

# Aplikace fuzzy a pravděpodobnostních automatů

Martin Jašek

12. září 2016 — ??

## Obsah

<b>1</b>	<b>Úvodní pojmy</b>	<b>4</b>
	1.1 Fuzzy teorie . . . . .	4
	1.2 Abecedy, řetězce, jazyky . . . . .	4
	1.3 Jazyky . . . . .	4
	1.4 Ostatní důležité pojmy . . . . .	5
10	1.4.1 If-Then pravidla . . . . .	6
	1.5 Další pojmy z úvodu do teorie automatů . . . . .	7
	1.5.1 Gramatiky . . . . .	7
<b>2</b>	<b>Definice fuzzy automatu</b>	<b>8</b>
	2.1 Koncept automatu . . . . .	9
	2.2 Nedeterministický bivalentní automat . . . . .	10
	2.3 Nedeterministický fuzzy automat . . . . .	10
	2.4 Reprezentace nedeterministického fuzzy automatu . . . . .	11
	2.5 Výpočet nedeterministického fuzzy automatu . . . . .	11
<b>3</b>	<b>Další varianty fuzzy automatů</b>	<b>14</b>
20	3.1 Zobecněný automat . . . . .	14
	3.2 Fuzzy automat s výstupem . . . . .	14
	3.3 Fuzzy automat jako stroj . . . . .	14
	3.4 Nedeterministický fuzzy automat s $\epsilon$ -přechody . . . . .	14
	3.5 Deterministický fuzzy automat . . . . .	14
	3.6 Zásobníkový fuzzy automat . . . . .	14
<b>4</b>	<b>Fuzzy automaty, jazyky, gramatiky a regulérní výrazy</b>	<b>14</b>
	4.1 Jazyk rozpoznávaný fuzzy automatem . . . . .	14
	4.2 Fuzzy a bivalentní regulérní jazyky . . . . .	15
	4.3 Fuzzy regulární výrazy . . . . .	15
30	<b>5 Rozpoznávání textových vzorů</b>	<b>15</b>
	5.1 Formální zavedení problému . . . . .	15
	5.2 Motivace k použití fuzzy automatů . . . . .	16
	5.3 Automat rozpoznávající $\omega$ . . . . .	16
	5.4 Podobnost symbolů . . . . .	18
	5.5 Fuzzy symboly . . . . .	20
	5.6 Editační operace . . . . .	21
	5.7 Deformovaný automat . . . . .	26

	5.8	Shrnutí . . . . .	27
	<b>6</b>	<b>Fuzzy tree automaty</b>	<b>28</b>
40	6.1	Zavedení . . . . .	28
	6.2	Stromy a pseudotermy . . . . .	28
	6.3	Fuzzy tree automat a jazyk jím rozpoznávaný . . . . .	30
	6.4	Použití fuzzy tree automatů . . . . .	33
	6.5	Detekce úplných $m$ -árních stromů . . . . .	33
	6.6	Složené geometrické útvary . . . . .	34
	<b>7</b>	<b>Buněčné fuzzy automaty</b>	<b>37</b>
	7.1	„Bivalentní“ buněčný automat . . . . .	37
	7.2	Buněčné fuzzy automaty . . . . .	39
	7.3	Obecně k aplikacím . . . . .	41
50	7.4	Problém městského růstu (urban growt problem) . . . . .	42
	7.5	Zpracování obrazu . . . . .	43
	7.6	Hledání hran . . . . .	46
	7.7	Ostraňování šumu . . . . .	48
	7.8	Rozpoznávání jednoduchých vzorů . . . . .	48
	<b>8</b>	<b>Strojové učení</b>	<b>49</b>
	8.1	Neuronové sítě . . . . .	50
	8.2	Konstrukce neuronové sítě . . . . .	51
	8.3	Fuzzy automaty a neuronovky . . . . .	52
	8.4	Učící se fuzzy automaty . . . . .	53
60	8.5	Metoda lisování dat . . . . .	53
	8.6	Reálné příklady použití . . . . .	54
	<b>9</b>	<b>Biologie a medicína</b>	<b>55</b>
	9.1	Rozpoznávání řetězců DNA . . . . .	55
	9.2	Simulace růstu mořských řas . . . . .	55
	9.3	Rozpoznávání vzorů v lékařských snímcích . . . . .	55
	9.4	Analýza zdravotního stavu pacienta . . . . .	55
	9.5	Detekce zhoubných nádorů . . . . .	56
	9.6	Analýza kardiogramu . . . . .	56
	9.7	Řízení protézy . . . . .	57
70	<b>10</b>	<b>Další aplikace</b>	<b>58</b>
	10.1	Pattern matching, další příklady . . . . .	58
	10.2	Detekce požárů . . . . .	58
	10.3	Automobilismus . . . . .	58
	10.4	Sledování vytížení el. sítě . . . . .	59
	10.5	Modelování emocí postav v počítačových hrách . . . . .	59
	10.6	Modelování emocí . . . . .	60
	10.7	Sledování cen investic . . . . .	61
	10.8	Sledování pohybu a aktivit osoby . . . . .	61
	10.9	Řídící systémy, fuzzy kontroléry . . . . .	61
80	10.10	Ostatní a nezařezeno . . . . .	62
	<b>11</b>	<b>Pravděpodobností automaty</b>	<b>62</b>

<b>12 Konkrétní příklady</b>	<b>63</b>
12.1 Automat jako „lepší“ reprezentace něčeho . . . . .	63
12.2 Rozpoznávání ručně psaného textu . . . . .	63
12.3 Detekce překlepů . . . . .	64

# 1 Úvodní pojmy

V této kapitole budou popsány základní koncepty a pojmy, bez kterých nebude možno se studiu fuzzy automatů věnovat. Bude zde tedy představen koncept fuzzy množin a základní pojmy z teorie automatů.

## 90 1.1 Fuzzy teorie

*(zde bude doplněno: Zde se důkladně rozepsat o motivaci a konceptech.) (zde bude doplněno: ZDE dát všechny ty příklady na „je teplo“, „je mladý“ a podob. !)*

Fuzzy množiny a relace budou po vzoru [13] nejčastěji malými řeckými písmeny (obdobně jako jejich členské (angl. „membership“) funkce). Množinu všech fuzzy podmnožin množiny  $S$  budeme značit  $\mathcal{F}(S)$ .

## 1.2 Abecedy, řetězce, jazyky

Definice a značení následujících pojmů jsou převzaty z [?].

### Abeceda

100 Základním pojmem při studiu automatů je abeceda. Abeceda je neprázdná a konečná množina symbolů a značí se  $\Sigma$ , případně jiným velkým písmenem řecké abecedy. Abecedou může být například „všechna malá písmena latinky“, nebo např. číslice 0 – 9.

### Řetězec

Posloupnost  $u = a_1 a_2 \dots a_n$  kde  $a_1, a_2, \dots, a_n \in \Sigma$  se nazývá řetězec  $u$  nad abecedou  $\Sigma$ . Číslo  $n$  je pak délka řetězce  $u$ , která se jinak značí  $|u|$ . Řetězec, který má nulovou délku, značíme  $\varepsilon$ .

Řetězec  $u \circ v = a_1 \dots a_n b_1 \dots b_m$  (častěji však  $uv$ ) se nazývá zřetězení (konkatenace) řetězců  $u = a_1 \dots a_n$  a  $v = b_1 \dots b_m$ . Přirozeně platí  $|uv| = n + m$ . Jako  $n$ -tá mocnina  $u^n$  řetězce  $u$  se označuje řetězec:

$$u^n = \begin{cases} \varepsilon & \text{pokud } n = 0 \\ uu^{n-1} & \text{jinak} \end{cases}$$

Symbolem  $\Sigma^*$  se značí množina všech řetězců nad abecedou  $\Sigma$  (včetně  $\varepsilon$ ). Symbol  $\Sigma^+$  pak značí všechny řetězce nad abecedou  $\Sigma$  vyjma  $\varepsilon$ .

## 1.3 Jazyky

110 Pojmem (formální) jazyk se označuje určitá vybraná množina  $L$  řetězců nad abecedou  $\Sigma$ , tj.  $L \subseteq \Sigma^*$ .

## Bivalentní jazyk

Nad jazyky  $L$ ,  $L_1$  a  $L_2$  nad abecedami  $\Sigma$ ,  $\Sigma_1$  a  $\Sigma_2$  se zavádí:

$$L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\} \quad \text{zřetězení (produkt)}$$

$$L^n = \begin{cases} \{\varepsilon\} & \text{pokud } n = 0 \\ LL^{n-1} & \text{jinak} \end{cases} \quad n\text{-tá mocnina}$$

$$L^* = \bigcup_{i=0}^{\infty} L^i \quad \text{Kleeneho uzávěr}$$

$$L^+ = \bigcup_{i=1}^{\infty} L^i \quad \text{pozitivní uzávěr}$$

Je-li jazyk konečná množina, tj, obsahuje konečně mnoho řetězců, nazýváme jej konečný. V opačném případě říkáme, že je jazyk nekonečný.

## Fuzzy jazyk

(zde bude doplněno: Doplnit) (zde bude doplněno: Fuzzy jazyk by měl přijít až po zavedení fuzzy teorie. Je to ok?)

## 1.4 Ostatní důležité pojmy

### Nedeterministický fuzzy automat s $\epsilon$ přechody

{def-NedFuzzAutEpsPre}

120 **Definice 1.1** (Nedeterministický fuzzy automat s  $\epsilon$  přechody). (zde bude doplněno: dohledat přesně, zkontrolovat a ozdrojovat) Nedeterministický fuzzy automat  $A$  je pětice  $(Q, \Sigma, \mu, \sigma, \eta)$ , kde  $Q$  je konečná množina stavů,  $\Sigma$  je abeceda,  $\mu$  je fuzzy přechodová funkce (fuzzy relace  $Q \times (\Sigma \cup \{\epsilon\}) \times Q \rightarrow [0, 1]$ ) a  $\sigma$  a  $\eta$  jsou po řadě fuzzy množiny nad  $Q$  počátečních, resp. koncových stavů.

(zde bude doplněno: Tady by asi bylo vhodné rozebřit  $\epsilon$ -uzávěry)

### Reprezentace fuzzy automatu

(zde bude doplněno: dohledat zdroje)

(zde bude doplněno: Značení fuzzy množiny  $\sigma = \{x/0.5\}$  vs.  $\sigma(x) = 0.5$ )

Přechodový diagram: Notace s lomítky např. zde: [14].

130 Tabulka: stav  $\times$  symbol nebo stav  $\times$  stav [15]?

### Konstrukce fuzzy automatu z konečného automatu

V praxi se často setkáme z problémem, kdy máme k dispozici konečný bivalentní automat avšak my potřebujeme pro naši práci fuzzy automat. Je tedy třeba zkonstruovat takový fuzzy automat, který rozpoznává odpovídající jazyk odpovídající jazyku rozpoznávaném naším bivalentním automatem.

Důležité je zmínit, že nelze zkonstruovat fuzzy automat, rozpoznávající stejný jazyk neboť fuzzy automat rozpoznává fuzzy jazyk, zatímco bivalentní automat klasický „bivalentní“ jazyk. Můžeme však sestavit automat takový, který přijímá řetězce ve stupni 0 nebo 1 podle toho, jestli je přijímal bivalentní automat.

Formálně řečeno, pro konečný (nedeterministický) bivalentní automat  $A$  budeme konstruovat nedeterministický fuzzy automat  $A'$  takový, že bude pro všechna  $x \in \Sigma^*$  splněna následující rovnost:

$$\mathcal{L}(A')(\omega) = \begin{cases} 1 & \text{pokud } \omega \in \mathcal{L}(A) \\ 0 & \text{pokud } \omega \notin \mathcal{L}(A) \end{cases}$$

140 **Poznámka 1.1.** *Postup budeme provádět pro nedeterministické automaty. To jednak proto, že nedeterministické automaty jsou obecnější, než deterministické, a navíc, protože jsou v praxi využívány častěji. (zde bude doplněno: ozdrojovat, klidně někde, kde rozebírám determinizmus vs. nedeterminizmus)*

Nyní se podíváme na to, jak výsledný fuzzy automat bude vypadat. Abeceda i množina stavů automatu zůstanou zachovány, lišit se tedy bude pouze množina počátečních a koncových stavů a přechodová funkce. (zde bude doplněno: fuzzy subset  $I$ ,  $F$  a  $\delta$ ? nebo tak něco, z teorie fuzzy množin?)

150 **Definice 1.2** (Fuzzy automat bivalentního automatu). *Mějme řetězec konečný nedeterministický automat  $A = (Q, \Sigma, \delta, I, F)$ . Pak nedeterministický fuzzy automat přijímající korespondující jazyk je automat  $A' = (Q, \Sigma, \mu, \sigma, \epsilon)$  kde pro všechna  $q_i, q_j \in Q$  a  $x \in \Sigma$ :*

{def-FuzzAutBivAut}

- $\sigma(q_i) = \begin{cases} 1 & \text{pokud } q_i \in I \\ 0 & \text{pokud } q_i \notin I \end{cases}$
- $\eta(q_i) = \begin{cases} 1 & \text{pokud } q_i \in F \\ 0 & \text{pokud } q_i \notin F \end{cases}$
- $\mu(q_i, x, q_j) = \begin{cases} 1 & \text{pokud } q_j \in \delta(q_i, x) \\ 0 & \text{pokud } q_j \notin \delta(q_i, x) \end{cases}$

(zde bude doplněno: rozebrat, jestli tento automat skutečně dělá to, co má? Asi by to chtělo)

(zde bude doplněno: vymyslet nějaký fakt pěkný příklad)

#### 1.4.1 If-Then pravidla

160 V mnoha aplikacích se setkáme se situací, kdy budeme potřebovat popsat určitou funkčnost či chování systému „lidským“ způsobem. Velmi častým způsobem, jak toho dosáhnout je s pomocí tzv. If-Then pravidel. Popis If-Then pravidel byl převzat z [?].

If-Then pravidlo je výrok je tvaru:

Jestliže  $x_1, \dots, x_n$ , pak  $y$

kde  $x_1, \dots, x_n, y$  jsou podvýroky. Za předpokladu, že jazyk (a později i struktura pro tento jazyk) jsou nám známy z kontextu, můžeme tento výrok zapsat jako formuli<sup>1</sup>:

$$x_1 \wedge \dots \wedge x_n \Rightarrow y$$

<sup>1</sup>Formule If-Then pravidel jsou vlastně Hornovské klauzule. Práce s If-Then pravidly se tak podobá logickému programování.

**Příklad 1.1.** *Následující výroky jsou If-Then pravidla:*

- *Jestliže je spínač sepnut, pak žárovka svítí*
- *Jestliže je věk( $p$ ) < 15, pak  $p$  je dítě*
- *Je-li  $x \in \mathbb{N}$ ,  $x > 2$ ,  $\nexists y < x : y|x$ , pak  $x$  je prvočíslo*

Určení pravdivosti pravidla, tedy jestli je  $y$  pravdivý nebo nepravdivý, se pak provede standardně, tj. určení pravdivosti formule pravidla ve struktuře. Pokud máme If-Then pravidel více, stačí, aby bylo pravdivé jen jedno z nich.

S fuzzy If-Then pravidly se obvykle pracuje mírně složitěji. Pravidla, tj. výroky se typicky uvádějí v mírně pozměněném tvaru, a to:

Jestliže  $x_1$  je  $X_1, \dots, x_n$  je  $X_n$ , pak  $y$  je  $Y$

případně

Jestliže  $x_1 = X_1, \dots, x_n = X_n$ , pak  $y = Y$

Symbole  $x_1, \dots, x_n, y$  označují termíny a symbole  $X_1, \dots, X_n, Y$  fuzzy množiny nad množinami těchto termínů. Značení  $x = X$  je ekvivalent zápisu  $X(x)$ , tj. „ $x$  je v množině  $X$  ve stupni  $X(x)$ “. Zapsáno jako formule tedy:

$$X_1(x_1) \wedge \dots \wedge X_n(x_n) \Rightarrow Y(y)$$

170 U fuzzy If-Then pravidel nemá smysl uvažovat jejich pravdivost jako takovou. Jejím výsledkem je totiž fuzzy množina.

**Příklad 1.2.** *Uvažujme systém, u kterého známe tlak a teplotu a popisujeme úroveň otevření ventilu. Označme  $p_L, p_N, p_H$  jako fuzzy množiny „tlak je nízký“, „tlak je normální“, „tlak je vysoký“ a dále  $t_N, t_H$  jako „teplota je v normě“ a „teplota je zvýšená“. Dále zavedme fuzzy množiny  $v_C$  a  $v_O$  jako „ventil je uzavřen“ a „ventil je otevřen“.*

*Pak pravidla*

*Pokud je tlak nízký a teplota zvýšená, pak ventil je uzavřen*

*Pokud je tlak vysoký a teplota v normě, pak ventil je otevřen*

*budou zapsána jako formule*

$$p_L(p) \wedge t_H(t) \Rightarrow v_C(v)$$

$$p_H(p) \wedge t_N(t) \Rightarrow v_O(v)$$

**Poznámka 1.2.** *Pokud máme fuzzy If-Then pravidel více, pak je spojujeme disjunkcí.*

## 1.5 Další pojmy z úvodu do teorie automatů

### 180 1.5.1 Gramatiky

Gramatiky (přesněji „formální gramatiky“) jsou nástroje pro popis jazyků. Gramatika určuje, jakým způsobem lze jazyk vygenerovat. Následující pojmy jsou převzaty z [?].

## Gramatika

**Definice 1.3** (Formální gramatika). *Jako formální gramatiku označujeme čtveřici  $(N, T, P, S)$ , kde*

1.  $N$  je abeceda tzv. *neterminálních symbolů*
2.  $T$  je abeceda tzv. *terminálních symbolů* ( $T \cap N = \emptyset$ )
3.  $P$  je množina *přechodových pravidel*, kde každé pravidlo je ve tvaru  $x \rightarrow y$ ,  
190  $x, y \in (N \cup T)^*$  a  $x$  obsahuje alespoň jeden *neterminál*
4.  $S$  je *počáteční neterminál* ( $S \in N$ )

Mějme gramatiku  $G = (N, T, P, S)$  takovou, že  $X \in N$ ,  $y \in T$  a  $X \rightarrow y \in P$ . Pak pro libovolné  $u, v \in (N \cup T)^*$  o řetězci  $uyv$  říkáme, že vznikl přímým odvozením z řetězce  $uXv$  (píšeme  $uXv \Rightarrow uyv$ ). Posloupnost  $w_1 \Rightarrow \dots \Rightarrow w_n$  řetězců  $w_1, \dots, w_n \in (N \cup T)^*$  zapisujeme  $w_1 \Rightarrow^* w_n$  a nazýváme *derivace*.

Jako jazyk generovaný gramatikou  $G$  značíme následující množinu:

$$L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$$

**Příklad 1.3.** Mějme gramatiku  $G = (N, T, P, S)$ , kde  $N = \{M\}$ ,  $T = \{a, b\}$ ,  
200  $P = \{S \rightarrow aM, M \rightarrow Sb, M \rightarrow b\}$ . Pak platí  $S \Rightarrow aM \Rightarrow aSb \Rightarrow aaMb \Rightarrow aabb$  a  $L(G) = \{a^n b^n \mid n \geq 1\}$ .

## Regulérní gramatika

200 Regulérní gramatika je speciálním případem gramatiky. Tento typ gramatiky má zásadní vliv pro studium automatů.

**Definice 1.4** (Regulérní gramatika). *Gramatika  $G = (N, T, P, S)$  se nazývá regulérní, pokud každé z jejích pravidel je ve tvaru  $X \Rightarrow aY$  nebo  $X \Rightarrow a$ , kde  $X, Y \in N$  a  $a \in T$  a dále platí, že  $S$  se nevyskytuje na pravé straně žádného pravidla.*

Jazyk generovaný regulérní gramatikou se nazývá *regulérní jazyk*.

**Příklad 1.4.** Regulérní gramatikou je například gramatika  $G = (N, T, P, S)$ ,  
kde  $N = \{R\}$ ,  $T = \{a, b\}$ ,  $P = \{S \rightarrow aR, R \rightarrow aR, R \rightarrow bR, R \rightarrow a\}$ .

210 Regulérní gramatiky (a jim odpovídající jazyky) mají spoustu zajímavých vlastností. Uvedeme zde pouze jednu důležitou vlastnost.

**Věta 1.1.** *Každý konečný jazyk  $L$  je regulérním jazykem.*

{the:FinRegLang}

*Důkaz.* Má-li být jazyk  $L$  regulérním pak pro něj musí existovat gramatika  $G$ , která jej generuje. Podrobnější důkaz lze najít v literatuře.  $\square$

## 2 Definice fuzzy automatu

V této kapitole bude podrobně popsán a formálně nadefinován „fuzzy automat“ a proces jeho výpočtu. Pro snazší ilustraci bude nejdříve popsán automat bivalentní a následně upraven do podoby fuzzy automatu. Popis bivalentního



automatu je převzat z [?], popis fuzzy automatu z [14] (pokud není uvedeno jinak).

220 Hned na úvod je třeba zdůraznit terminologii. Jak bývá u automatů zvykem, pojem „automat“ označuje souhrnně různé varianty automatů, typicky nedeterministický i deterministický, bez dalšího rozlišování. Vzhledem k tomu, že základním (a nejpoužívanějším) fuzzy automatem je automat nedeterministický, bude proto druhá polovina této kapitoly věnována právě nedeterministickému fuzzy automatu. Zbývající známé varianty fuzzy automatů budou poté shrnuty v následující kapitole.

*(zde bude doplněno: někde tady zdůraznit konečné automaty?)*

## 2.1 Koncept automatu

230 Automat se řadí mezi tzv. výpočetní modely. Výpočetní model je označení pro matematický formalismus, který popisuje určitý výpočet, algoritmus. Spolu s automaty (ve všech jejich variantách a modifikacích) se k výpočetním modelům řadí například také Turingovy stroje [?].

Automaty také často bývají nazývány jako stavové stroje. Na automat lze totiž nahlížet jako na určité zařízení. Toto zařízení je charakterizováno svým vnitřním stavem a v závislosti na vstupu se tento stav diskrétně mění, tj. můžeme tvrdit, že automat přechází od jednoho stavu k jinému.

Důležité je také dělení automatů na deterministické a nedeterministické. Obecná definice říká, že u deterministického automatu je jednoznačně dáno, do kterého stavu v každý moment výpočtu automat přejde. Naopak u nedeterministického tento předpoklad neplatí, tj. výpočet automatu se může octnout v 240 situaci, kdy má možnost přejít do dvou a více stavů „současně“.

Tuto situaci budeme u nedeterministických automatů reprezentovat tak, že automat se nebude nacházet vždy v jednom stavu, ale jeho aktuální stav bude popsán celou množinou stavů, ve kterých se nachází.

Automat tedy musí zcela určitě obsahovat stavy, ve kterých se při svém výpočtu může nacházet. Spolu s nimi je každý automat dán vstupy, se kterými dokáže pracovat. Vstupy pro automat budou řetězce nad nějakou danou abecedou. O popis přechodů mezi stavy se bude starat přechodová funkce.

250 Dále, u každého automatu musí být stanoven počáteční stav. Vzhledem k tomu, že se budeme bavit o automatu nedeterministickém, budeme uvažovat množinu počátečních stavů.

Automaty původně vznikly jako nástroje pro rozpoznávání určité třídy jazyků<sup>2</sup>. To znamená, že automat musí pro libovolný řetězec, zda-li do uvedeného jazyka patří nebo ne. U automatů je toto řešeno pomocí tzv. koncových stavů. Pokud výpočet automatu zkončí v koncovém stavu, pak je řetězec automatem zpracováváný přijat, v opačném případě zamítnut.

Nyní máme známou obecnou představu, jak by měl automat vypadat. Následuje tedy definice nedeterministického bivalentního automatu, spolu se stručným popisem jeho činnosti. Následně je pak odvozena definice nedeterministického fuzzy automatu a jeho výpočtu. 260

*(zde bude doplněno: potenční množina – bivalentní značím  $2^M$ , fuzzy  $\mathcal{F}(M)$ , nechtělo by to sjednotit a používat taky  $P(M)$ , resp.  $\mathcal{P}(M)$ ?)*

---

<sup>2</sup>Konkrétně, jedná se o tzv. regulérní jazyky, které budou popsány v kapitole *(zde bude doplněno: doplnit odkaz)*

## 2.2 Nedeterministický bivalentní automat

Nedeterministický bivalentní automat je definován následovně:

{def-NedBivAut}

**Definice 2.1** (Nedeterministický bivalentní automat). *Nedeterministický bivalentní automat  $A$  je pětice  $(Q, \Sigma, \mu, I, F)$ , kde  $Q$  je konečná množina stavů,  $\Sigma$  je abeceda,  $\mu : Q \times \Sigma \rightarrow 2^Q$  je přechodová funkce a  $I \subseteq Q$  a  $F \subseteq Q$  je množina počátečních a koncových stavů.*

(zde bude doplněno: reprezentace? příklad?)

270 Výpočet automatu, tedy zpracování vstupního řetězce, je definován jako posloupnost konfigurací. Konfigurace je přesný popis aktuálního stavu výpočtu (tzn. nezpracovaná část vstupního řetězce a množina stavů, ve kterých se automat nachází). Na počátku se výpočet nachází v tzv. počáteční konfiguraci, tj. nezpracovanou částí celého řetězce je celý vstupní řetězec a množinou stavů, ve kterých se automat nachází je celá množina  $I$ .

280 Automat čte ze vstupu postupně symboly. Pokud se automat nachází ve stavu  $q$  a právě přečteným symbolem je symbol  $a$ , pak automat přechází do stavu  $q'$  (a symbol  $a$  je ze vstupu odebrán) pokud  $q' \in \mu(q, a)$ . Vyprázdní-li takto automat celý vstup (na vstupu je prázdný řetězec), ocitá se v tzv. koncové konfiguraci. Pokud alespoň jeden ze stavů, ve kterém se automat nachází, je koncový, nazývá se tato konfigurace přijímací, v opačném případě zamítací.

Pokud výpočet automatu pro řetězec  $w$  zkončí přijímací konfigurací, říkáme, že automat řetězec přijímá. Pokud zkončí zamítací konfigurací, pak říkáme, že je řetězec zamítán. Jazyk rozpoznávaný automatem je pak množina takových řetězců  $w \in \Sigma^*$ , které automat přijímá.

290 Ekvivalentním způsobem, jak určit, zda-li je řetězec automatem přijímán či zamítán je pomocí rozšířené přechodové funkce  $\mu^* : 2^Q \times \Sigma^* \rightarrow 2^Q$ . Rozšířenou přechodovou funkci  $\mu^*$  tak lze číst „nachází-li se automat v množině stavů  $Q'$  a na vstupu je řetězec  $w$ , pak automat přejde do množiny stavů  $\mu^*(Q', w)$ “. (zde bude doplněno: uvést její předpis?)

Následuje popis, jak pojmy týkající se nedeterministického bivalentního automatu „zobecnit“ na odpovídající pojmy z oblasti fuzzy automatů.

## 2.3 Nedeterministický fuzzy automat

Stejně tak, jak fuzzy množiny jsou zobecněním klasických „bivalentních“ množin, dá se předpokládat, že i fuzzy automaty budou v určitém smyslu zobecněním klasických bivalentních automatů. A nebude tomu jinak ani u nedeterministického automatu.

300 Vzhledem k tomu, že automat je definován jako struktura, je zprvu třeba stanovit, které její části má smysl zobecňovat na fuzzy množiny (relace, funkce). Abeceda symbolů i množina stavů zcela určitě musí zůstat zachovány jako konečné bivalentní množiny. U přechodové funkce naopak očekáváme ostupňovanost (očekáváme možnost říkat „automat přejde do stavu  $q$  ve stupni  $d$ “). Co se počátečních a koncových stavů týče, někde se lze setkat s reprezentací bivalentní množinou (např. v. [8], [?]), jinde zase s fuzzy množinami ([20], [7], [?] [14])<sup>3</sup>. Vzhledem k tomu, že druhý jmenovaný, tj. automat s fuzzy množinou

<sup>3</sup>Jak bude ukázáno, mezi oběma druhy automatů existuje ekvivalence. Automaty s bivalentními počátečními a koncovými stavy se však jednodušeji reprezentují.

vstupních i výstupních stavů, je zřejmě obecnější, bude nadále uvažován pouze tento.

{def-ZaklDefNedFuzzAut}

**Definice 2.2** (Nedeterministický fuzzy automat). *Nedeterministický fuzzy automat  $A$  je pětice  $(Q, \Sigma, \mu, \sigma, \eta)$ , kde  $Q$  je konečná množina stavů,  $\Sigma$  je abeceda,  $\mu$  je fuzzy přechodová funkce (fuzzy relace  $Q \times \Sigma \times Q \rightarrow [0, 1]$ ) a  $\sigma$  a  $\eta$  jsou po řadě fuzzy množiny nad  $Q$  počátečních, resp. koncových stavů.*

## 2.4 Reprezentace nedeterministického fuzzy automatu

Nedeterministické fuzzy automaty se typicky reprezentují třemi způsoby. Prvním z nich je přechodová tabulka (např. [15]).

Jedná se o tabulku, která v řádcích obsahuje aktuální stavy a ve sloupcích následující stavy (tím je dána množina stavů). Poté buňka na řádku  $q$  a ve sloupci  $q'$  obsahuje  $\mu(q, q') \in \Sigma \rightarrow [0, 1]$ , tj. výčet symbolů a stupňů pravdivosti těchto přechodů. Dále tabulka obsahuje dva dodatečné sloupce pro určení stupně počátečního a koncového stavu každého stavu.

Tabulka však často může být rozsáhlá, proto se reprezentace tabulkou často nahrazuje maticovým způsobem (např. [?], [20]). Uvažujme označení stavů  $Q = \{q_0, \dots, q_n\}$ . Dále, přechodovou funkci  $\mu$  rozložíme na soubor funkcí  $\mu_x$  pro všechny  $x \in \Sigma$  takové, že  $\mu_x(q, q') = \mu(q, x, q')$ . Pro každý symbol  $x \in \Sigma$  vytvoříme matici tak, že na  $i$ -tém řádku a  $j$ -tém sloupci obsahuje hodnotu  $\mu(q_i, x, q_j)$ . Dále je přiložen vektor počátečních (koncových) stavů takový, že  $i$ -tá složka vektoru je rovna hodnotě  $\sigma(q_i)$  ( $\eta(q_i)$ ).

Posledním používaným způsobem je grafická reprezentace (např. [15], [14], [20]). Jedná se o orientovaný ohodnocený graf, kde:

- stavy automatu tvoří uzly grafu
- každý uzel stavu  $q$  je označen  $q/\eta(q)$
- hrana od uzlu  $q$  k uzlu  $q'$  je ohodnocena seznamem takových  $x/d$ , pro které platí  $\mu(q, x, q') = d$
- každý uzel  $q$  je označen šipkou vedoucí k tomuto uzlu ohodnocenou hodnotou  $\sigma(q)$

Pokud je některý ze stupňů roven 0, tak se v grafu zpravidla vynechává.

**Příklad 2.1.** (zde bude doplněno: vymyslet příklad)

## 2.5 Výpočet nedeterministického fuzzy automatu

Výpočet nedeterministického fuzzy automatu vychází z výpočtu nedeterministického bivalentního automatu. U bivalentního automatu jsme uvažovali, že automat se může nacházet ve více stavech současně, tj. součástí konfigurace výpočtu je množina stavů, ve kterých se automat nachází. Nedeterministický fuzzy automat se této koncepcí drží, jen ji obohacuje o odstupňovanost. Tedy, že množina stavů, ve kterých se automat může nacházet je fuzzy množinou. Tutu množinu budeme nazývat fuzzy stav.

{def-FuzzStav}

**Definice 2.3** (Fuzzy stav). *Mějme nedeterministický fuzzy automat  $A$ . Pak jako fuzzy stav označujeme každou fuzzy podmnožinu jeho stavů, tj.  $\hat{Q} \in \mathcal{F}(Q)$ .*

**Poznámka 2.1.** Fuzzy množiny počátečních a koncových stavů jsou ve své podstatě také fuzzy stavy.

Obdobně jako u bivalentního automatu, nezpracovaná část řetězce na vstupu spolu s (fuzzy) množinou stavů, ve kterých se automat nachází, označujeme jako konfigurace automatu. Posloupnost konfigurací nazýváme výpočtem. Formální definice výpočtu je však mírně složitější a pro jeho zavedení bude potřeba pár dalších pojmů, které budou nyní uvedeny.

**Definice 2.4** (Konfigurace nedeterministického fuzzy automatu). Mějme nedeterministický fuzzy automat  $A$ . Pak každý prvek  $(w, \hat{Q})$  relace  $\Sigma^* \times \mathcal{F}(Q)$  nazýváme konfigurace automatu  $A$ .

**Definice 2.5** (Aplikace fuzzy relace na fuzzy stav (tzv. t-kompozice)). Mějme nedeterministický fuzzy automat  $A$  a fuzzy stav  $\hat{Q}$ . Pak aplikací binární fuzzy relace  $R : Q \times Q \rightarrow [0, 1]$  na fuzzy stav  $\hat{Q}$  obdržíme fuzzy symbol  $\hat{Q} \circ R$  splňující pro každé  $p \in Q$ :  $(\hat{Q} \circ R)(p) = \max_{q \in Q} (\hat{Q}(q) \otimes R(q, p))$ .

(zde bude doplněno: přesunout do některé z úvodních kapitol?) (zde bude doplněno: t-kompozice má pár dalších zajímavých (a potřebných) vlastností (např. asociativitu))

**Značení.** Označme  $\mu[x](p, q) = \mu(p, x, q)$ .

**Definice 2.6** (Přechodová funkce fuzzy stavů). Mějme nedeterministický fuzzy automat  $A$ . Pak přechodová funkce fuzzy stavů je fuzzy relace  $\hat{\mu} : \mathcal{F}(F) \times \Sigma \rightarrow \mathcal{F}(F)$  taková, že pro každý fuzzy stav  $\hat{Q} \in \mathcal{F}(Q)$  a symbol  $x \in \Sigma$  je  $\hat{\mu}(\hat{Q}, x) = \hat{Q} \circ \mu[x]$ .

{def-PreFunFuzzStav}

**Definice 2.7** (Výpočet nedeterministického fuzzy automatu). Mějme nedeterministický fuzzy automat  $A$ . Každou posloupnost konfigurací  $(w_0, \hat{Q}_0), \dots, (w_m, \hat{Q}_m)$  splňující pro každé  $0 \leq i < m$

$$1. w_i = aw_{i+1} \text{ kde } a \in \Sigma$$

$$2. \hat{Q}_{i+1} = \hat{Q}_i \circ \hat{\mu}(\hat{Q}_i, a)$$

nazýváme výpočet automatu  $A$  z fuzzy stavu  $\hat{Q}_0$  při vstupu  $w_0$ .

Vidíme, že výpočet je definován rekurentně. Zápis můžeme přetransformovat do podoby rozšířené přechodové funkce [?].

**Definice 2.8** (Rozšířená přechodová funkce). Mějme nedeterministický fuzzy automat  $A$ . Pak rozšířená přechodová funkce je fuzzy relace  $\mu^* : Q \times \Sigma^* \times Q \rightarrow [0, 1]$  daná následujícím předpisem:

{def-RozPreFunFuzzStav}

$$1. \mu^*(q, \epsilon, q) = 1 \text{ pro všechna } q \in Q$$

$$2. \mu^*(q, ua, q') = \bigvee_{p \in Q} \mu^*(q, u, p) \otimes \mu(p, a, q') \text{ pro všechna } q, q' \in Q, u \in \Sigma^*, a \in \Sigma$$

Rozšířená přechodová funkce fuzzy stavů zřejmě plní funkci výpočtu automatu. Výraz  $\mu^*(q, w, q')$  odpovídá stupni, v jakém automat přejde při zpracování řetězce  $w$  ze stavu  $q$  do stavu  $q'$ .

Stupeň  $A(w)$ , v jakém je řetězec  $w$  automatem  $A$  přijat je dán jako nejvyšší stupeň pro všechny dvojice stavů  $p, q$ :

1. stupněm „stav  $q$  je počáteční ve stupni  $\sigma(q)$ “
2. stupněm „automat při vstupu  $w$  přejde ze stavu  $q$  do stavu  $q'$  ve stupni  $\mu^*(q, w, q')$ “
3. stupněm „stav  $q'$  je koncový ve stupni  $\eta(q')$ “

Můžeme tedy zapsat:

**Definice 2.9** (Řetězec přijímaný automatem). *Mějme nedeterministický fuzzy automat  $A$ . Pak řetězec  $w \in \Sigma^*$  je automatem  $A$  přijat ve stupni*

{def-RetPriAut}

$$A(w) = \max_{q, q' \in Q} (\sigma(q) \otimes \mu^*(q, w, q')(q) \otimes \eta(q')) \quad (1) \quad \{\text{eq-RetPriAut}\}$$

**Poznámka 2.2.** *V literatuře (např. [?] [?] [?]) se obvykle lze setkat s „techničtější“ zápisem ať už jen rozšířené přechodové funkce, tak  $A(w)$ . Pro řetězec  $w = a_0 \dots a_n$  rozvojem rekurence  $\mu^*$  a použitím asociativity  $\circ$  můžeme napsat:*

$$\begin{aligned} \mu^*(q, a_0 \dots a_n, q') &= \bigvee_{p_n \in Q} \left( \dots \bigvee_{p_0 \in Q} (1 \otimes \mu(p_0, a_0, p_1)) \dots \otimes \mu(p_n, a_n, q') \right) \\ &= \bigvee_{p_n \in Q} \dots \bigvee_{p_0 \in Q} \mu(p_0, a_0, p_1) \dots \otimes \mu(p_n, a_n, q') \\ &= \bigvee_{(p_n, \dots, p_0) \in Q^n} \mu(p_0, a_0, p_1) \dots \otimes \mu(p_n, a_n, q') \end{aligned}$$

Poté může být (1) zapsána jako:

$$A(a_0 \dots a_n) = \bigvee_{(p_n, \dots, p_0, q') \in Q^{n+1}} (\sigma(p_0) \otimes \mu(p_0, a_0, p_1) \dots \otimes \mu(p_n, a_n, q') \otimes \eta(q'))$$

Tento zápis intuitivněji popisuje výpočet automatu. Tento zápis totiž můžeme číst jako: „Řetězec je přijímán ve stupni  $A(a_0 \dots a_n)$  jestliže ze stavu  $p_0$ , který je počáteční ve stupni  $\sigma(p_0)$  přejde přečtením  $a_0$  do  $p_1$  ve stupni  $\mu(p_0, a_0, p_1)$ , ..., ze stavu  $p_n$  přečtením  $a_n$  do stavu  $q'$  ve stupni  $\mu(p_n, a_n, q')$ , a stav  $q'$  je koncový ve stavu  $\eta(q')$ .“

Z tohoto zápisu je také patrné, že výpočet  $A(a_0 \dots a_n)$  dle definice vyžaduje  $|Q|^{n+1}$  výpočtů, každý o délce  $1 + n + 1$  elementárních kroků. Výpočet má tedy exponenciální složitost vzhledem k délce vstupního řetězce.

Podobně, jak u bivalentních automatů, jazyk rozpoznávaný automatem je množina všech řetězců, které jsou tímto automatem rozpoznávané. U fuzzy automatu se však bude pochopitelně jednat o fuzzy jazyk.

{def-JazRozpAut}

**Definice 2.10** (Jazyk rozpoznávaný automatem). *Mějme nedeterministický fuzzy automat  $A$ . Pak fuzzy množinu  $L(A)(w) = A(w)$  nad univerzem  $\Sigma^*$  nazýváme fuzzy jazyk rozpoznávaný automatem  $A$ .*

(zde bude doplněno: Jazyk rozpoznávaný automatem značit  $\mathcal{L}$  nebo jen  $L$ ?  $A$  automat  $\mathcal{A}$ ,  $\mathbf{A}$ ,  $\mathbb{A}$  nebo jen  $A$ ?)

## 3 Další varianty fuzzy automatů

### 410 3.1 Zobecněný automat

Např. [15], v [?].

### 3.2 Fuzzy automat s výstupem

Např. zde: [1]. Zde [?][?][?][?] je bez koncových (zde bude doplněno: i „a počátečních“?) stavů.

### 3.3 Fuzzy automat jako stroj

Naopak, zde [?] je uveden automat „bez výstupu“ (nemá žádný výstup – ani formou koncových stavů ani formou výstpní funkce/abecedy).

### 3.4 Nedeterministický fuzzy automat s $\epsilon$ -přehcody

Je docela praktický a současně docela intuitivní. Např. zde: [?]. V [?] ho definují  
420 jako automat shodný s  $A$  jen místo  $\Sigma$  má  $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ .

### 3.5 Deterministický fuzzy automat

V [?] je definován jako deterministický automat, jen koncové stavy jsou fuzzy. V [?] je to deterministický automat, ale s fuzzy počátečními stavy, koncovými stavy i přechodovými funkcí.

Trošku jinak (pomocí jednotkových kardinalit) je uveden v [14].

### 3.6 Zásobníkový fuzzy automat

Někde jsem o něm četl. Je to s ním krapet složitější, co se bezkontextového jazyka týče.

Tady je pěkně popsán [?].

## 430 4 Fuzzy automaty, jazyky, gramatiky a regulérní výrazy

(zde bude doplněno: nějak to uvést. Budou pojmy jako regulérní jazyk a gramatika popsány v nějaké předchozí kapitole?)

(zde bude doplněno: pojem „Lattice language“)

(zde bude doplněno: značení: „Fuzzy množina  $\phi$ “ vs. „ $L$ -množina  $\phi : X \rightarrow L$ “; „fuzzy podmnožina“ vs. „fuzzy množina nad“)

### 4.1 Jazyk rozpoznávaný fuzzy automatem

Věta 6.3 [21] (pro lattice monoid, není to někde jen pro  $[0, 1]$ ?).

Dle definice 4 [14] je fuzzy regulární jazyk fuzzy podmnožina bivalentního.

440 Automat s bivalentní  $\mu$  and  $\eta$  (a fuzzy  $\sigma$ ) taky rozpoznává fuzzy regulární jazyk [9]. Neřešil něco takového i Bel? Jinak řečeno, support konečný automat [14].

Dále např. [?].

## 4.2 Fuzzy a bivalentní regulární jazyky

Univerzum fuzzy jazyka je regulární jazyk, pozorování 6.1 [21].

Stejně tak, zaříznutý jazyk ( $a$ -řez jazyka) je také regulární, věta 2.2 [9].

Pumping lemma pro fuzzy regulární jazyky, lemma 4-7 pro různé typy automatů [14].

Uzávěrové vlastnosti fuzzy regulárních jazyků, např. [9].

## 4.3 Fuzzy regulární výrazy

LiPed-FuzzFinAutFuzzRegExMembValLattOrdMon, definice 5.1, 5.2 (+ opsat důkaz, že  $[0, 1]$  je lattice monoid) [21].

Algoritmus převodu reg na aut, [20]. Ale zdá se mi to až moc složité.

# 5 Rozpoznávání textových vzorů

Rozpoznávání vzorů obecně je jednou z nejvýznamějších aplikací informatiky. V běžném životě se často setkáváme se situacemi, kdy je třeba v datech najít výskyt učitěho vzoru, popř. jeho další vlastnosti. Případně určit podobnost ke vzoru, nebo nejpodobnější vzor.

Typickým příkladem je např. detekce obličeje na fotografii, tedy rozpoznávání vzorů v obrazových datech. Vzory je však možné rozpoznávat v téměř jakýchkoliv datech, například textech, zvukových záznamech či výsedcích měření nebo pozorování.

Z pohledu teoretické informatiky je však základem vyhledávání vzorů v textových datech. Textová data, tedy řetězce, mají jednoduchou strukturu a lze s nimi snadno manipulovat. Na druhou stranu, jsou schopna reprezentovat nebo kódovat široké spektrum dat. Právě z tohoto důvodu je studium rozpoznávání textových vzorů klíčové pro zpracovávání jakýchkoliv dalších typů dat.

**Poznámka 5.1.** *Pokud nebude uvedeno jinak, pojem „rozpoznávání textových vzorů“ bude v této kapitole zkracován jen na „rozpoznávání vzorů“.*

## 5.1 Formální zavedení problému

Stejně tak, jak se mohou různit aplikace rozpoznávání vzorů, i samotný pojem „rozpoznávání vzorů“ bývá chápán různě. V nejzákladnější podobě se jedná o problém určení, zda-li pozorovaný řetězec odpovídá předem stanovenému vzoru. Vzorem bývá obvykle také řetězec, ale může jím být například regulární výraz. Také - může nás zajímat buď exaktní shoda pozorovaného řetězce se vzorem, nebo jen nějaká forma podobnosti.

V rozšířeném smyslu může být problém chápán jako klasifikace. Tedy, určení třídy, do které by měl pozorovaný řetězec spadat, typicky na základě podobnosti s vybranými reprezentanty jednotlivých tříd.

V této kapitole se však budeme zabývat pouze určováním podobnosti vzorového a pozorovaného řetězce. U každé instance problému budeme znát abecedu se kterou pracujeme a také vzor. Vzorem bude libovolný řetězec nad touto abecedou. Řešením tohoto problému pro nějaký, tzv. pozorovaný, vstupní řetězec

bude úroveň podobnosti tohoto řetězce s vzorovým. Jako podobnost zde budeme uvažovat reálné číslo z intervalu  $[0, 1]$ , kde 0 znamená úplnou rozdílnost a 1 úplnou shodu.

**Poznámka 5.2.** *Vzorový řetězec budeme v této kapitole vždy značit  $\omega$ , pozorovaný pak  $\alpha$ .*

490 Nyní máme zdefinován problém samotný, nicméně je třeba zdůraznit, že v jeho definici se používá vágní pojem „podobnost řetězců“. Podobnost řetězců je totiž pojem, který souvisí s konkrétní instancí problému a nelze jej nějak přesně, ale současně dostatečně obecně popsat. Jediné, co o podobnosti řetězců můžeme říct, je, že čím vyšší toto číslo je, tím by si měly být řetězce podobnější.

Například, budeme-li porovnávat vstup zadaný z klávesnice počítače oproti nějakému vzoru, je možné, že uživatel udělá překlep. V takovém případě bude vzorovému řetězci určitě více podobný řetězec obsahující dva překlepy (záměna symbolu za některý sousedící na klávesnici) než jiný, který se sice bude lišit jen v jednom symbolu, ale to takovém, který je na opačné straně klávesnice.

500 Obdobně, pokud budeme pracovat s abecedou malých a velkých písmen (majusku a minusku). Uvažujme vzorový řetězec `hello`. Řetězec `HELLO` se s ním neshoduje v ani jednom symbolu, ale přesto jejich podobnost může být blízka jedné.

## 5.2 Motivace k použití fuzzy automatů

Klasická teorie automatů vznikla jako nástroj pro zpracování textových řetězců. Z tohoto důvodu je rozpoznávání textových vzorů jejím základním výsledkem. Automaty obecně jsou nástroje sloužící pro rozhodování, zda-li řetězec odpovídá vzoru automatem reprezentovaném. Použití pro rozpoznávání řetězcového vzoru tak bude jen speciálním případem jejich užití.

510 V předchozí podkapitole jsme si stanovili, že řešením našeho problému je číslo z intervalu  $[0, 1]$ . Z tohoto důvodu nebude možné využít klasické bivalentní automaty. Fuzzy automaty pracují se stupněm pravdivosti, který by mohl s hodnotou podobnosti řetězců korespondovat. Navíc, v praxi se často setkáme s texty, které jsou nepřesné a nedokonalé. Fuzzy přístup by nám tak mohl pomoci na tyto nepřesnosti adekvátně reagovat.

*(zde bude doplněno: a co pravděpodobnostní?) (zde bude doplněno: Protože například: „pozorovaný řetězec se se vzorovým shoduje ve stupni  $x$ “ ale „je pravděpodobnost  $y$ , že uživatel zadal požadovaný řetězec“)*

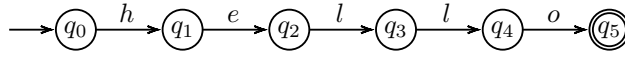
## 5.3 Automat rozpoznávající $\omega$

520 Klíčovým pro rozpoznávání vzorů (chceme-li využívat fuzzy automaty) je bivalentní automat rozpoznávající vzorový řetězec. Tedy automat takový, který přijímá jedinný řetězec  $\omega$  a všechny ostatní zamítá. Nyní si takovýto automat zkonstruujeme.

Uvažujme, že máme k dispozici vzorový řetězec  $\omega$  nad abecedou  $\Sigma$ . Označme  $\mathcal{L}(\omega)$  jako jednorvkový jazyk obsahující pouze řetězec  $\omega$ . Vzhledem k tomu, že jazyk  $\mathcal{L}(\omega)$  je konečný, je také regulérní a existuje tak konečný deterministický automat, který jej rozpoznává.

Automat bude v každém kroku konzumovat symboly ze vstupního řetězce a porovnávat je se symboly vzorového řetězce na odpovídajících pozicích. Pokud





Obrázek 1: Automat rozpoznávající **hello**

{diag-AutRozpHell}

530 dojde ke shodě na všech pozicích, automat dojde do koncového stavu a sledovaný řetězec přijme. Pokud se symboly shodovat nebudou, automat nebude mít definován žádný odpovídající přechod, kterým by pokračoval ve výpočtu, a řetězec tak zamítne.

Takovýto automat označme jako *automat rozpoznávající  $\omega$* .

**Definice 5.1** (Automat rozpoznávající  $\omega$  (deterministický)). *Mějme řetězec  $\omega$  délky  $n$  nad abecedou  $\Sigma$ . Automat rozpoznávající  $\omega$  je pak konečný automat  $A(\omega) = (Q, \Sigma, \delta, q_0, F)$  takový, že jeho množina stavů  $Q$  se sestává z právě  $n$  stavů  $q_0, \dots, q_n$ ,  $q_0$  je počáteční stav,  $F = \{q_n\}$  množina koncových stavů a  $\delta$  je přechodová funkce definována pro všechna  $0 \leq k < n$  následovně:*

$$\delta(q_k, a_k) = q_{k+1} \text{ kde } a_k \text{ je } k\text{-tý symbol řetězce } \omega$$

Tato definice automatu je vcelku intuitivní. K stejnému výsledku bychom došli, pokud bychom automat zkonstruovali konverzí gramatiky nebo regulérního výrazu.

**Příklad 5.1.** *Příklad automatu rozpoznávající řetězec  $\omega = \mathbf{hello}$  se nachází na obrázku 1.*

540 My však budeme potřebovat fuzzy automat rozpoznávající  $\omega$ . To znamená, že musíme nejdříve automat z předchozí definice převést na nedeterministický a poté dle definice 1.2 k němu zkonstruovat odpovídající fuzzy automat.

{def-AutRozpOme}

**Definice 5.2** (Automat rozpoznávající  $\omega$  (nedeterministický)). *Mějme řetězec  $\omega$  nad abecedou  $\Sigma$  z předchozí definice. Nedeterministický automat rozpoznávající  $\omega$  je pak konečný automat  $A'(\omega) = (Q, \Sigma, \delta, I, F)$  takový, že jeho množina stavů  $Q$  je stejná jako v předchozí definici, dále  $I = \{q_0\}$  je množina počátečních a  $F = \{q_n\}$  množina koncových stavů a  $\delta$  je přechodová funkce definována pro všechna  $0 \leq k < n$  následovně:*

$$\delta(q_k, a_k) = \begin{cases} \{q_{k+1}\} & \text{pokud je } a_k \text{ } k\text{-tý symbol řetězce } \omega \\ \emptyset & \text{jinak} \end{cases}$$

Následuje vytvoření fuzzy automatu.

{def-FuzzAutRozpOme}

**Definice 5.3** (Fuzzy automat rozpoznávající  $\omega$ ). *Mějme řetězec  $\omega$  nad abecedou  $\Sigma$  délky  $n$ . Fuzzy automat rozpoznávající  $\omega$  je pak automat  $A''(\omega)$  vytvořený z nedeterministického automatu rozpoznávající  $\omega$  (definice 5.2) dle definice 1.2. Bude to tedy automat  $A''(\omega) = (Q, \Sigma, \mu, \sigma, \epsilon)$  kde*

- $\sigma(q_0) = 1$  a  $\sigma(q_i) = 0$  pro všechna  $i > 0$
- $\epsilon(q_n) = 1$  a  $\epsilon(q_i) = 0$  pro všechna  $i < n$
- $\mu(q_k, a_k, q_{k+1}) = \begin{cases} 1 & \text{pokud je } a_k \text{ } k\text{-tý symbol řetězce } \omega \\ 0 & \text{jinak} \end{cases}$

550 Nyní máme k dispozici fuzzy automat, který ostře rozpoznává vzorový řetězec. V následujících podkapitolách následuje výčet několika technik, které tuto ostrost (pomocí dalších informací) odstraňují a nahrazují podobností.

## 5.4 Podobnost symbolů

Nezákladnější technika pro zanesení neostrého (stupňovitého) rozpoznávání je s využitím podobnostní relace symbolů. Tato technika byla přejata z [6]. Myšlenkou této techniky je, že symbol v pozorovaném řetězci může být snadno zaměněn za jiný, podobný, jemu odpovídající v řetězci vzorovém.

Pro realizaci této techniky je potřeba mít k dispozici fuzzy relaci  $s : \Sigma \times \Sigma \rightarrow [0, 1]$ . Tato relace popisuje podobnost dvojice symbolů. Tedy, je-li pro nějakou dvojici symbolů  $x, y \in \Sigma$   $s(x, y) = 0$ , pak se jedná o naprosto rozdílné symboly. Naopak, pokud bude  $s(x, y) = 1$ , pak se jedná o shodné symboly. Je zjevné, že by relace  $s$  měla být symetrickou a reflexivní. (zde bude doplněno: v článku to nepíše, ale měla by to být relace ekvivalence (Sym, Ref, Tra). Existuje něco, jako fuzzy relace ekvivalence?)

**Příklad 5.2.** Jako příklad podobnostní relace (nad abecedou písmen anglické abecedy) může posloužit například vzdálenost patřičných kláves na klávesnici. V takovém případě by určitě platilo kupříkladu  $s(a, s) > s(a, d) > s(a, l)$ . Protože klávesy  $A$  a  $S$  jsou si blíž (a tudíž symboly  $a$  a  $s$  jsou si „podobnější“) než například  $A$  a  $D$  či  $A$  a  $L$ .

Jiným příkladem může být například vizuální podobnost napsaných (malých psacích) písmen. V takovém případě by zřejmě platilo  $s(a, o) > s(m, t)$ , protože malá psací písmena  $a$  a  $o$  jsou si vizuálně podobnější než  $m$  a  $t$ , která vypadají úplně rozdílně.

Máme-li k dispozici relaci  $s$ , je nutné ji zakomponovat do automatu. Jak autoři uvádějí, tato technika může pracovat s libovolným konečným automatem. Podíváme se proto nejdříve, jak využít relaci  $s$  obecně. Následně ji aplikujeme na automat rozpoznávající  $\omega$ , čímž získáme nástroj pro podobnostní rozpoznávání textového vzoru.

{def-AutPracGS}

**Definice 5.4** (Automat pracující s  $s$ ). Uvažujme, že máme nedeterministický automat  $A$  a relaci podobnosti symbolů  $s$ . Pak k automatu  $A$  můžeme zkonstruovat fuzzy automat  $A'$ , který navíc pracuje s  $s$ . Takový automat bude zkonstruován dle definice 1.2 s tím rozdílem, že přechodová funkce  $\mu$  bude definována pro všechna  $q_i, q_j \in Q$  a  $x \in \Sigma$  následovně:

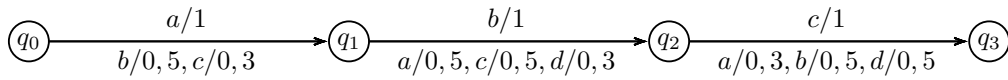
$$\mu(q_i, x, q_j) = \bigvee_{y \in \Sigma} (s(x, y) \wedge \delta_y(q_i, q_j))$$

$$\text{kde } \delta_x(q_i, q_j) = \begin{cases} 1 & \text{pokud } q_j \in \delta(q_i, x) \\ 0 & \text{pokud } q_j \notin \delta(q_i, x) \end{cases} \text{ pro všechna } q_i, q_j \in Q \text{ a } x \in \Sigma.$$

Definice je vcelku přímočará. Pro každý přechod ze stavu  $q_i$  do stavu  $q_j$  přes symbol  $x$ , procházíme přechody původního automatu. Obsahovala-li přechodová funkce původního automatu přechod ze stavu  $q_i$  přes symbol  $y$  do stavu  $q_j$ , pak je  $s(x, y) \wedge \delta_y(q_i, q_j)$  rovno podobnosti  $x$  a  $y$ . V opačném případě je roven nule. Hodnota tohoto výrazu je díky spojení přes všechny symboly maximalizována.

Nyní aplikujeme tento způsob konstrukce fuzzy automatu na automat rozpoznávající  $\omega$ .

**Definice 5.5** (Automat rozpoznávající  $\omega$  pracující s  $s$ ). Mějme abecedu  $\Sigma$ , řetězec  $\omega$  nad touto abecedou a fuzzy relaci  $s$  nad touto abecedou. Dle definice 5.2



Obrázek 2: Automat rozpoznávající **abc** pracující s  $s$

{diag-AutRozpABCPracGS}

590 můžeme zkonstruovat nedeterministický bivalentní automat  $A(\omega)$  rozpoznávající  $\omega$ . Jako automat rozpoznávající  $\omega$  pracující s  $s$  označme nedeterministický fuzzy automat  $A'(\omega)$ , který byl z automatu  $A(\omega)$  vytvořen podle definice 5.4.

(zde bude doplněno: neměl by se takovýto automat místo  $A'(\omega)$  značit třeba  $A_s(\omega)$ ?)

Následuje jednoduchý příklad takového automatu.

{ex-AutRozpOmePodSym}

**Příklad 5.3.** Mějme abecedu  $\Sigma = \{a, b, c, d\}$ . Dále uvažujme relaci podobnosti symbolů s takovou, že

- každý symbol je podobný sám sobě ve stupni 1
- každý symbol je podobný symbolu ve stupni 0,5 jedná-li se o symboly reprezentující sousedící písmena abecedy
- 600 • každý symbol je podobný symbolu ve stupni 0,3 jedná-li se o symboly reprezentující ob-jedno písmeno sousedící písmena abecedy
- všechny ostatní dvojice symbolů jsou si podobny ve stupni 0

Tuto relaci můžeme zapsat do matice (sloupce i řádky odpovídají po řadě symbolům  $a, b, c, d$ ):

$$s = \begin{pmatrix} 1,0 & 0,5 & 0,3 & 0,0 \\ 0,5 & 1,0 & 0,5 & 0,3 \\ 0,3 & 0,5 & 1,0 & 0,5 \\ 0,0 & 0,3 & 0,5 & 1,0 \end{pmatrix}$$

Nyní mějme vzorový řetězec  $\omega = abc$ . Pak můžeme podle předchozí definice sestavit automat  $A(\omega)$  rozpoznávající  $\omega$  pracující s  $s$ . Přechodový diagram takového automatu je na obrázku 2.

Tento automat evidentně rozpoznává řetězec **abc** ve stupni 1. Pokud v pozorovaném řetězci nahradíme symbol **a** za **b**, bude jej automat přijímat ve stupni 0,5. Pokud nahradíme **b** za **d**, bude jej automat přijímat ve stupni 0,3.

610 Pokud na začátek pozorovaného řetězce vložíme symbol **a** (tedy  $\alpha = aabc$ ), automat jej přijme ve stupni 0. Stejnětak, pokud odebereme symbol **c** z konce vzorového řetězce (tedy  $\alpha = ab$ ). Pokud vložíme symbol **a** na začátek a současně odebereme **c** z konce vzorového řetězce, obdržíme pozorovaný řetězec  $\alpha = aab$ . Tento řetězec bude přijat ve stupni 0,5 (zde bude doplněno:  $1 \otimes 0,5 \otimes 0,5$ , záleží tedy na  $\otimes$ ).

Z příkladu jasně vyplývá, že automat pracující s  $s$  je schopen akceptovat pouze náhradu symbolu jiným symbolem. Bude-li pozorovaný řetězec oproti vzorovému obsahovat vložený symbol nebo naopak z něj bude symbol odebrán, tento typ automatu selže. Na druhou stranu jeho princip i konstrukce jsou jednoduché a snadno se s nimi pracuje.

## 5.5 Fuzzy symboly

Fuzzy symbol je technika využívající podobnosti symbolů. Ve své podstatě se jedná o téže techniku jak v předchozí podkapitole, jen je na ni nahlíženo jinak. Oproti podobnosti symbolů je použití fuzzy symbolů komplikovanější, avšak umožňuje jednoduše tuto techniku kombinovat s jinými. Princip fuzzy symbolů byl přejat z [2].

Mějme abecedu  $\Sigma$  a relaci  $p$  podobnosti symbolů (stejně jako relace  $s$  v předchozí podkapitole). Fuzzy symbolem symbolu  $x \in \Sigma$  označujeme fuzzy množinu symbolů takových, které jsou podle relace  $p$  symbolu  $x$  „podobné“.

**Definice 5.6** (Fuzzy symbol). *Mějme abecedu  $\Sigma$  a fuzzy relaci  $p \subseteq \Sigma \times \Sigma$ . Pak pro každý symbol  $y \in \Sigma$  definujeme fuzzy symbol  $\tilde{y}$  symbolu  $y$  jako fuzzy množinu nad  $\Sigma$  takovou, že pro všechna  $x \in \Sigma$  platí*

$$\tilde{y}(x) = p(y, x)$$

Vzhledem k tomu, že fuzzy symbol máme definován pro všechny  $y \in \Sigma$ , můžeme množinu všech takových fuzzy symbolů nazvat abecedou fuzzy symbolů.

**Definice 5.7** (Abeceda fuzzy symbolů). *Mějme abecedu  $\Sigma$  a fuzzy symboly  $\tilde{y}$  pro všechna  $y \in \Sigma$ . Pak množinu všech těchto fuzzy symbolů nazvěme abeceda fuzzy symbolů abecedy  $\Sigma$  a označme  $\tilde{\Sigma}$ . Tedy  $\tilde{\Sigma} = \{\tilde{y} \mid y \in \Sigma\}$ .*

Máme-li abecedu fuzzy symbolů  $\tilde{\Sigma}$ , můžeme pracovat s řetězcí  $\tilde{\alpha} \in \tilde{\Sigma}^*$  nad touto abecedou. Ještě si však doplníme, jak vytvořit k řetězci  $\alpha \in \Sigma^*$  jemu odpovídající řetězec fuzzy symbolů  $\tilde{\alpha} \in \tilde{\Sigma}^*$ .

(zde bude doplněno: sjednotit značení  $\omega$  vs.  $\alpha$ , když se používá jen jeden obecný řetězec)

**Definice 5.8** (Řetězec fuzzy symbolů). *Mějme abecedu  $\Sigma$  a nějaký řetězec  $a_1 \dots a_n = \alpha \in \Sigma^*$ . Pak definujeme  $\tilde{\alpha} = \tilde{a}_1 \dots \tilde{a}_n$  jako řetězec fuzzy symbolů řetězce  $\alpha$ .*

V této fázi jsme schopni plnohodnotně pracovat s řetězcí fuzzy symbolů a konstruovat je z řetězců nad abecedou  $\Sigma$ . Nyní přejdeme k návrhu fuzzy automatu, který bude s fuzzy symboly pracovat. Stejně jako u podobnosti symbolů i fuzzy symboly mohou být aplikovány na libovolný typ automatu. Vytvoříme proto automat pracující s fuzzy symboly nejdříve obecně, pro libovolný automat  $A$ .

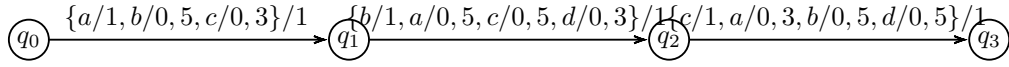
{def-AutPracFuzzSym}

**Definice 5.9** (Automat pracující s fuzzy symboly). *Mějme nedeterministický fuzzy automat  $A$ . Pak fuzzy automat  $\tilde{A}$  pracující s fuzzy symboly vytvoříme tak, že v definici automatu  $A$  nahradíme  $\Sigma$  za  $\tilde{\Sigma}$ .*

(zde bude doplněno: může to tak být? a co související pojmy)

Formální zavedení automatu pracujícího s fuzzy symboly je intuitivní, jedná se jen o formalitu. Abychom však využili potenciál fuzzy symbolů, je třeba pozměnit výpočet automatu. Proces jeho výpočtu se změní ve fázi výpočtu přechodové funkce fuzzy stavů. Připomeňme, že ta je definována (definice 2.6) jako fuzzy relace  $\hat{\mu}$  přiřazující každému fuzzy stavu  $V$  a fuzzy symbolu  $x$  fuzzy stav dle předpisu

$$\hat{\mu}(V, x) = V \circ \mu[x]$$



Obrázek 3: Automat rozpoznávající abc pracující s s

{diag-AutRozpOmePraFuzSym

Zde je zjevně nutné nahradit  $\mu[x]$  spojením přes všechny fuzzy symboly. Bude tedy vypadat následovně:

$$\hat{\mu}(V, x) = V \circ \bigvee_{y \in \Sigma} (\mu[x] \wedge \tilde{x}(y))$$

Tím, že je změna zakořeněna ve výpočtu automatu, nám umožňuje další práci se samotným automatem. Můžeme tedy bez problémů zkonstruovat automat rozpoznávající  $\omega$  pracující s fuzzy symboly.

**Definice 5.10** (Automat rozpoznávající  $\omega$  pracující s fuzzy symboly). *Mějme abecedu  $\Sigma$ , řetězec  $\omega$  nad touto abecedou a abecedu fuzzy symbolů  $\tilde{\Sigma}$ . Dle definice 5.3 můžeme zkonstruovat fuzzy automat  $A(\omega)$  rozpoznávající  $\omega$ . Následně pak podle definice 5.9 automat  $\tilde{A}(\omega)$  rozpoznávající  $\omega$  pracující s fuzzy symboly.*

Postup konstrukce takového automatu je opět vcelku intuitivní. Následuje demonstrace na příkladu.

**Příklad 5.4.** *Mějme abecedu  $\Sigma$ , vzorový řetězec a podobnostní relaci  $p = s$  stejné jako v příkladu 5.3. Na obrázku 3 je zobrazen diagram automatu  $\tilde{A}(\omega)$  rozpoznávající  $\omega$  pracující s fuzzy symboly.*

Co se týče vlastností automatů (rozpoznávajících  $\omega$ ) pracujících s fuzzy symboly, jejich charakteristika je vesměs stejná jako u automatů pracujících s podobností symbolů. Pouze, jak již bylo zmíněno v úvodu, nezasahují do struktury automatu jako takového.

## 5.6 Editační operace

Další technikou pro podobnostní porovnávání pozorovaného a vzorového řetězce je s využitím editačních operací. Tato technika byla přejata z [13]. Základní idea této techniky spočívá v trojici jednoduchých editačních operací, jejíž složením jsme schopni popsat transformaci pozorovaného řetězce na vzorový. Množství transformace pak udává podobnost pozorovaného a vzorového řetězce.

Následuje formální definice editačních operací a pojmů s nimi souvisejících. Následně přejdeme ke konstrukci automatu, který s nimi bude schopen pracovat.

**Definice 5.11** (Editační operace). *Mějme abecedu  $\Sigma$ , uvažujme množinu  $E = (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\}) \setminus \{(\epsilon, \epsilon)\}$ . Pak každou dvojici  $(x, y) = z \in E$  nazvěme editační operace. Speciálně pak, pro všechna  $x, y \in \Sigma$ ,  $(x, y) \in E$  znamená nahrazení symbolu  $x$  symbolem  $y$ ,  $(x, \epsilon) \in E$  znamená odebrání symbolu  $x$  a naopak  $(\epsilon, y) \in E$  pak vložení symbolu  $y$ . Navíc jako editační operaci uvažujme i všechny dvojice  $(x, x) \in E$  (pro každé  $x \in \Sigma$ ) symbolizující „žádnou editaci“.*

*Máme-li editační operaci  $(x, y) = z \in E$ , pak označme  $x = z^\downarrow$  a  $y = z^\uparrow$ .*

Editační operace jsou tedy tři a to náhrada symbolu, vložení symbolu a odebrání symbolu. Například řetězec hallo vznikl záměnou e za a v řetězci

hello. Obdobně, řetězec **hellow** vznikl přidáním **w** na konec a řetězec **helo** odebráním (prvního nebo druhého) symbolu **l**.

My však obvykle očekáváme, že došlo k více, než jedné jednoduché editaci. Je proto vhodné zavést koncept mnohanásobné editace. Jednotlivé editace za sebe seřadíme do posloupnosti v pořadí, v jakém mají být postupně aplikovány, a takovouto posloupnost nazvěme vyrovnáním řetězce  $\alpha$  na řetězec  $\omega$ .

Uvažujme nyní množinu  $E$  editačních operací jako abecedu. Pak každé vyrovnání  $\zeta$  řetězce  $\alpha$  na řetězec  $\omega$  (posloupnost  $z_1 z_2 \dots z_n$  symbolů  $z_1, z_2, \dots, z_n \in E$ ), tak můžeme považovat za řetězec nad abecedou  $E$ .

(zde bude doplněno: fakt  $E$  považovat za abecedu a  $G$  za jazyk? není to zbytečná komplikace? je to tam nutné?)

Například všechny tři následující řetězce jsou vyrovnáním řetězce **ahoj** na řetězec **hello**:

$$\begin{aligned}\zeta_1 &= (a, \epsilon)(h, h)(o, e)(j, l)(\epsilon, l)(\epsilon, o) \\ \zeta_2 &= (a, \epsilon)(h, \epsilon)(o, \epsilon)(j, \epsilon)(\epsilon, h)(\epsilon, e)(\epsilon, l)(\epsilon, l)(\epsilon, o) \\ \zeta_3 &= (a, h)(h, e)(o, l)(j, l)(\epsilon, o)\end{aligned}$$

Na tomto příkladu je vhodné si povšimnout, že obecně může existovat více než 1 vyrovnání mezi libovolnou dvojicí řetězců. Bude proto vhodné neuvažovat vyrovnání jednotlivá, ale množinu všech možných vyrovnání mezi dvojicí řetězců.

Podíváme-li se nyní jen na levé části editačních operací ve vyrovnání  $\zeta_1$  z předchozího příkladu, zjistíme, že jejich zřetěžením získáme řetězec  $\alpha$ :

$$(a, \epsilon)^\downarrow (h, h)^\downarrow (o, e)^\downarrow (j, l)^\downarrow (\epsilon, l)^\downarrow (\epsilon, o)^\downarrow = ahoj$$

Stejně tak, zřetěžením pravých částí editačních operací v  $\zeta_1$  získáme řetězec  $\omega$ :

$$(a, \epsilon)^\uparrow (h, h)^\uparrow (o, e)^\uparrow (j, l)^\uparrow (\epsilon, l)^\uparrow (\epsilon, o)^\uparrow = hello$$

Tato vlastnost nám udává, v jakém pořadí mají být editační operace aplikovány. Stejně tak nám odstraňuje nadbytečné editační operace (např. opakované přidávání a odebrání téže znaku, které by mohlo vést až k nekonečné posloupnosti editací). Proto nám tato vlastnost poslouží jako definiční pro formání zavedení vyrovnání řetězců.

**Definice 5.12** (Vyrovnání řetězců [13]). *Jako množinu všech vyrovnání  $G(\alpha, \omega)$  řetězce  $\alpha$  na řetězec  $\omega$  (kde  $(\epsilon, \epsilon) \neq \alpha, \omega \in \Sigma^*$ ) označme takovou množinu  $\{\zeta \in E^+ \mid \zeta \text{ splňuje vlastnosti 1., 2. i 3.}\}$*

1.  $\zeta = z_1 z_2 \dots z_r, z_i \neq (\epsilon, \epsilon)$  pro všechna  $i \in 1, \dots, r$ ,
2.  $z_1^\downarrow z_2^\downarrow \dots z_r^\downarrow = \alpha$
3.  $z_1^\uparrow z_2^\uparrow \dots z_r^\uparrow = \omega$

V tento okamžik máme formálně zavedena vyrovnání řetězců. Můžeme tedy přejít k práci s nimi. Ukážeme si způsob, jak pomocí vyrovnání řetězců spočítat podobnost dvojice řetězců. Na základě tohoto výpočtu pak sestavíme automat, který tento výpočet bude realizovat.

Pro určení podobnosti na základě vyrovnání řetězců budeme potřebovat znát míry pravdivosti editačních operací. Vstupem pro výpočet podobnosti řetězců

tak bude navíc binární fuzzy relace  $R$  nad množinou všech editačních operací ( $E$ ), udávající stupeň akceptovatelnosti každé z možných editačních operací. S touto znalostí můžeme nadefinovat relaci podobnosti řetězců, tzv. fuzzy míru dvojice řetězců  $\alpha$  a  $\omega$ .

720 (zde bude doplněno: Musí být  $R$  reflexivní a symetrická (def. automatu to vyžaduje, ale je to nutné?). A co  $T$ -tranzitivita?) (zde bude doplněno: Takové relaci se říká relace podobnosti (proximity relation))

{def-FuzzMir}

**Definice 5.13** (Fuzzy míra [13]). Mějme binární fuzzy relaci  $R$  nad  $\Sigma \cup \{\epsilon\}$ . Pak jako fuzzy míru mezi řetězci  $\alpha, \omega \in \Sigma^*$  (značenou  $S_{\Sigma, R, \otimes}$ ) označme fuzzy relaci danou následujícím předpisem:

$$S_{\Sigma, R, \otimes} = \begin{cases} 1 & \text{pokud } (\alpha, \omega) = (\epsilon, \epsilon) \\ \max_{\zeta \in G(\alpha, \omega)} (\bigotimes_{i=1}^{|\zeta|} R(\zeta_i)) & \text{pokud } (\alpha, \omega) \neq (\epsilon, \epsilon) \end{cases}$$

Definice fuzzy míry je vcelku intuitivní. Počítá se míra všech možných vyrovnaní z nichž se vybírá ta největší. Míra vyrovnaní se určuje jako t-norma ze všech  $R(\zeta_i)$ , tedy stupňů akceptovatelnosti jednotlivých editačních operací. Navíc, míra mezi dvojicí prázdných řetězců je dodefinována jako 1.

Označme  $\mathcal{L}(\omega)$  jako fuzzy jazyk řetězců „podobných“ řetězci  $\omega$  s podobností danou relací  $R$ . Takový jazyk pak můžeme nadefinovat pro všechna  $\alpha \in \Sigma^*$  následujícím předpisem

$$\mathcal{L}(\omega)(\alpha) = S_{\Sigma, R, \otimes}(\alpha, \omega)$$

Nyní zkonstruujeme nedeterministický fuzzy automat s  $\epsilon$ -přechody, který jazyk  $\mathcal{L}$  rozpoznává. (zde bude doplněno: rozpoznává vs. přijímá, pozor na to)

{def-AutRozpCall}

730 **Definice 5.14** (Automat rozpoznávající  $\mathcal{L}$  [13]). Mějme binární fuzzy relaci  $R$  nad  $\Sigma \cup \{\epsilon\}$  (stejná jako v definici 5.13). Pak pro vzorový řetězec  $a_1 a_2 \dots a_n = \omega \in \Sigma^*$  označme  $M_{\Sigma, R, \otimes}(\omega)$  automat rozpoznávající jazyk  $\mathcal{L}(\omega)$  dle definice 1.1, takový, že

1. množina stavů  $Q = \{q_0, q_1, \dots, q_n\}$
2. fuzzy přechodová funkce  $\mu$  pro všechny  $x \in \Sigma$ :
  - (a)  $\mu(q_i, q_i, x) = R(x, \epsilon)$  pro všechny  $q_i \in Q$  taková, že  $i = 0, \dots, n$
  - (b)  $\mu(q_i, q_{i+1}, x) = R(x, a_{i+1})$  pro všechny  $q_i, q_{i+1} \in Q$  taková, že  $i = 0, \dots, n-1$
  - (c)  $\mu(q, q', x) = 0$  pro všechny  $q, q' \in Q$  nesplňující předchozí dva body
  - (d)  $\mu(q_i, q_i, \epsilon) = 1$  pro všechny  $q_i \in Q$  taková, že  $i = 0, \dots, n$
  - 740 (e)  $\mu(q_i, q_{i+1}, \epsilon) = R(\epsilon, a_{i+1})$  pro všechny  $q_i \in Q$  taková, že  $i = 0, \dots, n-1$
  - (f)  $\mu(q, q', \epsilon) = 0$  pro všechny  $q, q' \in Q$  nesplňující předchozí dva body
3. množina počátečních stavů  $\sigma$ :  $\sigma(q_0) = 1$  a pro všechny ostatní  $q_0 \neq q' \in Q$ :  $\sigma(q') = 0$
4. množina koncových stavů  $\eta$ :  $\eta(q_n) = 1$  a pro všechny ostatní  $q_n \neq q' \in Q$ :  $\eta(q') = 0$

Máme nadefinován fuzzy automat rozpoznávající jazyk  $\mathcal{L}(\omega)$ . Bylo by však vhodné dokázat, že jazyk  $\mathcal{L}(\omega)$ , který tento automat rozpoznává je skutečně jazykem řetězců podobných řetězci  $\omega$  s podobností danou relací  $R$ . Vzhledem ke složitosti důkazu tohoto tvrzení se v této práci spokojíme pouze s ilustrací na příkladu.

**Věta 5.1.** *Mějme binární fuzzy relaci  $R$  nad  $\Sigma \cup \{\epsilon\}$  (stejná jako v definici 5.13) a vzorový řetězec  $\omega \in \Sigma^*$ . Pak pro automat  $M_{\Sigma, R, \otimes}(\omega)$  sestavený dle předcházející definice a fuzzy míru  $S_{\Sigma, R, \otimes}$  platí následující rovnost*

$$\mathcal{L}(M_{\Sigma, R, \otimes}) = \mathcal{L}(\omega)$$

*Důkaz.* Kompletní důkaz je k nalezení v [13]. □

(zde bude doplněno: sazba symbolů v matematickém módu vs. verbatim řetězce v textovém)

**Příklad 5.5.** *Uvažujme abecedu  $\Sigma = \{a, b, c\}$  a vzorový řetězec  $\omega = abc$ . Zkonstruujeme automat, který bude akceptovat ve stupni 0.5 náhradu symbolu  $x$  symbolem  $s$  ním v abecedě sousedícím. Navíc uvažujme vložení symbolu  $a$  ve stupni 0.2 a odebrání symbolu  $c$  ve stupni 0.1. Tedy, relace  $R$  bude vypadat následovně:*

$$R = \{(a, a)/1, (b, b)/1, (c, c)/1, \\ (b, a)/0.5, (a, b)/0.5, (c, b)/0.5, (b, c)/0.5, \\ (a, \epsilon)/0.2, (\epsilon, c)/0.1\}$$

Dle definice 5.14 můžeme sestavit automat  $M_{\Sigma, R, \otimes}$ . Jako  $\otimes$  použijme produktovou  $t$ -normu. Získáme tak automat  $M_{\Sigma, R, \otimes} = (Q, \Sigma, \mu, \sigma, \epsilon)$  takový, že:

1.  $Q = \{q_0, q_1, q_2, q_3\}$

2.  $\mu = \{$

(a)  $(q_0, q_0, a)/0.2, (q_1, q_1, a)/0.2, (q_2, q_2, a)/0.2, (q_3, q_3, a)/0.2,$

(b)  $(q_0, q_1, a)/1, (q_1, q_2, a)/0.5, \\ (q_0, q_1, b)/0.5, (q_1, q_2, b)/1, (q_2, q_3, b)/0.5, \\ (q_1, q_2, c)/0.5, (q_2, q_3, c)/1,$

(d)  $(q_0, q_0, \epsilon)/1, (q_1, q_1, \epsilon)/1, (q_2, q_2, \epsilon)/1, (q_3, q_3, \epsilon)/1,$

(e)  $(q_2, q_3, \epsilon)/0.1$

$\}$  (přechody dle bodů (b) a (d) v definici jsou s nulovým stupněm a ve výpisu jsou vynechány)

3.  $\sigma = \{q_0/1, q_1/0, q_2/0, q_3/0\}$

4.  $\eta = \{q_0/0, q_1/0, q_2/0, q_3/1\}$

(zde bude doplněno:  $\sigma = \{x/y, \dots\}$  by se mělo přepsat na  $\sigma(x) = y, \dots$ , ne?)

Přechodový diagram tohoto automatu je k nalezení na obrázku 4. V přechodovém diagramu jsou červeně zvýrazněny pravidla pro rozpoznávání  $\omega$ , ostatní pravidla (doplněna dle definice) jsou černá.



Nyní si na pár řetězcích zkuzme ukázat platnost věty 5.1. Dle definice 2.9 spočítáme stupeň, v jakém automat náš testovací rozpoznává řetězec  $\alpha$ :

$$\begin{aligned}
M_{\Sigma, R, \otimes}(\alpha) &= \max_{q \in Q} (\mu^*(\sigma, \alpha)(q) \otimes \eta(q)) \\
&= \max\{\mu^*(\sigma, \alpha)(q_0) \otimes \eta(q_0), \mu^*(\sigma, \alpha)(q_1) \otimes \eta(q_1), \\
&\quad \mu^*(\sigma, \alpha)(q_2) \otimes \eta(q_2), \mu^*(\sigma, \alpha)(q_3) \otimes \eta(q_3)\} \\
&= \max\{\mu^*(\sigma, \alpha)(q_0) \otimes 0, \mu^*(\sigma, \alpha)(q_1) \otimes 0, \\
&\quad \mu^*(\sigma, \alpha)(q_2) \otimes 0, \mu^*(\sigma, \alpha)(q_3) \otimes 1\} \\
&= \mu^*(\sigma, \alpha)(q_3)
\end{aligned}$$

- řetězec  $\alpha = abc$ : Určíme stupeň akceptance automatem:

$$M_{\Sigma, R, \otimes}(\alpha) = \mu^*(\sigma, abc)(q_3) = \hat{\mu}(\hat{\mu}(\hat{\mu}(\hat{\mu}(\sigma, \epsilon), a), b), c)(q_3) = \{q_3/1\}(q_3) = 1$$

A následně ověříme fuzzy míru. Množina všech vyrovnaní bude obsahovat například  $\zeta_1 = (a, a), (b, b), (c, c)$  či  $\zeta_2 = (a, \epsilon)(\epsilon, a)(b, \epsilon)(\epsilon, b)(c, \epsilon)(\epsilon, c)$ . Snadno zjistíme, že  $\bigotimes_{i=1}^{|\zeta|} R(\zeta_i)$  je maximální právě pro  $\zeta = \zeta_1$  a nabývá stupně 1. A tedy  $S_{\Sigma, R, \otimes}(\alpha) = 1$ .

- řetězec  $\alpha = ab$ :

$$M_{\Sigma, R, \otimes}(\alpha) = \mu^*(\sigma, ab)(q_3) = \hat{\mu}(\hat{\mu}(\hat{\mu}(\sigma, \epsilon), a), b)(q_3) = \{q_2/1, q_3/0.1\}(q_3) = 0, 1$$

$$S_{\Sigma, R, \otimes}(\alpha) = R(a, a) \otimes R(b, b) \otimes R(\epsilon, c) = 1 \otimes 1 \otimes 0.1 = 0, 1$$

- řetězec  $\alpha = bbb$ :

$$M_{\Sigma, R, \otimes}(\alpha) = \mu^*(\sigma, bbb)(q_3) = \dots = \{q_3/0, 25\}(q_3) = 0, 25$$

$$S_{\Sigma, R, \otimes}(\alpha) = R(b, a) \otimes R(b, b) \otimes R(b, c) = 0, 5 \otimes 1 \otimes 0, 5 = 0, 25$$

- řetězec  $\alpha = abaca$ :

$$M_{\Sigma, R, \otimes}(\alpha) = \mu^*(\sigma, abaca)(q_3) = \dots = \{q_3/0, 04\}(q_3) = 0, 04$$

$$S_{\Sigma, R, \otimes}(\alpha) = R(a, a) \otimes R(b, b) \otimes R(a, \epsilon) \otimes R(c, c) \otimes R(a, \epsilon) = 1 \otimes 1 \otimes 0, 2 \otimes 1 \otimes 0, 2 = 0, 04$$

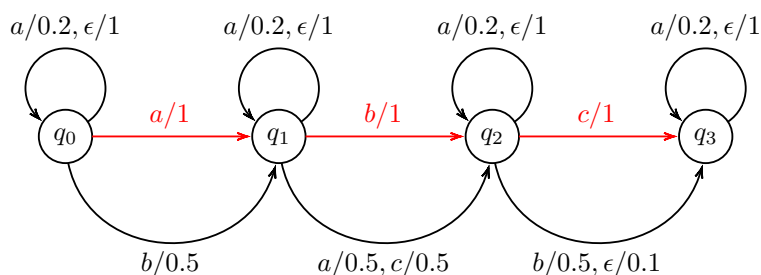
- řetězec  $\alpha = cba$ :

$$M_{\Sigma, R, \otimes}(\alpha) = \mu^*(\sigma, cba)(q_3) = \dots = (\emptyset)(q_3) = 0$$

$$S_{\Sigma, R, \otimes}(\alpha) = R(c, a) \otimes R(b, b) \otimes R(a, c) = 0 \otimes 1 \otimes 0 = 0$$

...

Je tedy zjevné, že výpočet automatu  $M_{\Sigma, R, \otimes}$  je v korespondenci s fuzzy mírou  $S_{\Sigma, R, \otimes}$ .



Obrázek 4: Přechodový diagram automatu z příkladu 5.5

{img-AutRozpCall}

780 Je vidět, že automat zkonstruován dle editačních operací je značně silný nástroj. Umožňuje nám velmi pohodlně popsat, jak moc mohou být konkrétní editační operace akceptovány. Editací operace vložení symbolu, náhrada symbolu a odebrání symbolu jsou pro popis modifikace vzorového řetězce přirozené.

790 Nevýhodou této techniky je, že jednotlivé editační operace jsou akceptovány bez ohledu na jejich výskyt v řetězci. Automat akceptuje nastanuvší editační operaci pokaždé, kde může nastat, ve stejném stupni. Často je však třeba v jiném stupni stejnou editační operaci přijímat např. na začátku a na konci řetězce pokaždé však v jiném stupni. Tento požadavek však automat zkonstruovaný pomocí editačních operací neumí zpracovat. Řešením může být například použití následující techniky.

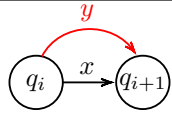
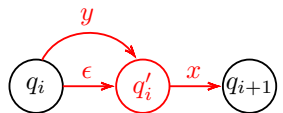
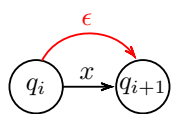
## 5.7 Deformovaný automat

Dalším ze způsobů, jak přijímat řetězec podobný vzorovému je s využitím deformovaného (fuzzy) automatu. Tato technika využívá tzv. deformovaného automatu, neboli automatu který byl stanoveným způsobem upraven, neboli deformován. Tato technika byla přejata z [7].

800 Jako deformace může být použita prakticky jakákoliv úprava automatu. Mějme fuzzy automat  $A$ . Provedením deformace  $x$  získáme deformovaný automat  $A'$ , který rozpoznává jiný jazyk, než automat  $A$ . Mezi nejzákladnější tři deformace patří náhrada symbolu, vložení symbolu před symbol a odebrání symbolu. Možných deformací existuje nekonečně mnoho. Mezi další deformace může patřit například (pro nějaké  $x, y, z \in \Sigma$  a  $i \geq 0$ ): „náhrada symbolu  $x$  na  $i$ -té pozici symbolu  $yz$ “, „odebrání všech výskytů symbolu  $x$ , které se nachází před symbolem  $y$ “ nebo „vložení sudého počtu symbolů  $y$  mezi symboly  $x$  a  $z$ “.

Provedeme-li deformaci fuzzy automatu rozpoznávající  $\omega$ , můžeme se na trojici základních deformací podívat konkrétně. Ukázka toho, jak by vypadal deformovaný automat po provedení jedné ze základních deformací je vyobrazeno v tabulce 1.

810 Je vidět, že deformace mohou být účinným nástrojem pro rozpoznávání modifikovaných pozorovaných řetězců. Na druhou stranu, deformování automatu vyžaduje znalost fungování automatů. Často také může nastat situace, kdy výsledný zdeformovaný automat bude více, než deformovaný automat rozpoznávající  $\omega$ , automatem reprezentující samostatný netriviální vzor.

Deformace	Význam deformace
<p>NÁHRADA symbolu na <math>i</math>-té pozici (symbolu <math>x</math>) symbolem <math>y</math>  <math>\delta' = \delta \cup \{(q_i, y, q_{i+1})\}, Q' = Q</math></p>	
<p>VLOŽENÍ symbolu <math>y</math> na <math>i</math>-tou pozici (před symbol <math>x</math>)  <math>\delta' = \delta \setminus \{(q_i, x, q_{i+1})\} \cup \{(q_i, \epsilon, q'_i), (q_i, y, q'_i), (q'_i, x, q_{i+1})\}, Q' = Q \cup \{q'_i\}</math></p>	
<p>ODEBRÁNÍ symbolu z <math>i</math>-té pozice (symbolu <math>x</math>)  <math>\delta' = \delta \cup (q_i, \epsilon, q_{i+1}), Q' = Q</math></p>	

Tabulka 1: Deformace deformovaného automatu  
(zde bude doplněno: pozor, toto je pro konečné automaty, ne pro fuzzy automaty!)

{tbl-DefAutDef}

## 5.8 Shrnutí

V této kapitole byl zaveden pojem rozpoznávání textových vzorů. Bylo ukázáno, že pro klasickou teorii automatů je to triviální problém, který však kvůli nepřesnostem reálných dat vyžaduje nasazení fuzzy automatů. Bylo představeno několik technik, které pomocí fuzzy automatů umožňují přijímání řetězců podobných vzorovému.

820 Nejjednodušší z nich, využívající relaci podobnosti symbolů, je vhodná na prosté nahrazování podobných symbolů. Technika fuzzy symbolů funguje na stejném principu, jen se liší ve formálním zavedení. Technika s využitím editačních operací umožňuje specifikovat stupeň akceptance základních editačních operací (vlození symbolu, odebrání symbolu, náhrada symbolu). Poslední technika, deformovaný automat, umožňuje libovolnou transformaci automatu, vedoucí až k libovolnému vzoru.

## 6 Fuzzy tree automaty

### 6.1 Zavedení

Fuzzy tree automaty jsou speciální třídou automatů, které jsou navrženy pro rozpoznávání dat, které mají v sobě obsaženu určitou stromovou strukturu. Jak bude ukázáno, fuzzy tree automaty tak mohou rozpoznávat vybrané bezkontextové jazyky.

Fuzzy tree automaty vznikly fuzzyfikací „klasických“ tree automatů. O „klasických“ tree automatech je možné se dočíst více informací např. v [?], popř. [12] a [?]. Problematicke fuzzy tree automatů se věnuje například [?], [18], [19], [?] a [?]. V této kapitole bude vycházeno z [?].

Zatímco běžné konečné (fuzzy) automaty pracují s řetězci symbolů, (fuzzy) tree automaty pracují se speciálními strukturami symbolů, tzv. stromy. Pro snadnější práci s nimi bylo navrženo zakódování do řetězců, kterým se říká pseudotermy. Oba tyto pojmy, a jejich vzájemný vztah budou rozebrány v následující podkapitole. Dále bude nadefinován fuzzy jazyk stromů a automat, fuzzy tree automat, který fuzzy jazyk stromů rozpoznává. Na závěr bude předloženo několik konkrétních ukázek využití fuzzy tree automatů.

Následující dvě podkapitoly budou doprovázeny příklady. Pro vyšší názornost se budou příklady vždy týkat syntaxe jednoduchého algebraického kalkulu. Tento kalkul bude disponovat dvěma proměnnými,  $x$  a  $y$ . Dále pak unárním operátorem  $S$  (symbolizující funkci „sinus“) a binárním operátorem  $M$  (symbolizujícím binární „mínus“, resp. „odečtení druhého argumentu od prvního“). Na závěr bude syntaxe našeho kalkulu fuzzyfikována, takže bude v určitém stupni pravdivosti možné považovat za výraz například  $S(x, y)$  nebo  $M(M(x))$ .

### 6.2 Stromy a pseudotermy

**Definice 6.1** (Doména stromu). *Mějme abecedu  $\Sigma$  uspořádanou pomocí  $\leq$ . Pak konečnou množinu  $U \subseteq \Sigma^*$  nazvěme doména konečného stromu, pokud splňuje následující podmínky:*

- jestliže  $w \in U$  a  $w = uv$  pak  $u \in U$  pro všechna  $u, v, w \in \Sigma^+$  (tj. množina je prefixově uzavřena)
- $wn \in U$  a  $m \leq n$  implikuje  $wm \in U$ , pro všechna  $w \in \Sigma^+$  a  $m, n \in \Sigma$

Na doménu stromu můžeme nahlížet jako na množinu řetězců, které formují prefixový strom. Množinu  $U$  tak lze rozložit na množinu  $\bar{U}$  listových uzlů

$$\bar{U} = \{w \in U \mid wu \notin U \text{ pro všechna } u \in \Sigma^+\}$$

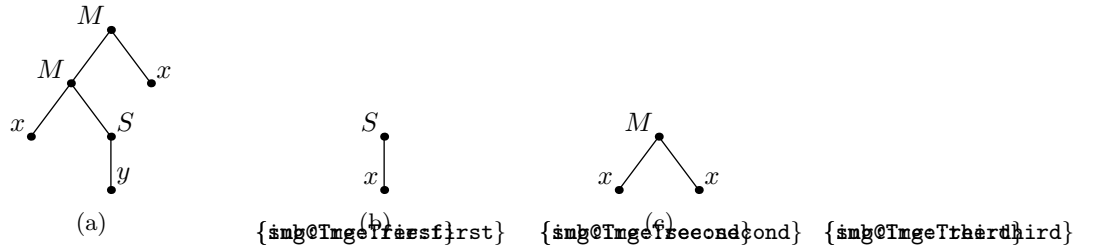
a množninu  $U \setminus \bar{U}$  vnitřních uzlů.

**Definice 6.2** (Částečně spořádaná abeceda). *Částečně spořádaná abeceda je dvojice  $(N, T)$ , kde  $N$  a  $T$  jsou dvě disjunktní konečné abecedy (tj.  $N \cap T = \emptyset$ ).*

**Definice 6.3** (Strom). *Strom  $t$  nad částečně spořádanou abecedou  $(N, T)$  je zobrazení z domény  $U$  stromu do  $(N \cup T)$  (psáno  $t : U \rightarrow (N, T)$ ) takové, že*

- $t(w) \in N$  pokud  $w \in U \setminus \bar{U}$
- $t(w) \in T$  pokud  $w \in \bar{U}$

{def:Tree}



Obrázek 5: Stromy k příkladu 6.1

{img:Tree}

Místo  $t(w)$  budeme psát jen  $t$ .

(zde bude doplněno: Takto definovaný strom však teoreticky může být nekonečný. Co s tím?) Strom  $t$  je tedy předpis pro „přejmenování“ uzlů prefixového stromu daného doménou  $U$ . Strom dle definice 6.3 je v korespondenci s pojmem „strom“ (resp. „kořenový strom“) z teorie grafů. Z tohoto důvodu si pro jednoduchost můžeme odpustit definici souvisejících pojmů z teorie grafů pro strom z definice 6.3. Můžeme tak stromy graficky zobrazovat, hovořit o jejich potomcích, podstromech, vnitřních a listových uzlech bez nutnosti formálního nadefinování.

870

**Příklad 6.1.** Označme  $T = \{x, y\}$  a  $N = \{S, M\}$ . Definujme doménu  $U_1$  stromu pro abecedu  $\Sigma = \mathbb{N}$  jako množinu řetězců  $U_1 = \{\epsilon, 1, 11, 12, 121, 2\}$ . Pak  $\overline{U_1} = \{11, 121, 2\}$ . Strom  $t_1 : U_1 \rightarrow (T, N)$  nad  $(T, N)$  pak může vypadat například takto:

{ex:Trees}

$$\begin{array}{lll} t_1(\epsilon) = M & t_1(1) = M & t_1(12) = S \\ t_1(11) = x & t_1(121) = y & t_1(2) = x \end{array}$$

Grafické znázornění stromu  $t_1$  je na obrázku 5a. Další ukázky stromů jsou na zbylých podobrázcích obrázku 5.

Stromy nám přirozeně reprezentují stromovou hierarchii. Pro nás bude ale občas vhodné mít lineární strukturu pro zápis téhož. Nadefinujeme si proto pseudotermy, protějšky termů predikátové logiky<sup>4</sup>.

**Definice 6.4** (Pseudoterm). Označme  $D_{(N,T)}^p$  nejmenší podmnožinu  $(N \cup T \cup \{(\cdot, \cdot)\})^*$  splňující následující podmínky<sup>5</sup>:

- $T \subset D_{(N,T)}^p$
- pokud  $n > 0$ ,  $A \in N$  a  $t_1, \dots, t_n \in D_{(N,T)}^p$ , pak  $A(t_1 \dots t_n) \in D_{(N,T)}^p$

880

Prvky množiny  $t^p \in D_{(N,T)}^p$  nazýváme pseudotermy.

**Poznámka 6.1.** Definice pseudotermu lze snadno přepsat do gramatiky. Vzhledem k tomu, že taková gramatika bude jistě bezkontextová, bude jazyk  $D_{(N,T)}$  bezkontextový. Tento fakt bude mít důsledek na konstrukci fuzzy tree automatu.

<sup>4</sup>Oproti termům predikátové logiky mají však jiný pohled na nulární funkory, které u pseudotermu neexistují

<sup>5</sup>Předpokládáme, že symboly závorek,  $($  a  $)$  nejsou součástí  $N \cup T$

**Příklad 6.2.** Pro částečně spořádanou abecedu  $(N, T)$  s předchozího příkladu můžeme za termy označit například:  $t_1^p = y$ ,  $t_2^p = S(x)$ ,  $t_3^p = M(xx)$ ,  $t_4^p = M(M(xS(y))x)$ .

Mezi stromy a pseudotermy platí vzájemně převoditelný vztah. To bude nyní dokázáno.

**Věta 6.1.** Pro každý strom  $t \in D_{(N,T)}$  nad částečně spořádanou abecedou  $(N, T)$  existuje odpovídající pseudoterm  $p(t)$ .

*Důkaz.* Existenci pseudotermu dokážeme podle toho, zda-li je  $t$  strom tvořený listovým nebo vnitřním uzlem. Je-li kořenový uzel stromu  $t$  listový, tj.  $t = a$ , kde  $a \in T$ , pak  $p(t) = a$ . V opačném případě, tj. reprezentuje-li kořen stromu  $t$  vnitřní uzel  $t = X$ , kde  $X \in N$ , pak  $p(t) = X(p(t_1) \dots p(t_n))$ , kde  $t_1, \dots, t_n$  jsou podstromy stromu  $t$ .  $\square$

**Věta 6.2.** Ke každému pseudotermu  $p(t) \in D_{(N,T)}^p$  existuje odpovídající strom  $t$ .

*Důkaz.* Opět dokážeme strukturálně:

- je-li pseudoterm atomický, tj.  $p(t) = a$ , kde  $a \in T$ , pak doménou stromu  $t$  je množina  $\{\epsilon\}$  a  $t(\epsilon) = a$
- pokud je pseudoterm ve tvaru  $p(t) = A(t_1^p \dots t_m^p)$ , pak doménou stromu  $t$  je množina  $\bigcup_{i \leq m} \{iw | w \in \text{domain}(t_i)\} \cup \{\epsilon\}$  a

$$t(w) = \begin{cases} A & \text{pokud } w = \epsilon \\ t_i(w') & \text{pokud je } w = iw' \text{ a } w \text{ je v doméně } t \end{cases}$$

$\square$

**Příklad 6.3.** Pseudoterm  $t_4^p$  z předchozího příkladu odpovídá stromu na obrázku 5a a pseudoterm  $t_3^p$  stromu 5c (a naopak).

Máme tedy prokázáno, že mezi pseudotermy a stromy platí vzájemná převoditelnost. Označíme si nyní množinu stromů jako jazyk a fuzzy množinu stromů jako fuzzy jazyk. Obdobným způsobem bychom mohli nadefinovat i jazyk pseudotermů, ale ten nebudeme potřebovat.

**Definice 6.5** (Fuzzy jazyk stromů). Fuzzy množinu  $\tau$  nad  $D_{(N,T)}$  nazvěme fuzzy jazyk stromů.

### 6.3 Fuzzy tree automat a jazyk jím rozpoznávaný

Začneme definicí fuzzy tree automatu.

**Definice 6.6** (Fuzzy tree automat). Fuzzy tree automat  $A$  je pětice  $(Q, T, N, \mu, F)$ , kde:

- $Q$  je konečná množina symbolů stavů
- $T$  je konečná množina terminálních symbolů uzlů

$\mu_x$	$q_1$	$q_2$
$\epsilon$	1	0
$\mu_y$	$q_1$	$q_2$
$\epsilon$	1	0

$\mu_S$	$q_1$	$q_2$
$q_1$	0	1
$q_2$	0	0,4
$q_1q_1$	0	0,3

$\mu_M$	$q_1$	$q_2$
$q_1$	0,1	0,8
$q_2$	0	0,5
$q_1q_1$	0	0,6
$q_1q_2$	0	1
$q_2q_1$	0	0,7

Tabulka 2: Příklad přechodové funkce  $\mu$  fuzzy tree automatu

{tab:MuOfFuzTreAut}

- $N$  je konečná množina neterminálních symbolů uzlů taková, že  $N \cap T = \emptyset$
- $\mu : (N \cup T) \rightarrow \{f | f : (Q \cup \{\epsilon\}) \times Q \rightarrow [0, 1]\}$  je fuzzy přechodová funkce, kde  $Q$  je konečná podmnožina  $Q^+$ . Pro  $X \in N$  je  $\mu(X) = \mu_X$ , kde  $\mu_X$  je zobrazení z  $Q \times Q$  do  $[0, 1]$ . Pro  $a \in T$  je  $\mu(a) = \mu_a$ , kde  $\mu_a$  je zobrazení z  $\{\epsilon\} \times Q$  do  $[0, 1]$ .
- $F \subseteq Q$  je množina koncových stavů.

Podívejme se nyní podrobněji na fuzzy přechodovou funkci  $\mu$ . Pro terminální symbol  $a \in T$  nám  $\mu_a$  definuje fuzzy stav, do kterého automat přejde při vstupu  $a$ . Pro neterminál  $X \in N$  nám definuje přechodovou funkci  $\mu_X(q_1 \dots q_k, q') = c$  s významem „pokud je na vstupu  $X$  a automat se nachází ve stavech  $q_1, \dots, q_k$ , pak automat přejde do stavu  $q'$  ve stupni  $c$ “.

**Příklad 6.4.** Uvažujme množiny  $N$  a  $T$  stejné, jako v předchozích příkladech. Stanovme  $Q = \{q_1, q_2\}$ . Fuzzy množinu  $F$  položme rovnu  $\{q_2\}$  a zobrazení  $\mu$  je zaznačeno v tabulce 2. Pak  $A = (Q, T, N, \mu, F)$  je fuzzy tree automatem.

Na přechodovou funkci se můžeme také podívat pohledem syntaktické analýzy zdola nahoru. Přechodové funkce  $\mu_X$  ( $X \in N$ ) realizují operaci „redukce“ a přechodové funkce  $\mu_a$  ( $a \in T$ ) operaci „přesun“. Můžeme tedy říci, že jazyk stromů (resp. jazyk jim odpovídajících pseudotermů) je fuzzy bezkontextový. Předtím je ale třeba ukázat, že fuzzy tree automaty skutečně přijímají fuzzy jazyky stromů.

**Definice 6.7** (Fuzzy přechodová funkce stromů). Pro strom  $t \in D_{(N,T)}$  definujeme fuzzy přechodovou funkci stromů jako zobrazení  $\mu_t : Q \rightarrow [0, 1]$  následovně:

- Pokud stromu  $t$  odpovídá pseudoterm  $p(t) = X(p(t_1) \dots p(t_k))$ , pak

$$\mu_t(q) = \mu_{X(t_1 \dots t_k)}(q) = \bigvee_{\substack{w \in Q \\ |w|=k}} \left( \mu_X(w, q) \wedge \bigwedge_{j=1}^k \mu_{t_j}(w_j) \right)$$

- pokud  $t = a$ , kde  $a \in T$ , pak  $\mu_t = \mu_a$ .

Fuzzy přechodová funkce stromů je obdobou fuzzy rozšířené přechodové funkce. Pokud je vstupní strom atomický ( $t = a$ ) je přechod realizován pomocí fuzzy přechodové funkce  $\mu_a$ . Pokud vstupní strom není atomický, je přechod realizován ve stupni, který je dán stupněm přechodu neterminálního symbolu a stupně příslušných podstromů.

Stupeň, ve kterém je strom  $t$  automatem přijímán, pak lze určit vztahem

$$A(t) = \bigvee_{q \in F} \mu_t(q)$$

**Definice 6.8** (Jazyk rozpoznávaný). (zde bude doplněno: značení fuzzy jazyka)  
Fuzzy množinu  $L(A)$  nad  $D_{(N,T)}$  danou předpisem

$$L(A) = \left\{ (t, c) \mid c = \bigvee_{q \in F} \mu_t(q) \right\}$$

nazvěme fuzzy jazyk rozpoznávaný fuzzy tree automatem.

**Příklad 6.5.** Uvažujme automat  $A$  z předchozího příkladu. Pak pro strom  $t_1$  (kde  $t_{1,1}$  značí levý podstrom kořene a  $t_{1,2}$  pravý podstrom) z obrázku 5c platí  $\mu_{t_1}(q_2)$ :

$$\begin{aligned} \mu_{t_1}(q_2) &= (\mu_M(q_1 q_1, q_2) \wedge \mu_{t_{1,1}}(q_1) \wedge \mu_{t_{1,2}}(q_1)) \\ &\quad \vee (\mu_M(q_1 q_2, q_2) \wedge \mu_{t_{1,1}}(q_1) \wedge \mu_{t_{1,2}}(q_2)) \\ &\quad \vee (\mu_M(q_2 q_1, q_2) \wedge \mu_{t_{1,1}}(q_2) \wedge \mu_{t_{1,2}}(q_1)) \\ &= (0, 6 \wedge 1 \wedge 1) \vee (1 \wedge 1 \wedge 0) \vee (0, 7 \wedge 0 \wedge 1) = 0, 6 \end{aligned}$$

Pak tedy  $A(t_1) = 0, 6$ . Pro stromy  $t_2$  a  $t_3$  z obrázku 5a a 5a platí  $A(t_2) = 0, 7$  a  $A(t_3) = 1$ .

Podobně jako u klasických automatů, i u tree automatů platí, že pro každý fuzzy jazyk stromů existuje automat, který tento jazyk rozpoznává.

**Věta 6.3.** Pro každý fuzzy jazyk stromů existuje fuzzy tree automat, který ho rozpoznává.

*Důkaz.* K dispozici v [?]. □

V praxi se však nejčastěji setkáme s automatem, který rozpoznává právě jeden strom. Snadnou modifikací takového automatu pak obdržíme automat, který nerozpoznává ostře jen jeden strom, ale v určitém nenulovém stupni také stromy jemu podobné.

**Definice 6.9** (Automat rozpoznávající strom). Mějme strom  $t \in D_{(N,T)}$  („vzor“) kde  $sub(t)$  značí množinu všech jeho podstromů. Pak jako fuzzy tree automat rozpoznávající strom  $t$  označme fuzzy tree automat  $A = (Q, N, T, \mu, F)$ , kde:

- $Q = \{q_{t'} \mid t' \in sub(t)\}$  (každému podstromu odpovídá jeden stav)
- $\mu_a(q_a) = 1$  pro všechna  $a \in T$
- $\mu_X(w, q_{t'}) = 1$  pro všechny  $t' \in sub(t)$ , kde  $w = q_1 \dots q_k$  a stromu  $t_i$  odpovídá pseudoterm  $p(t') = X(t_1 \dots t_k)$
- $F = \{q_t\}$  (koncovým stavem je stav odpovídající celému stromu  $t$ )

Takovýto automat zřejmě rozpoznává jazyk  $T = \{(t, 1)\}$ .

V následujících podkapitolách budou fuzzy tree automaty demonstrovány na konkrétních příkladech.



## 6.4 Použití fuzy tree automatů

970 Fuzzy tree automaty je možné použít všude tam, kde je třeba rozpoznávat určitým způsobem stromově strukturovaná data. Konečnost množiny  $Q$ , pro kterou je definována přechodová funkce  $\mu_X$  neterminálů  $X \in N$  však přináší omezení na aritu každého uzlu, která tak vždy musí být konečným číslem.

Konečnost množiny  $Q$  však teoreticky nemusí být nutná. Teoreticky by šlo jako vzory funkce  $\mu_X$  namísto konkrétních řetězců nad  $Q$  použít například regulární výraz popisující celou třídu takových řetězců (například  $(q_0 \mid q_1)^+$ ). To by však zkomplikovalo implementaci takového automatu a proto toto rozšíření nebude uvažováno.

980 Důležité však je, že automat umožňuje rozpoznávat rekurzivní stromy. Rekurze je dosaženo (opakovaným) přechodem ze stavu do téže stavu (v nenulovém stupni).

## 6.5 Detekce úplných $m$ -árních stromů

Základní technikou, jak lze využít fuzzy tree automaty je přímo práce se stromy. Ukážeme si, že s pomocí fuzzy tree automatů lze snadno rozpoznávat úplné  $m$ -ární stromy.

**Definice 6.10** (Úplný strom). *Úplný  $m$ -ární strom je takový strom, jehož každý uzel má buďto právě  $m$  potomků (vnitřní uzly) a nebo 0 (listové uzly).*

Vlastnost „být úplným  $m$ -árním stromem“ lze poměrně jednoduše fuzzyfikovat. Pro každý vnitřní uzel  $v$  o  $n$  potomcích určíme míru jeho úplnosti. Jinými slovy stupeň pravdivosti výroku „uzel  $v$  má  $m$  potomků“. Tuto míru označme  $\alpha_i$  a můžeme ji určit následujícím vztahem

$$\alpha_v = \frac{n}{m}$$

990 Pro strom  $t$  tvořený nelistovým uzlem  $v$  pak můžeme stupeň  $\alpha_t$  pravdivosti výroku „ $t$  je  $m$ -ární úplný strom“ stanovit jako minimum pravdivosti výroku „uzel  $n$  má  $m$ -potomků“ a pravdivosti „strom  $t'$  je  $m$ -ární úplný strom“ pro všechny jeho podstromy  $t' \in t$ .

Všechny atomické stromy tuto vlastnost splňují ve stupni 1. Můžeme tak napsat:

$$\alpha_t = \begin{cases} 1 & \text{pokud je } t \text{ tvořen listovým uzlem} \\ \alpha_v \wedge \bigwedge_{t' \in t} \alpha_{t'} & \text{pokud je } t \text{ tvořen nelistovým uzlem } v \end{cases}$$

Nyí je třeba vyjádřit tento problém v terminologii fuzzy tree automatů. Uvažujme, že strom  $t$  je tvořen vnitřními uzly  $I$  a listovými uzly  $o$ . Pak fuzzy jazyk  $m$ -árních úplných stromů bude fuzzy jazyk stromů nad  $D_{(N,T)}$ , kde  $N = \{I\}$  a  $T = \{o\}$ . Zbývá tedy navrhnout fuzzy tree automat, který takový jazyk bude rozpoznávat.

Automat bude mít jeden stav  $Q = \{q_1\}$  a přechodovou funkci  $\mu_o(q_1) = 1$  a

$$\mu_I(w, q_1) = \frac{|w|}{m}$$

pro všechna  $w \in \{q_1^i | 1 \leq i \leq m\}$ . Množina koncových stavů  $F$  bude rovna  $\{q_1\}$ .

Dokážeme nyní, že automat  $A = (Q, N, T, \mu, F)$  rozpoznává fuzzy jazyk úplných  $m$ -árních stromů.

**Věta 6.4.** *Fuzzy tree automat  $A = (Q, N, T, \mu, F)$ , kde  $Q$ ,  $N$ ,  $T$ ,  $\mu$  a  $F$  jsou popsány výše, rozpoznává fuzzy jazyk úplných  $m$ -árních stromů.*

*Důkaz.* Jazyk automatu  $A$  je roven  $L(A) = \{(t, c) | c = \mu_t(q_1)\}$  (automat  $A$  má jen jeden koncový stav). Hodnota  $\mu_t(q_1)$  závisí na tom, zda-li je  $t$  tvořen listovým uzlem nebo ne.

Je-li  $t$  tvořen listovým uzlem  $o$ , tj.  $t = o$ , pak  $\mu_t = \mu_o = 1$ .

Strom  $t$  je tvořen vnitřním uzlem  $v$  a  $p(t) = I(p(t_1) \dots p(t_k))$ . Existuje jedno jediné  $w \in Q$  takové, že  $|w| = k$ :  $w = q_1^k$ . Pak  $\mu_t = \mu_I(w, q_1) \wedge \bigwedge_{j=1}^k \mu_{t_j}(q_1)$ . Protože  $q \in Q$  a  $Q = \{q_1\}$ , pak  $\mu_I(w, q) = \mu_I(w, q_1) = \frac{k}{m} = \alpha_v$ .  $\bigwedge_{j=1}^k \mu_{t_j}(q_1)$  je infimum přes všech  $k$  podstromů stromu  $t$ , takže  $\bigwedge_{t' \in t} \mu_{t'}(q_1)$ .

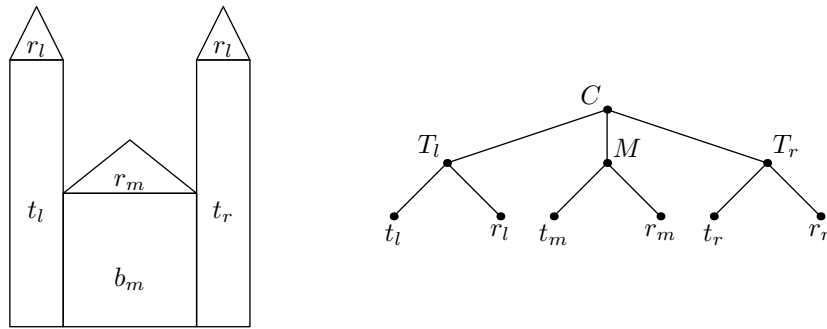
Předpokládejme, že platí  $\mu_t(q_1) = \alpha_t$  pro všechny atomické stromy  $t = a$ , kde  $a \in T$ . Pak  $\mu_t(q_1) = \alpha_v \wedge \bigwedge_{t' \in t} \mu_{t'}(q_1) = \alpha_v \wedge \bigwedge_{t' \in t} \alpha_{t'} = \alpha_t$  pro všechny stromy  $t$  tvořené vnitřními uzly. Pak tedy  $\mu_t(q_1) = \alpha_t$  pro všechny  $t \in D_{(N, T)}$ .  $\square$

Soubory automatu a ukázkových vstupních stromů jsou k nalezení v adresáři `fuzzy-tree-automata/test/data/m-ary-trees`.

## 6.6 Složené geometrické útvary

V [?] byl popsán způsob, jak pomocí fuzzy tree automatů rozpoznávat složené geometrické útvary. Složený geometrický útvar je chápán jako strom, jehož listové uzly reprezentují „primitivní geometrické objekty“ (čtverec, kruh, trojúhelník, aj.). Jeho vnitřní uzly pak popisují vzájemný vztah či vlastnost (např. vzájemnou polohu) jednotlivých podoblastí.

**Příklad 6.6.** Na obrázku 6 je vyobrazen složený geometrický útvar vyobrazující „budovu kostela“ a jemu odpovídající strom.



Obrázek 6: Příklad složeného geometrického tvaru a jeho stromu

{img:Geoms}

V příkladu, který autoři uvádějí, konstruují strom pro náčrt jednoduchého domu a následně kostela. Dům je tvořen čtvercem („budova“) a „nad ním“ se nachází rovnoramenný trojúhelník („střecha“). Kostel pak lze vyjádřit jako „dům, nad který se nachází kříž“.

Pro rozpoznávání takovýchto stromů fuzzy tree automatem je nutné tyto pojmy nejdříve formalizovat. Jakmile budeme mít pevně stanoveny jednotlivé pojmy, budeme moci provést jejich fuzzyfikaci a následně sestavit fuzzy tree automat, který stromy rozpoznává.

1030 Uvažujme primitivní geometrický útvar  $g$ . Konkrétní podoba útvaru  $g$  bude dána jeho typem. Například mnohoúhelníky budeme reprezentovat jako posloupnosti jejich vrcholů, kružnice bude reprezentována jako střed a poloměr. Připomeňme si nejdříve matematické definice některých základních primitivních geometrických tvarů. (zde bude doplněno: je to nutné zdrojovat?)

**Značení.** Pro mnohoúhelník  $g$  označme  $\alpha_X$  jako velikost úhlu při vrcholu  $X \in g$ .

**Definice 6.11** (Obdélník). Mějme čtyřúhelník  $g = ABCD$ . Pak tento čtyřúhelník je obdélník, pokud platí

$$\alpha_A = \alpha_B = \alpha_C = \alpha_D (= 90^\circ)$$

**Definice 6.12** (Čtverec). Mějme obdélník  $g = ABCD$ . Pak tento obdélník je čtverec, pokud

$$|AB| = |BC| = |CD| = |DA|$$

**Definice 6.13** (Rovnoramenný trojúhelník). Mějme trojúhelník  $g = ABC$ . Pak tento trojúhelník je rovnoramenný, pokud

$$\alpha_A = \alpha_B$$

Stranu  $AB$  nazýváme základna, strany  $AC$  a  $BC$  ramena.

Obdobným způsobem bychom ve výčtu mohli pokračovat. Pro užití fuzzy tree automatů však bude vhodné nehovořit o „útvary  $g$  je/není obdélník“, ale „útvary  $g$  je obdélníkem ve stupni  $c$ “.

Označme  $\varepsilon : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \rightarrow [0, 1]$  jako fuzzy ekvivalenci reálných čísel danou předpisem:

$$\varepsilon(x, y) = \begin{cases} \frac{x}{y} & \text{pokud } x \leq y \\ \frac{y}{x} & \text{pokud } x > y \end{cases}$$

1040 s tím, že  $\frac{0}{0} = 1$ . Zřejmě platí  $\varepsilon(x, x) = 1$  pro všechna  $x \in \mathbb{R}_0^+$ .

S využitím fuzzy ekvivalence  $\varepsilon$  tka můžeme předchozí tři definice „fuzzyfikovat“:

**Definice 6.14** („Fuzzy“ obdélník). Mějme čtyřúhelník  $g = ABCD$ . Pak tento čtyřúhelník je obdélníkem ve stupni  $\gamma_r(g)$ , kde

$$\gamma_r(g) = \varepsilon(\alpha_A, \alpha_B) \wedge \varepsilon(\alpha_B, \alpha_C) \wedge \varepsilon(\alpha_C, \alpha_D) \wedge \varepsilon(\alpha_D, \alpha_A)$$

**Definice 6.15** („Fuzzy“ čtverec). Mějme geometrický útvar  $g = ABCD$ , který je obdélníkem ve stupni  $\gamma_r(g)$ . Pak tento obdélník je čtvercem ve stupni  $\gamma_s(g)$ , kde

$$\gamma_s(g) = \gamma_r(g) \wedge \varepsilon(|AB|, |BC|) \wedge \varepsilon(|BC|, |CD|) \wedge \varepsilon(|CD|, |DA|) \wedge \varepsilon(|DA|, |AB|)$$

**Definice 6.16** („Fuzzy“ rovnoramenný trojúhelník). *Mějme trojúhelník  $g = ABC$ . Pak tento trojúhelník je rovnoramenný ve stupni  $\gamma_i(g)$ , kde*

$$\gamma_i(g) = \varepsilon(\alpha_A, \alpha_B)$$

Máme tedy fuzzifikovány vlastnosti primitivních geometrických útvarů. Nyní je třeba navrhnout fuzzifikace jejich vzájemných vztahů. Pro vztah „být nad“ máme například:

**Definice 6.17** (Vztah „být nad“<sup>6</sup>). *Mějme obdélník  $r = ABCD$  a trojúhelník  $q_t = EFG$ . Pak „trojúhelník  $r$  je nad obdélníkem  $q_t$  (a horní hrana  $q_r$  splývá se základnou  $t$ )“ právě tehdy, když:*

$$top(r) = base(t)$$

kde  $top(r) \in \{AB, BC, CD, DA\}$  je „horní strana“ obdélníku  $r$  a  $base(t) \in \{EF, FG, GA\}$  je základna trojúhelníku  $t$ .

*Mějme geometrický útvar  $r' = ABCD$ , který je obdélníkem ve stupni  $\gamma_r(r')$  a trojúhelník  $q'_t = EFG$ . Pak „trojúhelník  $r'$  je nad obdélníkem  $q'_t$  (a horní hrana  $q'_r$  splývá se základnou  $t'$ )“ ve stupni  $\gamma_T(r', t')$ , kde*

$$\gamma_T(r', t') = \gamma_r(r') \wedge \varepsilon(top(r'), base(t'))$$

a kde  $\varepsilon(XY, ZW) = \varepsilon(X, Z) \wedge \varepsilon(Y, W)$  je fuzzy ekvivalence úseček a  $\varepsilon(X, Y) = \bigwedge_i \varepsilon(X_i, Y_i)$  je fuzzy ekvivalence vrcholů.

1050 Máme tedy formálně popsány a fuzzifikovány vlastnosti primitivních geometrických tvarů a (alespoň jednu) vlastnost popisující složený geometrický tvar. Položme  $T = \{r, s, t, i\}$  jako terminály symbolizující obdélník, čtverec, (obecný) trojúhelník a rovnoramenný trojúhelník. Dále stanovme  $N = \{T\}$ . Pak pseudo-term  $p(t_H) = T(i, s)$  nad  $(N, T)$  symbolizuje dům popsáný výše.

Označme  $A_H = (Q, N, T, \mu, F)$  jako fuzzy tree automat rozpoznávající strom  $t_H$ . Označme  $A'_H = (Q, N, T, \mu', F)$ , kde  $\mu'$  vznikla z  $\mu$  nahrazením všech 1 (pro všechna  $X \in (N \cup T)$ ) výrazem  $\gamma_X$ .

**Příklad 6.7.** *Uvažujme složený geometrický útvar  $C$  z obrázku 6. Pak automat  $A'$  bude nějak vypadat. (zde bude doplněno: domyslet to nějak!) .*

1060 Takto vytvořený automat dokáže rozpoznávat geometrické tvary „podobné“ (ve smyslu relací  $\gamma$ ) vzorovému. Nevýhodou tohoto řešení je, že je pevně svázán s aritou (a pořadím potomků) uzlů vzorového stromu. Navíc, stupeň pravdivosti popisující vztah v uzlu  $U$  stromu je schopen kalkulovat pouze se svými potomky (a nikoliv například svými sousedy či předky). Obě tyto výhody se však smývají, pokud se bude pozorovaný strom od vzoru lišit jen málo.

I přes tyto nevýhody však lze fuzzy tree automaty použít na podobnostní rozpoznávání složených geometrických tvarů.

(zde bude doplněno: To samé dělá i v [?], akorát to dělá přes jazyky a gramatiky, ne přes automaty. Ale takových případů jsem měl víc, zmínit je v sekci problémy v metodice!)

1070

<sup>6</sup>Zde si dovoluujeme značné zjednodušení. Vztah „být nad“ by měl být popsán například s využitím porovnávání y-ových souřadnic bodů.

## 7 Buněčné fuzzy automaty

Buněčné (fuzzy) automaty jsou další z výpočetních modelů, které se svojí základní myšlenkou podobají (fuzzy) automatům. Oproti „klasickým“ (fuzzy) automatům mají však značně zajímavější možnosti uplatnění. Proto jim v této práci bude věnována samostatná kapitola.

### 7.1 „Bivalentní“ buněčný automat

Buněčné automaty a fuzzy buněčné automaty jsou výpočetní modely, které se koncepčně značně liší od klasických automatů. Dle [?] je jejich studium dokonce označováno za naprosto samostatné matematické paradigma. I přesto je v určitém smyslu možné je považovat za zobecnění „klasických“ deterministických automatů. Dá se totiž říci, že se jedná o  $n$ -dimenzionální mřížku tvořenou instancemi téže konečného automatu.

Přesné vymezení pojmu „buněčný automat“ se často různí. Některé definice uvažují nekonečnou mřížku (např. [?], [?], [?]) jiné zase konečnou (např. [?]). Také se často kromě klasické čtvercové mřížky pracuje s mřížkou trojúhelníkovou nebo šestiúhelníkovou (např. [?]).

Vzhledem k tomu, že úkolem této práce není studovat obecné vlastnosti buněčných automatů ale jen jejich fuzifikace a následné použití v praxi, bude zde nadefinován pouze standardní dvoudimenzionální buněčný automat (pracující na čtvercové mřížce). Právě tento typ automatu totiž našel v praxi největší uplatnění. V této práci se také pro jednoduchost omezíme jen na automat se čtvercovou mřížkou velikosti  $m$ .

Dvoudimenzionální buněčný automat je tedy mřížka  $m \times m$  tvořena buňkami  $c_{ij}$ ,  $i, j \in [1, m]$ . Každá buňka se nachází ve nějakém stavu  $q$  z množiny  $Q$ . Přechody mezi těmito stavy jsou realizovány přechodovými pravidly. Ty popisuje přechodová funkce  $\mu$ . Formálně tedy

**Definice 7.1** (Bivalentní buněčný automat). *Pro přirozené číslo  $m$  a konečnou množinu  $Q$  označme  $A_m = (Q, \mu)$  jako dvoudimenzionální buněčný automat o rozměrech  $m \times m$ , kde  $Q$  je konečná množina stavů a  $\mu$  přechodová funkce:*

$\mu : Q \times Q^k \rightarrow Q$  pro nějaké  $1 \leq k \leq m^2$ .

**Poznámka 7.1.** *Buněčný automat se obvykle definuje jen předpisem pro přechodovou funkci, resp. výpisem přechodových pravidel. My se však budeme držet konceptu klasických automatů a budeme buněčný automat definovat jako strukturu.*

**Značení.** *Kde to bude možné, budeme indexy  $i, j$  vynechávat a namísto  $c_{i,j}$  psát jen  $c$ . Fakt, že  $c$  je buňkou automatu  $A$  budeme značit  $c \in A$ . Fakt, že nějaká buňka  $c$  se nachází ve stavu  $q$  budeme značit  $c = q$ .*

Přechodová funkce  $\mu$  přiřazuje buňce  $c$  stav  $q'$  na základě aktuálního stavu  $q$  a stavu dalších, tzv. okolních,  $k$  buněk. Označme  $round(c_{ij})$  jako okolí buňky  $c_{ij}$ . Nejpoužívanějším okolím, které se používá, je Mooreovo okolí o poloměru 1, které je definováno jako sousedních 8 buněk buňky  $c_{ij}$ :

$$round(c_{i,j}) = (c_{i-1,j-1}, c_{i,j-1}, c_{i+1,j-1}, \\ c_{i-1,j}, c_{i+1,j}, \\ c_{i-1,j+1}, c_{i,j+1}, c_{i+1,j+1})$$

s tím, že nedefinované hodnoty ( $i, j < 1$  nebo naopak  $i, j > m$ ) za hranicemi mřížky se stanovují na nějakou pevně zvolenou hodnotu z  $Q$ .

Přechodová funkce  $\mu$  pak vypadá následovně:

$$\mu(c, \text{round}(c)) = f(c, \text{round}(c))$$

kde  $f$  je funkce přiřazující buňce  $c$  s okolními buňkami  $\text{round}(c)$  nový stav. V praxi se nejčastěji používá sada tzv. If-Then pravidel.

{ex:GameOfLife}

**Příklad 7.1.** Typickým příkladem buněčného automatu je tzv. Hra života (Game of Life) (např. [?]). Jedná se o jednoduchý simulátor živého organismu.

Hra života uvažuje dvoustavovou množinu stavů, tj.  $Q = \{0, 1\}$  a dvoudimenzionální mřížku ( $n = 2$ ). Je-li hodnota buňky  $c$  rovna 1 hovoříme, že je buňka „živá“, je-li rovna 0 nazýváme buňku „mrtvou“. Přechodová funkce  $\mu$  je dána následujícími pravidly:

1. Je-li buňka  $c$  živá a je v jejím okolí méně, než 2 živé buňky, buňka umírá (ve smyslu „samoty“)
2. Je-li buňka  $c$  živá a je v jejím okolí více, než 3 živé buňky, buňka umírá (ve smyslu „vyčerpání zdrojů“)
3. Je-li buňka  $c$  mrtvá, a v jejím okolí jsou přesně 4 živé buňky, je buňka oživena
4. Ve všech ostatních případech zůstává buňka buďto mrtvá nebo živá

Označme

$$\text{neighs}_{i,j} = \left( \sum_{k,l \in \{-1,0,+1\}} c_{i+k,j+l} \right) - c_{i,j}$$

jako počet živých buněk sousedících s  $c_{i,j}$ .

Pak můžeme přechodovou funkci  $\mu$  zapsat následovně:

$$\mu(c_{i,j}, \text{round}(c_{i,j})) = \begin{cases} 0 & \text{pokud } c_{i,j} = 1 \text{ a } \text{neighs}_{i,j} < 2 \\ 0 & \text{pokud } c_{i,j} = 1 \text{ a } \text{neighs}_{i,j} > 3 \\ 1 & \text{pokud } c_{i,j} = 0 \text{ a } \text{neighs}_{i,j} = 4 \\ c_{i,j} & \text{jinak} \end{cases}$$

Soubor všech stavů všech buněk se nazývá – podobně, jako u konečných automatů – konfigurace buněčného automatu.

**Definice 7.2** (Konfigurace buněčného automatu). Zobrazení  $C : [1, m] \times [1, m] \rightarrow Q$  se nazývá konfigurace buněčného automatu.

Podobně jako u klasických automatů můžeme hovořit o počáteční konfiguraci. Ta – na rozdíl od klasických automatů – může být libovolná. Koncová konfigurace však pro buněčné automaty neexistuje. Výpočet buněčného automatu totiž nemá stanoven žádný konec. Můžeme však uvažovat konfiguraci dosažitelnou. Pro její zavedení je však třeba nadefinovat výpočet buněčného automatu.

**Definice 7.3** (Krok výpočtu a výpočet buněčného automatu). *Binární relaci  $\vdash$  na množině konfigurací buněčného automatu nazvěme krok výpočtu, pokud  $C \vdash C'$  ( $(C, C') \in \vdash$ ) a pro všechny  $c_{i,j} \in A$  platí:*

$$C'(i, j) = \mu(C(i, j), \text{round}(c_{i,j}))$$

*Reflexivní a tranzitivní uzávěr  $\vdash^*$  relace  $\vdash$  nazvěme výpočtem buněčného automatu.*

**Definice 7.4** (Konfigurace dosažitelná). *Mějme konfiguraci  $C$  buněčného automatu. Pak o konfiguraci  $C'$  říkáme, že je dosažitelná z konfigurace  $C$ , pokud existuje výpočet z  $C$  k  $C'$ , tj.*

$$C \vdash^* C'$$

*V opačném případě říkáme, že konfigurace je nedosažitelná.*

Vzhledem k tomu, že přechodová funkce buněčného automatu je deterministická, je zřejmě deterministický i její výpočet, tj. pro každou konfiguraci  $C$  existuje právě jedna konfigurace  $C'$ , pro kterou platí  $C \vdash C'$ . Relace  $\vdash$  tak generuje posloupnost konfigurací. Očíslujeme-li si tuto posloupnost, pak počáteční konfigurace výpočtu bude  $C^{(0)}$  a posloupnost pak bude vypadat následovně:

$$C^{(0)} \vdash C^{(1)} \vdash C^{(2)} \vdash \dots$$

Nazýváme  $i$ -tý prvek této posloupnosti jako konfigurace  $i$ -té generace a samotné hodnoty  $i$  jako generace (hovoříme tak o nulté, první, druhé, ... generaci).

1140 Konfigurace buněčného automatu se často zobrazuje graficky. Zakresluje se jako bitmapa, kde jednotlivé pixely reprezentují stavy buněk na odpovídajících souřadnicích. Každému stavu je přiřazena barva, kterou je tento stav znázorněn. Pro zobrazení výpočtu se občas používá třírozměrné zobrazení (tj. voxelový obrázek), kde třetí rozměr odpovídá generaci.

**Příklad 7.2.** *Ukázka výpočtu automatu  $A_{100}$  realizující Hru života je na obrázku 7. Černá barva symbolizuje mrtvou buňku, bílá pak živou.*

## 7.2 Buněčné fuzzy automaty

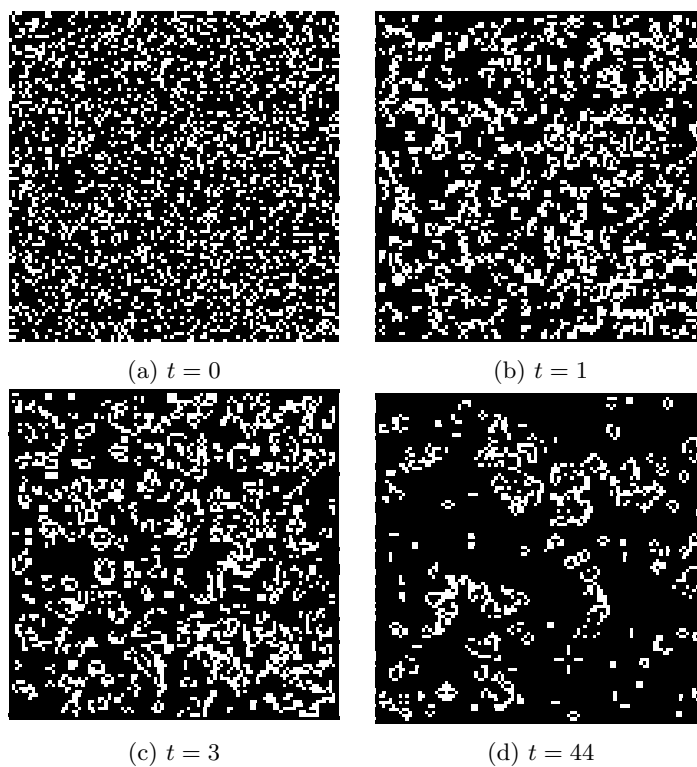
1150 Buněčný fuzzy automat není na rozdíl od klasického fuzzy automatu prostým zobecněním bivalentního buněčného automatu. Hlavním důvodem je bezesporu fakt, že přechodová funkce bivalentního buněčného automatu je deterministická a úplná. Tedy, že každá buňka vždy přejde ze stavu  $q$  do nějakého stavu  $q'$ . Buňka se tedy musí nacházet vždy v právě jednom stavu. Nemůže nastat situace, že by buňka přešla „do více stavů současně“<sup>7</sup> (přechodová funkce by nebyla deterministická) nebo nepřešla do žádného (přechodová funkce by nebyla úplná).

Vzhledem k tomu, že stejné chování budeme vyžadovat i u buněčného fuzzy automatu, „odstupňování“ přechodové funkce (zavední fuzzy přechodové funkce) by nemělo smysl<sup>8</sup>.

1160 Triviální způsob, jak zavést buněčný fuzzy automat je jako bivalentní buněčný automat, jehož množina stavů je rovna intervalu  $[0, 1]$ . Takovýto automat budeme označovat jako  $[0, 1]$ -buněčný fuzzy automat.

<sup>7</sup>Jak bude ukázáno, tento předpoklad lze obejít

<sup>8</sup>Teoreticky by bylo možné nahradit stav buňky fuzzy stavem buňky. Takováto úprava by však výrazně zvýšila výpočetní složitost výpočtu automatu a například také znesnadnila grafické znázornění jeho konfigurací a proto zde nebude uvažován.



Obrázek 7: Několik generací výpočtu buněčného automatu „Hra života“ s náhodnou počáteční konfigurací

{img:GameOfLife}



**Definice 7.5** ( $[0, 1]$ -buněčný fuzzy automat). *Jako  $[0, 1]$ -buněčný fuzzy automat velikosti  $m$  budeme označovat bivalentní buněčný automat  $A = (Q, \mu)$ , kde  $Q = [0, 1]$ .*

Jedná se tedy jen o speciální případ klasického bivalentního buněčného automatu. Každá buňka  $c$  takového automatu pak vždy splňuje následující tvrzení:

- „buňka  $c$  se nachází ve stavu 1“ ve stupni  $q$
- „buňka  $c$  se nachází ve stavu 0“ ve stupni  $1 - q$

Díky tomu je obejit předpoklad, že automat se smí nacházet pouze v jednom stavu současně.

Druhý způsob, jak definovat buněčný fuzzy automat, může pracovat s libovolnou množinou stavů. Využívá fuzzy If-Then pravidel, bude mu proto říkáno „buněčný automat s fuzzy logikou“.

**Příklad 7.3.** *Příkladem buněčného automatu s fuzzy logikou může být například následující automat. Množina stavů bude obsahovat přirozená čísla z intervalu  $[0, 150)$ . Stanovíme čtveřici fuzzy množin  $\varsigma_L$ ,  $\varsigma_M$ , a  $\varsigma_H$  ve významu „hodnota  $q$  je nízká“, „hodnota  $q$  je střední“ a „hodnota  $q$  je vysoká“. Pravidla automatu pak mohou vypadat následovně:*

- Pokud  $q_{i-1,j-1}^{(t)} = L$ , pak  $q_{i,j}^{(t+1)} = H$
- Pokud  $q_{i-1,j-1}^{(t)} = L$  nebo  $q_{i-1,j-1}^{(t)} = M$ , pak  $q_{i,j}^{(t+1)} = M$
- Pokud  $q_{i,j-1}^{(t)} = L$  a  $q_{i,j+1}^{(t)} = L$ , pak  $q_{i,j}^{(t+1)} = L$

### 7.3 Obecně k aplikacím

Buněčné automaty (obecně) nacházejí široké uplatnění v rozličných oblastech. Dle [?] se s nimi lze setkat v matematice, informatice, fyzice, biologii, společenských vědách, např. filozofii a umění. Používají se například pro simulace fyzikálních dějů (např. difuze, tok tekutin), krystalizace, biologickým, urbanistickým, environmentalistickým a geografickým simulacím či například ke generování fraktálů.

Co se buněčných fuzzy automatů týče, jejich aplikace nejsou tak rozšířené. Jedním z důvodů, proč tomu tak je, je velká podobnost bivalentních buněčných automatů a buněčných fuzzy automatů. Jak bylo uvedeno v předchozí podkapitole,  $[0, 1]$ -buněčný fuzzy automat je jen speciálním případem klasického bivalentního. Obdobně, práce s fuzzy If-Then pravidly je vlastně jen speciální případ klasických If-Then pravidel.

Například [?] používá čísla z intervalu  $[0, 1]$  jako vstupní informaci. [?] používá automat podobný buněčnému automatu s fuzzy logikou, který navíc pracuje se stavy na intervalu  $[0, 1]$ . Obdobně, [5] používá buněčné automaty v kombinaci s určitou formou fuzzy logiky (avšak automat napovažují za buněčný fuzzy automat). V [?] je použit automat pracující na intervalu  $[0, 255]$ , který by šel snadnou modifikací konvertován na  $[0, 1]$ -buněčný fuzzy automat. Automat v [?] vykazuje určitou náhodnost a bylo by tak možné považovat jej za pravděpodobnostní.

Další varianty buněčných automatů a jejich uplatnění jsou vyjmenovány v [?]. Ve všech případech se uvažují vždy dvoudimenzionální buněčné automaty.

Aplikace fuzzy automatů se dají rozdělit do dvou kategorií. Do první z nich spadají urbanistické simulace. Pomocí buněčných fuzzy automatů tak byly

1200 řešeny simulace hustoty dopravy [?], růstu mořských řas u pobřeží [?] či plánování elektrické rozvodné sítě. Zdaleka nejčastější však bylo nasazení buněčných fuzzy automatů na řešení problému městského růstu. Z tohoto důvodu bude tomuto problému věnována samostatná podkapitola.

Další oblastí, kde našly buněčné fuzzy automaty uplatnění je zpracování obrazu. Používají se například na zaostřování obrazů [?][?], hledání hran [?][?], vyhledávání vzorů [?][?] či odstranění šumu [?][?][?]. V následujících podkapitolách budou některé takové techniky rozebrány podrobněji.

Co se pravděpodobnostních buněčných automatů týče, ty nacházejí uplatnění všude tam, kde je třeba dodat náhodnost a nepravidelnost. Například pro problém 1210 simulace městského růstu [?] [?]. Dále pak například pro náhodné generátory, generátory šumu, simulace pohybu částic, difuze částic či nukleace (vznik krystalů) [?].

Před studiem samotných aplikací je nutné dodat, že aplikace zde popsané jsou pouze teoretickým popisem sestaveným na základě použitých zdrojů. Každá z těchto aplikací vyžaduje kalibraci parametrů (např. počet generací, návrh fuzzy množin) a přizpůsobení na míru konkrétní instance problému. Většina zdrojů proto automaticky kombinuje několik technik dohromady. Například [?] [?] využívají buněčné fuzzy automaty v kombinaci s neuronovými sítěmi, v [?] s genetickými algoritmy a [?] [?] s pomocí učícího se automatu. (zde bude do- 1220 plněno: dohledat, vysvětlit a ocitovat, co to je učící se automat) Většinou je také nutná hluboká znalost dané problematiky, nejlépe přítomnost experta.

## 7.4 Problém městského růstu (urban growt problem)

Problém městského růstu je problém z oblasti urbanistiky. Řešením tohoto problému je co nejpresnější predikce rozvoje městské zástavby na základě historických záznamů a současné situace. Ve zjednodušené podobě se nemusí jednat jen o růst městské zástavby, ale například nárůst vytíženosti silnic, kácení lesů nebo vytěženost ložisk. Stejně tak se nutně nemusí jednat o růst, ale obecný vývoj v čase. V této kapitole však budeme pro jednoduchost uvažovat stan- dardní problém, tedy městský růst.

1230 Buněčné fuzzy automaty byly již mnohokrát použity pro řešení tohoto problému. Pomocí těchto automatů byl například modelován rozvoj zástavby v městě Riyadh v Saudské Arábii [?], [?], regionu Helensvale v Austrálii [?], ostrova Sv. Lucie v Karibském moři [?], oblasti North Vancouver v Kanadě [?], oblasti Mesogia v Řecku [?], [?], oblasti Sanfranciského zálivu v Kalifornii [?] nebo části Tianhe města Guangzhou v jihovýchodní Číně [?]. Další literatura věnující se uplatnění (fuzzy) buněčných automatů při řešení problému městského růstu je k dispozici např. zde: [4], [?] a [?].

(zde bude doplněno: el. síť [?]? Doprava (toho mám taky povícero)? )

Pojďme se nyní podívat, jak se buněčné fuzzy automaty pro řešení tohoto 1240 problému používají. Základní idea pro nasazení fuzzy automatů je následující: Stav zástavby reprezentujeme jako konfiguraci fuzzy buněčného automatu. Pak růst zástavby bude odpovídat přechodům mezi těmito konfiguracemi.

V první fázi je nutné si sledovanou oblast („město“) rozdělit na dílčí parcely. Každé takové parcele pak bude odpovídat jedna buňka buněčného fuzzy automatu. U každé parcely je třeba zjistit rozličné ukazatele, např.:

- je-li parcela zastavěna (popř. jak moc)

- typ zástavby na parcele (např. rodinné domy, obytné domy, komerční prostory, prostory pro rekreaci, dopravní stavby, průmyslová zóna)
- jak moc žije na parcele obyvatel
- 1250 • jak moc vysoké budovy stojí na parcele
- jak velké produkuje parcela znečištění ovzduší/hluku

Nejčastěji se jako ukazatel používá informace o zastavěnosti parcely. Ta se totiž dá poměrně snadno stanovit s pomocí satelitních snímků sledované oblasti.

Dále je třeba získat ukazatele, které mají vliv na růst zástavby. Mezi takového ukazatele patří například:

- vzdálenost parcely od centra města (popř. škol, nákupních center, ...)
- dopravní obslužnost parcely (vzdálenost od hlavní silnice nebo zastávek hromadné dopravy)
- atraktivita lokality (výhled na město, okolní zástavba, ...)
- 1260 • stavební podmínky (podloží, záplavová zóna, terén, ...)

Za povšimnutí stojí, že některé ukazatele jsou neměnné v čase (např. vzdálenost od centra města nebo podloží).

Následně je možné sestavit přechodová pravidla. Ta mohou být například:

- Je-li vzdálenost parcely od centra města malá, pak růst zástavby bude velký
- Je-li vzdálenost od hlavní silnice velmi malá, pak růst zástavby bude malý a množství hluku bude velké
- Je-li vzdálenost od hlavní silnice velmi velká, pak růst zástavby bude malý
- Není-li lokalita atraktivní, pak růst zástavby bude malý

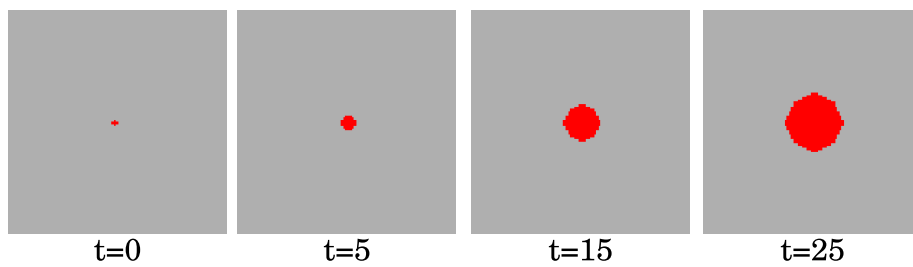
1270 Další ukázky pravidel jsou např. v [?], [11] a[?]. Ukázku toho, jak se chová automat při různých počátečních konfiguracích lze spatřit na obrázku 8.

Na obrázku 9 je k vidění konkrétní ukázka městského růstu. Na obrázku jsou pro porovnání zobrazeny jak vypočtené stavy zástavby, tak i skutečné (upravené satelitní snímky). Na snímcích je patrné, že simulace rozvoje mezi lety 1987 a 1997 dosáhla poměrně přesných výsledků. Stejně tak, v simulaci růstu mezi lety 1997 a 2005 naznačuje jen malý rozvoj. Při simulaci od roku 1987 do roku 2005 už jsou patrné větší odlišnosti (simulace nevyprodukovala tak výrazný růst, jaký doopravdy nastal).

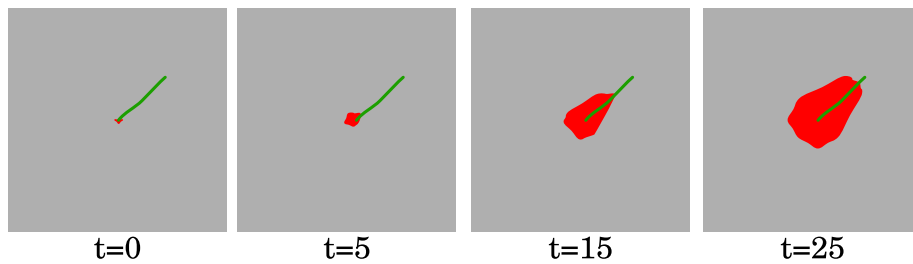
## 7.5 Zpracování obrazu

1280 Zpracování obrazu je v dnešní době velmi populární informatická disciplína. Jak bude ukázáno, nasazení buněčných fuzzy automatů zde nachází značné uplatnění.

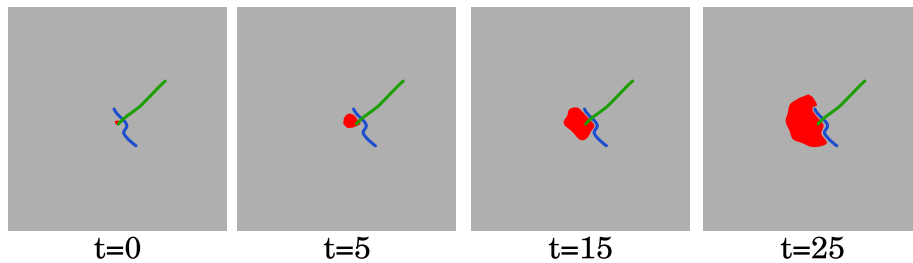
Uvažujme obraz jako mřížku  $m \times m$  pixelů s odstíny šedi jako hodnotami od 0 do 1. Hodnota 0 značí černou, hodnota 1 bílou. Jinými slovy, barva (resp.



(a) Růst města bez dalších dodatečných podmínek



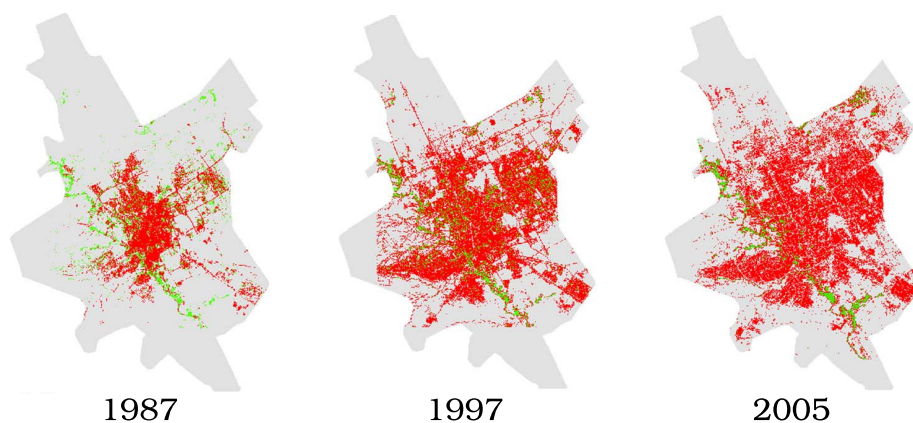
(b) Růst města podél silnice



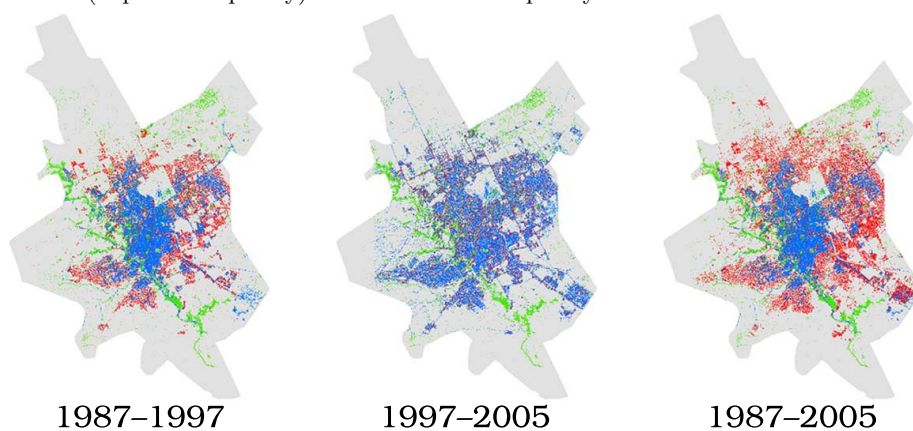
(c) Růst města bržděn řekou

Obrázek 8: (převzato z [?], upraveno) Ukázky chování automatu při různých počátečních konfiguracích. Šedě jsou znázorněny prázdné parcely, červeně zástavba, zeleně hlavní silnice a modře řeky.

{img-VarTransRuls}



(a) Skutečný stav zástavby v uvedeném roce. Červená značí zástavbu, zelená přírodní oblasti (např. vodní plochy) a šedá nezastavěné plochy.



(b) Simulovaný stav zástavby. Modrá značí zástavbu na počátku sledovaného období, červená značí (novou) zástavbu na konci sledovaného období, zelená přírodní oblasti (např. vodní plochy) a šedá nezastavěné plochy.

Obrázek 9: (převzato z [?]) Ukázky simulace městského růstu

{img-UrbGroProSample}



Obrázek 10: Ukázky jednoduchých filtrů. Vlevo původní obrázek, uprostřed obrázek po 5 generacích jednoduchého rozostřovacího filtru a v pravo obrázek po 8 generacích filtru pro zvýraznění tvarů ( $\epsilon = 1, 1$ ).

{img:Filters}

stupeň šedi) pixelu odpovídá stupni pravdivosti tvrzení „pixel má bílou barvu“. Tato skutečnost nám umožňuje pracovat s obrazem pomocí fuzzy logiky.

Každý obraz tak můžeme považovat za konfiguraci buněčného fuzzy automatu s množinou stavů  $Q = [0, 1]$ . Návrhem vhodné přechodové funkce tak můžeme vytvořit automat, který provádí určitou operaci pro úpravu obrazu. Typickou operací je tzv. obrazový filtr, který obrazu  $m \times m$  přiřazuje obraz  $m \times m$ .

Speciálním případem takového automatu je automat realizující konvoluční metodu [?]. Konvoluce je v základu obrazový filtr, který přiřazuje (novou) hodnotu pixelu na základě váženého součtu (stávající) hodnoty pixelu a hodnot pixelů sousedních. Váhy bývají reprezentovány tzv. konvoluční maticí. Například matice

$$B = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

je konvoluční maticí jednoduchého rozostření. Aplikuje se následujícím způsobem:

$$c'_{i,j} = \frac{1}{S} \sum_{k,l \in \{-1,0,+1\}} B_{k+1,l+1} c_{i+k,j+l}$$

kde  $S$  je součet hodnot v matici  $B$ , tj. 16.

Další ukázkou grafického filtru pracujícího s využitím buněčného fuzzy automatu je například filtr pro zvýraznění tvarů. Je daný následujícím předpisem

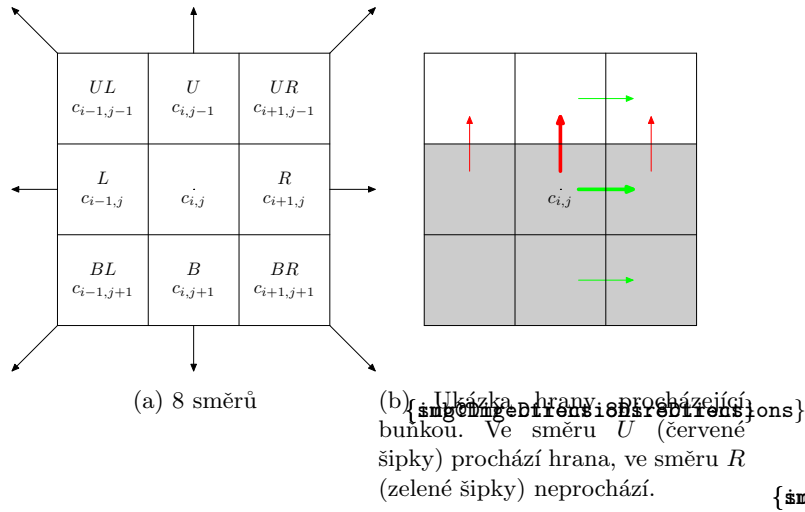
$$c'_{i,j} = \max(0, \min(1, \begin{cases} \epsilon(c_{i,j} + 1) - 1 & \text{pokud } c_{i,j} > \text{neighs}_{i,j} \\ \epsilon c_{i,j} & \text{pokud } c_{i,j} < \text{neighs}_{i,j} \\ c_{i,j} & \text{pokud } c_{i,j} = \text{neighs}_{i,j} \end{cases}))$$

kde  $\epsilon > 1$  je parametr udávající agresivitu zvýrazňování a  $\text{neighs}_{i,j}$  je součet hodnot okolních buněk (viz příklad 7.1).

Ukázky aplikací obou filtrů jsou k nalezení na obrázku 10. V následujících podkapitolách budou prezentovány některé další (pokročilejší) techniky zpracování obrazu využívající buněčné fuzzy automaty.

## 7.6 Hledání hran

Hledání hran je jednou ze základních technik zpracování obrazu. Hledání hran je často klíčové pro rozpoznávání vzorů v obrazech. V dnešní době existuje značné



množství technik pro rozpoznávání hran [?]. V [?] je popsán poměrně elegantní způsob, jak hledání hran vyřešit pomocí buněčných fuzzy automatů.

Označme osmici směrů dle obrázku 11a. Množinu těchto směrů nazvěme  $dim$ . Dále označme  $c_X$  (kde  $X \in dim$ ) jako sousední buňku buňky  $c$  ve směru  $X$ .

Autoři vycházejí z následující úvahy: Prochází-li buňkou  $c$  hrana ve směru  $X \in dim$ , pak má buňka  $c_X$  výrazně jinou barvu, než buňka  $c$  (viz obrázek 11b). Prochází-li buňkou  $c$  hrana v alespoň jednom směru  $X \in dim$ , pak můžeme říci, že buňka obsahuje hranu.

Nadefinujeme fuzzy relaci „buňky  $c$  a  $c'$  mají zcela rozdílnou barvu“ předpisem:

$$\Delta(c, c') = 1 - |c - c'|$$

1310 Označme  $\Delta'$  jako doplněk k  $\Delta$ , tedy „buňky  $c$  a  $c'$  mají zcela shodnou barvu“. Pak můžeme stanovit fuzzy množiny  $\epsilon_X$  (pro všechny  $X \in dim$ ) ve smyslu „buňkou  $c$  prochází hrana ve směru  $X$ “. Pro  $X = U$  by pravidla vypadala následovně:

- Pokud  $\Delta'(c_L, c_{UL})$ ,  $\Delta'(c, c_U)$  a  $\Delta'(c_R, c_{UR})$  pak  $\epsilon_U(c) = 0$
- Pokud  $\Delta(c_L, c_{UL})$ ,  $\Delta(c, c_U)$  a  $\Delta(c_R, c_{UR})$  pak  $\epsilon_U(c) = 1$

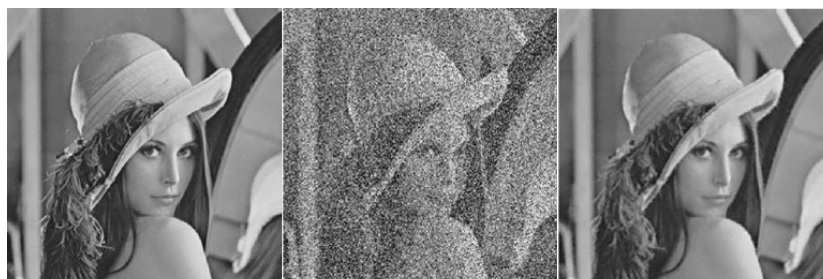
Obdobným způsobem by se dodefinovaly zbývající fuzzy množiny  $\epsilon_X$ . Následně lze nadefinovat fuzzy množinu  $\epsilon$  ve smyslu „buňkou  $c$  prochází hrana“ pomocí pravidel:

- Pokud  $c = \epsilon_U$  pak  $\epsilon = 1$
- ...
- Pokud  $c = \epsilon_{UR}$  pak  $\epsilon = 1$
- Jinak  $\epsilon = 0$

1320

Pomocí těchto pravidel tak lze sestavit buněčný fuzzy automat s fuzzy logikou, který rozpoznává hrany.

Dle [?] může být hodnota stupně „buňkou  $c$  prochází hrana“ použita jako parametr  $\alpha$  tzv. Gas Diffusion Modelu, jednu z technik zaostřování obrazu.



(a) Vstupní obraz (b) Zashumněný obraz (c) Obraz s odstraněným šumem

Obrázek 12: (převzato z [?]) Ukázka odstraňování šumu

{img:Noises}

## 7.7 Ostraňování šumu

Odstraňování šumu je další častý problém, který je třeba při zpracování obrazů řešit. Pro studium technik odstraňování šumu se používá zashumnění tzv. impulzním šumem popř. šumem „sůl a pepř“. Zashumnění impulzním šumem nahradí stanovený počet pixelů náhodnými barvami. Šumění „sůl a pepř“ pak narazuje pixely buď bílou (1) nebo černou (0) barvou.

V [?] je prezentována jednoduchá avšak efektivní technika, která kombinuje klasický bivalentní buněčný automat s buněčným fuzzy automatem.

V první fázi je klasickým buněčným automatem šum detekován. Buňka obsahuje šum, pokud rozdíl její barvy od průměrné barvy jejích sousedů překračuje stanovenou mez. Tato mez může být stanovena statisticky, například na základě směrodatné odchylky barev pixelů celého obrazu. V druhé fázi je aplikován buněčný fuzzy automat, který buňky obsahující šum nahradí hodnotami spočtenými z jejich okolí.

Jak autoři poukazují, tato technika je na odstraňování šumu velmi efektivní. Ukázky výsledků jsou na obrázku 12.

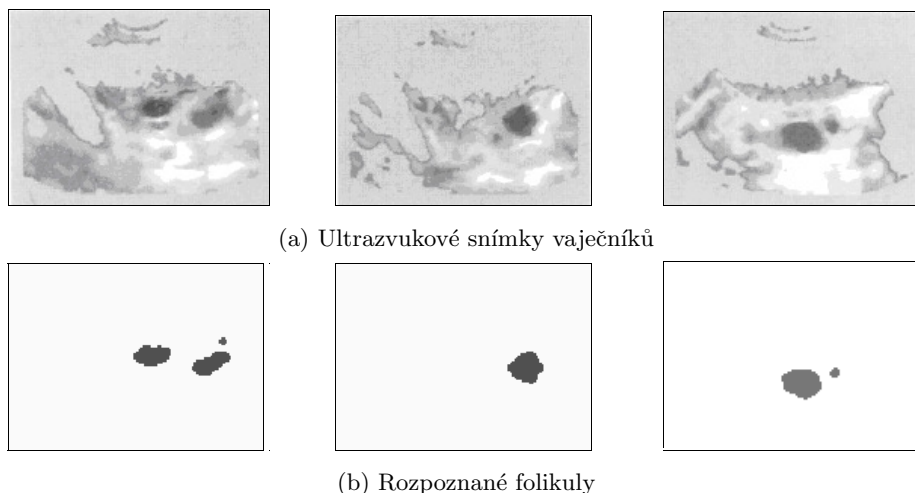
Velmi podobný způsob odstranění šumu je popsán v [?]. Zde však operace detekce šumu a jeho odstranění provádějí v jednom kroku.

## 7.8 Rozpoznávání jednoduchých vzorů

Rozpoznávání vzorů je další z častých způsobů práce s obrazy. Obecně je problém definován (obdobně, jako rozpoznávání textových vzorů v kapitole (zde bude doplněno: ref na kapitolu) ) jako problém určení, zda-li obraz obsahuje předem stanovený vzor či ne. Obvykle nás také zajímá, kde přesně se vzor v obraze vyskytuje.

Rozpoznávání vzorů v obrazech je však značně komplikované i pro buněčné fuzzy automaty. V [?] je popsán způsob, který popisuje rozpoznávání vzorů v obrazu velikosti  $1 \times m$  pomocí (jednodimenzionálního) buněčného automatu. Pomocí fuzzy množin reprezentující různé stupně šedi jsou sestavena pravidla, která popisují jak vzorový obraz, tak obrazy jemu podobné. Množina přechodových pravidel tak vyjmenovává téměř všechny možné kombinace hodnot fuzzy množin pro všechny buňky v okolí. Velikost přechodové funkce je tak exponenciální vzhledem k velikosti okolí buňky. Vzhledem k tomu, že okolí buňky je vlastně





Obrázek 13: (převzato z [?], upraveno) Ukázky rozpoznávání folikul

1360 předpisem pro vzor, je tato technika nepoužitelná pro vzory větší než jednotky pixelů.

Zcela jiný přístup používají v [?]. Vzor nepovažují jako konkrétní kombinaci odstínů barev, ale jako část obrazu splňující určité vlastnosti.

Autoři metodu doporučují na vyhledávání vzorů v lékařských snímcích (např. ultrazvuk, Röntgen). Techniku demonstrují na ultrazvukových snímcích vaječníků. Poukazují na to, že folikula<sup>9</sup> je na snímku tvořena jednodlitou svrnou stejné barvy, obklopena ostatní tkání (barevné přechody). Ukázka několika snímků je k dispozici na obrázku 13a. Obecně tak lze hovořit o „popředí“ (folikula) vystupující z „pozadí“.

1370 K nalezení popředí používají dvojici buněčných fuzzy automatů. První automat určuje hodnotu „buňka je kandidát na buňku folikuly“. Druhý automat pak buňky, které nebyly označeny jako kandidáti, ostraňuje (nastavuje na 0).

Přechodová funkce prvního automatu je poměrně složitá, pracuje s 5 stupni šedi a třemi stupni kandidatury, takže zde nebude rozebírána. Druhý automat pak funguje elementárně. Buňky, jež nejsou označeny jako dostatečné kandidáty na folikuly, jsou odstraněny, ostatní ponechány.

Na obrázku 13 jsou k nahlédnutí ukázky rozpoznávaných folikul pomocí této techniky.

## 8 Strojové učení

1380 Strojové učení je technika pro přibližné řešení komplikovaných či složitých problémů. Spočívá v tom, že namísto přímého hledání algoritmu, který požadovaný problém řeší, navrhujeme algoritmus, jehož vstupem je popis problému (ve vhodném formátu) a výstupem je algoritmus pro řešení problému. Pro popis problému se typicky používá deklarativní přístup, tj. „co má být při určitém vstupu výsledkem“ namísto „jak ze vstupu spočítat výsledek“.

<sup>9</sup>Folikula je dutinka ve vaječníku, v níž probíhá zrání vajíčka. (zde bude doplněno: citovat: <http://lekarske.slovniky.cz/pojem/folikul>)

Je třeba mít na paměti, že správnost „vygenerovaného algoritmu“ závisí na správnosti popisu řešeného problému. Vzhledem k tomu, že strojové učení se používá pro komplikované problémy, popis problému vkládaný do patřičné techniky strojového učení bude zřejmě zjednodušený. Tím vzniká prostor pro nepřesnost řešení.

1390 Na druhou stranu, s použitím fuzzy či pravděpodobnostního přístupu můžeme tento fakt vzít v potaz. Vygenerovaný algoritmus tak sice nebude fungovat správně, nicméně jeho výsledkem bude moci být například „ $y$  je správným řešením s pravděpodobností  $p$ “ či „ $z$  je správným řešením ve stupni  $d$ “. Propojení strojového učení s fuzzy nebo pravděpodobnostního přístupu by tak mělo být přínosem. Navíc, jak bude ukázáno, může být i přirozené.

Mezi nejznámější techniky strojového učení patří například neuronové sítě, rozhodovací stromy či shlukování. Následuje popis neuronových sítí. Ty mají totiž nejlepší předpoklady pro provázání s fuzzy automaty a následné využití v praxi.

## 1400 8.1 Neuronové sítě

Umělé neuronové sítě (dále jen „neuronové sítě“) jsou jednou z nejzákladnějších technik strojového učení. Neuronové sítě jsou inspirovány zpracováním informací pomocí nervových buněk známých z biologie. Vygenerovaný algoritmus je tvořen sítí tzv. neuronů, které jsou propojeny v zpravidla rozsáhlou síť. Tato síť obsahuje spoustu parametrů, které jsou nastaveny tak, aby síť tvořila algoritmus řešící požadovaný problém.

Následuje formální zavedení uvedených pojmů. Popis neuronových sítí je převzat z [?]. Označme  $X$  jako tzv. signální množinu, tj. množinu všech možných hodnot, kterých mohou nabýt signály přenesené prostřednictvím spojení. Dále 1410 označme  $S$  jako množinu stavů neuronů.

V tomto textu budeme jako signální množinu  $X$  pokládat reálný interval  $(0, 1)$  (v souladu s fuzzy teorií), jako množinu  $S$  stavů pak množinu všech reálných čísel. Dále budeme uvažovat pouze neurony diskrétní (v čase). Říkáme, že diskrétní neuron  $s$  nabývá v čase  $t \in \mathbb{N}$  stavu  $s(t)$ . Dále označme  $y(t)$  jako výstup neuronu  $s$  v čase  $t$ .

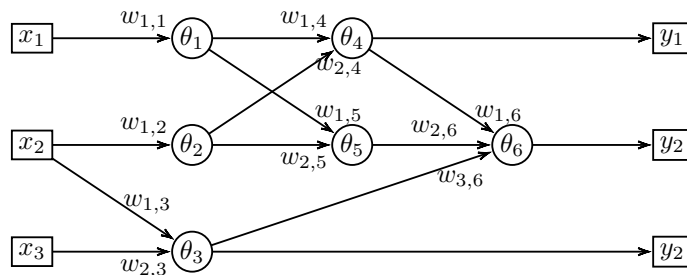
**Definice 8.1** (Diskrétní neuron). *Jako diskrétní neuron označujeme dvojici funkcí  $f : X^n \times S \rightarrow S$  (přechodová funkce) a  $g : S \rightarrow X$  (výstupní funkce) tak, že:*

$$\begin{aligned} s(t) &= f(x_1(t-1), \dots, x_n(t-1), s(t-1)) \\ y(t) &= g(s(t)) \end{aligned}$$

**Poznámka 8.1.**  $x_1, \dots, x_n$  v předchozí definici značí signální spojení, a to buď ze vstupů sítě k neuronům, nebo mezi jednotlivými neurony.

**Příklad 8.1.** Pro jeden z nejznámějších, tzv. McCulloch-Pittsovův neuron, vypa-  
dají funkce  $f_j$  a  $g_j$  neuronu  $s_j$  následovně:

$$\begin{aligned} f_j(x_{1,j}(t), \dots, x_{n,j}(t), s_j) &= \sum_{i=1}^n w_{i,j} x_{i,j}(t) \\ g_j(x) &= \begin{cases} 0 & \text{pokud } x < \theta_j \\ 1 & \text{pokud } x \geq \theta_j \end{cases} \end{aligned}$$



Obrázek 14: Ukázka neuronové sítě

{img:SimplNeuNet}

kde  $w_{1,j}, \dots, w_{n,j}$  jsou tzv. *synaptické váhy* neuronu  $s_j$  a  $\theta_j$  je jeho tzv. *prahová hodnota*.

**Definice 8.2** (Neuronová síť). Jako neuronovou síť označme strukturu  $(X, S, N, I, O, o)$ , kde  $X$  (signální množina) a  $S$  (množina stavů) byly popsány výše,  $N$  je množina neuronů,  $I \subseteq X^m$  je množina vstupních spojení a  $O \subseteq X^n$  množina výstupních spojení a  $o : N \rightarrow O$  zobrazení propojující neurony s výstupními spojeními.

**Definice 8.3** (Rekurentní a přímočará neuronová síť). Obsahuje-li neuronová síť cyklus, tj. existuje v ní alespoň jeden neuron s splňující

$$s(t) = f(s_1(t-1), \dots, s_2(t-1+k), \dots, s_n(t-1), s(t-1))$$

kde  $k \geq 1$ , pak se síť nazývá *rekurentní*. V opačném případě se nazývá *přímocará*.

**Příklad 8.2.** Mějme  $I = \{x_1, x_2, x_3\}$ ,  $I = \{y_1, y_2, y_3\}$  a  $o(s_4) = y_1$ ,  $o(s_6) = y_2$ ,  $o(s_3) = y_3$ . Pak pro množinu  $N = \{s_1, s_2, s_3, s_4, s_5, s_6\}$  McCulloch-Pittsových neuronů znázorněných na obrázku 14 tvoří  $(X, S, N, I, O, o)$  přímočarou neuronovou síť.

## 8.2 Konstrukce neuronové sítě

Existují dva základní způsoby, jak zkonstruovat neuronovou síť. Prvním je pomocí znalostní báze a druhý pomocí trénovacích dat. Jak znalostní báze, tak trénovací data nám zde plní účel popisu problému, který má neuronová síť řešit. V praxi se často používá kombinace obou přístupů.

Uvažujme, že máme Množinu  $P$  If-Then pravidel „Jestliže  $x_{j,1}, \dots, x_{j,n}$  pak  $y_j$ “. Pak pro tuto množinu můžeme vytvořit neuronovou síť tak, že každému pravidlu  $j$  vytvoříme následující neuron: (zde bude doplněno: doplnit pravidlo pro disjunkce, tj. když je více pravidel s „pak  $y$ “)

$$y_j(t) = s_j(t) = x_{j,1}(t-1) \wedge \dots \wedge x_{j,n}(t-1)$$

Takto vytvořená neuronová síť bude schopna pracovat jak s klasickými If-Then pravidly, tak i s fuzzy If-Then pravidly.

Druhým způsobem, jak zkonstruovat neuronovou síť, je pomocí trénovacích dat. Tato technika se dále dělí na učení s učitelem a učení bez učitele. V tomto textu se spokojíme s popisem učení s učitelem.

Hovoříme-li o trénovacích datech v souvislosti s učením s učitelem, pak jako trénovací data považujeme konečné zobrazení, které vstupům problému

přiřazuje správné nebo očekávané výsledky. Na počátku se sestaví neuronová síť s náhodnými parametry a tzv. učení se její parametry postupně modifikují tak, aby byla schopna správně přiřazovat vstupům z trénovací množiny odpovídající výstupy.

(zde bude doplněno: *backpropagation algorithm* (běžný algoritmus učení)?)

V praxi je možné oba přístupy kombinovat. Typicky se tedy sestaví neuronová síť pomocí znalostní báze, která je následně pomocí trénovacích dat přizpůsobena skutečným datům. Navíc, díky znalostní bázi neuronová síť je hned má hned v prvních fázích učení určitý přehled o řešeném problému a její učení tak může probíhat značně efektivněji. (zde bude doplněno: *fakt?*)

### 8.3 Fuzzy automaty a neuronovky

Bylo zjištěno, že fuzzy automat může být konvertován do neuronové sítě. Této problematice se věnují např. v [16], [3] [1], [?], [?], [?], [?].

Ve zjednodušené podobě se bude jednat o neuronovou síť, která bude realizovat jen přechodovou funkci, tj. přechod mezi stavy na základě vstupu. Vstupem neuronové sítě bude zakódování fuzzy stavu, v jakém se automat nachází a symbol na vstupu automatu. Výsledkem výpočtu pak bude informace, do jakého/jakých stavů by měl automat přejít, tj. opět fuzzy stav.

Na základě neuronové sítě realizující přechodovou funkci pak může být sestavena neuronová síť, která si sama pamatuje fuzzy stav, v jakém se výpočet nachází. Vzhledem k tomu, že taková neuronová síť bude muset být rekurentní, bude její struktura složitější a nad rámec této práce. Vstupem takové neuronové sítě pak bude v jednotlivých časových kvantech vstupní symboly a na výstupu číslo z intervalu  $(0, 1)$  reprezentující stupeň, v jakém je zadaný řetězec automatem přijímán.

Sestavíme-li neuronovou síť pomocí přechodové funkce  $\mu$  nám známého automatu  $A$  budeme generovat dvojice  $((q, d), a), (q', d')) \in \mu^*$  (zde bude doplněno: *je to tak?*) tj. „je-li automat ve stavu  $q$  ve stupni  $d$  a na vstupu je symbol  $a$ , pak přejde do stavu  $q'$  ve stupni  $d'$ “. Například v [1] autoři demonstrují učení na automatu o dvou stavech a nad dvojprvkovou abecedou. Již po 5 trénovacích dvojicích se síť naučila rozpoznávat s 95% přesností a po 432 pokusech rozpoznávala se 100% přesností (při toleranci stupně pravdivosti  $1.25 \cdot 10^{-7}$ ).

Lze však použít i konstrukce s pomocí znalostní báze. V takovém případě je třeba zformulovat If-Then pravidla síť popisující. Pro každý přechod

$$\delta((q, d), a) = (q', d')$$

tedy sestavíme If-Then pravidlo

Je-li automat ve stavu  $q$  ve stupni  $d$  a na vstupu je symbol  $a$ , pak automat přejde do stavu  $q'$  ve stupni  $d'$

Na základě této znalostní báze může být navržena neuronová síť implementující přechodovou funkci automatu.

Opačný vztah, tj. jak z neuronové sítě zpět získat fuzzy automat je popsán např. v [?], [?].

Poněkud elegantní způsob, jak z neuronové sítě vyextrahovat automat je popsán v [?]. Autoři uvažují neuronovou síť jako nástroj, který rozpoznává (v odstupňované pravdivosti)  $(n\text{-dimenzionální})$  podprostory prostoru  $(0, 1)^n$ . Každou souřadnici v tomto prostoru pak rozkládají symboly abecedy automatu,

tj. každému vektoru v tomto prostoru přiřazují řetězec nad abecedou automatu. Vzniklá struktura, kterou je vlastně kd-tree, pak tvoří automat.<sup>10</sup> *(zde bude doplněno: Zapsat to nějak aspoň trochu lidsky, je např. potřeba doplnit práci s cykly)* ?

## 8.4 Učí se fuzzy automaty

V předchozích kapitolách byly prezentovány techniky, jak konvertovat fuzzy automat na neuronovou síť a zpět. Konverzí automatu na neuronovou síť obdržíme nástroj, který nám umožňuje učení, tj. adaptaci modelu na jiná data. Zpětnou extrakcí automatu tak můžeme obdržet automat, který rozpoznává jiný jazyk, a to jazyk který odpovídá trénovacím datům.

Tento postup lze však s určitou újmou na obecnosti zjednodušit. Namísto konverze automatu na neuronovou síť, její modifikaci a následnou konverzi zpět, je možné automat modifikovat napřímo. Této technice se říká učící se automat. Této technice se věnují např. v [?], [?], [?], [?]. V [5], [?] a [?] je použit buněčný fuzzy učící se automat. V [7] používají pro učení pokročilejší techniku založenou na genetických algoritmech. Obdobně, v [?] učním generují kompletně nový automat (tedy přeučením náhodného).

Základní myšlenka techniky učících se automatů je obvykle, že jednotlivé stupně pravdivosti, které figurují v instanci automatu, mohou být pozměněny. V rozšířené podobě je pak možné automat rozšiřovat o kompletně nové stavy a přechody.

## 8.5 Metoda lisování dat

*(zde bude doplněno: zkusit nějaký známý dataset, něco z toho, co jsme dělali v KMI/ZZD)* Autor práce prezentuje novou, jednoduchou, techniku strojového učení. Výhodou této techniky je, že je založena čistě na teorii fuzzy automatů, není třeba žádné další techniky strojového učení.

Vstupem této techniky je (konečná a neprázdná) množina trénovacích dat ve formě jazyka  $L$  nad známou abecedou  $\Sigma$  a koeficient přesnosti  $\sigma$ . Výstupem je poté fuzzy automat  $A_{L,\sigma}$ , který jazyk  $L$  přijímá tak, že pro všechna  $w \in L$  platí:

$$A_{L,\sigma}(w) \geq \sigma$$

tj. že řetězec  $w$  je automatem přijímán alespoň ve stupni  $\sigma$ . U řetězců, které do jazyka  $L$ <sup>11</sup> nepatří (tj.  $w' \in \Sigma^* \setminus L$ ) by posléze měl automat být schopen určit jak moc se jazyku  $L$  podobají, tj. určit stupeň  $A_{L,\sigma}(w')$ .

Postup konstrukce automatu  $A_{L,\sigma}$  je následující:

1. Máme jazyk  $L$ , který je konečný. Je tedy zřejmě i regulérní.
2. K jazyku  $L$  sestavíme nedeterministický konečný automat  $A_L$  rozpoznávající  $L$ . Takový automat bude v nejhorším případě obsahovat  $\sum_{w \in L} |w|$  stavů.
3. K automatu  $A_L$  vytvoříme nedeterministický fuzzy automat  $A'_L$ .

<sup>10</sup>Tímto způsobem vygenerovaný automat je automat dle definice *(zde bude doplněno: doplnit ten, co má ostré přechody, ale fuzzy koncové stavy)*

<sup>11</sup>Jazyk  $L$  může být klidně fuzzy jazyk, následný postup se pak jen patřičně upraví

4. Fuzzy minimalizací automatu  $A'_L$  se stupněm  $\sigma$  získáme hledaný automat  $A_{L,\sigma}$ .

Klíčovým bodem této techniky je minimalizace automatu. Technika předpokládá, že jazyk trénovacích dat v sobě obsahuje jeden nebo více jednoduchých regulérních jazyků (ať už jako podmnožiny nebo jako podřetězce vybraných řetězců). Části automatu reprezentující tyto „podjazyky“ budou minimalizací zmenšeny. Ve výsledku se tak dá očekávat značné zmenšení velikosti automatu.

Navíc, vygenerovaný automat poměrně přehledně popisuje strukturu ve vstupním jazyce. Technika tak může být použita pro odtržení šumu a nepřesností v jazyce  $L$ .

Technika lisování dat bude naimplementována a experimentálně ověřena. (zde bude doplněno: naprogramovat, ověřit) Jednou z možností, kde by mohla tato technika být uplatněna, by bylo určování síly přihlašovacího hesla. Vstupem by byla databáze uživatelských hesel s informací od experta, která hesla jsou silná a která slabá. Ze silných hesel by poté byl sestaven jazyk, na který by bylo aplikováno lisování dat. Výsledkem by byl automat, který rozpoznává silná hesla. Tato aplikace však nebyla naimplementována, vzhledem k tomu, že je z bezpečnostních důvodů nemožné získat databázi uživatelských hesel.

Další z možných aplikací je určování dělitelnosti čísel. Uvažujme abecedu symbolů číslic, tj.  $\Sigma = \{0, \dots, 9\}$  a jazyk  $L \subseteq S^*$  čísel dělitelných čtyřmi. Při velikosti jazyka  $x$  a koeficientu  $\sigma = y$  se podařilo automatu dosáhnout z úspěšného rozpoznávání čísel dělitelných 4 ve stupni vyšším, než  $w\%$ . (zde bude doplněno: naprogramovat, ověřit)

## 8.6 Reálné příklady použití

V [?] používají učící se fuzzy automaty v mírně pozměněné formě k řízení výkonu při obloukovém sváření. Abeceda v tomto případě obsahuje dva symboly, „zvýšení výkonu“ a „snížení výkonu“. Automat byl sestaven tak, aby správně rozpoznával vhodné posloupnosti zvýšení a snížení výkonu za účelem zvýšení kvality sváru. V téže publikaci demonstrují obdobným způsobem uplatnění pro řízení robota pohybujícího se v neznámém prostředí. Pomocí dvojice symbolů pro zvýšení výkonu motorů a snížení výkonu motorů si kladou za cíl navržení systému pro řízení robota. Tento systém má za cíl řídit přidávání a ubírání výkonu tak, aby se robot, bez ohledu na terén, pohyboval konstantní rychlostí.

V [?] je fuzzy automat použit pro návrh řídicích systému. Řídicí systém je stavový stroj, který je překlápěn mezi diskrétními stavy pomocí událostí. Množina možných událostí tak tvoří abecedu a řídicí systém může být reprezentován automatem. Použitím nepřesností je tak možno získat fuzzy automat. (zde bude doplněno: osamostatnit definici řídicího systému? Např. do úvodu, kde budu zmiňovat, co všechno se podobá FA?) S pomocí znalosti očekávaného chování systému je pak možné pomocí učícího automatu sestavit adekvátní řídicí systém.

V [?] jsou učící automaty použity jako agenti v hře s nulovým součtem, která nemá jasné ekvilibrium. Automaty se postupným hraním tahů učí a vylepšují svoji strategii tak, aby bylo dosaženo ekvilibria.

V [?] navrhuje používat učení fuzzy automatů pro konstrukci logických klopných obvodů. Tj. obvodů, které se na základě vstupů překlápí mezi různými diskrétními stavy. Sami autoři uvádějí, že bylo výhodné pracovat s učícím fuzzy automatem a teprve poté jej „diskretizovat“, tj. konvertovat na kýžený bivalentní.

V [7] a [?] používají učící automat na vylepšení rozpoznávání textu v obrazu. Jako trénovací data jsou použity známé (již dříve rozpoznané) texty a trénování zde plní účel přizpůsobení se reálným vytištěným textům. V [5] a [?] používají učící buněčný fuzzy automat pro zpřesnění detekce hran v obrazech. *(zde bude doplněno: Odkázat na patřičné kapitoly v textu)*

1570 Autor sám přichází s myšlenkou, že kombinace fuzzy automatů a strojového učení má poměrně značný potenciál pro uplatnění v praxi. Kombinace fuzzy automatů, které poměrně jednoduše, avšak poměrně restriktivně popisují určitý jazyk mohou být pomocí strojového učení snadno upraveny na mocnější nástroje. Důvod, proč se však nepoužívají vidí v současném boomu neuronových sítí jako takových, které utlačují všechny ostatní techniky strojového učení.

## 9 Biologie a medicína

Biologie a medicína jsou odvětví, které zpravidla disponují velikým množstvím dat, které je třeba zpracovat. Typicky v datech získaných nějakým sledováním najít určité vzory. Fuzzy automaty mohou být v této oblasti nápomocny.

Vzhledem k tomu, že většina těchto aplikací vyžaduje zásah experta na danou problematiku, budou tyto aplikace rozbrány pouze teoreticky.

### 9.1 Rozpoznávání řetězců DNA

1580 Mělo by to jít, něco málo na to mám. [?]

### 9.2 Simulace růstu mořských řas

[?]. *(zde bude doplněno: odkázat se na urban growth problem)* .

### 9.3 Rozpoznávání vzorů v lékařských snímcích

Viz buněčné automaty a jejich jednoduché rozpoznávání vzorů.

A pak ještě toto: [?].

### 9.4 Analýza zdravotního stavu pacienta

V [?], [?], [?] a [?] je popisován způsob, jak pomocí fuzzy automatů sledovat zdravotní stav pacienta.

1590 Technika pracuje s automatem s fuzzy If-Then pravidly. Stavy automatu v tomto případě reprezentují choroby (popř. stavy jedné choroby), přechodová funkce pak přechody mezi nimi. Přechodová pravidla, navržená expertem, pak popisují přechody při různých událostech či akcích (např. medikace).

Důležitým předpokladem, který automat musí splňovat, je tzv. vlastnost pozdržení vrcholu *(zde bude doplněno: nadefinovat někde v úvodech?)* . Tato vlastnost říká, že výpočet automatu pro libovolný vstup nikdy nekončí v prázdném fuzzy stavu, tj. výpočet se vždy bude muset nacházet v nenulovém stupni v alespoň některém stavu.

*(zde bude doplněno: Příklad?)*

1600 Fuzzy stav automatu v [?] nazývají „fuzzy chorobný syndrom“. Fuzzy chorobný syndrom je vlastně popis zdravotního stavu pacienta v určitý okamžik.

Tato technika tak může sloužit ke porovnávání simulovaného a skutečného stavu (např. po medikaci, zákroku) pacienta a případně včas reagovat na odchylku. Výhodou této techniky je, že značně zjednodušuje sledování více diagnóz současně.

V [?] tuto techniku používají pro sledování systolického krevního tlaku a množství krevního cukru. V [?] používají podobnou techniku pro sledování těla při sportu (konkrétně tělní teplotu, dehydrataci a srdeční tep).

V [?] se fuzzy automaty používají pro diagnózu srdečních chorob. Po spuštění automat na základě pohlaví a věkové skupiny přejde do odpovídajícího podautomatu. Ten si poté sám žádá měření různých parametrů (aktuální tepová frekvence, aktuální variabilita srdeční frekvence) v závislosti na tom, kdy je který pro proces diagnózy aktuálně významný. V případě stanovení rizika je riziko ohlášeno a automat se vrací do počátečního stavu a proces se spouští znovu (aby diagnózu ověřil).

*(zde bude doplněno: nadefinovat u různých typů automatů Fuzzy automat (klasický nedeterministický) s If-Then pravidly) (zde bude doplněno: On je vlastně i rozdíl mezi fuzzy automatem s If-Then pravidly a bivaletním automatem s fuzzy If-Then pravidly)*

*(zde bude doplněno: terminologie u proměnných (resp. indexů proměnných), česky (triangle, age, distance) vs. česky?)*

## 1620 9.5 Detekce zhoubných nádorů

V [?] popř. [?] využívají učící se fuzzy automat pro rozpoznávání zhoubných (maligních) nádorových buněk. Vstupem této techniky je mikroskopický snímek z buněk z prsní tkáně, výstupem pak nalezené maligní buňky. Využívají faktu, že nezhoubné buňky mají na snímcích obvykle symetričtější tvary a méně „skvrn“.

Proces rozpoznávání probíhá následujícím způsobem:

- Snímek je převeden do odstínů šedi.
- Jednotlivé buňky na snímku jsou izolovány do samostatných obrazů. Následně jsou zpracovávány všechny obrazy postupně.
- Na obraze jsou rozpoznány plochy stejné nebo podobné barvy.
- 1630 • Z ploch je na základě relace „plocha  $X$  obsahuje plochu  $Y$ “ zkonstruován strom ploch.
- Strom je následně zakódován do řetězce a rozpoznán fuzzy automatem.

Stromy jsou do řetězců kódovány jako posloupnost čísel, kde každé číslo odpovídá počtu potomků jednotlivých uzlů stromu. Automat byl sestaven učením z veřejné databáze snímků. Autoři poukazují, že tato technika je na rozpoznávání účinnější, než vybrané z dalších technik.

*(zde bude doplněno: ukázka?)*

## 9.6 Analýza kardiogramu

Např. [?], popř. i [?]. Ale je to obecně zpracování signálu, popsat někde jinde?



## 9.7 Řízení protézy

V [?] (popř. i [?] *(zde bude doplněno: to by šlo ale obecně použít pro zpracování signálu, ne?)*) používají fuzzy automaty pro modelování fungování protézy dolní končetiny. Pomocí akcelerometrů sledují pohyb patřičné náhrady a pomocí známých vzorů spouští automat, který určuje, v jaké fázi pohybu protéza je.

## 10 Další aplikace

### 10.1 Pattern matching, další příklady

Kódování výstupu z nějakého systému do řetězce a jeho pattern matching [17].

Toto [10] vypadá taky zajímavě, nějak pattern matchinguje nějaký signál či co.

1650 V [?] nejspíš používají dvouúrovňový pattern matching pro vyhledávání vzoru v dlouhém textu. Nejdřív sestaví jednoduchý automat pro přibližné rozpoznání něčeho, co se vzoru podobá (ale je to jednodušší?) např. na každé stránce. Ty stránky, které prošly v nenulovém stupni pak prohledá pořádně a určí konkrétní výskyty. Asi.

### 10.2 Detekce požárů

V [?] a [?] používají fuzzy automaty pro pattern matching ve videosekvenci, konkrétně pro detekci požárů.

1660 V první fázi se snímek rozdělí na několik regionů a na základě barvy (teplé světlé barvy) se určí, zda-li jednotlivé regiony mohou být plamenem (tzv. kandidáti). Pro každého kandidáta je pak sestaven fuzzy automat o čtyřech stavech  $q_{VL}, q_L, q_H, q_{VH}$ . Pokud se automat regionu nachází ve stavu  $q_{VL}$ , pak platí, že „region je velmi málo pravděpodobné, že je tvořen plamenem“. Obdobně pro  $q_L$  („málo pravděpodobné“),  $q_H$  („hodně pravděpodobné“) a  $q_{VH}$  („velmi pravděpodobné“). Abecedou událostí jsou pak kombinace dalších atributů (světlo, pohyb v určitém směru, vlnění) spočtených z předchozích snímků. Přejítové pravidlo pak může být např. „pokud jsi ve stavu  $q_H$  a došlo k velkému posunu směrem nahoru a malému snížení svitu, pak přejdi do stavu  $q_{VH}$ “. Přejítovou funkci pak navrhli statistickým pozorováním známých videosekvencí s požáry<sup>12</sup>.

1670 Autoři tuto techniku experimentálně ověřili a ukázalo se, že požár detekuje správně ve vyšším množství případů, než některé vybrané další metody. Stejně tak, oproti jiným metodám, technika mnohem méně rozpoznávala požár tam, kde nebyl. (zde bude doplněno: vložít obrázek s výsledkem?)

(zde bude doplněno: zavést pojem *lingvistická veličina* (aka takový to fuzzy výrok/fuzzy množina)? L. A. Zadeh, „The concept of linguistic variable and its application to approximate reasoning“, *Information sciences*, 1975.)

### 10.3 Automobilismus

1680 Automat (pochopitelně event-driven) pro monitoring manévrů řidiče (předjíždění, zatáčení doleva v křižovatce, pouštění chodce na přechodu, a podob.) [?]. Pomocí fuzzy stavových proměnných (rychlost, zrychlení, natočení volantu, blinkry ano/ne) se stanovují pravidla, např. odbočení v křižovatce: blinkr levý – lehce doleva (volitelně) – zpomalení (volitelně) – velké otočení volantem doleva – vypnutí blinkru.

V [?] např. sledují řízení auta pomocí fuzzy automatů. Popisují techniku, kdy na vstupu je proud dat z čidel, ty se fuzzyfikují („teplota je nízká“, „prudce brzdit“, „mírně zatáčí doleva“), patřičný automat pak rozpoznává známé vzory. Tedy podobné jako [?]. Akorát tady namísto otáčení volantu a sešlápnutí pedálu měřili pomocí trojice akcelerometrů (pro tři osy,  $x, y$  a  $z$  intenzity akcelerace v

<sup>12</sup>Dá se říci, že automat vznikl pomocí učení s učitelem

1690 patří k ose, a to jestli je pozitivní, negativní nebo neutrální). Používají automat s výstupem, který tak může vybrané přechody hlásit („ted’ se jede rovně“, „právě byla ostrá zatáčka doleva“). Získá se tak docela příjemný popis trasy.

V [?] také popisují řídicí systém, avšak zavádějí distribuovaný model.

## 10.4 Sledování vytížení el. sítě

V [?] používají fuzzy automaty pro sledování vytížení el. rozvodné sítě. Technika předpokládá znalost spotřebičů v síti a nástroj pro monitorování odběru el. proudu. Výstupem je pak rozpis, kdy byl jaký spotřebič zapnut a kolik odebíral energie.

1700 Technika využívá faktu, že každý spotřebič se může nacházet v různých stavech. Například vysoušeč vlasů se může nacházet v jednom z těchto tří stavů: vypnut, fouká studený vzduch, fouká teplý vzduch. Každému takovému stavu může být asociována konkrétní hodnota odběru el. energie. Pokud známe výkyvy spotřeby proudu při změnách těchto stavů, můžeme spotřebiči sestavit fuzzy automat. Událostmi, které automat řídí mohou být např. „spotřeba velmi výrazně klesá“, „spotřeba je konstantní“ či „spotřeba lehce stoupá“.

1710 Nyní uvažujme elektrickou síť s více spotřebiči. Předpokládejme, že v každou časovou jednotku dojde vždy pouze k jedné změně stavu jednoho spotřebiče. Pak nastane-li v síti událost  $x$  (např. pokles spotřeby) (*zde bude doplněno: fuzzy událost? událost nastane ve stupni  $d$ ? ještě jinak?*), pak na základě znalosti stavů všech spotřebičů můžeme určit nejpravděpodobnější přechod (tj. jaký spotřebič přešel do jakého stavu), který nastal. Sledováním sítě tak můžeme neinvazivně sledovat, které spotřebiče, kdy a jak zatěžují síť.

## 10.5 Modelování emocí postav v počítačových hrách

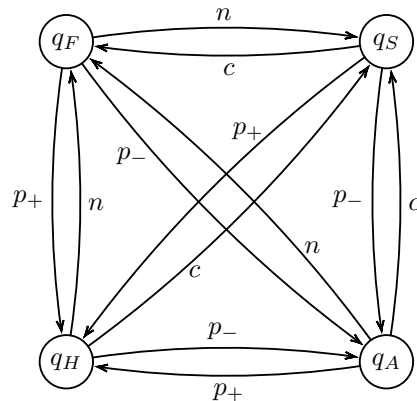
V [?] je použit fuzzy automat pro modelování nálady hráče ve hře šachy. Automat je tvořen čtveřicí stavů reprezentující jednotlivé „elementární“ nálady, konkrétně „zamyšlený“, „překvapený“, „radostný“ a „rozzuřený“. Přechodová abeceda je tvořena následujícími nastanuvšími tahy:

- tah, který hráče dostal do výhody
- tah, který dostal hráče do nevýhody
- tah, po kterém se hráč octl v šachu (popř. matu)
- tah, kdy se nestalo nic z předchozích

1720 Výhoda a nevýhoda, resp. stupeň v jakém platí „hráč se dostal do výhody, resp. nevýhody“ lze snadno spočítat jako rozdíl ohodnocení šachovnice. Ohodnocení šachovnice lze určit například jako výžený součet figur hráče dle jejich síly. Tah zakončený šachem může být nadefinován dvojstavově nebo např. jako „stupeň matu“, tj. pro šach např. 0.5, pro mat 1 (jinak 0).

Následně jsou navržena přechodová pravidla. Například, dostane-li se hráč v rozzuřeném stavu do výhodné pozice, přejde do radostného stavu. Následně, pokud se v dalším tahu nestane nic zajímavého, přechází do zamyšleného stavu. Kompletní automat je k vidění na obrázku 15.

1730 Na základě aktuálního fuzzy stavu (kombinace různých stupňů elementárních nálad) automatu je upravována mimika hráče a jeho obličej je zobrazován



Obrázek 15: Automat pro modelování nálad hráče ve hře šachy. Stavy  $q_F$ ,  $q_S$ ,  $q_H$  a  $q_A$  reprezentují po řadě náladu „zamyšlený“, „překvapený“, „radostný“ a „rozzuřený“. Symboly  $p_+$  a  $p_-$  značí přechod k výhodné, resp. nevýhodné situaci pro hráče,  $c$  symbolizuje šach a  $n$  tah bez žádné změny situace na šachovnici. Převzato z [?], upraveno.

{img:ChessEmosFA}



Obrázek 16: Ukázka výrazů ve tváři hráče při ztrátě pěšce (vlevo) a při sebrání dámy (vpravo). Převzato z [?].

{img:ChessEmosScreens}

v počítačové hře. Ukázka některých herních situací je na obr. 16. Pro lepší uživatelský dojem autoři navíc implementovali plynulé přechody mezi stavy.

(zde bude doplněno: zmínit v úvodu, že pokud u event automatu není uplatnitelný žádný přechod, automat zůstává ve stávajícím stavu (a neukončuje svůj výpočet, jak je obvyklé)) (zde bude doplněno: If-Then pravidla jsou prý tzv. Takagi-Sugeno-Kang pravidla (dle [?])) (zde bude doplněno: Minimálně u event-driven automatů uvést, že je vhodné nějak rozpoložít stupně. Tedy, aby např. fuzzy stav měl v součtu vždycky 1.)

## 10.6 Modelování emocí

V [?] je prezentován prototyp systému pro modelování lidských emocí. Systém je založen na faktu, že velký vliv na emoce člověka má počasí. V jejich modelu (pro snazší sběr dat) pak konkrétně teplota vzduchu a množství světla. Pro reprezentaci emocí používají tzv. Plutchikovo kolo emocí. Jedná se o emoční model, který reprezentuje emoce pomocí dvojice ukazatelů, aktivace a ohodnocení. Např. pozitivní aktivace a negativní ohodnocení začí odpor či opovržení,

negativní aktivace při neutrálním ohodnocení znamená strach.

1750 Autoři používají lingvistické proměnné „negativní“, „neutrální“ a „pozitivní“ a jak pro aktivaci, tak pro ohodnocení. V jejich zjednodušeném modelu vstup do systému (data z teploměru a světelného čidla) nejprve transformují na odpovídající aktivaci (průměrná teplota znamená pozitivní aktivaci, nízká nebo naopak vysoká teplota pak negativní) a ohodnocení (čím je světleji, tím je ohodnocení pozitivnější).

Následně konstruují automat. Automat má 9 stavů, každé kombinaci lingvistických proměnných aktivace a ohodnocení odpovídá jeden. Poté definují přechody mezi nimi. Má-li dojít ke zvýšení (snížení, ponechání) hodnoty příslušné lingvistické proměnné, pak automat přejde k stavu odpovídajícímu vyššímu (nižšímu, stejnému) stupni této lingvistické proměnné. Například při vstupu „aktivace je pozitivní a ohodnocení je neutrální“ při stavu „aktivace je negativní a ohodnocení je pozitivní“ přejde do stavu „aktivace je neutrální a ohodnocení je pozitivní“.

1760 Z fuzzy stavu pak lze zpětně zkonstruovat, v jakém emocionálním rozpoložení se člověk aktuálně nachází. Autoři poukazují, že při pozměnění konfigurace (úprava definic fuzzy množin pro lingvistické proměnné) lze chování systému přizpůsobit povaze člověka (např. klidná povaha, pesimista a podob.), což může výsledky značně zpřesnit.

## 10.7 Sledování cen investic

Zde [?], automat „fuzzy-sleduje“ vzory v časovém průběhu výše ceny na trhu (např. „cena prudce vzrostla“). Na základě toho pak sleduje stav, v jakém stavu se cena nachází (např. těsně před výrazným růstem).

## 10.8 Sledování pohybu a aktivit osoby

1770 V [?] je prezentován postup, jak pomocí sledování různých parametrů (pohyb, získaný akcelerometrem, a el. vodivost kůže) určit pravděpodobnou činnost (nicnedělání, chůze, práce, odpočinek) osoby. Činnosti odpovídají stavům automatu a na základě parametrů se určí přechody. Navíc, je pravděpodobnější, že člověk ze stavu nicnedělání přejde do stavu práce než do stavu relaxace.

Stejnou techniku využívají v [?]. Zde však detekují činnosti: chůze, práce u sebe, hovor s kolegy, dávání si kávy, míting, a to pomocí polohy zařízení (chytrý telefon) osoby (určené triangulací Wifi sítě) a polohy těla.

1780 Poloha těla pakl může být buď, že osoba stojí, sedí nebo jde. Tyto hodnoty jsou opět sledovány fuzzy automatem (přechody jsou realizovány na základě dat z akcelerometru, pohybového čidla, a gyroskopu (senzoru náklonu).

## 10.9 Řídící systémy, fuzzy kontroléry

Vztah fuzzy automatů a řídicích systémů pracujících na bázi ontologického řízení jsou popsány např. zde [?][?][?]. Ontologické řídicí systémy se vyznačují popisem ontologických předpokladů. Ontologické předpoklady jsou popisy stavu modelovaného systému, na jejich základě systém provádí rozhodování. Systémy ovšem bývají často tak složité, že je nemožné zanést do znalostní báze všechny možné předpoklady. Z tohoto důvodu může u systému dojít k tzv. porušení ontolo-

gických předpokladů (např. při chybě či nečekaném vnějším podnětu). Dalo by se říci, že systém se takto dostane do nekonzistentního stavu.

1790 Je-li systém modelován klasickým, bivalentním automatem (*zde bude doplněno: on ani tak není klasický bivalentní, ale event-driven*) je pravděpodobné, že takovouto situaci nebude schopen ošetřit. Mohl by se tak ve snaze provést synchronizaci reálného a modelového stavu např. zacyklit mezi dvojicí diskrétních stavů.

## 10.10 Ostatní a nezařezeno

V [?] popisují fuzzy programy. Fuzzy program je prakticky deklarativní program, fuzzy reguérní výraz, který popisuje, jak mají vypadat výsledky.

V [?] používají fuzzy automat pro vylepšení (zefektivnění) human-robot interakce. Nikde však neuvádějí, jak konkrétně.

1800 V [?] definují fuzzy lexikální analyzátor. Funguje na bázi klasického pattern matching, jen s informací, ve kterém koncovém stavu skončil. (*zde bude doplněno: mimochodem, v [?] je poměrně pěkně popsána teorie (deterministické a nedeterministické automaty, regexp i reg. jazyky).*)

## 11 Pravděpodobností automaty

V [?] a [?] zavádějí pravděpodobnostní automat s pamětí. Každý stav si pamatuje něco jako své prefixy. Takže je to tak trochu automat pracující s celými slovy. Spolu s učením ho používají ke korekci textu bible. A posléze pro segmentaci DNA.

V [?] používají pravděpodobnostní automaty na epigenetiku. Víc fakt netuším.

1810 V [?] a [?] používají aritmetický automat (automat, který v každém přechodu může něco udělat) pro počítání počtu výskytů vzoru v řetězci. To pak využívají na určování proteinů (protein je rozštěpen a jednotlivé peptidy jsou pomocí hmotnostního spektrometru analyzovány a hmotnosti předány automatu).

V [?] modelují pomocí pravděpodobnostního automatu počítačovou síť (zvlášť uzly a kanály, jejich matice pak matematicky spojují do jedné velké matice „matice automatu sítě“). Stav uzlu jsou délky front v bufferech u jednotlivých kanálů. (U kanálů nevím.) Výsledkem je návrh optimální směrovací tabulky pro celou síť.

1820 V [?] pomocí pravděpodobnostních automatů analyzují aspektová tvrzení (tj. zda-li je tvrzení míněno pozitivně, negativně, neutrálně nebo s rozporem). Stav automatu jsou všechna slova z celé databáze, vstupní abeceda taktéž. Počáteční stav je první slovo tvrzení, koncový pak poslední. Pravidla jsou, pochopitelně, sestavena učením. Přechody automatu mohou popisovat fráze či použití stejných slov v různém kontextu. Každý stav má pak přiřazenu polaritu, zkomibnováním všech polarit napříč celým tvrzením (napříč všemi stavy) se pak obdrží celková polarita tvrzení.

1830 V [?] používají automat jako generátor strojového člověka, konkrétně jako pracovníka banky. Generuje dotazy a, a na základě podnětů od člověka získává informace, co a jak chce (v jejich příkladě, bankovním pracovníku, jestli chce člověk vybrat peníze, nebo provést transakci z účtu na účet). (*zde bude doplněno: Tak ale tohle je spíš práce pro gramatiky, ne?*)

V [?] (a možná v [?]) de facto překládají fonetickou abecedu na angličtinu (některá slova se totiž mohou vyslovovat různě, např. „data“ a „dejta“ a naopak, dvě slova shodně, např. „red“ a „red“). Z databáze sestavují automaty pro jednotlivá slova a ty na principu automatu pro lexikální analyzátor slývají do jednoho obrovského.

V [?] používají pravděpodobnostní automat jako jednu komponentu procesu pro parsování „MIDI“ dat ze zvukového záznamu.

1840 V [?] kombinují editační operace s pravděpodobnostními automaty pro pattern matching. Tentorkát však přirozeného jazyka.

Mějme pravděpodobnostní automat. Pak v [?] studují, jak najít nejpravděpodobnější řetězec, tj. řetězec s nejvyšší pravděpodobností přijetí.

V [?] používají pravděpodobnostní automat pro reprezentaci databáze hudebních skladeb. Tj. že abeceda jsou tóny a pravděpodobnosti přechodů pravděpodobnosti, v jakých se v testovacím datasetu vyskytoval za aktuálním tónem tón na vstupu automatu.

V [?] popisují, jak pomocí statistických dat sestavit automat, který něco modeluje (nastanuvší události a výsledný stav). Např., v jejich případě, skóre ve hře, jejich pravděpodobnosti a výsledný stav hry (prohra, výhra).

1850 V [?] poukazují na použití pravděpodobnostních automatů jako jednu z alternativ pro jednu z částí procesu sledování aktivit člověka pomocí automatu (podobně, jak u fuzzy automatů výše). Uvádějí však další aplikace pro tuto techniku (biometrika, sémantická analýza videa, bezpečnost, uživatelská rozhraní, syntéza animací).

V [?] zpracovávají signál (rozpoznávají). Vypadá to ale na nějaký dvojrozměrný, nebo co.

V [?] pomocí automatů parsují reference z textu a rozpoznávajív nich strukturu.

V [?] zpracovávají přirozený jazyk. Vypadá to jako tree.

1860 V [?] zřejmě zpracovávají signál.

[?] vypadá jako lisování dat.

V [?] používají pravděpodobnostní automat pro rozpoznávání pravděpodobnostních stromů reprezentujících chování (události, např. systémová volání) počítačových aplikací.

V [?] používají zřejmě něco na způsob lexikálního analyzátoru.

V [?] popisují velmi zajímavý problém - šíření nákazy. A to pomocí pravděpodobnostního buzněčného automatu.

Dále pak ještě [?], ale to chce načíst pořádně.

## 12 Konkrétní příklady

### 1870 12.1 Automat jako „lepší“ reprezentace něčeho

Markovův řetěz (probabilistický (dvoj)stavový systém). Petriho sítě (petri nets). Rozhodovací stromy. Řídící systémy, událostní systémy (ale není to to samé, co petriho sítě?). [?]

### 12.2 Rozpoznávání ručně psaného textu

[7], [?] a podob. ...

### 12.3 Detekce překlepů

[8] až na konci, Fuzzy Aho-Corasick algorithm (popř. dohledat jiný zdroj).  
[?]



## Reference

- 1880 [1] M. C. Pegalajar A. Blanco, M. Delgado. Fuzzy automaton induction using neural network. *International Journal of Approximate Reasoning*, 2001.
- [2] Javier Echanobe José R. Gonzáles Mendívil José R. Garitagoitia Carlos F. Alastrueya. Deformed systems for contextual postprocessing. *Fuzzy Sets and Systems*, 1998.
- [3] C. L. Giles Ch. W. Omlin, K. K. Thornber. Fuzzy finite-state automata can be deterministically encoded into recurrent neural networks. *University of Technical Report*, 1996.
- [4] Roger White Conrad Power, Alvin Simms. Hierarchical fuzzy pattern matching for the regional comparison of land use maps. *International Journal of Geographical Information Science*, 2010.
- 1890 [5] Prof. Sagar A. More Dhiraj Kumar Patel. Edge detection technique by fuzzy logic and cellular learning automata using fuzzy image processing. *International Conference on Computer Communication and Informatics*, 2013.
- [6] H. A. Girijamma Dr. V. Ramaswamy. Conversion of finite automata to fuzzy automata for string comparison. *International Journal of Computer Applications*, 2012.
- [7] J. J. Astrain; J. R. Garitagoitia; J. R. Gonzalez De Mendivil; J. Villadangos; F. Farina. Approximate string matching using deformed fuzzy automata: A learning experience. *Springer Science & Business Media*, 2004.
- 1900 [8] Abdulwahed Almarimi Gabriela Andrejková and Asmaa Mahmoud. Approximate pattern matching using fuzzy logic. *CEUR Workshop Proceedings*, 2003.
- [9] S.S. Yau G.F. DePalma. Fractionally fuzzy grammars with application to pattern recognition. *US-Japan Seminar on Fuzzy Sets and their Applications*, 1974. článek jako e-book: <http://bit.ly/2cumxjz>, výcuc v MorMal-FuzzAutAndLangs 10.7.
- [10] Gracián Triviño Gonzalo Bailadora. Pattern recognition using temporal fuzzy automata. *Fuzzy Sets and Systems*, 2010.
- 1910 [11] Lefteris A. Mantelas Thomas Hatzichristos and Poulicos Prastacos. A fuzzy cellular automata modeling approach – accessing urban growth dynamics in linguistic terms. *Computational Science and Its Applications*, 2010.
- [12] Kou-Yuan Huang. *Syntactic Pattern Recognition for Seismic Oil Exploration*. World Scientific, 2002.
- [13] J.R. Garitagoitia J. Astrain, J.R. González de Mendívil. Fuzzy automata with  $\epsilon$ -moves compute fuzzy measures between strings. *Fuzzy Sets and Systems* 157, 2005.

- 1920 [14] José R. Garitagoitia José R. González de Mendívil. Fuzzy languages with infinite range accepted by fuzzy automata: Pumping lemma and determination procedure. *Fuzzy sets and Systems*, 2014.
- [15] S. C. Kremer M. Doostfateme. New directions in fuzzy automata. *International Journal of Approximate Reasoning*, 2004.
- [16] Hasan Ahmed Manuj Darbari and Vivek Kr. Singh. Application of fuzzy automata theory and knowledge based neural networks for development of basic learning model. *Computer Technology and Application 2*, 2011.
- [17] Gerasimos G. Rigatos. Fault detection and isolation based on fuzzy automata. *Information Sciences*, 2009.
- [18] M. M. ZAHEDI S. MOGHARI and R. AMERI. New direction in fuzzy tree automata. *Iranian Journal of Fuzzy Systems*, 2011.
- 1930 [19] Mukta N. Joshi S. R. Chaudhari. A note on fuzzy tree automata. *International Journal of Computer Applications*, 2012.
- [20] Miroslav Stamenkovic, Aleksandar; Ciric. Construction of fuzzy automata from fuzzy regular expressions. *Fuzzy Sets and Systems*, 2012.
- [21] Witold Pedrycz Yongming Li. Fuzzy finite automata and fuzzy regular expressions with membership values in lattice-ordered monoids. *Fuzzy Sets and Systems*, 2005.