

# 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 Pravděpodobnostní počet . . . . .	5
	1.3 Fuzzy vs. pravděpodobnostní přístup . . . . .	5
	1.4 Popis v přirozeném jazyce . . . . .	5
10	1.5 Základní pojmy z teorie automatů . . . . .	7
<b>2</b>	<b>Definice fuzzy automatu</b>	<b>8</b>
	2.1 Koncept automatu . . . . .	8
	2.2 Nedeterministický bivalentní automat . . . . .	9
	2.3 Nedeterministický fuzzy automat . . . . .	10
	2.4 Reprezentace nedeterministického fuzzy automatu . . . . .	10
	2.5 Výpočet nedeterministického fuzzy automatu . . . . .	11
<b>3</b>	<b>Variety fuzzy automatů</b>	<b>13</b>
	3.1 Deterministický fuzzy automat . . . . .	13
	3.2 Nedeterministický fuzzy automat s $\epsilon$ -přechody . . . . .	14
20	3.3 Zobecněný automat . . . . .	14
	3.4 Fuzzy automat s výstupem . . . . .	14
	3.5 Stavový stroj . . . . .	14
	3.6 Událostmi řízený fuzzy automat . . . . .	14
	3.7 Zásobníkový fuzzy automat . . . . .	15
<b>4</b>	<b>Další modely podobné fuzzy automatům</b>	<b>15</b>
	4.1 Fuzzy tree automaty . . . . .	15
	4.2 Stromy a pseudotermy . . . . .	16
	4.3 Fuzzy tree automat a jazyk jím rozpoznávaný . . . . .	18
	4.4 Buněčné fuzzy automaty . . . . .	20
30	4.5 „Bivalentní“ buněčný automat . . . . .	20
	4.6 Buněčné fuzzy automaty . . . . .	24
<b>5</b>	<b>Užitečné techniky pro práci s fuzzy automaty</b>	<b>24</b>
	5.1 Fuzzy automat bivalentního automatu . . . . .	24
	5.2 Fuzzy automat rozpoznávající $\omega$ . . . . .	25
	5.3 Podobnost symbolů . . . . .	26
	5.4 Editační operace, deformovaný automat . . . . .	26
	5.5 Učící se fuzzy automaty . . . . .	29

	5.6	Fuzzy automaty a neuronové sítě . . . . .	29
	<b>6</b>	<b>Obecně k rozpoznávání</b>	<b>30</b>
40	6.1	Detekce úplných $m$ -árních stromů . . . . .	30
	6.2	Lexikální analyzátor . . . . .	31
	6.3	Rozpoznávání ručně psaného textu . . . . .	31
	6.4	Detekce překlepů . . . . .	31
	6.5	Korekce textu pomocí pravděpodobnostních automatů . . . . .	31
	6.6	Fuzzy programy . . . . .	32
	6.7	Pravděpodobnost výskytu vzoru v řetězci . . . . .	32
	6.8	A dále ještě ... . . . .	32
	6.9	Pravděpodobnostní rozpoznávání fonetické abecedy . . . . .	32
	6.10	Parsování MIDI ze zvuku . . . . .	32
50	6.11	Reprezentace zvuků . . . . .	32
	6.12	Pattern matching přirozeného jazyka . . . . .	32
	6.13	Nejpravděpodobnější řetězec . . . . .	33
	6.14	PATtern matching signálů . . . . .	33
	6.15	Pattern matching dvourozměrného signálu . . . . .	33
	6.16	Parsování referencí . . . . .	33
	6.17	Pravděpodobnostní rozpoznávání přirozeného textu . . . . .	33
	<b>7</b>	<b>Modelování a simulace</b>	<b>34</b>
	7.1	Automobilismus . . . . .	34
	7.2	Sledování vytížení el. sítě . . . . .	34
60	7.3	Modelování emocí postav v počítačových hrách . . . . .	34
	7.4	Modelování emocí . . . . .	35
	7.5	Sledování a modelování aktivit počítačových programů . . . . .	36
	7.6	Sledování cen investic . . . . .	37
	7.7	Sledování pohybu a aktivit osoby . . . . .	37
	7.8	Řídící systémy, fuzzy kontroléry . . . . .	37
	7.9	Predikce průběhu hry . . . . .	37
	7.10	Pravděpodobnostní sledování aktivit člověka . . . . .	37
	7.11	Problém městského růstu (urban growt problem) . . . . .	38
	7.12	A co na to učící? . . . . .	40
70	7.12.1	Něco . . . . .	40
	7.12.2	Něco . . . . .	41
	7.12.3	Něco . . . . .	41
	7.12.4	Něco . . . . .	41
	7.13	Modelování vytíženosti počítačové sítě . . . . .	41
	7.14	Sledování aspektů tvrzení . . . . .	41
	7.15	Generátor strojového člověka . . . . .	41
	7.16	Metoda lisování dat . . . . .	42
	<b>8</b>	<b>Biologie a medicína</b>	<b>43</b>
	8.1	Rozpoznávání řetězců DNA . . . . .	43
80	8.2	Simulace růstu mořských řas . . . . .	43
	8.3	Rozpoznávání vzorů v lékařských snímcích . . . . .	43
	8.4	Analýza zdravotního stavu pacienta . . . . .	43
	8.5	Detekce zhoubných nádorů . . . . .	44
	8.6	Analýza kardiogramu . . . . .	44

	8.7	Řízení protézy . . . . .	44
	8.8	Modelování buněčných sítí . . . . .	44
	8.9	Modelování šíření infekční nemoci . . . . .	45
	<b>9</b>	<b>Zpracování obrazu</b>	<b>45</b>
	9.1	Konvoluce . . . . .	45
90	9.2	Hledání hran . . . . .	46
	9.3	Ostraňování šumu . . . . .	47
	9.4	Rozpoznávání jednoduchých vzorů . . . . .	48
	9.5	Složené geometrické útvary . . . . .	49
	9.6	Detekce požárů . . . . .	52
	9.7	A co na tu učící? . . . . .	52
	9.7.1	Něco . . . . .	52

# 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, pravděpodobnostního počtu a základní pojmy z teorie automatů.

## 1.1 Fuzzy teorie

Fuzzy teorie je poměrně nový pohled (oficiálně byl publikován v roce 1965 (zde bude doplněno: ocitovat Zadeh: Fuzzy sets) ) na matematický svět. Uvažuje, že pravdivost tvrzení nenabývá jen hodnot „pravda“ a „nepravda“, ale libovolného stupně pravdivosti (splňujícího určitá kritéria). Stejně tak, množiny (klasické neboli bivalentní) mohou prvek buďto obsahovat nebo neobsahovat. Fuzzy teorie přichází s konceptem fuzzy množin, tedy množin, které mohou prvky obsahovat jen částečně.

Svět kolem nás je jen málokdy dvojstavový, mít možnost okolí popisovat ve stupních pravdivosti je tak více, než přirozené. Například, tvrzení „student učivo umí“ lze těžko ohodnotit pravdou či nepravdou, pokud student z testu získal 50% bodů. Naopak, říci, že „student učivo umí z 1/2“ je mnohem přirozenější.

Fuzzy teorie se toto uvažování snaží formalizovat matematicky. Na úvod si nadefinujeme fuzzy množinu.

**Definice 1.1** (Fuzzy množina). *Mějme libovolnou neprázdnou množinu  $U$  (tzv. univerzum). Pak zobrazení  $\rho_A : U \rightarrow [0, 1]$  nazvěme členská funkce. Množinu  $A$  nazýváme fuzzy množina nad  $U$ . Pro libovolné  $x \in U$  říkáme, že prvek  $x$  náleží do množiny  $A$  ve stupni  $\rho_A(x)$ .*

**Poznámka 1.1.** Členská funkce nemusí být nutně zobrazení do intervalu  $[0, 1]$ . Interval může být nahrazen svazem s 0 a 1, pokud splňuje ještě některá další kritéria. Studium těchto vlastností však není účelem této práce a proto bude vynecháno. Navíc, práce s intervalem  $[0, 1]$  je poměrně výhodná z důvodu snadné implementace na počítači a proto ve všech praktických aplikacích (které jsou smyslem této práce) se bude používat právě tento interval.

**Značení.** Abychom rozlišili „klasické“ bivalentní množiny od fuzzy množin, budou fuzzy množiny v tomto textu značeny malými řeckými písmeny.

Nyní se podíváme na základní množinové operace nad fuzzy množinami.

**Definice 1.2** (Operace nad fuzzy množinami). *Mějme dvě fuzzy množiny  $\pi$  a  $\rho$  nad univerzem  $U$ . Pak definujeme jejich průnik, sjednocení, rovnost a inkluzi (relaci „býti podmnožinou“) jako:*

$$\begin{aligned}\pi \cup \rho &= \pi(x) \vee \rho(x) \\ \pi \cap \rho &= \pi(x) \wedge \rho(x) \\ \pi = \rho &= \pi(x) = \rho(x) \\ \pi \subseteq \rho &= \pi(x) \leq \rho(x)\end{aligned}$$

pro všechna  $x \in U$ .

Připomeňme, že operátory  $\wedge \vee$  na reálném intervalu  $[0, 1]$  jsou definovány jako min a max. Operátor  $\leq$  značí přirozené uspořádání čísel.

Množinu všech fuzzy množin nad univerzem  $U$  (tedy protějšek potenční množiny u „klasických“ množin) budeme značit  $\mathcal{F}(U)$ . Obdobně jako u „klasických“ množin lze definovat kartézský součin a relace, můžeme definovat kartézský součin fuzzy množin a fuzzy relace.

**Definice 1.3** (Kartézský součin fuzzy množin). *Mějme fuzzy množiny  $\rho_1, \dots, \rho_n$  nad univerzy  $U_1, \dots, U_n$ . Pak kartézský součin těchto fuzzy množin, psáno  $\rho_1 \times \dots \times \rho_n$  nebo  $\prod_{i=1}^n \rho_i$  je definován jako zobrazení  $U_1 \times \dots \times U_n \rightarrow [0, 1]$ .*

**Definice 1.4** (Fuzzy relace). *Jako fuzzy relaci na fuzzy množinách množinách  $\rho_1, \dots, \rho_n$  nazýváme libovolnou fuzzy podmnožinu kartézského součinu těchto fuzzy množin.*

(zde bude doplněno:  $t$ -norma a konorma) .

## 1.2 Pravděpodobnostní počet

(Dle potřeby rozepsat více.) (zde bude doplněno: Pravděpodobnostní počet, rozepsat)

(zde bude doplněno: Doplnit příklady)

Z pohledu pravděpodobnostních automatů je nejzákladnějším pojmem pravděpodobnostního počtu pravděpodobnostní míra. Pravděpodobnostní míra, neboli pravděpodobnost, je reálné číslo z intervalu  $[0, 1]$ .

## 1.3 Fuzzy vs. pravděpodobnostní přístup

{subsec:FuzzyVsProb}

V předchozích dvou podkapitolách byly zavedeny základní pojmy z oblasti fuzzy teorie a pravděpodobnostního počtu. Je vidět, že jak stupeň pravdivosti, tak pravděpodobnostní míra jsou obě reálná čísla z intervalu  $[0, 1]$ . Oba tyto pojmy totiž popisují určitou formu neurčitosti, každá však jinou. Často však může docházet k jejich záměně.

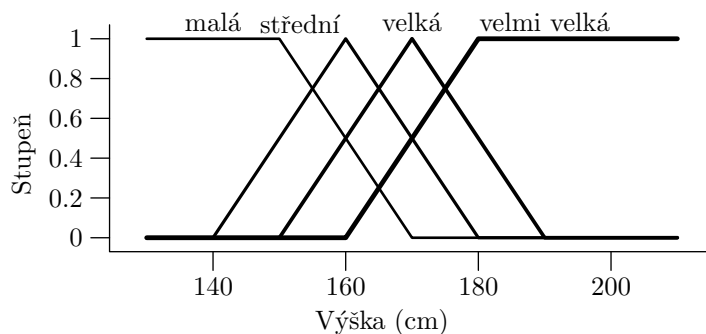
Je však třeba mít na paměti, že tyto pojmy jsou zcela rozdílné. Stupeň pravdivosti popisuje, jak moc je tvrzení pravdivé, zatímco pravděpodobnost, jak moc je pravděpodobné, že tvrzení pravdivé bude.

Názorně je to vidět na příkladě: Trvzení „Já jsem vysoký“ může nabývat různých stupňů pravdivosti, avšak pravděpodobnost se nemění. Naopak tvrzení „Zítra ráno mi ujede autobus“ může popisovat buď zcela pravdivou skutečnost, nebo zcela nepravdivou, nicméně má smysl uvažovat, s jakou pravděpodobností tento jev nastane.

Občas se stává, že lze uplatnit oba přístupy. Například trzení „Venku je pěkné počasí“. Na jednu stranu se můžeme bavit o „pěknosti“ počasí (tj. jak moc je venku pěkně). Ovšem na druhou stranu můžeme uvažovat, že autor výroku není seznámen s aktuálním počasím a proto může určovat s jakou pravděpodobností venku skutečně je pěkně. Právě z takovýchto situací pramení časté záměny těchto dvou přístupů.

## 1.4 Popis v přirozeném jazyce

Vzhledem k tomu, že účelem této práce je studovat uplatnění fuzzy a pravděpodobnostních automatů v praxi, je nutné zavést nástroj pro přemostění „propasti“ mezi fuzzy teorií (pravděpodobnostním počtem) a popisem v přirozeném jazyce.



Obrázek 1: Graf významových funkcí štítků „malá“, „střední“, „velká“ a „velmi velká“

{fig:lingVarsMeansChart}

Pravděpodobnostní počet k tomuto účelu používá jevy. Je poměrně přirozené říkat „Pravděpodobnost jevu  $e$  je  $p$ “.

Pro popis stupňů pravdivosti by bylo možné používat „Tvzení  $t$  je pravdivé ve stupni  $d$ “. Bude však výhodnější použít tzv. lingvistické proměnné [?]. Každá lingvistická proměnná  $\alpha$  je tvořena univerzem  $U$  hodnot a množinou lingvistických štítků  $T$ . Pro každý štítek  $X \in T$  pak definuje jeho význam  $M(X)$ , fuzzy množinu nad  $U$  (tj. funkci, která hodnotě z univerza přiřadí stupeň pravdivosti). Fakt, že lingvistická proměnná  $\alpha$  nabývá „hodnoty“  $X$  budeme zapisovat  $\alpha = X$ .

**Příklad 1.1.** Mějme lingvistickou proměnnou  $\alpha$  s významem „výška osoby“. Univerzem hodnot jsou všechna nezáporná reálná čísla. Lingvistické štítky jsou „malá“, „střední“, „vysoká“ a „velmi velká“. Význam jednotlivých štítků je vyobrazen na obrázku 1.

Osoba  $X$  má výšku  $h$ . Namísto „Platnost tvrzení „osoba  $X$  má střední výšku“ je ve stupni  $M(\text{střední})(h)$ “ můžeme říkat „osoba  $X$  má střední výšku“. Symbolicky zapsáno:  $\alpha = \text{střední}$ .

Pro popis složitějších zákonitostí je potřeba pokročilejší nástroj. Vztahy tak budeme popisovat pomocí tzv. fuzzy IF–THEN pravidel. Klasické bivalentní IF–THEN pravidlo je výrok: je tvaru:

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

kde  $x_1, \dots, x_n, y$  jsou podvýroky<sup>1</sup>.

**Příklad 1.2.** 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

Fuzzy IF–THEN pravidla pak budou vypadat následovně:

Jestliže  $\alpha_1 = x_1, \dots, \alpha_n = x_n$ , pak  $\beta = y$

<sup>1</sup>Zapišeme-li IF–THEN pravidlo jako formuli, obdržíme Hornovskou klauzuli. Můžeme tak určit pravdivost jejího důsledku  $y$  na základě pravdivosti předpokladů  $x_1, \dots, x_n$ .

kde  $\alpha_1, \dots, \alpha_n, \beta$  jsou lingvistické proměnné a  $x_1, \dots, x_n, y$  jsou jejich štitky.

**Příklad 1.3.** Uvažujme systém, u kterého známe tlak (lingvistická proměnná  $\alpha_t$ ) a teplotu (lingvistická proměnná  $\alpha_p$ ) a popisujeme úroveň otevření ventilu (lingvistická proměnná  $\alpha_v$ ). Označme štitky  $p_L$  („tlak je nízký“),  $p_N$  („tlak je normální“),  $p_H$  („tlak je vysoký“) a dále  $t_N$  („teplota je v normě“) a  $t_H$  („teplota je zvýšená“). Konečně zavedme štitky  $v_O$  („ventil je uzavřen“) a  $v_C$  („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*

*mohou být přepsána jako*

*Jestliže  $\alpha_p = p_L, \alpha_t = t_H$  pak  $\alpha_v = v_C$*

*Jestliže  $\alpha_p = p_H, \alpha_t = t_N$  pak  $\alpha_v = v_O$*

## 200 1.5 Základní pojmy z teorie automatů

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

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

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  $\varepsilon$ ).

210 Symbol  $\Sigma^+$  pak značí všechny řetězce nad abecedou  $\Sigma$  vyjma  $\varepsilon$ .

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

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.

Podobně, jako jsme si zavedli fuzzy množinu jako protějšek „klasické“ množiny, můžeme nadefinovat i fuzzy jazyk. Fuzzy jazyk nad abecedou  $\Sigma$  je její libovolná fuzzy podmnožina, tj.  $\lambda \in \mathcal{F}(\Sigma^*)$ . *(zde bude doplněno: a co probabilistický jazyk?)*

## 2 Definice fuzzy automatu

220 V této kapitole bude podrobně popsán a formálně nadefinován „fuzzy automa-  
mat“ 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 [?] (pokud není uvedeno  
jinak).

Hned na úvod je třeba zdůraznit terminologii. Jak bývá u automatů zvy-  
kem, 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 nedeterminis-  
tický, bude proto druhá polovina této kapitoly věnována právě nedeterminis-  
tickému fuzzy automatu. Zbývající známé varianty fuzzy automatů budou poté  
230 shrnuty v následující kapitole.

*(zde bude doplněno: někde tady zdůraznit konečné (ve smyslu konečnosti automatu)  
automaty?)*

### 2.1 Koncept automatu

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  
radí například také Turingovy stroje [?].

240 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 nedeter-  
ministického tento předpoklad neplatí, tj. výpočet automatu se může octnout v  
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.

250 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 abece-  
dou. O popis přechodů mezi stavy se bude starat přechodová funkce.

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 libovoný 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.

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

## 2.2 Nedeterministický bivalentní automat

Nedeterministický bivalentní automat je definován následovně:

**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.*

{def-NedBivAut}

(zde bude doplněno: reprezentace? příklad?)

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  $I$ .

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  $q' \in \mu(q, a)$ . Vyprázdnil-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.

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.

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

## 2.3 Nedeterministický fuzzy automat

300 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.

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. [?], [?]), jinde zase s fuzzy množinami ([?], [?], [?])<sup>3</sup>. 310 Vzhledem k tomu, že druhý jmenovaný, tj. automat s fuzzy množinou 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. [?]).

320 Jedná se o tabulku, která v řádcích obsahuje aktuální stavy a ve sloupcích následující stavy (tím je dána množina stavů). Poté buňka na řádce  $q$  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. [?], [?]). 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 330 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. [?], [?], [?]). 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  $\sigma(q)$

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

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.

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

{def-FuzzStav}

**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 1.** ?? 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  $w_0$ .*

380

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

{def-RozPreFunFuzzStav}

**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:*

1.  $\mu^*(q, \epsilon, q) = 1$  pro všechna  $q \in Q$
2.  $\mu^*(q, ua, q') = \bigvee_{p \in Q} \mu^*(q, u, p) \otimes \mu(p, a, q')$  pro všechna  $q, q' \in Q, u \in \Sigma^*, a \in \Sigma$

390

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:

{def-RetPriAut}

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

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

**Poznámka 2.2.** *V literatuře (např. [?] [?] [?]) se obvykle lze setkat s „techničtějším“ 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:

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

400

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 rozpoznávány. 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 Varianty fuzzy automatů

V této kapitole budou uvedeny další typy fuzzy automatu. Kromě nedeterministického fuzzy automatu, kterému byla věnována předchozí kapitola, se v souvislosti s fuzzy automaty často mluví s o dalších výpočetních modelech. A právě ty jsou předmětem této kapitoly.

Vzhledem k tomu, že se obvykle jedná o modifikace elementární, nebudou rozebírány do hloubky. Podrobnější informace o těchto automatech jsou k nalezení v literatuře.

#### 3.1 Deterministický fuzzy automat

V teorii „klasických“ bivalentních automatů se rozlišuje automat deterministický a nedeterministický. Deterministický automat je charakterizován tím, že každý krok jeho výpočtu je jednoznačně určen. U nedeterministického tento předpoklad platit nemusí. Právě z tohoto důvodu je deterministický automat klíčovým pojmem v oblasti „klasických“ bivalentních automatů, zatímco nedeterministický jeho zobecnění.

U fuzzy automatů tomu je přesně naopak. Deterministický fuzzy automat se obvykle definuje pouze jako speciální případ nedeterministického. Podobně jako u nedeterministického automatu existuje několik různých definic. V [?] je definován jako deterministický bivalentní automat, jen koncové stavy jsou fuzzy. Dle [?] je to deterministický bivalentní automat, ale s fuzzy počátečními stavy, koncovými stavy i přechodovými funkcemi.

V [?] definují deterministický fuzzy automat jako nedeterministický fuzzy automat s tím, že je doplněn o omezení přechodové funkce, že z každého stavu při každém vstupním symbolu automat může přejít do nejvýše jednoho stavu.

Deterministické fuzzy automaty mají význam především pro implementace. Jak bylo zmíněno v předchozí kapitole, (zde bude doplněno: zmínit složitost (popř. ověřit, že byla zmíněna)) časová složitost výpočtu nedeterministického fuzzy automatu je exponenciální vzhledem k délce vstupu. U deterministického automatu je tato složitost lineární. Bohužel, zpravidla bývá vykoupena mnohonásobně vyšším počtem stavů automatu.

U všech tří citovaných definic deterministického fuzzy automatu jsou uvedeny algoritmy pro tzv. determinizaci automatu, tedy převod nedeterministického fuzzy automatu na jemu odpovídající deterministický. Algoritmus podle [?] byl naimplementován (*zde bude doplněno: opsat ho?*).

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

450 Nedeterministický fuzzy automat s  $\epsilon$ -přechody je rozšíření nedeterministického fuzzy automatu. Toto rozšíření je realizováno pomocí tzv.  $\epsilon$ -přechodů. „Klasický“ přechod automatu při svém výpočtu realizuje, je-li na vstupu symbol odpovídající symbolu pravidla. Oproti tomu  $\epsilon$ -přechod však automat může realizovat kdykoliv, bez ohledu na symbol na vstupu.

Takovýto automat nalézá uplatnění především v praktických aplikacích. S jeho pomocí lze velmi elegantně vyřešit např. rozdvojení výpočtu nebo přesun výpočtu do jiného stavu, a to bez ohledu na vstup. Formální definice tohoto automatu lze nalézt např. v [?] či [?].

### 3.3 Zobecněný automat

460 V kapitole 1.3 byl rozebírán koncepční rozdíl mezi stupněm pravdivosti a pravděpodobností. V teorii fuzzy/pravděpodobnostních automatů existují snahy tyto dva přístupy sloučit. Vniklo proto několik definic tzv. zobecněného automatu, např. [?] či Max-Min automat [?].

Hodnota reprezentující neucitost (tj. stupeň pravdivosti nebo pravděpodobnost) se u těchto automatů nazývá širším pojmem „váha“. Tomuto druhu automatů se proto někdy říká automat s váhami.

### 3.4 Fuzzy automat s výstupem

470 Rozšířením klasického automatu lze získat automat s výstupem. Automat kromě přechodové funkce disponuje také výstupní přechodovou funkcí, která při přechodu mezi stavy odesílá na výstup symboly. Automat s výstupem tak obvykle nemá koncové stavy. S fuzzy automaty s výstupem se lze setkat např. v [?][?][?][?][?].

### 3.5 Stavový stroj

V určitém smyslu opakem fuzzy automatu s výstupem je stavový stroj. Oproti němu totiž výstupní schopnosti běžného fuzzy automatu nerozšiřuje, ale naopak ubírá. Stavový stroj je totiž výpočetní model, který nedisponuje žádnou formou výstupu<sup>4</sup>. U stavových strojů není totiž důležitý výsledek výpočtu, ale jeho průběh.

Takovýto automat je uveden např. v [?].

### 3.6 Událostmi řízený fuzzy automat

{subsec:FuzzEvMach}

Jako událostmi řízený fuzzy automat se obvykle označuje další třída výpočetních modelů, které zobecňují fuzzy automaty. Vycházejí z myšlenky, že přechodová

<sup>4</sup>V určitém smyslu však může být za výstup považován stav (resp. fuzzy stav), ve kterém výpočet skončil.

funkce může být zapsána jako fuzzy IF–THEN pravidlo. Slovní popis činnosti přechodu  $(q, a, q', d) \in \mu$  je totiž následující:

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

480 Událostmi řízené fuzzy automaty tak uvažují, že přechodová funkce může být libovolná množina fuzzy IF–THEN pravidel. *(zde bude doplněno: doplnit pár citací)* Takovýto automat má velké praktické uplatnění, protože popis pomocí fuzzy IF–THEN pravidel je mnohem přirozenější a flexibilnější, než pomocí symbolů.

**Poznámka 3.1.** *V souvislosti s automaty budeme často používat pojem „abecda událostí“. Tímto pojmem budeme souhrnně nazývat všechny „události“, tj. logistické proměnné, které se v přechodové funkci vyskytují.*

### 3.7 Zásobníkový fuzzy automat

Podobně jako v „klasické“ teorii automatů, existuje i zásobníkový fuzzy automat. Jedná se o nedeterministický fuzzy automat, který je navíc vybaven tzv. zásobníkem. Zásobník je struktura typu Li–Fo tvořena symboly tzv. zásobníkové  
490 abecedy. Přechodová funkce automatu pak kromě symbolu na vstupu sleduje také symbol na vrcholu zásobníku a při přechodu kromě změny stavu také provádí vložení, odebrání či nahrazení symbolu na vrcholu zásobníku.

Zásobníkový fuzzy automat se však od „klasického“ v některých oblastech liší<sup>5</sup>.

Definice zásobníkového automatu je např. v [?].

## 4 Další modely podobné fuzzy automatům

V této kapitole bude navázáno na kapitolu předcházející. Budou zde představeny výpočetní modely, které se také automatům podobají. Oproti automatům v  
500 předchozí kapitole se však od „běžného“ nedeterministického automatu liší více a proto je třeba pro jejich zavedení doplnit některé související pojmy.

### 4.1 Fuzzy tree automaty

Fuzzy tree automaty jsou speciální třídou automatů, které jsou navrženy pro rozpoznávání dat, které mají v sobě obsaženu určitou stromovou strukturu.

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

510 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

<sup>5</sup>Vlastnost, že zásobníkový automat rozpoznává bezkontextový jazyk *(zde bude doplněno: dohledat, ocitovat)* u zásobních fuzzy automatů obecně neplatí

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  $M(M(x))$ .

## 4.2 Stromy a pseudotermy

**Definice 4.1** (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 4.2** (Částečně spořádaná abeceda). *Částečně spořádaná abeceda je dvojice  $(N, T)$ , kde  $N$  a  $T$  jsou dvě disjunktí konečné abecedy (tj.  $N \cap T = \emptyset$ ).*

{def:Tree}

**Definice 4.3** (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  $w \in \bar{U}$

Místo  $t(w)$  budeme psát jen  $t$ .

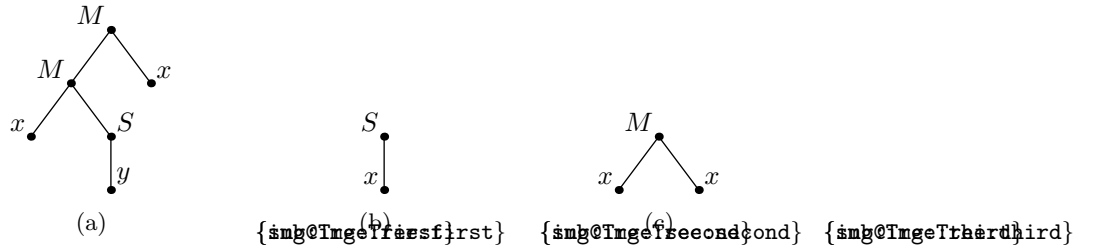
(zde bude doplněno: Takto definovaný strom však teoreticky může být nekonečný. Co s tím?) Strom  $t$  je tedy předpis pro „přejmenování“ uzlů prefixového stromu daného doménou  $U$ . Strom dle definice 4.3 je v korespondenci s pojmem „strom“ (resp. „kořenový strom“) z teorie grafů. Z tohoto důvodu si pro jednoduchost můžeme odpustit definici souvisejících pojmů z teorie grafů pro strom z definice 4.3. Můžeme tak stromy graficky zobrazovat, hovořit o jejich potomcích, podstromech, vnitřních a listových uzlech bez nutnosti formálního nadefinování.

{ex:Trees}

**Příklad 4.1.** 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  $\bar{U}_1 = \{11, 121, 2\}$ . Strom  $t_1 : U_1 \rightarrow (T, N)$  nad  $(T, N)$  pak může vypadat například takto:

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





Obrázek 2: Stromy k příkladu 4.1

{img:Tree}

Grafické znázornění stromu  $t_1$  je na obrázku 2a. Další ukázky stromů jsou na zbylých podobrázcích obrázku 2.

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>6</sup>.

**Definice 4.4** (Pseudoterm). Označme  $D_{(N,T)}^p$  nejmenší podmnožinu  $(N \cup T \cup \{(\cdot, \cdot)\})^*$  splňující následující podmínky<sup>7</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  $A(t_1 \dots t_n) \in D_{(N,T)}^p$

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

**Poznámka 4.1.** 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.

**Příklad 4.2.** 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 4.1.** 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 4.2.** Ke každému pseudotermu  $p(t) \in D_{(N,T)}^p$  existuje odpovídající strom  $t$ .

*Důkaz.* Opět dokážeme strukturálně:

<sup>6</sup>Oproti termům predikátové logiky mají však jiný pohled na nulární funktory, které u pseudotermu neexistují

<sup>7</sup>Předpokládáme, že symboly závorek,  $( \ a \ )$  nejsou součástí  $N \cup T$

- 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  $\bigcup_{i \leq m} \{iw \mid w \in \text{domain}(t_i)\} \cup \{\epsilon\}$  a

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

□

**Příklad 4.3.** Pseudoterm  $t_4^p$  z předchozího příkladu odpovídá stromu na obrázku 2a a pseudoterm  $t_3^p$  stromu 2c (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.

580

**Definice 4.5** (Fuzzy jazyk stromů). Fuzzy množinu  $\tau$  nad  $D_{(N,T)}$  nazvěme fuzzy jazyk stromů.

### 4.3 Fuzzy tree automat a jazyk jím rozpoznávaný

Začneme definicí fuzzy tree automatu.

**Definice 4.6** (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
- $N$  je konečná množina neterminálních symbolů uzlů taková, že  $N \cap T = \emptyset$
- $\mu : (N \cup T) \rightarrow \{f \mid f : (\mathcal{Q} \cup \{\epsilon\}) \times Q \rightarrow [0, 1]\}$  je fuzzy přechodová funkce, kde  $\mathcal{Q}$  je konečná podmnožina  $Q^+$ . Pro  $X \in N$  je  $\mu(X) = \mu_X$ , kde  $\mu_X$  je zobrazení z  $\mathcal{Q} \times Q$  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.

590

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  $c$ “.

600

**Příklad 4.4.** 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 1. Pak  $A = (Q, T, N, \mu, F)$  je fuzzy tree automatem.

$\mu_x$	$q_1$	$q_2$
$\epsilon$	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
$q_1 q_1$	0	0,3

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

Tabulka 1: Příklad přechodové funkce  $\mu$  fuzzy tree automatu

{tab:MuOfFuzTreAut}

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 jím odpovídajících pseudotermů) je fuzzy bezkontextový. Předtím je ale třeba ukázat, že fuzzy tree automaty skutečně přijímají fuzzy jazyky stromů.

**Definice 4.7** (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 stupňů příslušných podstromů.

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

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

**Definice 4.8** (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 4.5.** *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 2c 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 2a a 2a platí  $A(t_2) = 0,7$  a  $A(t_3) = 1$ .

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

**Věta 4.3.** *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 4.9** (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:*

- $Q = \{q_{t'} | t' \in sub(t)\}$  (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  $T = \{(t, 1)\}$ .

## 4.4 Buněčné fuzzy automaty

Buněčné (fuzzy) automaty jsou další z výpočetních modelů, které se svojí základní myšlenkou podobají (fuzzy) automatům. Oproti „klasickým“ (fuzzy) automatům 640 mají však značně zajímavější možnosti uplatnění. Proto jim v této práci bude věnována samostatná kapitola.

## 4.5 „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 je možné je považovat za zobecnění „klasických“ deterministických automatů. Dá se totiž říci, že se jedná o  $n$ -dimenzionální mřížku tvořenou instancemi téže konečného automatu.

650 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  $m$ .

660 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 4.10** (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  $1 \leq k \leq m^2$ .*

670 **Poznámka 4.2.** *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:

$$\mu(c, round(c)) = f(c, round(c))$$

kde  $f$  je funkce přiřazující buňce  $c$  s okolními buňkami  $round(c)$  nový stav. V praxi se nejčastěji používá sada tzv. IF-THEN pravidel.

**Příklad 4.6.** *Typickým příkladem buněčného automatu je tzv. Hra života (Game of Life) (např. [?]). Jedná se o jednoduchý simulátor živého organismu.*

680 *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*

{ex:GameOfLife}

4. Ve všech ostatních případech zůstává buňka buďto mrtvá nebo živá

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

**Definice 4.11** (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 4.12** (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 4.13** (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.*

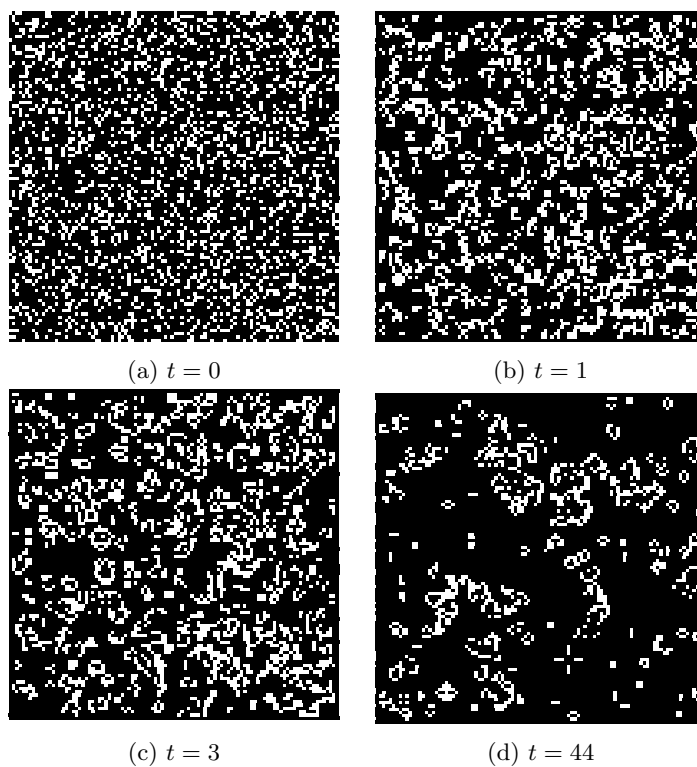
Vzhledem k tomu, že přechodová funkce buněčného automatu je deterministická, je zřejmě deterministický i její výpočet, tj. pro každou konfiguraci  $C$  existuje právě jedna konfigurace  $C'$ , pro kterou platí  $C \vdash C'$ . Relace  $\vdash$  tak generuje posloupnost konfigurací. Očíslujeme-li si tuto 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 konfiguraci  $i$ -té generace a samotné hodnoty  $i$  jako generace (hovoříme tak o nulté, první, druhé, ... generaci).

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 4.7.** *Ukázka výpočtu automatu  $A_{100}$  realizující Hru života je na obrázku 3. Černá barva symbolizuje mrtvou buňku, bílá pak živou.*



Obrázek 3: Několik generací výpočtu buněčného automatu „Hra života“ s náhodnou počáteční konfigurací

{img:GameOfLife}

## 4.6 Buněčné fuzzy automaty

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>8</sup> (přechodová funkce by nebyla deterministická) nebo nepřešla do žádného (přechodová funkce by nebyla úplná).

720

Vzhledem k tomu, že stejné chování budeme vyžadovat i u buněčného fuzzy automatu, „odstupňování“ přechodové funkce (zavední fuzzy přechodové funkce) by nemělo smysl<sup>9</sup>.

Lze se tak setkat s definicí buněčného fuzzy automatu jakožto buněčného automatu pracujícího s množinou stavů rovnu intervalu  $[0, 1]$ . Častěji však bývá buněčný fuzzy automat definován jako buněčný automat, jehož přechodová funkce není určena „klasickými“ IF–THEN pravidly, ale fuzzy IF–THEN pravidly.

730

**Příklad 4.8.** *Příkladem buněčného fuzzy automatu může být například následující automat. Množina stavů bude obsahovat přirozená čísla z intervalu  $[0, 150)$ . Mějme lingvistickou proměnnou  $\varsigma$  a její čtyři štítky  $\varsigma_L$ ,  $\varsigma_M$ , a  $\varsigma_H$ . 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  $q_{i,j}^{(t+1)} = L$

(zde bude doplněno: desetinná čísla s desetinnou čárkou, ne tečkou!!!!)

## 5 Užitečné techniky pro práci s fuzzy automaty

740

Fuzzy automaty či jim podobné modely zavedené v předchozích kapitolách jsou z pohledu praktických aplikací pouze teoretické modely. V reálných aplikacích je často nutné pracovat s těmito modely spíše jen jako s kostrou, od které je poté odvozen výsledný model nebo algoritmus.

V této kapitole budou některé ze základních technik, jak fuzzy automaty přiblížit reálným aplikacím, odprezentovány. Demonstrovány budou na nedeterministickém fuzzy automatu, nicméně není problém aplikovat je na jakýkoliv jiný typ fuzzy automatu.

### 5.1 Fuzzy automat bivalentního automatu

V některých situacích máme k dispozici „klasický“ bivalentní automat<sup>10</sup> a potřebujeme k němu sestavit odpovídající fuzzy automat. Odpovídající ve smyslu, že řetězce,

<sup>8</sup>Jak bude ukázáno, tento předpoklad lze obejít

<sup>9</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.

<sup>10</sup>Zde je vhodné připomenout, že automat můžeme získat například také konverzí regulárního výrazu nebo regulární gramatiky



750 které bivalentní automat přijímá (zamítá) musí fuzzy automat přijímat ve stupni 1 (0).

Uvažujme automat  $A = (Q, \Sigma, \mu, I, F)$  z definice 2.1. Pak fuzzy automat  $A' = (Q, \Sigma, \mu', \sigma, \eta)$  odpovídající tomuto automatu je následující:

- množina stavů  $Q$  a abeceda  $\Sigma$  zůstávají stejné
- přechodová funkce  $\mu'$  je dána předpisem (pro všechny  $q, q' \in Q$  a  $x \in \Sigma$ ):

$$\mu'(q, x, q') = \begin{cases} 1 & \text{pokud } (q, x, q') \in \mu \\ 0 & \text{pokud } (q, x, q') \notin \mu \end{cases}$$

- množina počátečních stavů  $\sigma(q) = 1$  pro všechny stavy  $q \in I$ , jinak 0
- množina koncových stavů  $\eta(q) = 1$  pro všechny stavy  $q \in F$ , jinak 0

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

## 5.2 Fuzzy automat rozpoznávající $\omega$

760 Další ze základních pomůcek pro práci s fuzzy automaty je automat, který v jednotkovém stupni rozpoznává pouze jeden konkrétní řetězec nad nějakou známou abecedou  $\Sigma$ . Tento řetězec budeme v této a některých dalších podkapitolách nazývat „vzorový“ a značit  $\omega$ .

Pro konstrukci automatu rozpoznávající řetězec omega bychom mohli použít následující postup:

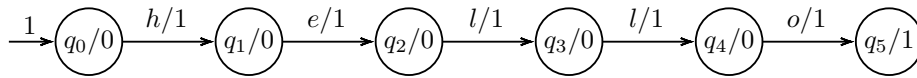
1. Známe řetězec  $\omega$  nad abecedou  $\Sigma$
2. Označme  $L(\omega)$  jako jednoprvkový fuzzy jazyk obsahující řetězec  $\omega$  ve stupni 1
3. Jazyk  $L(\omega)$  je konečný a tudíž i regulérní. (zde bude doplněno: ozdrojovat, nebo se odkázat do úvodu, platí to i pro fuzzy jazyky?)
- 770 4. Můžeme tak k němu sestavit nedeterministický fuzzy automat  $A(\omega)$ , který jazyk  $L(\omega)$  rozpoznává.

My však použijeme jinou, intuitivnější, úvahu. Automat bude v každém kroku konzumovat symboly ze vstupního řetězce a porovnávat je se symboly vzorového řetězce na odpovídajících pozicích. Pokud 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. Formálně pak:

780 **Definice 5.1** (Fuzzy automat rozpoznávající  $\omega$ ). *Mějme řetězec  $\omega$  nad abecedou  $\Sigma$  délky  $n$ . Fuzzy automat rozpoznávající  $\omega$  je pak automat  $A(\omega) = (Q, \Sigma, \mu, \sigma, \eta)$  kde*

- $\sigma(q_0) = 1$  a  $\sigma(q_i) = 0$  pro všechna  $i > 0$
- $\eta(q_n) = 1$  a  $\eta(q_i) = 0$  pro všechna  $i < n$

{def-FuzzAutRozpOme}



Obrázek 4: Automat rozpoznávající **hello**

{diag-AutRozpHell}

$$\bullet \mu(q_k, a_k, q_{k+1}) = \begin{cases} 1 & \text{pokud je } a_k \text{ } k\text{-tý symbol řetězce } \omega \\ 0 & \text{jinak} \end{cases}$$

**Příklad 5.1.** Příklad fuzzy automatu rozpoznávající řetězec  $\omega = \mathbf{hello}$  se nachází na obrázku 4.

### 5.3 Podobnost symbolů

V této a následujících podkapitolách budou popsány základní techniky, jak chování automatu upravit tak, aby více odpovídalo reálným aplikacím. Například, máme fuzzy automat, který rozpoznává určitý fuzzy jazyk. Řetězce, které se určitým způsobem „podobají“ řetězcům toho jazyka, však tento jazyk obvykle obsahuje v nulovém stupni. Vzniká zde tedy snaha přizpůsobit automat tak, aby i takovéto řetězce v určitém nenulovém stupni přijímal.

Základním nástrojem, jak toho dosáhnout je pomocí podobnosti symbolů. Některé symboly mohou totiž vykazovat určitou formu podobnosti a tudíž možné zaměnitelnosti. Například znaky ASCII tabulky, které se liší v jednom bitu. Podobnost symbolů však nemusí být klasická „bivalentní“ relace, ale je výhodné použít fuzzy relaci. V takovém případě by podobnost znaků ASCII tabulky mohla být formulována následovně: „znaky  $x$  a  $y$  se liší o  $\delta$  bitů ve stupni  $1 - \delta/8$ “.

Formálně se tedy jedná o fuzzy relaci  $\simeq : \Sigma \times \Sigma \rightarrow [0, 1]$ . Tato relace zjevně musí splňovat symetrii ( $\simeq(x, x) = 1$ ) a reflexivitu ( $\simeq(x, y) = \simeq(y, x)$ ) (pro všechny  $x, y \in \Sigma$ ). (zde bude doplněno: a tranzitivita?) Fuzzy relace podobnosti symbolů pak může být součástí automatu. V takovém případě dojde ke modifikaci výpočtu automatu takovým způsobem, že ve značení ?? a definici 2.8 bude nahrazen výraz  $\mu(q, x, q')$  následujícím výrazem

$$\bigvee_{y \in \Sigma} \mu(q, y, q') \otimes \simeq(x, y)$$

Tento zápis říká, že při vstupu  $x$  automat má zkontrolovat všechny ostatní symboly  $y$ , které by mohly být se symbolem  $x$  podobné a v takovém případě realizovat přechod přes symbol  $y$ . Takto modifikovaný automat tak umožňuje namísto symbolu  $x$  „rozpoznat“ jiný symbol. Můžeme tak říci, že automat umožňuje ve vstupu náhradu symbolu.

Další informace a příklady k této technice jsou k nalezení v [?]. V [?] je pak tato technika prezentována v mírně pozměněné formě. Symboly nahrazují tzv. fuzzy symboly, což jsou fuzzy množiny symbolů danému symbolu podobné.

### 5.4 Editační operace, deformovaný automat

Další technikou pro modifikaci chování automatu je využití editačních operací. Tato technika byla přejata z [?]. Základní idea této techniky spočívá v trojici jednoduchých editačních operací, jejíž složením jsme schopni popsat požadovanou

transformaci vstupu na automatem reprezentovaný vzor. „Množství“ transformace pak udává podobnost pozorovaného a vzorového řetězce.

Editační operace nám určují elementární (na úrovni symbolů) transformace z řetězce na jiný řetězec. Jak bylo zmíněno, tyto elementární transformace jsou tři, a to:

1. vložení symbolu
2. odstranění symbolu
3. nahrazení symbolu jiným symbolem

820

Aplikováním posloupnosti těchto operací na vstupní řetězec tak můžeme obdržet řetězec, který odpovídá vzoru a je tak automatem přijímán. Posloupnost editačních operací budeme nazývat vyrovnání řetězců.

Pracujme nyní s abecedou  $\Sigma$ . Editací operace budeme zapisovat jako uspořádané dvojice  $(a, b) \in (\Sigma \cup \{\epsilon\})^2 \setminus (\epsilon, \epsilon)$ . Vložení symbolu  $x$  značme  $(\epsilon, x)$ , odstranění symbolu  $x$  pak  $(x, \epsilon)$  a náhradu symbolu  $x$  symbolem  $y$  pak  $(x, y)$ . Pak všechny tři posloupnosti editačních operací jsou vyrovnáním řetězce **ahoj** na řetězec **hello**:

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

Označme  $E$  jako množinu všech editačních operací, tj.  $E = (\Sigma \cup \{\epsilon\})^2 \setminus (\epsilon, \epsilon)$ . Budeme-li chtít pro každou editační operaci stanovit stupeň, v jakém má být provedena, použijeme fuzzy relaci  $P : E \rightarrow [0, 1]$ . Následovat bude popis tzv. „deformace“ automatu, tedy procedury, která pro fuzzy relaci editačních operací  $R$  a fuzzy automat  $A$  zkonstruuje fuzzy automat  $A'$ , který editační operace reflektuje.

830

**Definice 5.2** (Deformovaný automat). *Mějme binární fuzzy relaci  $R : E \rightarrow [0, 1]$  a nedeterministický fuzzy automat  $A = (Q, \Sigma, \mu, \sigma, \eta)$ . Pak jako deformovaný automat označme nedeterministický fuzzy automat  $A' = (Q, \Sigma, \mu', \sigma, \eta)$  s  $\epsilon$ -přechody, kde  $Q$ ,  $\Sigma$ ,  $\sigma$  a  $\eta$  zůstávají stejné a fuzzy přechodová funkce  $\mu'$  je dána:*

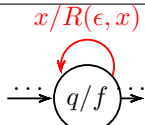
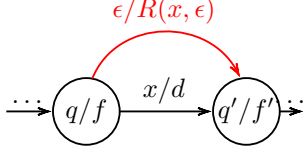
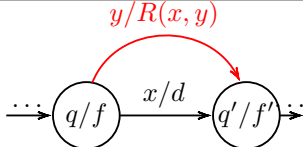
{def-AutRozpCall}

$$\begin{aligned} \mu' = & \{(q, x, q', d) | (q, x, q', d) \in \mu\} && (\text{původní přechody}) \\ & \cup \{(q, y, q, R(\epsilon, y)) | q \in Q, y \in \Sigma\} && (\text{vložení symbolu}) \\ & \cup \{(q, \epsilon, q', R(x, \epsilon)) | (q, x, q', d) \in \mu\} && (\text{odstranění symbolu}) \\ & \cup \{(q, y, q', R(x, y)) | (q, x, q', d) \in \mu\} && (\text{nahrazení symbolu}) \end{aligned}$$

Význam jednotlivých složek přechodové funkce (deformací) je názorně zobrazen v tabulce 2. Příklad deformovaného automatu je na obrázku 5. (zde bude doplněno: Okomentovat ten příklad!)

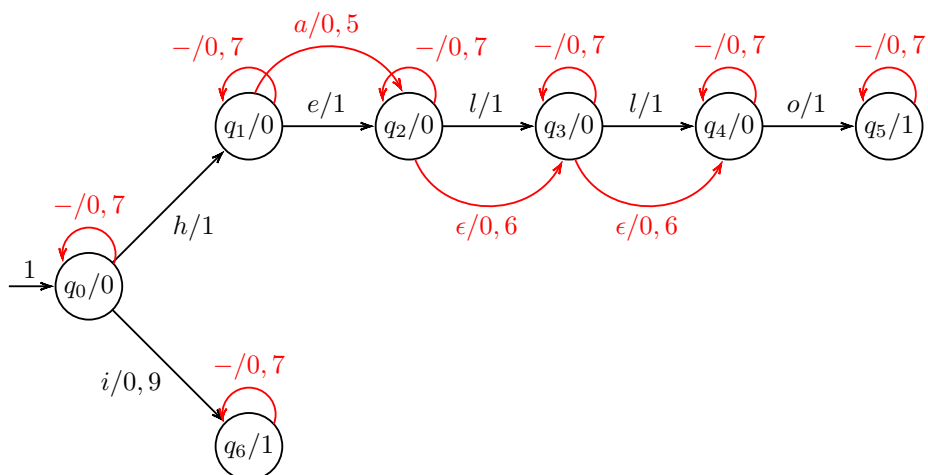
Je vidět, že pomocí editačních operací jsme schopni popsat libovolnou transformaci. Deformovaný automat tak může být schopen rozpoznat v nenulovém stupni jakoukoliv transformaci řetězce, který původní automat přijímal. Je také

840

Editační operace	Deformace (nový přechod v automatu)
Vložení symbolu $y$	
Odebrání symbolu $x$	
Narazení symbolu $x$ symbolem $y$	

Tabulka 2: Deformace automatu. Černé jsou znázorněny přechody původního automatu, červeně pak nově přidané

{tbl:DefAutDef}



Obrázek 5: Ukázka deformovaného automatu (červené přechody byly přidány deformací)

{img:DefAut}

vidět, že deformovaný automat nedokáže ošetřit pokročilejší transformace, jako je vložení právě jednoho symbolu, vložení symbolu na určité pozici či nahrazení podřetězce jiným podřetězcem. Takovéto deformace automatu je zpravidla potřeba provést ručně. Další informace ohledně editačních operací a deformovaných automatů lze nalézt např. v [?].

## 5.5 Učící se fuzzy automaty

Učící se automat je koncept pro propojení fuzzy automatů a strojového učení. Strojové učení je moderní přístup pro řešení obtížných úkolů. Návrh či přizpůsobení výpočetního modelu, který perfektně modeluje data z reálného světa je typický takovýto úkol. Takovým modelem může být i fuzzy automat. Velmi často se můžeme setkat s tím, že navržený fuzzy automat zcela nekoresponduje se vstupními daty (a jednoduché techniky pro ošetření těchto odchylek nebyly dostatečné).

Důležitým předpokladem pro strojové učení je velké množství vstupních, tzv. trénovacích, dat. Strojové učení totiž data obvykle zpracovává na značně nízké úrovni, takže toto nízké množství informace je třeba kompenzovat velkým množstvím vstupů.

V této kapitole bude prezentován koncept učícího se fuzzy automatu. Jedná se o všeobecný koncept techniky, jak přizpůsobit chování automatu reálným datům. Praxe ukazuje, že učící automaty tento problém dokáží velmi efektivně řešit. Setkat se s nimi můžeme např. v [?], [?], [?], [?], [?], [?], [?] nebo [?]. Vzhledem k tomu, že konkrétní výpočty, resp. algoritmy pro učení automatu bývají poměrně komplikované, spokojíme se pouze s obecným popisem principu.

Učení fuzzy automatů je obvykle založeno na tzv. shlukování. Mějme fuzzy automat  $A$  rozpoznávající jazyk  $\mathcal{L}(A)$ . Tento jazyk tvoří shluky. Shluk je podmnožina jazyka  $\mathcal{L}(A)$ , jejíž řetězce jsou si navzájem podobnější, než s řetězci ostatních shluků. Nyní si vezmeme libovolný řetězec  $\omega$ , který do jazyka  $\mathcal{L}(A)$  nepatří. Pak můžeme najít řetězec  $\omega'$  z jazyka  $\mathcal{L}(A)$ , který je řetězci  $\omega$  určitým způsobem nejpodobnější, tj. patřil by do jeho shluku. Pak můžeme automat (resp. části automatu, kterámají vliv na rozpoznávání řetězce  $\omega$ ) upravit tak, aby tento shluk obsahoval i řetězec  $\omega$ .

Konkrétní techniky jsou uvedeny v literatuře. *(zde bude doplněno: reálný ocitovaný příklad z některého z článků)*

## 5.6 Fuzzy automaty a neuronové sítě

Umělé neuronové sítě (dále jen „neuronové sítě“) jsou jednou z nejzákladnějších a nejpoužívanějších technik strojového učení. Neuronové sítě jsou inspirovány zpracováním informací pomocí nervových buněk. Neuronová síť je tvořena sítí tzv. neuronů, které jsou propojeny zpravidla v rozsáhlou síť. Tato síť obsahuje spoustu parametrů a vah, které jsou nastaveny tak, aby síť řešila patřičný problém. Při výpočtu se sítí šíří tzv. vzruchy (typicky reálná čísla), které jsou neurony přepočítávány. Více o neuronových sítích je dispozici např. v [?].

Způsoby, jak sestavit neuronovou síť existují dva (v praxi se však obvykle tyto přístupy kombinují). A to pomocí trénovacích dat nebo pomocí znalostní báze. Konstrukce pomocí trénovacích dat je podobná, jako v předchozí podkapitole. Tedy, že vstupní data se postupně nechávají vyhodnocovat neuronovou

sítí a ta je posléze přizpůsobována. Konstrukce pomocí znalostní báze je pak založena na kódování znalostní báze (IF–THEN pravidel) do vah neuronové sítě.

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

890 V jednodušších případech se jedná pouze o neuronovou síť, která realizuje jen přechodovou funkci automatu. Tedy neuronovou síť, jejímž vstupem je fuzzy stav, ve kterém se automat nachází, a symbol na vstupu a výstupem fuzzy stav, do kterého automat přejde. Existují však i způsoby, jak pomocí neuronové sítě reprezentovat celý automat. (zde bude doplněno: najít, ve kterém článku to bylo?)

Možnosti, jak takovou neuronovou síť sestavit, jsou stejné, jako při návrhu obecné neuronové sítě. Lze tedy využít jak konstrukci pomocí trénovacích dat, tak pomocí znalostní báze. Použití trénovacích dat spočívá ve vygenerování dostatečného množství kombinací fuzzy stavů a vstupních symbolů. Pomocí automatu je určen „výstupní“ fuzzy stav, který je spolu se vstupem předán neuronové síti k učení. Při použití znalostní báze jsou přechodová pravidla přepsána  
900 do fuzzy IF–THEN pravidel, jak bylo uvedeno v kapitole 3.6.

Stejně tak existují způsoby, jak konvertovat neuronovou síť zpět na fuzzy automat. Postup je popsán např. v [?], [?].

## 6 Obecně k rozpoznávání

### 6.1 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 6.1** (Úplný strom). *Úplný  $m$ -ární strom je takový strom, jehož každý  
910 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

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

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:

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

Nýl 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

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

920 pro všechna  $w \in \{q_1^i \mid 1 \leq i \leq m\}$ . 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 6.1.** *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  $L(A) = \{(t, c) \mid c = \mu_t(q_1)\}$  (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  $\mu_t = \mu_o = 1$ .

930 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  $\mu_I(w, q) = \mu_I(w, q_1) = \frac{k}{m} = \alpha_v \cdot \bigwedge_{j=1}^k \mu_{t_j}(q_1)$  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`.

## 6.2 Lexikální analyzátor

V [?] definují fuzzy lexikální analyzátor. Funguje na bázi klasického pattern matching, jen s informací, ve kterém koncovém stavu skončil. (zde bude doplněno: 940 mimořádně, v [?] je poměrně pěkně popsána teorie (deterministické a nedeterministické automaty, regexy i reg. jazyky).)

## 6.3 Rozpoznávání ručně psaného textu

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

## 6.4 Detekce překlepů

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

## 6.5 Korekce textu pomocí pravděpodobnostních automatů

950 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.

## 6.6 Fuzzy programy

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

## 6.7 Pravděpodobnost výskytu vzoru v řetězci

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  
960 hmotnostního spektrometru analyzovány a hmotnosti předány automatu).

## 6.8 A dále ještě ...

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

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

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.

## 970 6.9 Pravděpodobnostní rozpoznávání fonetické abecedy

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.

## 6.10 Parsování MIDI ze zvuku

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

## 6.11 Reprezentace zvuků

980 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.

## 6.12 Pattern matching přirozeného jazyka

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



### 6.13 Nejpravděpodobnější řetězec

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

### 990 6.14 PAttern matching signálů

V [?] a ještě někde se věnují tomu, že rozkouskovat signál umožní jeho pattern matching.

### 6.15 Pattern matching dvourozměrného signálu

V [?] je popisován způsob, jak rozpoznávat dvojrozměrný signál (mřížku signálů). Používá se k tomu model podobný pravděpodobnostním buněčným automatům. Na rozdíl od nich však tento model lépe podchycuje stavy a přechody mezi nimi na úkor větší složitosti.

Vstupní signály jsou diskrétně kvantovány v čase a následně i v úrovni. Dále je stanoveno zobrazení, které každé buňce s daným okolím přiřadí určitý stav automatu. Pro každou buňku je poté vytvořen pravděpodobnostní automat s přechodovou funkcí ve tvaru „Pravděpodobnost, že automat přejde do stavu  $q''$  za předpokladu, že nyní se nachází ve stavu  $q$  a na vstupu je vzorek odpovídající stavu  $q'$ , je rovna  $p$ “.

Autoři tuto techniku demonstrují na problému detekce poruch u materiálu. Měřený kus testovaného materiálu je zatěžován stále větším tlakem a postupně proměřován ultrazvukovými signály. Při namáhání materiálu dochází k jeho systematickému poškození. Změna chování materiálu je detekována ultrazvukovými senzory a předány automatu, který v nich rozpozná vznikající poruchu.

### 6.16 Parsování referencí

1010 V [?] pomocí automatu parsují reference z textu a rozpoznávají v nich strukturu. Reference v textu mají totiž pevnou strukturu. Např. dle normy ČSN ISO 690:2017 (*zde bude doplněno: ocitovat: <https://sites.google.com/site/novaiso690/>*) :

Tvůrce. *Název publikace*. Vedlejší názvy. Vydání. Další tvůrce. Místo: nakladatel, rok. Počet stran. Edice, číslo edice. ISBN.

Problém je však ten, že některé části (např. počet stran) jsou nepovinné. Dále také, např. rok vydání a počet stran jsou číselné údaje a může tak dojít ke zmatení. Běžný automat proto někdy může nalést více, než jednu shodu se vzorem. Proto používají „pravděpodobnostní“ (jedná se spíše o fuzzy než o pravděpodobnostní automat) automat, který dokáže určit nejbližší shodu.

### 1020 6.17 Pravděpodobnostní rozpoznávání přirozeného textu

V [?] zpracovávají přirozený jazyk. Vzhledem k podstatě gramatiky přirozeného jazyka využívají pravděpodobnostní tree automaty. (*zde bude doplněno: příklad gramatického stromu*) ?

## 7 Modelování a simulace

### 7.1 Automobilismus

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

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

### 7.2 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.

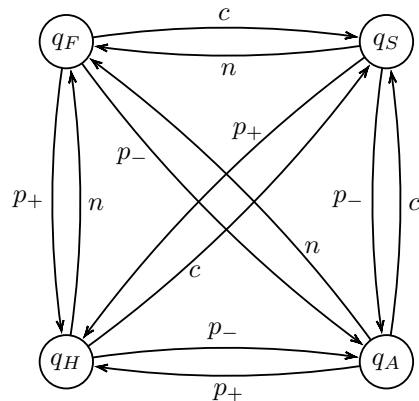
Technika využívá faktu, že každý spotřebič se může nacházet v různých stavech. Například vysoušeč vlasů se může nacházet v jednom z těchto tří stavů: vypnut, fouká studený vzduch, fouká teplý vzduch. Každému takovému stavu může být přiřazena konkrétní hodnota odběru el. energie. Pokud známe výkyvy spotřeby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

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.

### 7.3 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



Obrázek 6: 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}

- tah, po kterém se hráč ocitl v šachu (popř. matu)
- tah, kdy se nestalo nic z předchozích

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 6.

Na základě aktuálního fuzzy stavu (kombinace různých stupňů elementárních nálad) automatu je upravována mimika hráče a jeho obličej je zobrazován v počítačové hře. Ukázka některých herních situací je na obr. 7. 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.)

## 7.4 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 snažší 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 ohodno-



Obrázek 7: Ukázka výrazů ve tváři hráče při ztrátě pěšce (vlevo) a při sebrání dámy (vpravo). Převzato z [?].

{img:ChessEmosScreens}

cení. Např. pozitivní aktivace a negativní ohodnocení začí odpor či opovržení, negativní aktivace při neutrálním ohodnocení znamená strach.

Autoři používají lingvistické proměnné „negativní“, „nutrální“ 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 konstruuji 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 pozitivní“ přejde do stavu „aktivace je neutrální a ohodnocení je pozitivní“.

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.

## 7.5 Sledování a modelování aktivit počítačových programů

V [?] prezentují nástroj založený na pravděpodobnostních automatech, jehož úkolem je určovat nejpravděpodobnější aktivitu počítačového programu. Autoři mají k dispozici softwarový nástroj pro výpis nízkourovňového chování (vybraných elementárních akcí) počítačového programu (systémová volání, změny na programovém zásobníku). Sledováním těchto výpisů při známém chování programu mohou stanovit vzory chování programu. Pokud jsou tyto vzory vloženy do pravděpodobnostního automatu, je možné obdržet automat modelující chování programu. Lze tak například z automatu vyčíst, že pokud program otevřel soubor, pak následující elementární akcí bude s nejvyšší pravděpodobností alokace paměti na haldě.

## 7.6 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).

## 7.7 Sledování pohybu a aktivit osoby

1130 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 určují 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.

1140 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).

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

1150 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.

## 7.9 Predikce průběhu hry

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).

## 7.10 Pravděpodobnostní sledování aktivit člověka

1160 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).

## 7.11 Problém městského růstu (urban growth problem)

Problém městského růstu je problém z oblasti urbanistiky. Řešením tohoto problému je co nejpřesnější 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 standardní problém, tedy městský růst.

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: [?], [?] a [?].

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

Pojďme se nyní podívat, jak se buněčné fuzzy automaty pro řešení tohoto 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 parcely 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
- jak moc vysoké budovy stojí na parcele
- jak velké produkuje parcela znečištění ovzduší/hluku

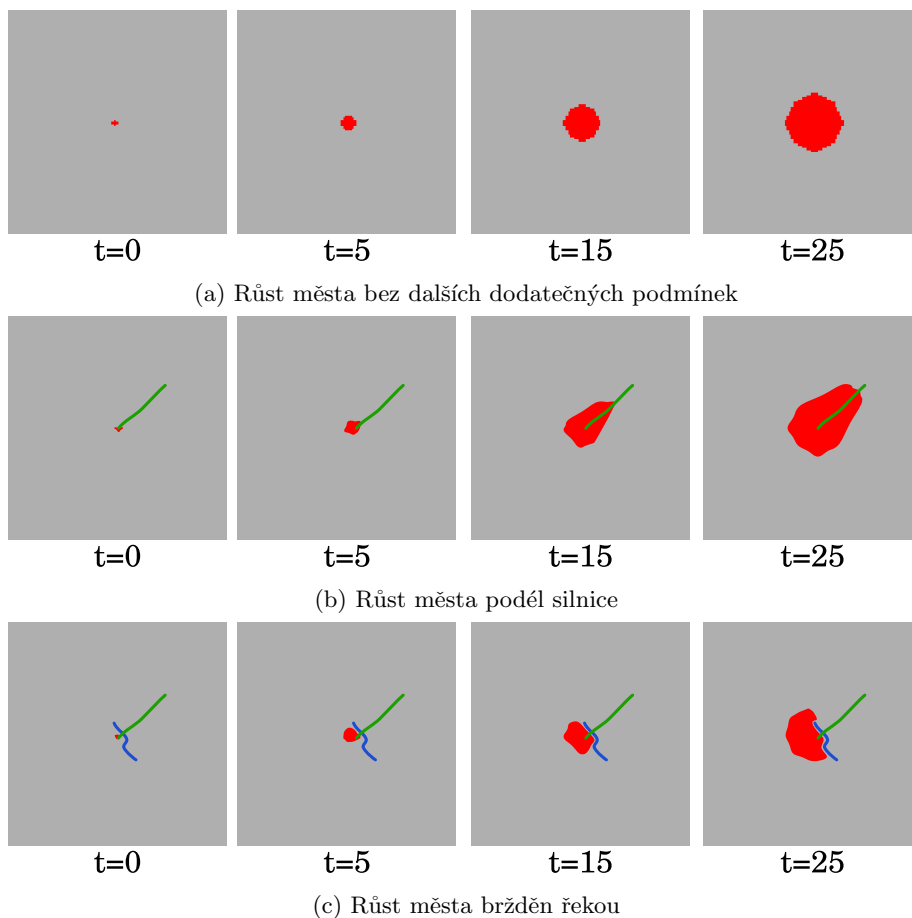
Nejčastěji se jako ukazatel používá informace o zastavěnosti parcely. Ta se totiž dá poměrně snadno stanovit s pomocí satelitních snímků sledované oblasti.

Dále je třeba získat ukazatele, které mají vliv na růst zástavby. Mezi takovéto 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, ...)
- 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:



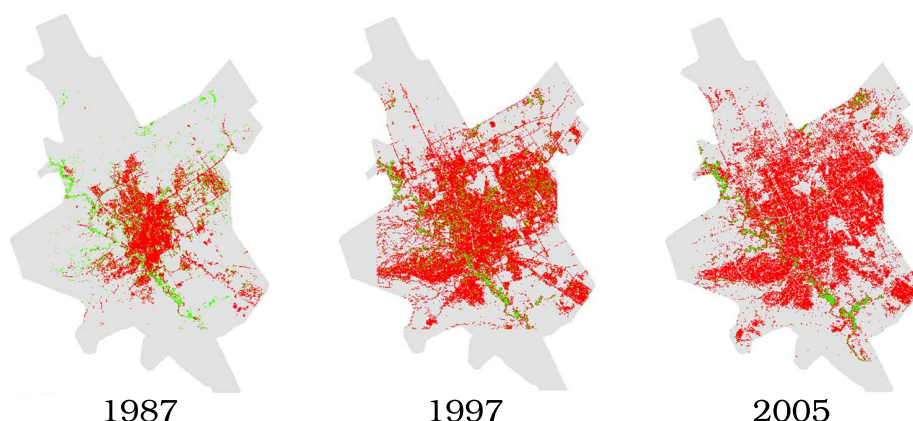
Obrázek 8: (převzato z [?], upraveno) Ukázky chování automatu při různých počátečních konfiguracích. Šedě jsou znázorněny prázdné parcely, červeně zástavba, zeleně hlavní silnice a modře řeky.

{img-VarTransRuls}

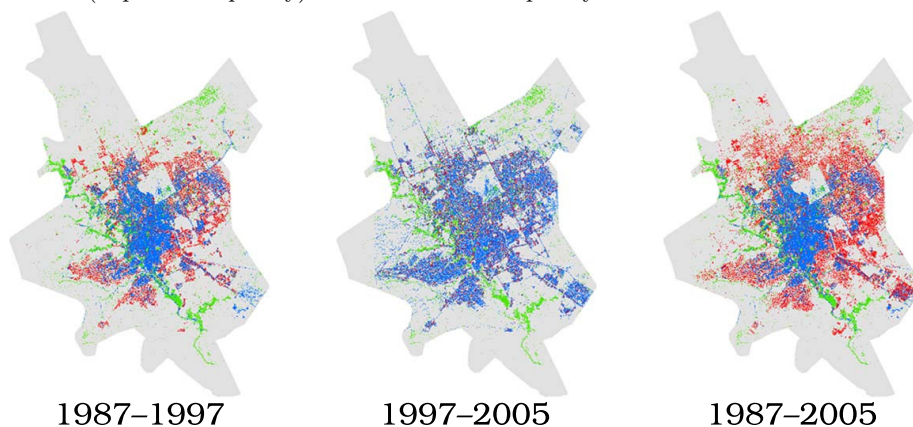
- Je-li vzdálenost parcely od centra města malá, pak růst zástavby bude velký
- Je-li vzdálenost od hlavní silnice velmi malá, pak růst zástavby bude malý a množství hluku bude velké
- Je-li vzdálenost od hlavní silnice velmi velká, pak růst zástavby bude malý
- Není-li lokalita atraktivní, pak růst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

Další ukázky pravidel jsou např. v [?], [?] 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



(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}

1220 už jsou patrné větší odlišnosti (simulace nevyprodukovala tak výrazný růst, jaký  
doopravdy nastal).

## 7.12 A co na to učící?

### 7.12.1 Něco

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  
1230 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.

### 7.12.2 Něco

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.

### 7.12.3 Něco

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.

### 7.12.4 Něco

V [?] navrhuji 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.

## 7.13 Modelování vytíženosti počítačové sítě

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

## 7.14 Sledování aspektů tvrzení

V [?] pomocí pravděpodobnostních automatů analyzují aspektová tvrzení (tj. zda-li je tvrzení míněno pozitivně, negativně, neutrálně nebo s rozporem). Stavy 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.

## 7.15 Generátor strojového člověka

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?)*

## 7.16 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:

- 1280 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  $A'_L$ .
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.

1290 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 odtrnění 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.

1300 Další z možných aplikací je určování dělitelnosti čísel. Uvažujme abecedu symbolů číslic, tj.  $\Sigma = \{0, \dots, 9\}$  a jazyk  $L \subseteq \Sigma^*$  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)

Podobnou techniku používají v [?], jen s pravděpodobnostními automaty.

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

## 8 Biologie a medicína

1310 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.

### 8.1 Rozpoznávání řetězců DNA

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

### 8.2 Simulace růstu mořských řas

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

### 8.3 Rozpoznávání vzorů v lékařských snímcích

Viz buněčné automaty a jejich jednoduché rozpoznávání vzorů.

1320 A pak ještě toto: [?].

### 8.4 Analýza zdravotního stavu pacienta

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

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).

1330 Důležitým předpokladem, který automat musí splňovat, je tzv. vlastnost pozdržení vrcholu *(zde bude doplněno: nadefinovat někde v úvodě?)* . Tato vlastnost říká, že výpočet automatu pro libovolný vstup nikdy nezkončí v prázdném fuzzy stavu, tj. výpočet se vždy bude muset nacházet v nenulovém stupni v alespoň některém stavu.

*(zde bude doplněno: Příklad?)*

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.

1340 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 automatu 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).

1350 *(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 (triangel, age, distance) vs. česky?)*

## 8.5 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 nezahobné buňky mají na snímcích obvykle symetričtější tvary a méně „skvrn“.

1360 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.
- Z ploch je na základě relace „plocha  $X$  obsahuje plochu  $Y$ “ zkonstruován strom ploch.
- Strom je následně zakódován do řetězce a rozpoznán fuzzy automatem.

1370 Stromy jsou do řetězců kódovány jako posloupnost čísel, kde každé číslo odpovídá počtu potomků jednotlivých uzlů stromu. Automat byl sestaven učním z veřejné databáze snímků. Autoři poukazují, že tato technika je na rozpoznávání účinnější, než vybrané z dalších technik.

*(zde bude doplněno: ukázka?)*

## 8.6 Analýza kardiogramu

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

## 8.7 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.

## 1380 8.8 Modelování buněčných sítí

V [?] používají pravděpodobnostní automaty na epigenetiku. Princip je podobný, jak u buněčných automatů, jen množina „buněk“ není pevně daná. Jinými slovy neuvažuje se mřížka živých a neživých buněk, ale množina jen těch živých, která se postupně rozrůstá (popř. odumírá). Každá buňka se nachází v některém stavu z množiny stavů (jako je např. „buňka se zrodila“, „buňka je

připravena k dělení“, „buňka je rodičovskou buňkou jiné buňky“). Přejížděcí pravidla poté v periodických časových kvantech provádějí přechody mezi těmito stavy. Například je-li buňka „připravena k dělení“, pak v dalším kroku provede dělení, tj. vytvoří novou buňku ve stavu „buňka se zrodila“ a buňka samotná přechází do stavu „buňka je rodičovskou buňkou jiné buňky“. Díky využití pravděpodobnostních automatů se takovéto modely mohou značně více přiblížit reálným buněčným sítím.

## 8.9 Modelování šíření infekční nemoci

V [?] používají systém podobný pravděpodobnostnímu buněčnému automatu pro modelování šíření infekčních nemocí. Každá buňka je buďto prázdná nebo se v ní nachází osoba. Osoba může být ve stavu „nenakažena“ nebo „nakažena“. Nachází-li se v okolí „nenakažené“ osoby  $q$  alespoň jedna nakažená osoba, pak s pravděpodobností  $p$  přejde při dalším časovém kroku do stavu „nakažena“, jinak zůstává ve stavu „nenakažena“.

Autoři navíc do modelu zanášejí pohyb jedinců, tj. že s určitou pravděpodobností se může osoba na buňce  $c$  přesunout na některou sousední buňku  $c'$  je-li neobsazena. Změnou parametrů systému (jednotlivých pravděpodobností) tak lze nasimulovat vymícení choroby nebo naopak vznik epidemie.

## 9 Zpracování obrazu

### 9.1 Konvoluce

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. 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:

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



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}

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

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

kde  $\epsilon > 1$  je parametr udávající agresivitu zvýrazňování a  $neighs_{i,j}$  je součet hodnot okolních buněk (viz příklad 4.6).

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.

## 9.2 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  $X$ .

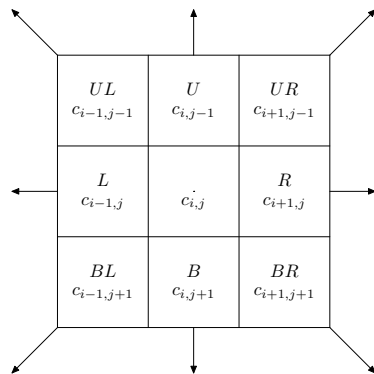
Autoři vycházejí z následující úvahy: Prochází-li buňkou  $c$  hrana ve směru  $X \in dim$ , pak má buňka  $c_X$  výrazně jinou barvu, než buňka  $c$  (viz obrázek 11b). Prochází-li buňkou  $c$  hrana v alespoň jednom směru  $X \in dim$ , pak můžeme říci, že buňka obsahuje hranu.

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

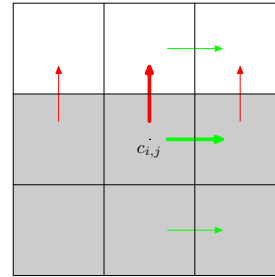
$$\Delta(c, c') = 1 - |c - c'|$$

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  $\epsilon_U(c) = 0$



(a) 8 směrů



(b) Ukázka hrany procházející buňkou. Ve směru  $U$  (červené šipky) prochází hrana, ve směru  $R$  (zelené šipky) neprochází.

$\{\text{sm@DmgeDfrensifEdgeEdges}\}$

- 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  $\epsilon = 0$

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

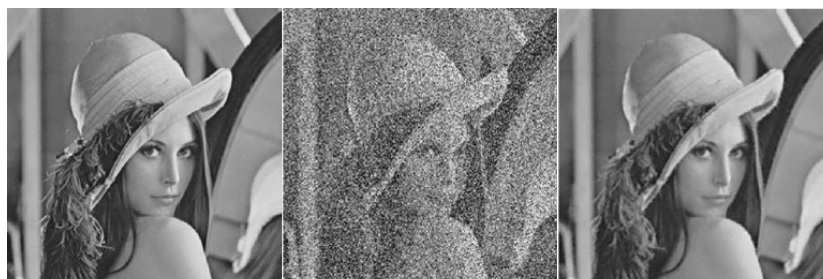
Dle [?] může být hodnota stupně „buňkou  $c$  prochází hrana“ použita jako parametr  $\alpha$  tzv. Gas Diffusion Modelu, jednu z technik zaostřování obrazu.

### 9.3 Ostraňování šumu

Odstraňování šumu je další častý problém, který je třeba při zpracování obrazů řešit. Pro studium technik odstraňování šumu se používá zašumnění tzv. impulzním šumem popř. šumem „sůl a pepř“. Zašumnění 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 jejich 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.



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

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

{img:Noises}

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  
1470 detekce šumu a jeho odstranění provádějí v jednom kroku.

## 9.4 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  
1480 doplněno: ref na kapitole) ) 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.  
1480 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 předpisem pro vzor, je tato technika nepoužitelná pro vzory větší než jednotky pixelů.

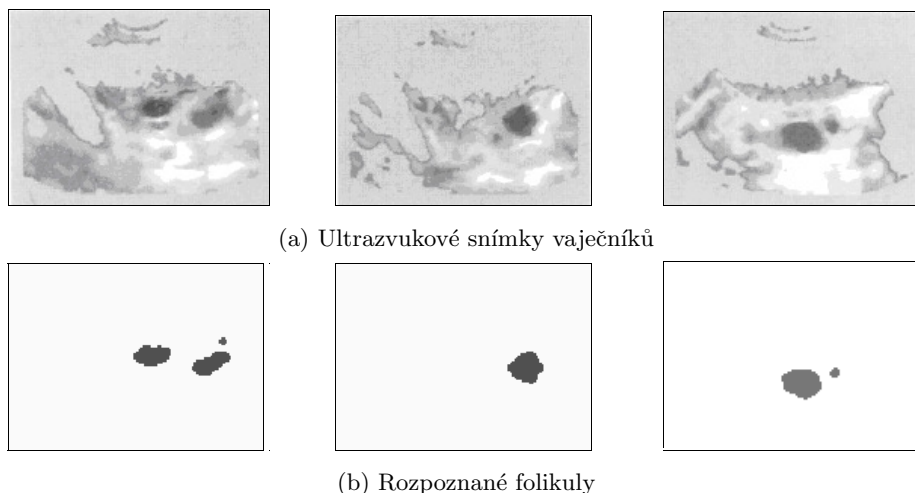
Zcela jiný přístup používají v [?]. Vzor nepovažují jako konkrétní kombinaci odstínů barev, ale jako část obrazu splňující určité vlastnosti.

Autoři metodu doporučují na vyhledávání vzorů v lékařských snímcích (např.  
1490 ultrazvuk, Röntgen). Techniku demonstrují na ultrazvukových snímcích vaječníků. Poukazují na to, že folikula<sup>12</sup> je na snímku tvořena jednolitou 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“.

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).

<sup>12</sup>Folikula je dutinka ve vaječníku, v níž probíhá zrání vajíčka. (zde bude doplněno: ocitovat: <http://lekarske.slovníky.cz/pojem/folikul>)





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

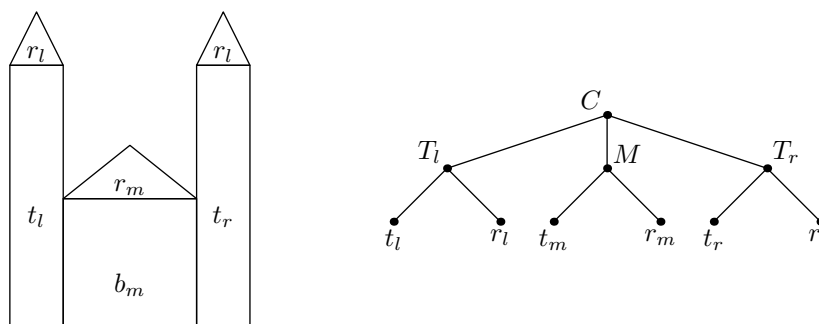
1500 Přechodová funkce první automatu je poměrně složitá, pracuje s 5 stupni šedi a třemi stupni kandidatury, takže zde nebude rozebírána. Druhý automat pak funguje elementárně. Buňky, jež nejsou označeny jako dostatečné kandidáty na folikuly, jsou odstraněny, ostatní ponechány.

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

## 9.5 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 podobjektů.

1510 **Příklad 9.1.** Na obrázku 14 je vyobrazen složený geometrický útvar vyobrazující „budovu kostela“ a jemu odpovídající strom.



Obrázek 14: Příklad složeného geometrického tvaru a jeho stromu

V příkladu, který autoři uvádějí, konstruuji 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 kterým 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.

1520 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 9.1** (Obdélník). Mějme čtyřúhelník  $g = ABCD$ . Pak tento čtyřúhelník je obdélník, pokud platí

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

**Definice 9.2** (Čtverec). Mějme obdélník  $g = ABCD$ . Pak tento obdélník je čtverec, pokud

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

**Definice 9.3** (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:

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

1530 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 9.4** („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 9.5** („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 9.6** („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 9.7** (Vztah „být nad“<sup>13</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.

1540 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 9.2.** *Uvažujme složený geometrický útvar  $C$  z obrázku 14. Pak automat  $A'$  bude nějak vypadat. (zde bude doplněno: domyslet to nějak!) .*

1550 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!)

1560

<sup>13</sup>Zde si dovoluujeme značné zjednodušení. Vztah „být nad“ by měl být popsán například s využitím porovnávání y-ových souřadnic bodů.

## 9.6 Detekce požárů

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

1570 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  $q_V L$ , 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 (svit, pohyb v určitém směru, vlnění) spočtených z předchozích snímků. Přejížděcí 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  $q_{VH}$ “. Přejížděcí funkci pak navrhli statistickým pozorováním známých videosekvencí s požáry<sup>14</sup>.

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: vložit obrázek s výsledkem?)*

## 9.7 A co na tu učící?

### 1580 9.7.1 Něco

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

---

<sup>14</sup>Dá se říci, že automat vznikl pomocí učení s učitelem