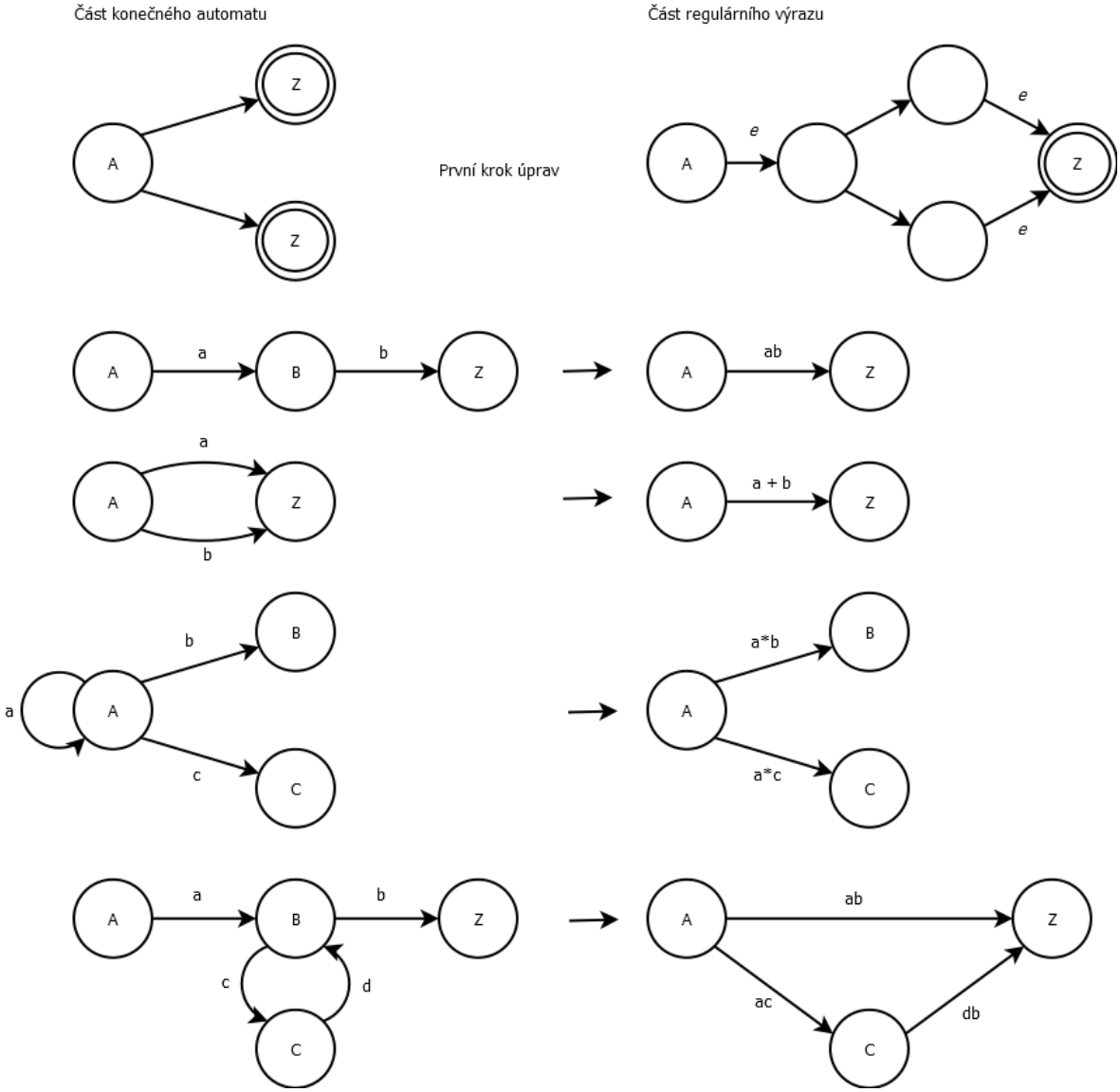
Obrázek 30 - Pravidla pro úpravu RV na KA



Zdroj: vlastní

Při opačném převodu, tedy převodu konečného automatu na regulární výraz, je postup obdobný, ale obrácený. Cílem je mít automat pouze se dvěma stavy – počátečním a koncovým. Výchozí automat těchto stavů může mít více, proto se první úpravou přidá vlastní počáteční a koncový stav s hranou *e*. Následují úpravy podle pravidel na obrázku 31, kterými se postupně zbavujeme hran a stavů, dokud nezůstanou pouze v prvním kroku vytvořené stavy. Ohodnocená hrana mezi nimi je výsledný regulární výraz.

Obrázek 31 - Pravidla pro převod KA na RV



Zdroj: vlastní

10.4 Převod regulární gramatiky na konečný automat a zpět

Protože ke každé regulární gramatice *G* existuje konečný automat *A* takový, že L (A) = L (G), jsme schopni vytvořit takový KA, který bude rozpoznávat stejný jazyk, jako gramatika *G*. Jsou tedy navzájem převoditelné. Postupy převodů si vypíšeme teoreticky a v části s praktickými příklady je převedeme do praxe. K vytvoření automatu potřebujeme následující pravidla:

- Stavy Q automatu *A* se ztotožní s neterminálními symboly *N* gramatiky *G*. Ke stavům automatu *A* se přidá ještě stav *K*, který neodpovídá žádnému neterminálnímu symbolu gramatiky. Množina stavů *Q* se tedy rovná $N \cup \{K\}$, $K \neq N$
- Počáteční stav q_0 je shodný s počátečním symbolem *S* gramatiky *G*
- Abeceda Σ bude shodná s terminálními symboly *T*
- Množina koncových stavů *F* je určena
    a) $F = \{S, K\}$ když $(S \rightarrow e) \in P$
    b) $F = \{K\}$ když $(S \rightarrow e) \notin P$
- Přechodová funkce δ se vytváří podle následujících pravidel
    a) $\delta(B, a) = C$ pokud $B \rightarrow aC \in P$
    b) $\delta(B, a) = K \wedge K \in F$ pokud $B \rightarrow a \in P$

(Kozel, 2008)

Pro převod zpátky z konečného automatu $A = (Q, \Sigma, \delta, q_0, F)$ na regulární gramatiku $G = (N, T, P, S)$ nám poslouží obdobná pravidla – neterminální symboly budou stavy automatu ($N = Q$), terminální symboly budou tvořit abecedu automatu ($T = \Sigma$), počáteční symbol je roven počátečnímu stavu ($S = q_0$). Přepisovací pravidla odvodíme z přechodových funkcí následovně

- $B \rightarrow aC \in P$ když $\delta(B, a) = C$
- $B \rightarrow a \in P$ když $\delta(B, a) = K \wedge K \in F$
- $S \rightarrow e \in P$ když $q_0 \in P$

(Kozel, 2008)

10.5 Praktický příklad

10.5.1 Regulární výraz

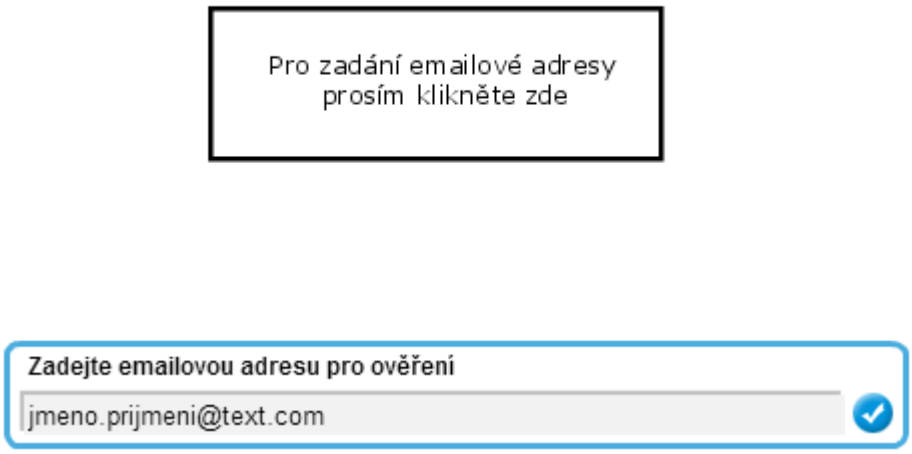
S regulárními výrazy se můžeme setkat již prakticky všude, například při zadávání emailové adresy do internetového formuláře při objednávce zboží. Formulář musí adresu zkontrolovat regulárním výrazem tak, aby obsahovala libovolný počet znaků abecedy,

zavináč, znovu libovolnou posloupnost znaků, tečku a libovolný počet znaků. Některé z regulárních výrazů jdou ještě dále a kontrolují například, jestli je první zadaný symbol speciálním znakem abecedy nebo celkový počet znaků. V tomto příkladu se však zaměříme pouze na první případ a zapíšeme regulární výraz.

RV = $(a + b + .. + z)^*$ @ $(a + b + .. + z)^*$. $(a + b + .. + z)^*$

Ve vytvořené komponentě stačí zadat emailovou adresu, respektive textový řetězec, který bude akceptován jako správný, pokud bude ve formátu *text@dalsi_text.txt,* jak to ilustruje obrázek 32.

Obrázek 32 - Ověření emailové adresy

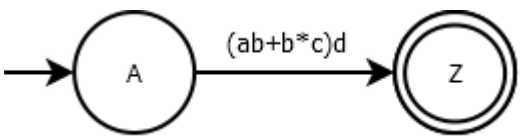


Zdroj: vlastní

10.5.2 Převod regulárního výrazu na konečný automat a zpět

Je dán regulární výraz RV = (ab + b*c)d. Tento výraz převedeme na konečný automat v několika krocích podle pravidel na obrázku číslo 30.

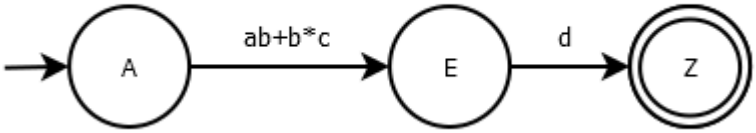
Obrázek 33 - Regulární výraz



Zdroj: vlastní

Nejdříve se zbavíme závorky podle prvního pravidla převodu. Tím nám vznikne nový stav *E*.

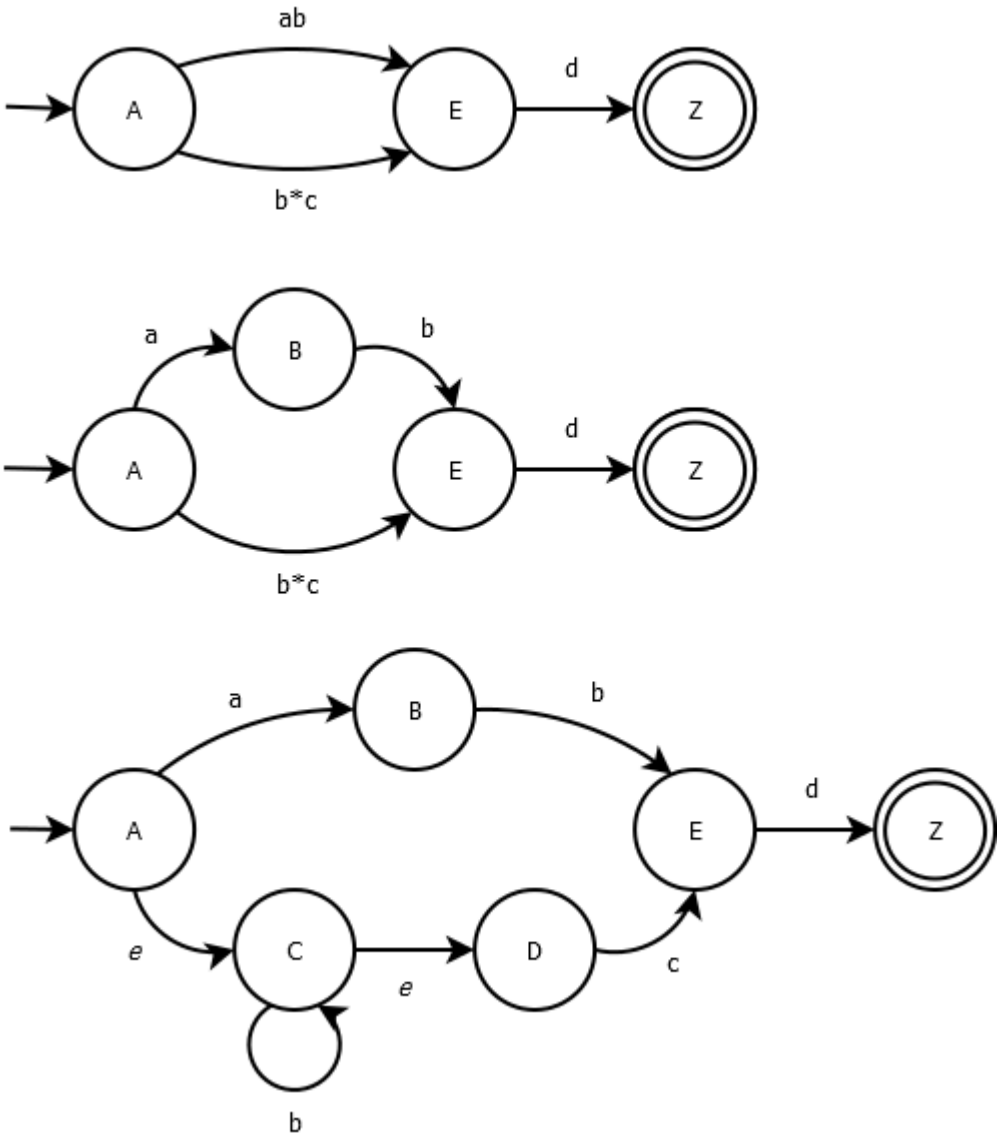
Obrázek 34 - Převod RV na KA 1



Zdroj: vlastní

Ze stavu *E* do *Z* vede hrana ohodnocená pouze jedním symbolem. Tím je převod v této části ukončen. Zbývá převést RV mezi stavy *A* a *E*.

Obrázek 35 - Převod RV na KA 2

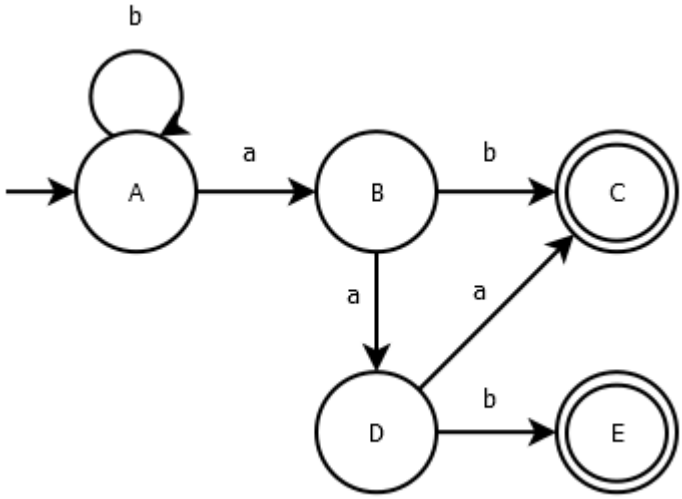


Zdroj: vlastní

Postupnou aplikací všech potřebných pravidel jsme rozebrali regulární výraz a převedli ho na konečný automat. Jak je vidět, každá hrana je nyní ohodnocena jedním symbolem, automat obsahuje počáteční a koncový stav a takto sestavený automat přijímá slova vyhovující regulárnímu výrazu RV = (ab + b*c)d.

Pro obrácený převod, tedy z konečného automatu na regulární výraz, použijeme pravidla z obrázku 31. Máme automat *A*, který je reprezentován stavovým diagramem.

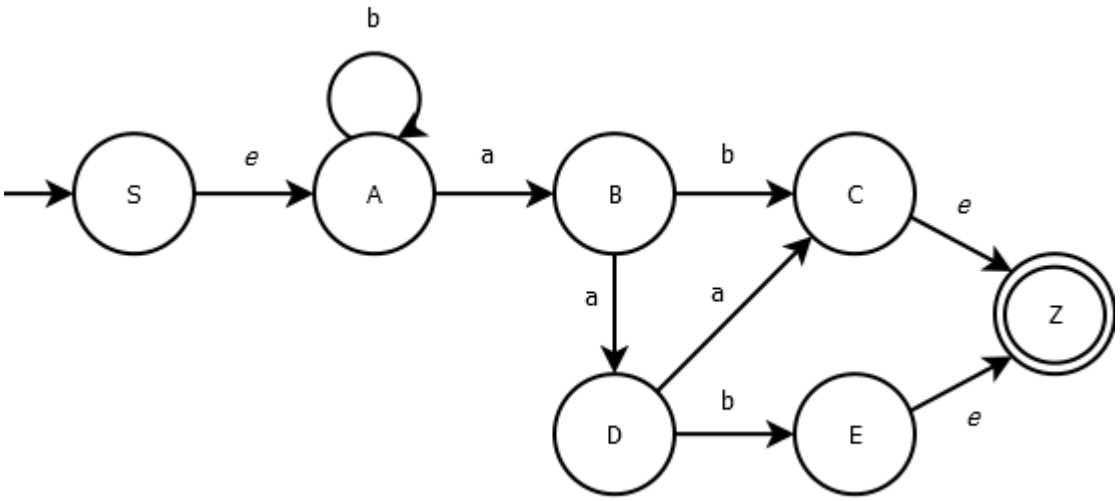
Obrázek 36 - Automat pro převod na RV



Zdroj: vlastní

V prvním kroku přidáme nový počáteční stav a sjednotíme dalším novým stavem dva stavy koncové. Nové stavy připojíme hranou ohodnocenou prázdným řetězcem *e*.

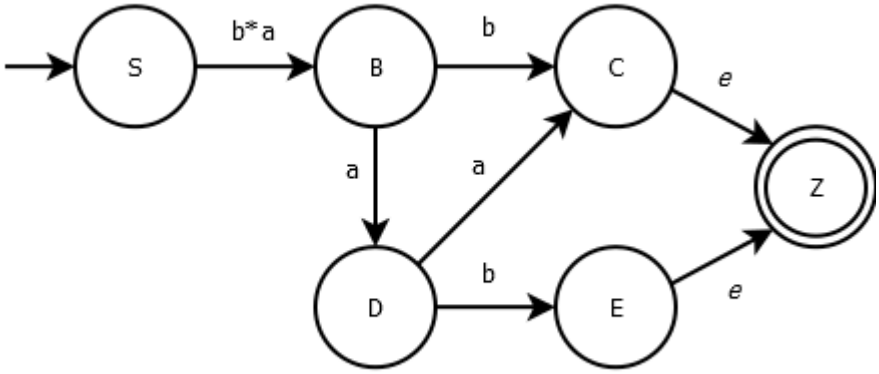
Obrázek 37 - Převod KA na RV 1



Zdroj: vlastní

Postupně eliminujeme původní stavy podle pravidel převodu a nahrazujeme je příslušnými ohodnocenými hranami. Po odebrání stavu *A* tak dostaneme diagram v tomto tvaru

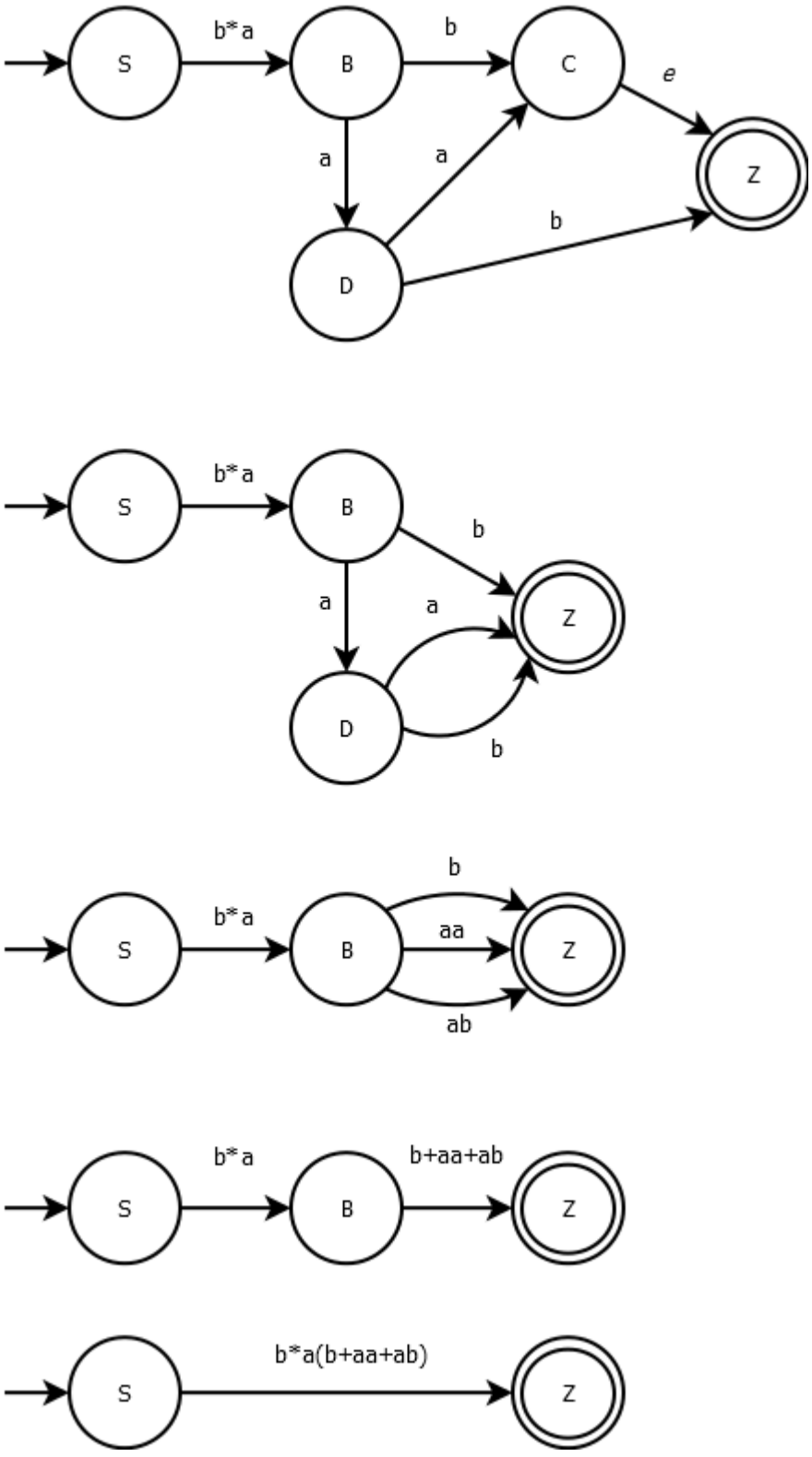
Obrázek 38 - Převod KA na RV 2



Zdroj: vlastní

Další stavy eliminujeme v pořadí *E, C, D, B*, až nám zůstane pouze počáteční a koncový stav s jedinou ohodnocenou hranou, což je výsledný regulární výraz RV = $b^*a(b+aa+ab)$.

Obrázek 39 - Převod KA na RV 3



Zdroj: vlastní

10.5.3 Převod regulární gramatiky na konečný automat a zpět

Pro převod využijeme pravidla, která jsme si definovali v teoretické části kapitoly 10. Regulární gramatiku G = ({S, A, B}, {a, b, c, d}, P, S) s přepisovacími pravidly

$S \rightarrow aS \mid dB$

$A \rightarrow d \mid aS$

$B \rightarrow bB \mid cA$

převedeme na konečný automat A = ({S, A, B, K}, {a, b, c, d}, δ, S, {K}). Stavy automatu jsou dány neterminálními symboly gramatiky s jedním novým, koncovým stavem *K*. Počáteční stav automatu je shodný s počátečním symbolem gramatiky. Terminální symboly zapíšeme jako znaky abecedy. Pro převod použijeme tabulkový zápis, ve kterém záhlaví řádků tvoří stavy automatu a záhlaví sloupců vstupní abeceda. Pokud přepisovací pravidlo gramatiky nemá na pravé straně neterminální symbol, přechodová funkce povede do nového stavu *K*.

Tabulka 8 - Převod RG na KA

	a	**b**	**c**	**d**
→ S	S			B
A	S			K
B		B	A	
← K				

Zdroj: vlastní

Převedení konečného automatu na regulární gramatiku funguje reverzním způsobem. Pokud přechodová funkce vede do koncového stavu, zapíše se kromě standardního převodu i pravidlo bez neterminálního symbolu. V případě, že počáteční stav je zároveň i koncovým, budou přepisovací pravidla obsahovat navíc přepis S → *e*. Ukážeme si to na příkladu automatu A = ({q_0, q_1, q_2, q_3}, {0, 1}, δ, q_0, {q_2, q_3}).

Tabulka 9 - Převod KA na RG

	0	**1**
→ **q_0**	q_1	q_2
q_1	q_0	q_3
q_2	q_2	q_1
← **q_3**	q_1	q_2

Zdroj: vlastní

Podle pravidel převodu bude výsledná gramatika G = ({q_0, q_1, q_2, q_3}, {0, 1}, P, S) obsahovat tyto přepisovací pravidla.

$q_0 \rightarrow 0q_1 \mid 1q_2 \mid 1$

$q_1 \rightarrow 0q_0 \mid 1q_3 \mid 1$

$q_2 \rightarrow 0q_2 \mid 1q_1 \mid 0$

$q_3 \rightarrow 0q_1 \mid 1q_2 \mid 1$

11 BEZKONTEXTOVÉ GRAMATIKY A JEJICH VYUŽITÍ

11.1 Bezkontextové gramatiky

Už jsme si ukázali praktickou důležitost u regulárních gramatik a jejich návaznosti na konečné automaty. Občasnou slabinou se však může stát právě onen konečný počet stavů. Na první pohled jednoduchý jazyk $L = (0^i1^i, i \geq 0)$ není regulární. Konečný automat si není schopen pamatovat „nekonečnou“ posloupnost jednoho ze symbolů a tu poté opakovat pro druhý symbol. Řešení nabízí právě bezkontextové gramatiky s využitím zásobníkových automatů. Z kapitoly o gramatikách už známe i definici. Jedná se o typ 2 z hierarchie podle Chomského. Bezkontextová gramatika G = (N, T, P, S) obsahuje tedy množinu neterminálů *N*, terminálů *T*, počáteční neterminál *S* a přepisovací pravidla *P* ve tvaru $A \rightarrow \beta$, kde β je řetězec složený z terminálů a neterminálů. Z definice přepisovacího pravidla vidíme, že neterminál na levé straně je vždy samotný, jeho přepsání je tedy nezávislé na kontextu řetězce, proto bezkontextová gramatika. Jazyk generovaný gramatikou *G* budeme značit L (G). Jedná se o takovou množinu slov *w* nad abecedou *Σ*, která je možné vygenerovat přepisovacími pravidly z kořene *S*. Je možné, že jazyk tvořený bezkontextovou gramatikou bude regulární. Je třeba si ale uvědomit, že i když každá regulární gramatika je zároveň bezkontextová, obráceně toto tvrzení neplatí. (Jančar, 2010, s. 127 – 130)

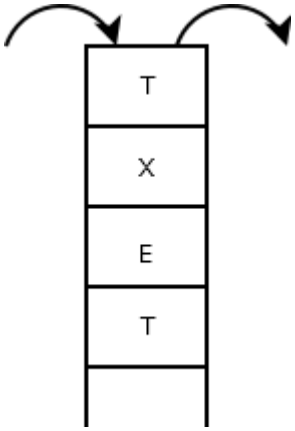
V praxi můžeme bezkontextové jazyky vidět mezi programovacími jazyky jako součást kompilátorů. Ke každému párovému klíčovému slovu syntaxe tak máme zajištěné upozornění v případě, že nám jedno slovo z páru bude chybět. Při psaní HTML kódu budeme informováni o chybějícím ukončujícím tagu, v jazyce Java například o neukončené proceduře slovem *end*.

11.2 Zásobníkový automat

Stejně jako existuje vztah mezi konečnými automaty a regulárními gramatikami, existuje vztah mezi zásobníkovým automatem a bezkontextovou gramatikou. Platí proto, že pro libovolnou bezkontextovou gramatiku *G* lze zkonstruovat zásobníkový automat *Z*, pro který platí Z (R) = L (G). Zásobníkový automat obsahuje jednu potenciálně

neomezenou paměť, do které lze ukládat mezivýsledky a která funguje na bázi LIFO. LIFO je zkratka pro *last in first out* metodu, která umožňuje automatu pracovat vždy s naposledy vloženým údajem (obrázek 40). Změna stavu automatu závisí na aktuálním stavu, na přečteném symbolu a na údaji v zásobníku. Při tom může automat uložit do zásobníku další znak či řetězec. Zásobníkový automat přijme slovo v případě, že končí v některém z koncových stavů, jako u klasických konečných automatů, nebo pokud je zásobník prázdný. Díky této paměti jsme nyní schopni vyřešit problém s jazykem $L = (0^i1^i, i \geq 0)$. (Vaníček, Papík, Pergl, Vaníček, 2007, s. 201 - 203)

Obrázek 40 - Princip LIFO



Zdroj: vlastní

Definujme tedy zásobníkový automat jako uspořádanou sedmici $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde

- Q je neprázdná konečná množina stavů
- Σ je neprázdná konečná množina vstupních symbolů
- Γ je neprázdná konečná množina zásobníkových symbolů (zásobníková abeceda)
- δ je přechodová funkce, jedná se o zobrazení množiny $Q \times (\Sigma \cup \{e\}) \times \Gamma \rightarrow (Q \times \Gamma^*)$
- q_0 je počáteční stav, $q0 \in Q$
- Z_0 je počáteční zásobníkový symbol, $Z_0 \in \Gamma$
- F je množina koncových stavů, $F \subseteq Q$

(Vavrečková, 2007, s. 22)

Přechodové funkce δ zapisujeme jako přiřazení trojici (q, a, A) množinu dvojic (q_i, γ_i). Stavy q a q_i jsou z množiny Q, a je vstupní symbol, A symbol na vrcholu zásobníku a γ_i řetězec, který se zapíše do zásobníku. Do zásobníku je řetězec zapsán zprava doleva, na vrcholu tedy bude nejlevější znak řetězce. (Černá, Křetínský, Kučera, 2002, s. 77)

11.3 Praktický příklad

Využití zásobníkového automatu si v praxi ukážeme na matematickém příkladu, který obsahuje i několikanásobně vnořené závorky. Při kontrole musí automat příklad vzít a zjistit, zda použití závorek odpovídá normám a zda jich je správný počet. Do zásobníkové paměti se budou ukládat otevírací části závorek, při zjištění uzavírací části bude závorka z paměti odebrána. Automat umí rozlišovat tři typy závorek – složenou, hranatou a kulatou. Každá má svoji prioritu. Složená závorka s nejvyšší prioritou může obsahovat ostatní druhy závorek, nemůže ale být ukončena, dokud nejsou ostatní závorky v ní obsažené také ukončeny. Hranaté závorky mají střední prioritu, mohou se vyskytovat uvnitř složených závorek, ale mohou být ukončeny pouze v případě, že kulaté závorky uvnitř jsou také ukončené. Kulaté závorky v sobě nemohou obsahovat žádné jiné a mají proto nejnižší prioritu.

Uživatel pomocí ovládacích prvků programu nejdříve zapíše celý matematický příklad, včetně operátorů a závorek. Stisknutím tlačítka pro kontrolu dojde k vyhodnocení příkladu z hlediska správnosti zadání závorek. V případě nalezení chyby bude uživatel upozorněn.

Pro potřeby praktického příkladu je dán zásobníkový automat A = ({q0, q1, q2q, q3}, {0 .. 9, +, -, *, /, (,), [,], {, }}, {S, H, K}, δ, q_0, Z_0, q_0). Zásobníková abeceda je písmenné ohodnocení závorek ve výrazu. Složená závorka s nejvyšší prioritou je reprezentována symbolem *S*, hranatá *H* a kulatá *K*. Přechodové funkce automatu jsou tyto:

$\delta(q_0, (, Z_0) = (q_1, KZ_0)$ $\quad$ $\delta(q_1,), K) = (q_0, e) \mid (q_2, e) \mid (q_3, e)$

$\delta(q_0, [, Z_0) = (q_2, HZ_0)$ $\quad$ $\delta(q_2,], H) = (q_0, e) \mid (q_3, e)$

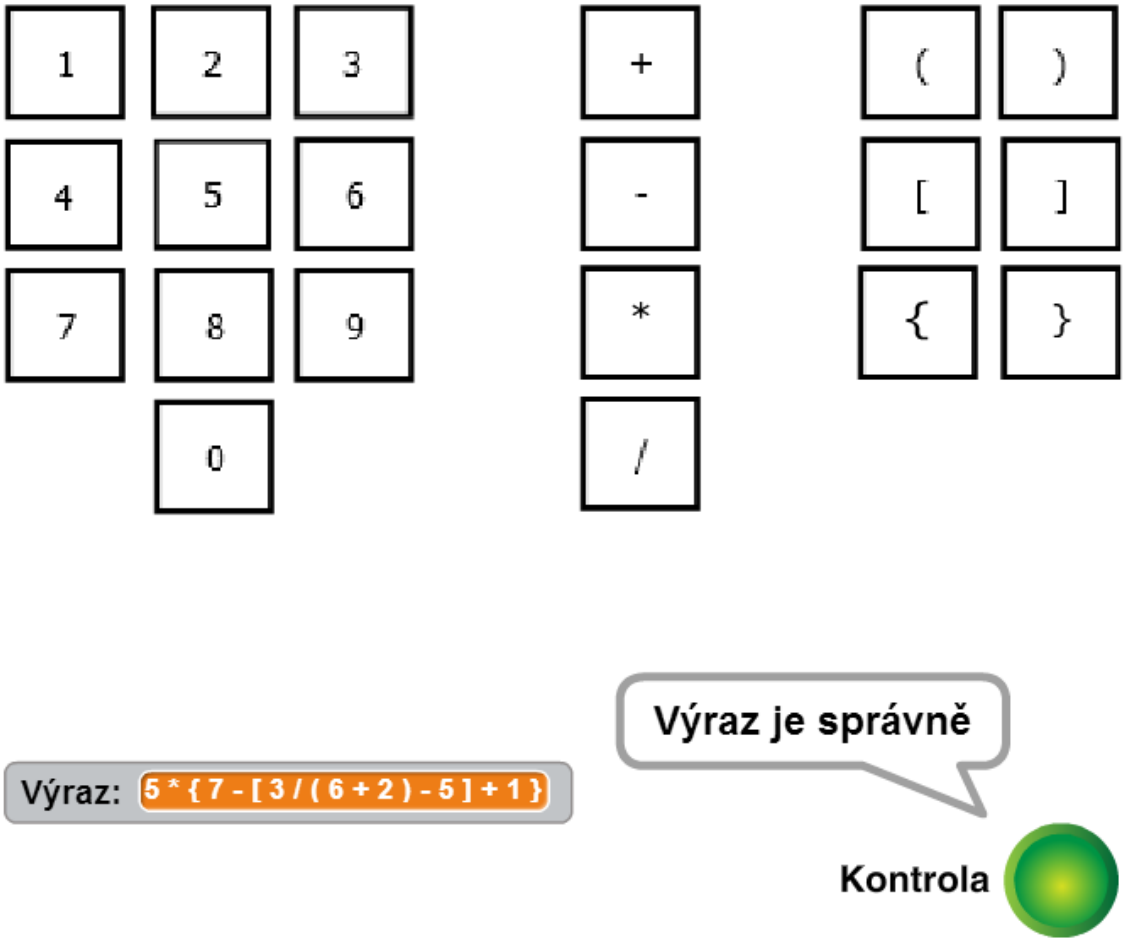
$\delta(q_0, \{, Z_0) = (q_3, SZ_0)$ $\quad$ $\delta(q_2, (, H) = (q_1, KH)$

$\delta(q_0, e, Z_0) = (q_0, e)$ $\delta(q_3, \}, S) = (q_0, e)$

$\delta(q_3, [, S) = (q_2, HS)$ $\delta(q_3, (, S) = (q_1, KS)$

Na obrázku 41 je vidět zpracování zásobníkového automatu určujícího správnost použití závorek ve výrazu. Automat tedy nekontroluje jeho matematickou správnost. Je možné zadat na první pohled nesmyslný zápis znaků, pokud však bude obsahovat závorky ve správném pořadí a počtu, bude automatem přijat jako bezchybný.

Obrázek 41 - Ověření správného zadání závorek



Zdroj: vlastní

ZÁVĚR

V současné době se prakticky denně setkáváme s konečnými automaty a jejich projevy v okolí. Tyto projevy podvědomě registrujeme jako správný způsob fungování věcí, neuvědomujeme si už, že za tím stojí právě konečné automaty.

Cílem práce bylo navrhnout komponenty pro kurz předmětu Konečné automaty a formální jazyky tak, aby demonstrovali jednoduchým interaktivním způsobem využití automatů v praktickém životě. Pro toto zpracování byl využit programovací jazyk Scratch, který vhodně kombinuje jednoduchost a grafickou stránku výsledné animace. Tím bylo zajištěno vytvoření uživatelsky příjemných příkladů z nudných, složitých a na první pohled často těžko pochopitelných definic. Na těchto příkladech si student předmětu sám může vyzkoušet použití probírané látky v praxi.

Ve výsledku tedy dostáváme osm vytvořených komponent na základě sylabu předmětu KAFJB. Pro příklady převodů automatů na regulární výrazy a gramatiky a pro převod z nedeterministického automatu na deterministický není komponenta vytvořena. Jedná se pouze o opakující se aplikaci převodních pravidel, která je důkladně popsána v příslušných kapitolách. Také redukce a ekvivalence automatu je popsána pouze teoreticky, dávaje tak návod k použití těchto technik pro libovolný vytvořený automat před uvedením do ostrého provozu.

Zdrojovému kódu komponent není těžké porozumět. Důležité části jsou navíc opatřeny komentáři pro možnou budoucí úpravu. Pedagog využívající tyto animace proto může jednoduchou změnou některé části přizpůsobit výsledný vzhled a funkčnost svým potřebám.

RESUMÉ

Main goal of this bachelor thesis is designing components for Finite automata and formal languages course. These components have to demonstrate usage of finite automata in real life in an easy and interactive way.

In the teoretical part are definitions of finite automata notions and descriptions of practical examples.

In the practical part are created components. It was used programming language Scratch for components processing. It conveniently combines simplicity with graphic animation. There were created user friendly examples from individual definitions by using Scratch. On these examples can students themselves try to use study curriculum in practice. There are eight components – one of them for a different chapter. Chapters dealing with transfers are solved only in theory as instructions for use.

SEZNAM POUŽITÉ LITERATURY

CHYTIL, Michal. *Automaty a gramatiky*. 1.vyd. Praha: Státní nakladatelství technické literatury, 1984. 336 s. Matematický seminář SNTL.

ČERNÁ, Ivana, KŘETÍNSKÝ, Mojmír, KUČERA, Antonín. *Automaty a formální jazyky I* [online]. Verze 1.3. Brno: Masarykova univerzita. Fakulta informatiky, 2002 [cit. 2015-05-20]. Dostupné z: http://is.muni.cz/elportal/estud/fi/js06/ib005/Formalni_jazyky_a_automaty_I.pdf

JANČAR, Petr. *Teoretická informatika* [online]. 1.vyd. Ostrava: Vysoká škola báňská - Technická univerzita, 2010 [cit. 2015-05-04]. ISBN 978-80-248-1487-2. Dostupné z: http://www.cs.vsb.cz/jancar/TEORET-INF/ti-text.2010-01-20.pdf

SIPSER, Michael. *Introduction to the theory of computation*. 3rd ed. Massachusett: Cengage Learning, 2013. 458 p. ISBN 978-1-133-18781-3.

KOCUR, Pavel. *Úvod do teorie konečných automatů a formálních jazyků*. [online]. Plzeň: Západočeská univerzita, 2000 [cit. 2015-05-18]. Dostupné z: http://www.kvd.zcu.cz/cz/materialy/kafj/_kafj1.html

VANÍČEK, Jiří, PAPÍK, Martin, PERGL, Robert, VANÍČEK, Tomáš. *Teoretické základy informatiky*. 1.vyd. Praha: Kernberg, 2007. 431 s. ISBN 978-80-903962-4-1.

VAVREČKOVÁ, Šárka. *Teorie jazyků a automatů II* [online]. Opava: Slezská univerzita – Ústav informatiky, 2007 [cit. 2015-06-14]. Dostupné z: http://vavreckova.zam.slu.cz/obsahy/tja2/skripta/tja2_celek.pdf

KOZEL, Tomáš. *Teorie konečných automatů, regulárních gramatik, jazyků a výrazů* [online]. Hradec Králové: Univerzita Hradec Králové, 2008 [cit. 2015-04-11]. Dostupné z: http://iris.uhk.cz/tein/index.html

XAVIER, S.P. Eugene. *Theory of automata, formal languages and computation*. New Delhi: New Age International Ltd., Publishers, 2005. ISBN 978-81-224-2334-1.

HABIBALLA, Hashim. *Gramatiky a jazyky* [online]. Ostrava: Ostravská univerzita, 2013 [cit. 2015-05-20]. Dostupné z: http://www1.osu.cz/home/habibal/kurzy/GRAJA.pdf

HOPCROFT, John E., Rajeev MOTWANI a Jeffrey D. ULLMAN. *Introduction to automata theory, languages, and computation*. 2nd ed., Boston: Addison-Wesley, 2001. ISBN 0-201-44124-1.

JANČAR, Petr. *Studijní opora k předmětu Teoretická informatika: speciálně v oblasti jazyků a automatů* [online]. Ostrava: 2003 [cit. 2015-05-05]. Dostupné z: http://www.cs.vsb.cz/jancar/TJAA/tjaa_2p.pdf

KŘETÍNSKÝ, Mojmír. *Automaty a formální jazyky I.* [Přednáška]. Praha: ČVUT, 1991.

PŘIBÁŇ, Tomáš. *Konečné automaty a formální jazyky* [Přednáška]. Plzeň: ZČU, 14.04.2014.

PAVLÁSEK, Ondřej. Grafy a grafové algoritmy. Brno, 2010. Bakalářská práce. Vysoké učení technické v Brně. Fakulta podnikatelská.

Hanojské věže. In: *www.hlavolamy.info* [online]. 2012 [cit. 2015-04-06]. Dostupné z: http://www.hlavolamy.info/news/hanojska-vez/.

SEZNAM POUŽITÝCH ZKRATEK

KAFJB – konečné automaty a formální jazyky

KA – konečný automat

DKA – deterministický konečný automat

NKA – nedeterministický konečný automat

RV – regulární výraz

RG – regulární gramatika

LIFO – last in, first out

SEZNAM TABULEK

SEZNAM PŘÍLOH

Na přiloženém kompaktním disku jsou vloženy tyto přílohy – komponenty:

2_DKA – komponenta pro kapitolu Deterministický konečný automat bez výstupu

3_KlasifikacniAutomat – komponenta pro kapitolu Klasifikační automat

4_NKA – komponenta pro kapitolu Nedeterministický konečný automat

6_Mealy_Moore – komponenta pro kapitolu Automaty s výstupem

8_Jazyky – komponenta pro kapitolu Formální jazyky

9_Gramatiky – komponenta pro kapitolu Gramatiky

10_RV – komponenta pro kapitolu Regulární výrazy v konečných automatech a formálních jazycích

11_Zasobnik – komponenta pro kapitolu Bezkontextové gramatiky a jejich využití