

# Aplikace fuzzy a pravděpodobnostních automatů

Martin Jašek

12. září 2016 — ???

## Obsah

<b>1</b>	<b>Úvod</b>	<b>4</b>
<b>2</b>	<b>Úvodní pojmy</b>	<b>4</b>
	2.1 Fuzzy množiny a fuzzy logiky . . . . .	4
	2.2 Pravděpodobnostní počet . . . . .	6
	2.3 Fuzzy vs. pravděpodobnostní přístup . . . . .	7
10	2.4 Popis v přirozeném jazyce . . . . .	7
	2.5 Základní pojmy z teorie automatů . . . . .	9
<b>3</b>	<b>Definice fuzzy automatu</b>	<b>10</b>
	3.1 Koncept automatu . . . . .	10
	3.2 Nedeterministický bivalentní automat . . . . .	11
	3.3 Nedeterministický fuzzy automat . . . . .	11
	3.4 Reprezentace nedeterministického fuzzy automatu . . . . .	12
	3.5 Výpočet nedeterministického fuzzy automatu . . . . .	12
	3.6 Pravděpodobností automat . . . . .	15
<b>4</b>	<b>Varianty fuzzy automatů</b>	<b>16</b>
20	4.1 Deterministický fuzzy automat . . . . .	16
	4.2 Nedeterministický fuzzy automat s $\epsilon$ -přechody . . . . .	17
	4.3 Zobecněné automaty . . . . .	17
	4.4 Fuzzy automat s výstupem . . . . .	17
	4.5 Stavový stroj . . . . .	17
	4.6 Událostmi řízený fuzzy automat . . . . .	18
	4.7 Zásobníkový fuzzy automat . . . . .	18
<b>5</b>	<b>Další modely podobné fuzzy automatům</b>	<b>18</b>
	5.1 Fuzzy tree automaty . . . . .	18
	5.2 Stromy a pseudotermy . . . . .	19
30	5.3 Fuzzy tree automat a jazyk jím rozpoznávaný . . . . .	21
	5.4 „Bivalentní“ buněčný automat . . . . .	23
	5.5 Buněčné fuzzy automaty . . . . .	25

	<b>6 Užitečné techniky pro práci s fuzzy automaty</b>	<b>27</b>
	6.1 Fuzzy automat bivalentního automatu . . . . .	27
	6.2 Fuzzy automat rozpoznávající $\omega$ . . . . .	28
	6.3 Podobnost symbolů . . . . .	28
	6.4 Editací operace, deformovaný automat . . . . .	29
	6.5 Učící se fuzzy automaty . . . . .	31
	6.6 Fuzzy automaty a neuronové sítě . . . . .	32
40	<b>7 Rozpoznávání, klasifikace a korekce textových dat</b>	<b>32</b>
	7.1 Výpočet podobnosti řetězců . . . . .	33
	7.2 Klasifikace a korekce textových řetězců . . . . .	33
	7.3 Detekce překlepů . . . . .	34
	7.4 Fuzzy lexikální analyzátor . . . . .	34
	7.5 Pravděpodobnostní rozpoznávání přirozeného textu . . . . .	34
	7.6 Dvouúrovňové vyhledávání v dlouhém textu . . . . .	34
	7.7 Parsování referencí . . . . .	34
	<b>8 Další oblasti pro rozpoznávání</b>	<b>35</b>
	8.1 Rozpoznávání signálů . . . . .	35
50	8.2 Rozpoznávání dvourozměrného signálu . . . . .	36
	8.3 Rozpoznávání ručně psaného textu . . . . .	36
	8.4 Rozpoznávání gest . . . . .	36
	8.5 Práce se zvukem . . . . .	36
	8.6 Fuzzy programy . . . . .	37
	8.7 Metoda lisování dat . . . . .	37
	8.8 Detekce úplných $m$ -árních stromů . . . . .	38
	<b>9 Modelování a simulace</b>	<b>38</b>
	9.1 Automobilismus . . . . .	38
	9.2 Monitorování elektrických a počítačových sítí . . . . .	39
60	9.3 Teorie her, počítačové hry . . . . .	40
	9.4 Interakce s člověkem . . . . .	41
	9.5 Sledování pohybu a aktivit osoby . . . . .	42
	9.6 Průmyslové řídicí systémy, fuzzy kontroléry . . . . .	42
	9.7 Problém městského růstu . . . . .	43
	9.8 Další uplatnění . . . . .	44
	<b>10 Zpracování obrazu</b>	<b>44</b>
	10.1 Konvoluce . . . . .	47
	10.2 Hledání hran . . . . .	47
	10.3 Ostraňování šumu . . . . .	49
70	10.4 Rozpoznávání jednoduchých vzorů . . . . .	50
	10.5 Složené geometrické útvary . . . . .	50
	10.6 Detekce požárů . . . . .	53
	<b>11 Biologie a medicína</b>	<b>53</b>
	11.1 Rozpoznávání řetězců DNA . . . . .	53
	11.2 Biologické simulace . . . . .	54
	11.3 Analýza zdravotního stavu pacienta . . . . .	54
	11.4 Analýza lékařských snímků . . . . .	55

	11.5 Další aplikace . . . . .	56
	<b>12 Implementace vybraných problémů</b>	<b>56</b>
80	12.1 Základní informace . . . . .	56
	12.2 Korekce překlepů . . . . .	57
	12.3 Rozpoznávání ručně psaného textu . . . . .	57
	12.4 Detekce úplných $m$ -árních stromů . . . . .	58
	12.5 Simulace spotřeby elektrického produktu . . . . .	59
	12.6 Zpracování obrazu . . . . .	59
	12.7 Metoda lisování dat . . . . .	59

# 1 Úvod

Fuzzy automaty jsou výpočetní modely, které propojují teorii automatů s fuzzy množinami. Obdobně, pravděpodobnostní automaty spojují teorii automatů s pravděpodobnostním počtem.

Všechny tyto tři oblasti matematiky (resp. informatiky) jsou v současné době velmi hluboce prozkoumány. Především fuzzy množiny jsou tak dnes stále více propojovány s dalšími oblastmi informatiky, jako jsou například řídicí systémy[?], formálně-konceptuální analýza[?] nebo právě teorie automatů.

Motivací pro propojení teorie automatů s fuzzy množinami (popř. pravděpodobnostním počtem) je vcelku jasná. Jak fuzzy množiny, tak pravděpodobnostní počet, zavádějí (resp. poměrně elegantně dokáží popsat) různé formy nepřesnosti, neurčitosti, nejistoty či vágnosti. Přesně s těmito pojmy se setkáváme v běžném životě. Provázání fuzzy množin a pravděpodobnostního počtu s teorií automatů tak může sloužit jako určitý „můstek“, který nám usnadní nasazení automatů v praxi.

Úkolem této práce je prezentovat základy teorie fuzzy a pravděpodobnostních automatů, a především jejich možná uplatnění v praxi. Práce prezentuje vybrané aplikace fuzzy a pravděpodobnostních automatů, tematicky rozdělených dle oblastí, do kterých spadají.

Následuje zavedení základních pojmů z oblasti fuzzy množin, pravděpodobnostního počtu a teorie automatů. Poté budou formálně zavedeny pojmy „fuzzy automat“ a „pravděpodobnostní automat“ spolu s některými souvisejícími pojmy. Poté následují kapitoly věnované samotným aplikacím fuzzy a pravděpodobnostních automatů. Na závěr je popsána softwarová implementace vybraných problémů.

## 2 Úvodní pojmy

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

### 2.1 Fuzzy množiny a fuzzy logiky

Pomocí matematiky můžeme formalizovat svět kolem nás. Zatímco matematika je založena na přesnosti, většina okolního světa lze přesně popsat jen s těží. Při popisu okolního světa pravujeme mnohem častěji s vágními pojmy jako „málo“, „téměř“ či „trochu“ než s jednoznačným „ano“ a „ne“. 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. V případech jako je tento mohou fuzzy množiny přinést značné zpřehlednění.

Podobně jako například predikátová logika tvoří matematický nástroj pro popis logického uvažování, fuzzy množiny (případně fuzzy logiky od nich odvozené) jsou matematický nástroj pro práci s nepřesnými pojmy<sup>1</sup>. Oproti „klasické“ množině, která prvek buďto obsahuje nebo neobsahuje, fuzzy množině může prvek náležet v určitém stupni, který se nachází někde mezi „nenáležet vůbec“ a „náležet plně“. Obdobně, pravdivost tvrzení (tedy formule vyjádřené v

<sup>1</sup>Slovo „fuzzy“ v angličtině znamená „nejasný“, „rozmazaný“, „neostrý“

130 přirozeném jazyce) ve fuzzy logice nemusí být pouze pravdivé či nepravdivé, ale může nabývat libovolného stupně pravdivosti.

Fuzzy množiny jsou tedy matematickým nástrojem. Následuje jejich definice. Definice fuzzy množin a související pojmy jsou přejaty z [?].

**Definice 2.1** (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  $\rho_A(x)$ .*

140 V této práci se dopouštíme významného zjednodušení. Množina stupňů pravdivosti, kterou zde uvažujeme jako interval  $[0, 1]$ , může být mnohem obecnější. Takové zobecnění by však jednak vyžadovalo hlubší popis, který je u intervalu  $[0, 1]$  intuitivní. Navíc, použití intervalu  $[0, 1]$  je pro člověka přirozené, 0 odpovídá „nulové pravdivosti“ a 1 „plné pravdivosti“. V neposlední řadě, použití intervalu  $[0, 1]$  je výhodné také z implementace na počítači.

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

Nyní se podíváme na základní množinové operace nad fuzzy množinami.

**Definice 2.2** (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:*

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

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 a  $=$  jejich rovnost.

150 Kromě operací s fuzzy množinami ještě budeme pracovat s operacemi t-norma a t-konorma, což jsou binární operace na množině  $[0, 1]$ .

**Definice 2.3** (T-norma a t-konorma). *Binární operace  $\otimes$  (t-norma) na množině  $[0, 1]$  je asociativní, komutativní, monotonní operátor s neutrálním prvkem 1.*

*Binární operace  $\oplus$  (t-konorma) na množině  $[0, 1]$  je asociativní, komutativní, monotonní operátor s neutrálním prvkem 0 takový, že:*

$$a \oplus b = 1 - ((1 - a) \otimes (1 - b))$$

splňuje pro všechna  $a, b \in [0, 1]$ .

Operace t-norma a t-konorma si lze představit jako zobecnění disjunkce a konjunkce. V tabulce 1 jsou uvedeny nejpoužívanější tři t-normy a jim odpovídajícím t-konormy.

160 Množinu všech fuzzy množin nad univerzem  $U$  (tedy protějšek potenční množiny u „klasických“ množin) budeme značit  $\mathcal{F}(U)$ . Posledním z důležitých pojmů je fuzzy relace.

**Definice 2.4** (Fuzzy relace). *Jako fuzzy relaci na fuzzy množinách  $U_1, \dots, U_n$  nazýváme fuzzy množinu nad  $U_1 \times \dots \times U_n$  (tj. nad jejich kartézským součinem).*

\*

Název	t-norma	t-konorma
Gödelova	$a \otimes b = \min(a, b)$	$a \otimes b = \max(a, b)$
Produktová	$a \otimes b = ab$	$a \otimes b = a + b - ab$
Łukasiewiczova	$a \otimes b = \max(0, a + b - 1)$	$a \otimes b = \min(a + b, 1)$

Tabulka 1: Tři nejpoužívanější t-normy a jim odpovídající t-konormy

{tbl:Norms}

## 2.2 Pravděpodobnostní počet

Pravděpodobnost je další forma neurčitosti. Pozorujeme-li nějaký jev, pak pravděpodobnost vyjadřuje, jak moc je jisté, že tento jev nastane i v budoucnu.

Z pohledu aplikací pravděpodobnostních automatů pro nás bude pravděpodobnost – podobně, jako stupeň pravdivosti – reálný interval  $[0, 1]$  s některými dalšími vlastnostmi. Definice a vlastnosti pravděpodobností zde budou uvedeny ve zjednodušené podobě a jsou převzaty z [?].

**Definice 2.5** (Prostor jevů, jev). *Označme neprázdnou množinu  $\Omega$  jako prostor jevů, její prvky jako elementární jevy a její podmnožiny jako jevy.*

**Definice 2.6** (Pravděpodobnost elementárních jevů). *Pravděpodobnost (resp. pravděpodobnostní míra) elementárního jevu je zobrazení  $P : \Omega \rightarrow [0, 1]$ , (tj. zobrazení, které každému elementárnímu jevu  $\omega \in \Omega$  přiřazuje jeho pravděpodobnost  $P(\omega)$ ) takové, že  $\sum_{\omega \in \Omega} P(\omega) = 1$ .*

**Definice 2.7** (Pravděpodobnostní míra). *Mějme prostor jevů  $\Omega$  a pravděpodobnost  $P$  elementárních jevů z tohoto prostoru. Pak jako pravděpodobnost (pravděpodobnostní míra) jevu  $A \subset \Omega$  nazýváme zobrazení  $P : 2^\Omega \rightarrow [0, 1]$  dané následujícím předpisem:*

$$P(A) = \sum_{\omega \in A} P(\omega)$$

Libovolná pravděpodobnostní míra  $P$  pro každý prostor jevů  $\Omega$  a libovolné jevy  $A, B \subseteq \Omega$  dále splňuje následující vlastnosti:

1.  $P(\emptyset) = 0$  (tzv. nemožný jev)
2.  $P(\Omega) = 1$  (tzv. jistý jev)
3.  $P(\bar{A}) = 1 - P(A)$
4.  $P(A \cap B) = P(A)P(B)$
5.  $P(A \cup B) \leq P(A) + P(B)$
6.  $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

Dalším důležitým pojmem je podmíněná pravděpodobnost. Zjednodušeně řečeno, podmíněná pravděpodobnost nám říká, s jakou pravděpodobností nastal jev  $A$  jestliže víme, že nastal jev  $B$ .

**Definice 2.8** (Podmíněná pravděpodobnost). *Mějme pravděpodobnostní prostor  $\Omega$  a dva jevy  $A, B \subseteq \Omega$  z tohoto prostoru tak, že  $P(B) > 0$ . Podmíněná pravděpodobnost jevu  $A$  za podmínky, že nastal jev  $B$  je pravděpodobnost  $P(A|B) = \frac{P(A \cap B)}{P(B)}$ .*

**Příklad 2.1.** Uvažujme příklad s klasickou šestistěnnou hrací kostkou. Prostor jevů je  $\Omega = \{1, 2, 3, 4, 5, 6\}$ . Označme  $A = \{2, 4, 6\}$  jako jev „padlo sudé číslo“ a  $B = \{5, 6\}$  jako „padlo číslo větší než 4“.

Pravděpodobnost jevu  $A$  je  $\frac{1}{2}$ , pravděpodobnost jevu  $B$  je  $\frac{1}{3}$  a  $P(A|B) = \frac{1}{2}$  (tj. pravděpodobnost, že padlo sudé číslo za předpokladu, že padlo číslo větší než 4).

## 2.3 Fuzzy vs. pravděpodobnostní přístup

{subsec:FuzzyVsProb}

200 V předchozích dvou podkapitolách byly zavedeny základní pojmy z oblasti fuzzy množin 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, avšak každá jinou.

Popisujeme-li nějaký jev, pak stupeň pravdivosti udává, jak moc jev odpovídá jeho popisu. Například tvrzení „osoba  $X$  je vysoká“ může být pravdivé ve stupni 1, pokud je osoba vysoká 180 centimetrů a víc a ve stupni 0, 5, pokud osoba měří 160 centimetrů. Popisujeme tedy skutečnost, která je nám známá, avšak při jejím popisu používáme vágní pojem („vysoká“). Ohodnocení stupněm pravdivosti tak záleží na skutečné výšce osoby.

210 Naopak, tvrzení „zítra mi ujede autobus“ je popsáno přesně (nepoužívá žádné vágní pojmy), avšak nejistota je zde skryta v pozorovaném jevu. Zda-li tento jev nastane (a tvrzení je tudíž pravdivé) nebo nenastane nám není známo. Můžeme se však bavit o pravděpodobnosti, s jakou nastane.

Matoucí může být tvrzení „uživatel píše slovo  $X$ “. Na jednu stranu může být chápáno ve smyslu pravděpodobnosti. Tedy, že na základě určitých pozorování predikujeme s jakou pravděpodobností slovo, které uživatel píše, je slovem  $X$ . Na druhou stranu toto tvrzení může být chápáno tak, že víme, (se stoprocentní jistotou) jaké slovo uživatel píše, a pak můžeme studovat stupeň pravdivosti, v jakém je psané slovo slovem  $X$ .

220 Poněkud jinou situací je tvrzení „zítra bude pěkné počasí“. Toto tvrzení totiž obsahuje jak vágní pojem „pěkné“, tak nejistotu (nevíme, jaké bude počasí). Využívá tedy obou přístupů současně.

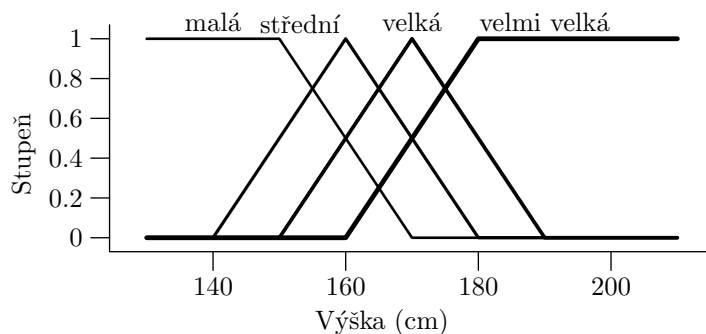
Při návrhu systému pracujícího s neurčitostí je tedy nutné se vždy zamyslet, jestli jeho neurčitost tkví spíše v nepřesném označení nebo naopak v nepřesné znalosti popisované situace.

**Poznámka 2.1.** *Existují snahy fuzzy a pravděpodobnostní přístup sjednotit. Takovýmto „stupněm“ se obvykle říká „váhy“. Systém pracující s vahami však nemá žádný logický základ a není tak vhodné jej používat v reálných aplikacích. Uplatnění však může najít pro teoretické studium společných vlastností jak pravděpodobností tak stupňů pravdivosti.*

## 230 2.4 Popis v přirozeném jazyce

Vzhledem k tomu, že účelem této práce je studovat uplatnění fuzzy a pravděpodobnostních automatů v praxi, je nutné seznámit se s nástroji pro propojení matematického světa (tj. fuzzy množin a pravděpodobnostního počtu) a popisu v přirozeném jazyce.

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



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

{fig:lingVarsMeansChart}

Pro popis stupňů pravdivosti by bylo možné používat „Tvrzení  $t$  je pravdivé ve stupni  $d$ “. Taková formulace však zní poněkud kostrbatě. Mnohem elegantněji lze podobná tvrzení vyjádřit pomocí tzv. lingvistických proměnných [?]. Každá lingvistická proměnná  $\alpha$  je tvořena univerzem  $U$  hodnot a množinou lingvistických štítků  $T$ . Pro každý štítek  $X \in T$  pak definuje jeho význam  $M(X)$  (významovou funkci), 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 2.2.** Mějme lingvistickou proměnnou  $\alpha$  s významem „výška osoby“. Univerzem hodnot jsou všechna nezáporná reálná čísla. Lingvistické štítky jsou „malá“, „střední“, „vysoká“ a „velmi velká“. Významové funkce jednotlivých štítků jsou vyobrazeny na obrázku 1.

Nyní mějme osobu  $X$ . 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 ve tvaru:

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

kde  $x_1, \dots, x_n, y$  jsou podvýroky<sup>2</sup>.

**Příklad 2.3.** 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  $\beta = y$

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

<sup>2</sup>Zapíšeme-li IF–THEN pravidlo jako formuli, obdržíme Hornovskou klauzuli. Můžeme tak určit pravdivost jejího důsledku  $y$  na základě pravdivosti předpokladů  $x_1, \dots, x_n$ .



**Příklad 2.4.** 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 štítky  $p_L$  („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 štítky  $v_O$  („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  $\alpha_v = v_O$*

## 2.5 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  $\varepsilon$ ). 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.  $\lambda \in \mathcal{F}(\Sigma^*)$ . \*

### 3 Definice fuzzy automatu

V této kapitole bude podrobně popsán a formálně definován „fuzzy automat“ a proces jeho výpočtu. Pro snazší ilustraci bude nejdříve popsán automat bivalentní a následně upraven do podoby fuzzy automatu. Popis bivalentního automatu je převzat z [?], popis fuzzy automatu z [?] (pokud není uvedeno jinak).

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

\*

#### 3.1 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 řadí například také Turingovy stroje [?].

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

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

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

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

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. To znamená, že automat musí pro libovolný řetězec určit, 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.

### 3.2 Nedeterministický bivalentní automat

Nedeterministický bivalentní automat je definován následovně:

{def-NedBivAut}

330 **Definice 3.1** (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.*

\*

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  $I$ .

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

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

350 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  $\mu^*(Q', w)$ “. \*

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

### 3.3 Nedeterministický fuzzy automat

360 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>. 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 3.2** (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.*

### 3.4 Reprezentace nedeterministického fuzzy automatu

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

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

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

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

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

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

**Příklad 3.1.** \*

### 3.5 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

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

fuzzy automat se této koncepcce drží, jen ji obohacuje o odstupňovanost. Tedy, že množina stavů, ve kterých se automat může nacházet je fuzzy množinou. Tutu množinu budeme nazývat fuzzy stav.

{def-FuzzStav}

**Definice 3.3** (Fuzzy stav). *Mějme nedeterministický fuzzy automat  $A$ . Pak jako fuzzy stav označujeme každou fuzzy podmnožinu jeho stavů, tj.  $\hat{Q} \in \mathcal{F}(Q)$ .*

**Poznámka 3.1.** *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 výpočet. Formální definice výpočtu je však mírně složitější a pro jeho zavedení bude potřeba pár dalších pojmů, které budou nyní uvedeny.

**Definice 3.4** (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 3.5** (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 stav  $\hat{Q} \circ R$  splňující pro každé  $p \in Q$ :  $(\hat{Q} \circ R)(p) = \bigoplus_{q \in Q} (\hat{Q}(q) \otimes R(q, p))$ .*

**Značení 1.** ?? Označme  $\mu[x](p, q) = \mu(p, x, q)$ .

{def-PreFunFuzzStav}

**Definice 3.6** (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  $\hat{\mu}(\hat{Q}, x) = \hat{Q} \circ \mu[x]$ .*

**Definice 3.7** (Výpočet nedeterministického fuzzy automatu). *Mějme nedeterministický fuzzy automat  $A$ . Každou posloupnost konfigurací  $(w_0, \hat{Q}_0), \dots, (w_m, \hat{Q}_m)$  splňující pro každé  $0 \leq i < m$*

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

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

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

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

{def-RozPreFunFuzzStav}

**Definice 3.8** (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:*

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

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

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

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

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

Můžeme tedy zapsat:

**Definice 3.9** (Řetězec přijímaný automatem). *Mějme nedeterministický fuzzy automat  $A$ . Pak řetězec  $w \in \Sigma^*$  je automatem  $A$  přijat ve stupni*

{def-RetPriAut}

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

**Poznámka 3.2.** 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  $\mu^*$  můžeme napsat (poznamenejme, že  $\mu^*(q, \epsilon, p_0) = 1$  pokud  $q = p_0$ , jinak 0):

$$\begin{aligned} \mu^*(q, a_0 \dots a_n, q') &= \\ &= \bigoplus_{p_n \in Q} \left( \dots \bigoplus_{p_0 \in Q} (\mu^*(q, \epsilon, p_0) \otimes \mu(p_0, a_0, p_1)) \dots \otimes \mu(p_n, a_n, q') \right) = \\ &= \bigoplus_{p_n \in Q} \dots \bigoplus_{p_1 \in Q} (\mu(q, a_0, p_1) \otimes \mu(p_1, a_1, p_2) \otimes \dots \otimes \mu(p_n, a_n, q')) = \\ &= \bigoplus_{(p_n, \dots, p_1) \in Q^n} \mu(q, a_0, p_1) \otimes \mu(p_1, a_1, p_2) \otimes \dots \otimes \mu(p_n, a_n, q') \end{aligned}$$

Poté může být (1) zapsána jako:

$$A(a_0 \dots a_n) = \bigoplus_{(q, p_n, \dots, p_1, q') \in Q^{n+1}} (\sigma(q) \otimes \mu(q, a_0, p_1) \otimes \mu(p_1, a_1, p_2) \dots \dots \otimes \mu(p_n, a_n, q') \otimes \eta(q'))$$

Tento zápis intuitivněji popisuje výpočet automatu. Tento zápis totiž můžeme chápat tak, že automat projde všechny  $(n+1)$  prvkové posloupnosti stavů  $q, q_n, \dots, q_1, q$  (tj. všechny možné cesty v grafu automatu) pro každou z nich spočítá stupeň, v jakém by byl automatem přijat a vybere tu s nejvyšším stupněm.

Vzhledem k tomu, že počet cest je roven  $|Q|^{n+1}$  a každá cesta je tvořena  $n+1$  stavy, automat při svém výpočtu musí navštívit  $|Q|^{n+1}(n+1)$  stavů. Časová složitost je exponenciální vzhledem k délce vstupního řetězce<sup>4</sup>.

<sup>4</sup>Výpočet však může být optimalizován. Pokud některý přechod není definován (tj. automat by jej realizoval v nulovém stupni) můžou být cesty, procházející tímto přechodem při výpočtu vynechány.

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

\*

### 3.6 Pravděpodobnostní automat

470 Pravděpodobnostní automat je opět jen určitým rozšířením „klasického“ automatu. Očekáváme od něj, že přechody mezi stavy budou prováděny s určitou pravděpodobností. Tedy, že nachází-li se automat v určitém stavu  $a$  na vstupu je určitý symbol, stav, do kterého automat přejde, bude vybrán náhodně. Přechodová funkce pak bude určovat, s jakou pravděpodobností do kterého stavu automat má přejít.

Je vidět, že automat se tedy pokaždé bude nacházet vždy pouze v jednom stavu. Pravděpodobnostní automat bude tedy přirozené uvažovat jako deterministický. Z tohoto důvodu bude mít automat jen jeden počáteční stav. Množina koncových stavů bude „klasická“ podmnožina stavů, stav tedy bude vždy buďto koncový nebo nekoncový.

480 Následuje definice pravděpodobnostního automatu. Byla převzata z [?].

**Definice 3.11** (Pravděpodobnostní automat). *Jako pravděpodobnostní automat označujeme strukturu  $(\Sigma, Q, \mu, q_0, F)$ , kde  $\Sigma$  je abeceda,  $Q$  je neprázdná množina stavů,  $q_0 \in Q$  je počáteční stav,  $F \subseteq Q$  je množina koncových stavů a  $\mu : Q \times \Sigma \rightarrow (Q \rightarrow [0, 1])$  je pravděpodobnostní přechodová funkce<sup>5</sup>, která každému  $q \in Q$  a  $a \in \Sigma$  přiřazuje pravděpodobnost  $P_{q,a} : Q \rightarrow [0, 1]$  takovou, že:*

$$\sum_{q' \in Q} P_{q,a}(q') = 1$$

Přechodovou funkci lze číst následujícím způsobem. Por libovolné  $(q, a, q', p) \in \mu$ : „jestliže se automat nachází ve stavu  $q$  a na vstupu je symbol  $a$ , do stavu  $q'$  přejde s pravděpodobností  $p$ “. Přejdeme nyní k popisu výpočtu automatu. Očíslujme si stavy  $Q = \{q_0, q_1, \dots, q_n\}$ . Dále označme  $A(a)$  jako čtvercovou matici velikosti  $n \times n$  symbolu  $a \in \Sigma$  tak, že:

$$B(a)_{i,j} = P_{a,i}(j)$$

Dále definujme matici  $B(\omega)$ , kde  $a_1 \dots a_m = \omega \in \Sigma^*$  jako

$$B(\omega) = B(a_1) \times \dots \times B(a_m)$$

kde operátor  $\times$  značí násobení matic. Každý prvek  $B(\omega)_{i,j}$  pak symbolizuje pravděpodobnost, s jakou automat přejde ze stavu  $q_i$  do stavu  $q_j$ , je-li na vstupu řetězec  $\omega$ . Pak  $\sum_{q_k \in F} A(\omega)_{0,k}$  je pravděpodobnost, že automat přejde z počátečního stavu ( $q_0$ ) do některého z koncových stavů.

<sup>5</sup>V literatuře (např. v [?]) se můžeme setkat s alternativní definicí. Přechodová funkce je definována jako podmíněná pravděpodobnost, že automat přejde do stavu  $q'$  za předpokladu, že se nachází ve stavu  $q$  a na vstupu je  $a$ .

**Definice 3.12** (Pravděpodobnost přijetí řetězce). *Mějme pravděpodobnostní automat  $A = (\Sigma, Q, \mu, q_0, F)$  a řetězec  $\omega \in \Sigma^*$ . Pak*

$$p_A(\omega) = \sum_{q_k \in F} B(\omega)_{0,k}$$

kde  $B(\omega)$  je matice popsána výše, je pravděpodobnost přijetí řetězce  $\omega$  automatem  $A$ .

Jazyk řetězců rozpoznávaných automatem je pak určen tzv. řezem (tj. minimální pravděpodobností, s jakou je řetězec přijímán).

**Definice 3.13** (Jazyk rozpoznávaný pravděpodobnostním automatem). *Pravděpodobnostní automat  $A = (\Sigma, Q, \mu, q_0, F)$  rozpoznává s řezem  $0 \leq \lambda < 1$  jazyk*

$$L_{A,\lambda} = \{\omega \in \Sigma^* \mid \lambda < p_A(\omega)\}$$

490 \*

## 4 Varianty fuzzy automatů

V této kapitole budou uvedeny další typy fuzzy automatů. 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 obvykle elementární, nebudou rozebírány do hloubky. Podrobnější informace o těchto automatech jsou k nalezení v literatuře.

**Poznámka 4.1.** *Vybranné varianty automatů se mohou kombinovat. Například automat může být současně deterministický, událostmi řízený i zásobníkový.*

### 4.1 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.

V klasické teorii automatů je nedeterministický automat považován za zobecnění deterministického. U fuzzy automatů tomu je přesně naopak. Deterministický fuzzy automat se obvykle definuje pouze jako speciální případ nedeterministického.

510 Podobně jako u nedeterministického automatu existuje několik různých definic. V [?] je definován jako deterministický bivalentní automat, jen koncové stavy jsou fuzzy množinou. 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 jeho přechodová funkce omezena tak, 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, časová složitost výpočtu nedeterministického fuzzy automatu je exponenciální vzhledem k délce vstupu. U deterministického automatu je tato složitost lineární. Obvykle však 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.

\*

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

Nedeterministický fuzzy automat s  $\epsilon$ -přechody je rozšíření nedeterministického fuzzy automatu. Toto rozšíření je realizováno pomocí tzv.  $\epsilon$ -přechodů. „Klasický“ přechod je automatem realizován, 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 [?].

## 4.3 Zobecněné automaty

V kapitole 2.3 byl rozebírán rozdíl mezi stupněm pravdivosti a pravděpodobností. Ve snaze zobecnit tyto dva přístupy na tzv. váhy vzniklo proto několik různých automatů pracujících s váhami. Označují se např. zobecněný automat, automat s váhami či např. Max-Min automat. O takovýchto automatech se lze dočíst např. v [?], [?] či [?].

## 4.4 Fuzzy automat s výstupem

Rozšířením klasického automatu lze získat automat s výstupem. Automat kromě přechodové funkce (které se pak říká „vstupní“) disponuje také výstupní přechodovou funkcí, která při přechodu mezi stavy odesílá na výstup symboly. Takové automaty generují řetězce, takže obvykle nemívají koncové stavy. S fuzzy automaty s výstupem se lze setkat např. v [?][?][?][?][?].

## 4.5 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>6</sup>. U stavových strojů není totiž důležitý výsledek výpočtu, ale jeho průběh.

Takovýto automat je uveden např. v [?] nebo [?].

<sup>6</sup>V určitém smyslu však může být za výstup považován stav (resp. fuzzy stav), ve kterém výpočet zkončil.

## 4.6 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 funkce může být zapsána jako fuzzy IF–THEN pravidla. 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  $d$

Událostmi řízené fuzzy automaty pak tyto pravidla nahrazují libovolnými fuzzy IF–THEN pravidel. Takovýto automat má velké praktické uplatnění, protože popis pomocí fuzzy IF–THEN pravidel je mnohem přirozenější a flexibilnější, než pomocí symbolů. Setkat se s nimi můžeme např. v [?], [?] nebo [?].

560 **Poznámka 4.2.** *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.*

## 4.7 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 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. Definice 570 zásobníkového automatu je uvedena např. v [?].

# 5 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 předchozí kapitole se však od „běžného“ nedeterministického automatu liší více a proto je třeba pro jejich zavedení doplnit některé související pojmy.

## 5.1 Fuzzy tree automaty

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

580 Fuzzy tree automaty vznikly zobecněním „klasických“ bivalentních 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 [?].

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

590 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  $M(M(x))$ .

## 5.2 Stromy a pseudotermy

**Definice 5.1** (Doména stromu). *Mějme abecedu  $\Sigma$  uspořádanou pomocí  $\leq$ . Pak konečnou množinu  $U \subseteq \Sigma^*$  nazvěme doména konečného stromu, pokud splňuje*  
600 *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 5.2** (Částečně spořádaná abeceda). *Částečně spořádaná abeceda je dvojice  $(N, T)$ , kde  $N$  a  $T$  jsou dvě disjunktní konečné abecedy (tj.  $N \cap T = \emptyset$ ).*

{def:Tree}

**Definice 5.3** (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  $w \in \bar{U}$

610

*Místo  $t(w)$  budeme psát jen  $t$ .*

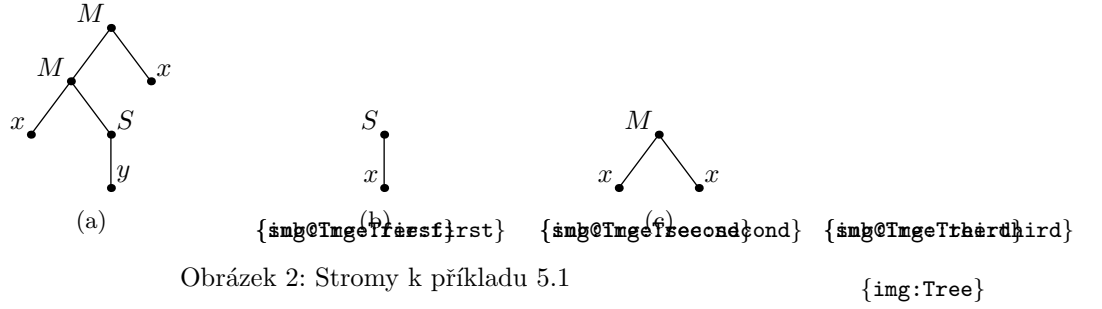
\* Strom  $t$  je tedy předpis pro „přejmenování“ uzlů prefixového stromu daného doménou  $U$ . Strom dle definice 5.3 je v korespondenci s pojmem „strom“ (resp. „kořenový strom“) z teorie grafů. Z tohoto důvodu si pro jednoduchost můžeme odpustit definici souvisejících pojmů z teorie grafů pro strom z definice 5.3. Můžeme tak stromy graficky zobrazovat, hovořit o jejich potomcích, podstrozech, vnitřních a listových uzlech bez nutnosti formálního nadefinování.

{ex:Trees}

**Příklad 5.1.** *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:*

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

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



Obrázek 2: Stromy k příkladu 5.1

620 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>7</sup>.

**Definice 5.4** (Pseudoterm). *Označme  $D_{(N,T)}^p$  nejmenší podmnožinu  $(N \cup T \cup \{(\cdot, \cdot)\})^*$  splňující následující podmínky<sup>8</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 5.1.** *Definice pseudotermu lze snadno přepsat do gramatiky. Vzhle-*  
*dem k tomu, že taková gramatika bude jistě bezkontextová, bude jazyk  $D_{(N,T)}$*   
630 *bezkontextový. Tento fakt bude mít důsledek na konstrukci fuzzy tree automatu.*

**Příklad 5.2.** *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 5.1.** *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.  $t = a$ ,  
640 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 5.2.** *Ke každému pseudotermu  $p(t) \in D_{(N,T)}^p$  existuje odpovídající strom*  
 *$t$ .*

*Důkaz.* Opět dokážeme strukturálně:

- je-li pseudoterm atomický, tj.  $p(t) = a$ , kde  $a \in T$ , pak doménou stromu  $t$   
je množina  $\{\epsilon\}$  a  $t(\epsilon) = a$

<sup>7</sup>Oproti termům predikátové logiky mají však jiný pohled na nulární funktory, které u  
pseudotermu neexistují

<sup>8</sup>Předpokládáme, že symboly závorek, ( a ) nejsou součástí  $N \cup T$

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

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

□

**Příklad 5.3.** 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.

**Definice 5.5** (Fuzzy jazyk stromů). Fuzzy množinu  $\tau$  nad  $D_{(N,T)}$  nazvěme fuzzy jazyk stromů.

### 5.3 Fuzzy tree automat a jazyk jím rozpoznávaný

Začneme definicí fuzzy tree automatu.

**Definice 5.6** (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  $N \cap T = \emptyset$
- $\mu : (N \cup T) \rightarrow \{f | f : (\mathcal{Q} \cup \{\epsilon\}) \times Q \rightarrow [0, 1]\}$  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.

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

**Příklad 5.4.** Uvažujme množiny  $N$  a  $T$  stejné, jako v předchozích příkladech. Stanovme  $Q = \{q_1, q_2\}$ . Fuzzy množinu  $F$  položme rovnu  $\{q_2\}$  a zobrazení  $\mu$  je zaznačeno v tabulce 2. Pak  $A = (Q, T, N, \mu, F)$  je fuzzy tree automat.

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.

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

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

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

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

{tab:MuOfFuzTreAut}

**Definice 5.7** (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 5.8** (Jazyk rozpoznávaný). *\* 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 5.5.** *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  $\mu_{t_1}(q_2)$ :*

$$\begin{aligned} \mu_{t_1}(q_2) &= (\mu_M(q_1q_1, q_2) \wedge \mu_{t_{1,1}}(q_1) \wedge \mu_{t_{1,2}}(q_1)) \\ &\quad \vee (\mu_M(q_1q_2, q_2) \wedge \mu_{t_{1,1}}(q_1) \wedge \mu_{t_{1,2}}(q_2)) \\ &\quad \vee (\mu_M(q_2q_1, q_2) \wedge \mu_{t_{1,1}}(q_2) \wedge \mu_{t_{1,2}}(q_1)) \\ &= (0,6 \wedge 1 \wedge 1) \vee (1 \wedge 1 \wedge 0) \vee (0,7 \wedge 0 \wedge 1) = 0,6 \end{aligned}$$

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  $A(t_3) = 1$ .

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

**Věta 5.3.** *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 5.9** (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  $T = \{(t, 1)\}$ .

## 5.4 „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.

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

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

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

**Definice 5.10** (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  $1 \leq k \leq m^2$ .*

**Poznámka 5.2.** 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.

740 **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  $c_{ij}$ :

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

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

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

{ex:GameOfLife}

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

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

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

**Definice 5.11** (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. Konečná 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 5.12** (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:*

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

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

**Definice 5.13** (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 konfigurace  $i$ -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.

780 **Příklad 5.7.** *Ukázka výpočtu automatu  $A_{100}$  realizující Hru života je na obrázku 3. Černá barva symbolizuje mrtvou buňku, bílá pak živou.*

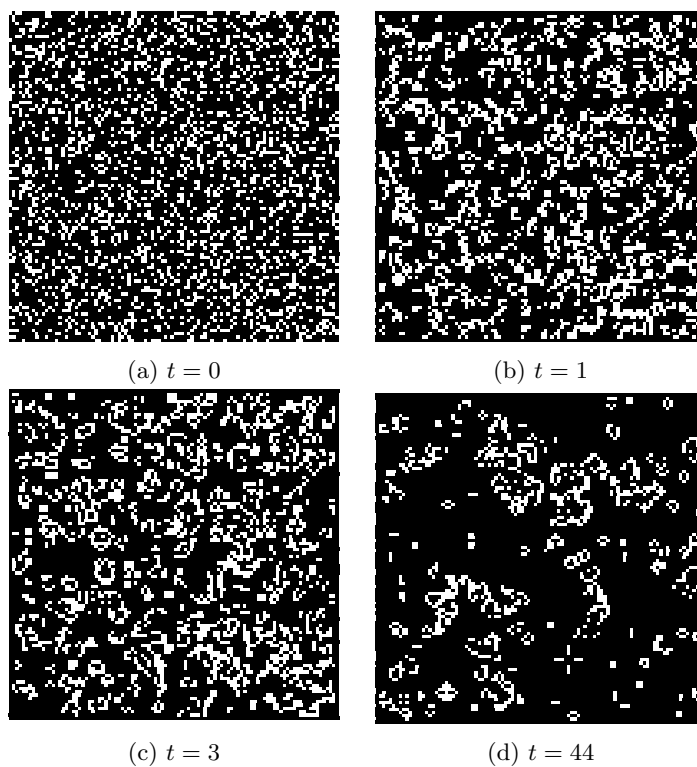
## 5.5 Buněčné fuzzy automaty

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>9</sup> (přechodová funkce by nebyla deterministická) nebo nepřešla do žádného (přechodová funkce by nebyla úplná).

790 Vzhledem k tomu, že stejné chování budeme vyžadovat i u buněčného fuzzy automatu, „odstupňování“ přechodové funkce (zavední fuzzy přechodové funkce) by nemělo smysl<sup>10</sup>.

<sup>9</sup>Jak bude ukázáno, tento předpoklad lze obejít

<sup>10</sup>Teoreticky by bylo možné nahradit stav buňky fuzzy stavem buňky. Takováto úprava by však výrazně zvýšila výpočetní složitost výpočtu automatu a například také znesnadnila grafické znázornění jeho konfigurací a proto zde nebude uvažován.



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

{img:GameOfLife}

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.

**Příklad 5.8.** *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  $q_{i,j}^{(t+1)} = L$

\*

## 6 Užitečné techniky pro práci s fuzzy automaty

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.

### 6.1 Fuzzy automat bivalentního automatu

V některých situacích máme k dispozici „klasický“ bivalentní automat<sup>11</sup> a potřebujeme k němu sestavit odpovídající fuzzy automat. Odpovídající ve smyslu, že řetězce, které bivalentní automat přijímá (zamítá) musí fuzzy automat přijímat ve stupni 1 (0).

Uvažujme automat  $A = (Q, \Sigma, \mu, I, F)$  z definice 3.1. Pak fuzzy automat  $A' = (Q, \Sigma, \mu', \sigma, \eta)$  odpovídající tomuto automatu je následující:

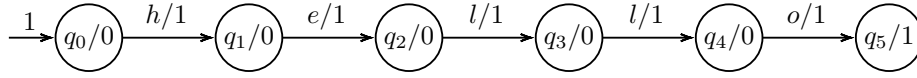
- množina stavů  $Q$  a abeceda  $\Sigma$  zůstávají stejné
- přechodová funkce  $\mu'$  je dána předpisem (pro všechny  $q, q' \in Q$  a  $x \in \Sigma$ ):

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

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

\* \*

<sup>11</sup>Zde je vhodné připomenout, že automat můžeme získat například také konverzí regulárního výrazu nebo regulární gramatiky



Obrázek 4: Automat rozpoznávající hello

{diag-AutRozpHell}

## 6.2 Fuzzy automat rozpoznávající $\omega$

{sec:FuzAutRozpOme}

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

830

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 regulární. \*
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:

840

{def-FuzzAutRozpOme}

**Definice 6.1** (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*

850

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

**Příklad 6.1.** *Příklad fuzzy automatu rozpoznávající řetězec  $\omega = \text{hello}$  se nachází na obrázku 4.*

## 6.3 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  $x, y \in \Sigma$ ). Fuzzy relace podobnosti symbolů pak může být součástí automatu. V takovém případě dojde ke modifikaci výpočtu automatu takovým způsobem, že ve značení ?? a definici 3.8 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.

## 6.4 Editační operace, deformovaný automat

{sec:DefAut}

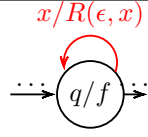
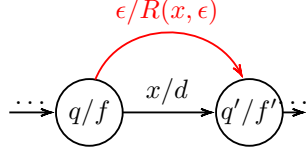
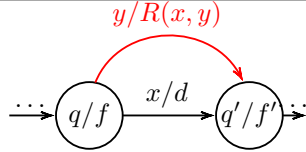
Další technikou pro modifikaci chování automatu je využití editačních operací. Tato technika byla přejata z [?]. Základní idea této techniky spočívá v trojici jednoduchých editačních operací, jejíž složením jsme schopni popsat požadovanou transformaci vstupu na automatem reprezentovaný vzor. „Množství“ 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

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  $\Sigma$ . Editační 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

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

Tabulka 3: Deformace automatu. Černě jsou znázorněny přechody původního automatu, červeně pak nově přidané

{tbl:DefAutDef}

tři posloupnosti editačních operací jsou vyrovnáním řetězce **ahoj** na řetězec **hello**:

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

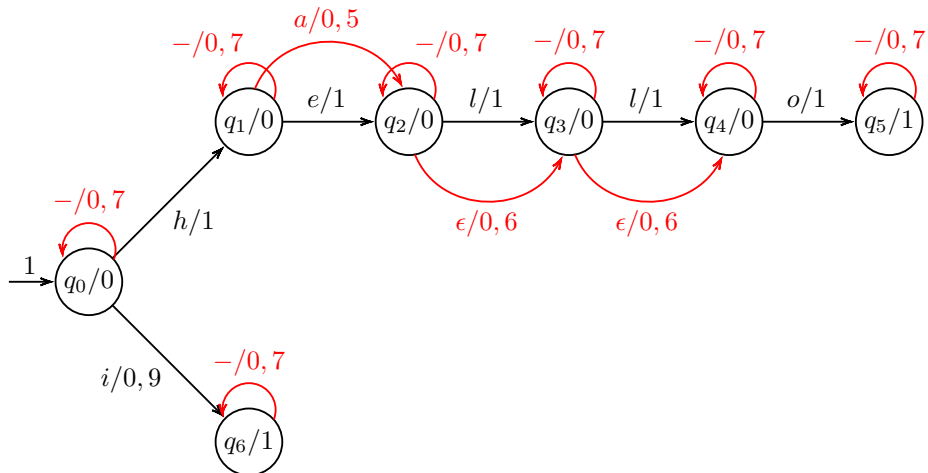
Označme  $E$  jako množinu všech editačních operací, tj.  $E = (\Sigma \cup \{\epsilon\})^2 \setminus (\epsilon, \epsilon)$ . Budeme-li chtít pro každou editační operaci stanovit stupeň, v jakém má být provedena, použijeme fuzzy relaci  $P : E \rightarrow [0, 1]$ . Následovat bude popis tzv. „deformace“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

{def-AutRozpCall}

**Definice 6.2** (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:*

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

Význam jednotlivých složek přechodové funkce (deformací) je názorně zobrazen v tabulce 3. Příklad deformovaného automatu je na obrázku 5. \*



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

{img:DefAut}

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 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  
910 potřeba provést ručně. Další informace ohledně editačních operací a deformovaných automatů lze nalést např. v [?].

## 6.5 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 do-  
920 statečné).

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.

930 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. \*

## 6.6 Fuzzy automaty a neuronové sítě

940 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 síť 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 kapitole. Tedy, že vstupní data se postupně nechávají vyhodnocovat neuronovou 950 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 [?], [?] [?], [?], [?], [?], [?].

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.\*

960 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 do fuzzy IF–THEN pravidel, jak bylo uvedeno v kapitole 4.6.

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

## 7 Rozpoznávání, klasifikace a korekce textových dat

970

Fuzzy (pravděpodobnostní) automat je nástroj, který v základu slouží k rozpoznávání řetězců. To znamená, že určuje, jak moc řetězce odpovídají automatem určenému vzoru.

Na tento koncept lze dále navázat. Máme-li vzorů více, můžeme určit, kterému z nich je vstup nejpodobnější. V takovém případě hovoříme o klasifikaci. Případně



se pokusit odchylku od nejpodobnějšího vzoru odstranit, v takovém případě hovoříme o korekci.

V této kapitole budou prezentovány některé reálné aplikace fuzzy a pravděpodobnostních na problém rozpoznávání, klasifikace a korekce a to textových dat. Využití fuzzy 980 automatu pro rozpoznávání (a případně klasifikace a korekci) netextových dat se věnuje následující kapitola. Navíc, rozpoznávání obrazových dat je popsáno v kapitole \*.

## 7.1 Výpočet podobnosti řetězců

Uvažujme, že máme dva (různé) řetězce,  $w$  a  $w'$  (nad společnou abecedou  $\Sigma$ ). Jak moc jsou si „podobné“? Jinými slovy, je třeba navrhnout zobrazení  $f : \Sigma^* \times \Sigma^* \rightarrow [0, 1]$ , které jim přiřadí stupeň „podobnosti“.

Existuje několik technik, jak podobnost řetězců určit. Například Hammingova vzdálenost, která je definována jako počet rozdílných symbolů na odpovídajících pozicích (např.  $H(abcd, bacc) = 3$ ). Hammingova vzdálenost však 990 nedokáže například reflektovat řetězce různých délek.

Pro určení podobnosti řetězců lze použít i fuzzy automaty. Sestavíme-li automat  $A_w$  rozpoznávající řetězec  $w$  (viz kapitola 6.2) a poté automat upravíme (např. pomocí deformací dle kapitoly 6.4), získáme nástroj pro určování podobnosti řetězců  $w$  a  $w'$ . Podobnostní relace pak bude vypadat následovně:

$$f(w, w') = A_w(w')$$

Používání fuzzy automatů pro výpočet podobnosti řetězců bývá v praxi poměrně časté. Setkat se s ním můžeme např. v [?], [?], [?], [?], [?].

## 7.2 Klasifikace a korekce textových řetězců

Jak již bylo uvedeno, fuzzy (a pravděpodobnostní) automaty lze využít pro klasifikaci a následnou korekci (obecných) textových řetězců.

Probablistické klasifikaci se věnují např. v [?].

V [?] a [?] zavádějí pravděpodobnostní automat v mírně modifikované podobě<sup>12</sup>. Tento automat nechává učit text a poté s jeho pomocí opravují pozměněný text bible. Techniku používají také, pro segmentaci DNA, která je podrobněji 1000 popsána v kapitole \*.

Další variantu automatu<sup>13</sup> používají v [?] a [?]. Automat tak dokáže počítat počítat počet výskytů vzorů ve vstupním řetězci. Toho pak využívají pro rozpoznávání proteinů. Sledovaný vzorek proteinu nejdříve chemicky rozštěpí na jednotlivé peptidy. Ty jsou hmotnostním spektrometrem analyzovány a naměřené hodnoty jsou kódovány do symbolů. Z naměřených symbolů jsou posléze sestaveny řetězce, které jsou vyhodnocovány automatem.

V [?] a [?] pomocí pravděpodobnostních automatů snaží překládat slova fonetické abecedy na „běžnou“ (angličtinu). Dá se totiž předpokládat, že vyřčené slovo nemuselo být řečeno zcela zprávně a navíc mohlo být upraveno určitým 1010 akcentem. Z databáze tak (podobně, jako při učení) sestavují automaty pro jednotlivá slova a ty spouštějí oproti vstupu.

<sup>12</sup>Zavádějí automat s pamětí. Automat si kromě aktuálního stavu a nezpracovaného vstupu pamatuje také vstup již zpracovaný. Jeho výpočet tak může být v určitém smyslu kontextově závislý

<sup>13</sup>Zde se jedná o tzv. aritmetický pravděpodobnostní automat. Automat při přechodech provádí elementární algebraické operace, v jejich případě sčítání.

### 7.3 Detekce překlepů

{subs:DetTyp}

Pomocí fuzzy automatů lze poměrně snadno detekovat a případně se pokoušet opravit překlepy. Využívá se při tom fakt, že překlep obvykle znamená vložení či náhradu symbolem, který je na klávesnici počítače v okolí očekávaného symbolu. Například řetězec `hr11lo` zcela určitě vznikl překlepnutím při psaní `hello`, než kupříkladu `ahoj`. Je tak možné napsané slovo opravit. Tuto techniku popisují v [?], popř. [?].

### 7.4 Fuzzy lexikální analyzátor

1020 Lexikální analyzátor je základní součást všech překladačů (nejen) programovacích jazyků. Jeho vstupem je konečný proud znaků (ten můžeme chápat jako řetězec) a výstupem pak sekvence tzv. tokenů. Token je dvojice tvořená typem tokenu a jeho hodnotou. Tokeny může být například „číslo 42“, „identifikátor `foo`“, „operátor `+`“ či „symbol konce příkazu `;`“.

Lexikální analyzátory obvykle rozpoznávají regulérní jazyky. Vzory pro jednotlivé typy tokenů bývají obvykle zapisovány pomocí regulérních výrazů. Při chodu lexikálního analyzátoru je vstup spouštěn s automaty jednotlivých vzorů a nejdelsí shoda je použita jako výstup. Nevýhodou tohoto přístupu je, že nijak ne-reflektuje malé, bezvýznamné chyby ve vstupu. Například vstup `iif` by zřejmě 1030 měl být reprezentován jako token „klíčové slovo `if`“ než jako „identifikátor `iif`“. Tyto problémy se snaží odstranit nahrazením „klasického“ lexikálního analyzátoru fuzzy Lexikálním analyzátozem.

Fuzzy lexikální analyzátor je popsán v [?]. Oproti „klasickému“ lexikálnímu analyzátoru používá pro popis vzorů tokenů fuzzy regulérní výrazy. Výstupní token je pak zvolen ten, který nejvíce odpovídá vzoru.

\*

### 7.5 Pravděpodobnostní rozpoznávání přirozeného textu

Pravděpodobnostní automaty lze uplatnit při zpracování přirozeného jazyka. Jako jazyk je zde chápána množina všech slov, jedná se tedy o syntaktickou 1040 analýzu vět v přirozeném (např. anglickém) jazyce. Přirozené jazyky bývají obvykle bezkontextové. Je proto nutné použít buď pravděpodobnostní tree automaty [?] nebo zásobníkové pravděpodobnostní automaty [?].

### 7.6 Dvouúrovňové vyhledávání v dlouhém textu

V [?] používají dvouúrovňové rozpoznávání vzorů v dlouhém textu. Nejdříve sestaví jednoduchý (např. co do počtu stavů) automat pro přibližné rozpoznání vzoru. Tento rychlý automat je aplikován na každý větší blok textu, např. na každou stránku. Stránky, které prošly v nenulovém stupni (tj. je pravděpodobné, že obsahují vzor) pak prohledá přesným automatem a určí konkrétní výskyty.\*

\*

### 1050 7.7 Parsování referencí

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\*:

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.

## 8 Další oblasti pro rozpoznávání

V předchozí kapitole bylo prezentováno pár příkladů, kdy lze pomocí fuzzy a pravděpodobnostních automatů rozpoznávat, klasifikovat a případně opravovat textová data. V této kapitole bude na tyto problémy navázáno a budou rozšířeny na rozpoznávání a případně následnou klasifikaci a korekci netextových dat.

Základní myšlenka těchto technik vždy spočívá ve způsobu, jak vstupní data zakódovat do symbolů a řetězců. Data, která transformujeme do řetězců, pak můžeme předat příslušnému fuzzy nebo pravděpodobnostnímu automatu a pracovat s nimi jako v předchozí kapitole. Některé ukázky takových uplatnění zde budou prezentovány.

### 8.1 Rozpoznávání signálů

{subs:RozpSign}

Rozpoznávání signálů je jedna ze základních možností, jak využít rozpoznávání. Věnují se tomu např. v [?], [?], [?] a [?]. Signál, jakožto spojitý průběh jedné proměnné (času) nejprve kvantujeme v čase. Obdržíme tak posloupnost po sobě jdoucích vzorků (typicky číselných hodnot). Každý takový vzorek můžeme snadno zakódovat do odpovídajícího symbolu. Takovými symboly může být například „hodnota je vysoká“, „hodnota je střední“ či „hodnota je nízká“. Pokud však symboly nahradíme ligvistickými štítky, můžeme s těmito symboly pracovat na fuzzy úrovni. Následně můžeme sestavit automat, který vzor z těchto fuzzy symbolů dokáže zpracovávat.

\*

\*

V [?] tuto techniku testují na detekci závady na elektromotoru. Vstupem je úhel natočení rotoru, jehož sinus by měl kopírovat sinusovku. Při poruše se však průběh zdeformuje, což automat rozpozná. Stejně tak používají tuto techniku pro rozpoznávání vzorů v datech z elektrokardiogramu (EKG). Problematicke analýzy signálu EKG je věnována kapitola \*. V [?] je rozpoznávání signálů demonstrováno na rozpoznávání gest rukou. To je podrobněji rozebráno v kapitole \*.

V [?] pomocí pokročilejších technik monitorují teplotu taženého profilu v ocelárně. Příliš nízká, vysoká (resp. některé specifické kombinace) teplota může způsobit, že se tažený profil zlomí. Monitorováním teploty a příslušným přizpůsobováním parametrů tažení lze zlomům předcházet. Automat autoři doporučují konstruovat učením.

Velmi podobnou techniku používají v [?]. Autoři zde podobným způsobem modelují vzory ve vývoji cen na burze.

## 8.2 Rozpoznávání dvourozměrného signálu

V [?] je popisován způsob, jak rozpoznávat dvojrozměrný signál (mřížku signálů), např. data z termokamery\*. Používá se k tomu model podobný pravděpodobnostním buněčným automatům. Na rozdíl od nich však tento model lépe ošetřuje 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.

## 8.3 Rozpoznávání ručně psaného textu

{subs:RecHandWrit}

Problém ručně psaného textu je v současné době velmi významný. V závislosti na konkrétní situaci může být vstupem buď přímo bitmapa (např. sken), nebo již vektorizovaný obrázek obsahující informace o cestě, kterou hrot pera při psaní písmene udělal.

Uvažujme tedy, že máme k dispozici již cestu. Analýzou průběhu cesty, tj. rozkladem na podcesty (např. na základě směru či délky) získáme segmenty. Každému segmentu může být posléze přiřazen symbol (dán typicky jeho směrem). Celé cestě pak může být přiřazen řetězec, který může být následně rozpoznán automatem. \*

Této technice se věnují např. v [?] [?]. V obou případech poukazují, že pro řádově lepší výsledky rozpoznávání je vhodné systém vylepšit určitou formou strojového učení.

Tato technika pro rozpoznávání ručně psaného textu byla naimplementována. Popis implementace a získané výsledky je popsán v \*.

## 8.4 Rozpoznávání gest

Problém rozpoznávání gest je ve své podstatě rozšíření rozpoznávání ručně psaného textu. Gestem je zde myšlen pohyb ruce nebo rukou vytvářející určitý tvar (např. trojúhelník).

Při rozpoznávání gest je však značně komplikovanější konverze vstupních dat na posloupnost symbolů. Vstupem může být např. videosekvence \* [?], nebo např. data z akcelerometrů umístěných na ruce [?].

## 8.5 Práce se zvukem

Vzhledem k tomu, že zvuk je ve své podstatě analogový signál, může s ním být pracováno jako se signálem. To znamená, že může být rozpoznáván tak, jak bylo popsáno v kapitole 8.1. V [?] a [?] popisují způsob, jak takové automaty zkonstruovat.

## 1140 8.6 Fuzzy programy

V [?] popisují fuzzy programy. Fuzzy program je prakticky deklarativní program, fuzzy reguérní výraz, který popisuje, jak mají vypadat výsledky. Fuzzy automaty tak mohou sloužit jako interprety takovýchto programovacích jazyků. \*

## 8.7 Metoda lisování dat

\* 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>14</sup> nepatří (tj.  $w' \in \Sigma^* \setminus L$ ) by posléze měl automat být schopen určit  
1150 jak moc se jazyku  $L$  podobají, tj. určit stupeň  $A_{L,\sigma}(w')$ .

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

1. Máme jazyk  $L$ , který je konečný. Je tedy zřejmě i regulérní.
2. K jazyku  $L$  sestavíme nedeterministický konečný automat  $A_L$  rozpoznávající  $L$ . Takový automat bude v nejhorším případě obsahovat  $\sum_{w \in L} |w|$  stavů.
3. K automatu  $A_L$  vytvoříme nedeterministický fuzzy automat  $A'_L$ .
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).  
1160 Části automatu reprezentující tyto „podjazyky“ budou minimalizací zmenšeny. Ve výsledku se tak dá očekávat značné zmenšení velikosti automatu.

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

Technika lisování dat bude naimplementována a experimentálně ověřena. \*  
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  
1170 poté byl sestaven jazyk, na který by bylo aplikováno lisování dat. Výsledkem by byl automat, který rozpoznává silná hesla. Tato aplikace však nebyla naimplementována, vzhledem k tomu, že je z bezpečnostních důvodů nemožné získat databázi uživatelských hesel.

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

<sup>14</sup>Jazyk  $L$  může být klidně fuzzy jazyk, následný postup se pak jen patřičně upraví

Při velikosti jazyka  $x$  a koeficientu  $\sigma = y$  se podařilo automatu dosáhnout z úspěšného rozpoznávání čísel dělitelných 4 ve stupni vyšším, než  $w\%$ . \*

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

## 8.8 Detekce úplných $m$ -árních stromů

{subs:DetComTrees}

1180 Fuzzy tree automaty mohou být použity pro velmi elegantní určení úplnosti stromu. Úplný  $m$ -ární strom je takový strom, jehož každý uzel má buďto právě  $m$  potomků (vnitřní uzly) a nebo 0 (listové uzly). Tato technika pochází z autorovy invence.

Zaveďme lingvistickou proměnnou  $C$  „vnitřní uzel  $v$  o  $n$  potomcích je vnitřní uzel úplného  $m$ -árního stromu“ a její štítek „ano, je“. Pak významovou funkci  $C(v)$  tohoto štítku může nadefinovat následovně:

$$C(v) = \frac{n}{m}$$

Pro listové uzly můžeme  $C(v)$  nadefinovat vždy jako 1. Vzhledem k tomu, že nyní známe hodnotu lingvistické proměnné  $C$  pro všechny uzly  $v$  stromu, můžeme určit, jak moc je úplný  $m$ -ární celý strom. K tomu použijeme fuzzy tree automat.

1190 Uvažujme, že strom  $t$  je tvořen vnitřními uzly  $I$  a listovými uzly  $o$ . Pak fuzzy jazyk  $m$ -árních úplných stromů bude fuzzy jazyk stromů nad  $D_{(N,T)}$ , kde  $N = \{I\}$  a  $T = \{o\}$ . Zbývá tedy navrhnout fuzzy tree automat, který takový jazyk bude rozpoznávat.

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

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

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

Takto navržený automat zřejmě rozpoznává  $m$ -ární úplné stromy. Pro demonstraci byla tato technika naimplementována. Soubory automatu a ukázkových vstupních stromů jsou k nalezení v adresáři `fuzzy-tree-automata/test/data/m-ary-trees`.

## 9 Modelování a simulace

1200 Modelování a simulace je odvětví jehož účelem je získat přehled o chování určitého systému. Jakmile takový model máme, můžeme například simulovat některé jevy, které by mohly nastat. Případně, naopak, předpovídat, který jev z nejvyšší pravděpodobnosti nastane.

Fuzzy a pravděpodobnostní automaty v určitých situacích mohou posloužit jako takovéto modely. Podmínkou obvykle bývá, aby se modelovaný systém nacházel v určitém diskrétním stavu, popř. více stavech současně. Vzhledem k obvykle složitější struktuře dat, se kterými systém pracuje, bude velká část těchto systémů založena na událostmi řízených automatech.

### 9.1 Automobilismus

Automobily jsou v dnešní době zařízení vybavené značným množstvím palubní elektroniky. Je výhodné, aby elektronika měla přehled o tom, co se automobilem děje, případně, co se v nejbližších sekundách bude dít.

1210 V [?] a [?] jsou prezentovány způsoby, jak jednoduše modelovat manévry řidiče. Mezi takové manévry patří např. předjíždění, zatáčení doleva v křižovatce nebo pouštění chodce na přechodu. Model využívá sadu lingvistických proměnných jako např. „aktuální rychlost“, „zrychlení“, „natočení volantu“ nebo „směrová světla jsou zapnuta“. Následně mohou být stanoveny vzory jednotlivých manévru, např. při odbočování vlevo:

1. řidič zapnul levý blinkr
2. řidič otočil volantem lehce doleva (může být vynecháno)
3. řidič srovnal volant (může být vynecháno)
4. řidič zpomalil
- 1220 5. řidič výrazně otočil volantem doleva
6. řidič vypnul levý blinkr

Takto navrženému vzoru může být sestaven automat, který jej rozpoznává.

Podobný způsob je popsán v [?]. V tomto případě však jako vstupní data sloužila měření z trojice akcelerometrů (pro všechny tři osy,  $x$ ,  $y$  a  $z$ ). Lingvistické proměnné pak sledovaly, zda-li bylo zrychlení ve směru příslušné osy kladné, neutrální či záporné. Vzory, které tento systém sledoval byly např. „dlouhá táhlá zatáčka vlevo“ nebo „přejezd přes horizont“.

\*

## 9.2 Monitorování elektrických a počítačových sítí

{subs:MonElComNet}

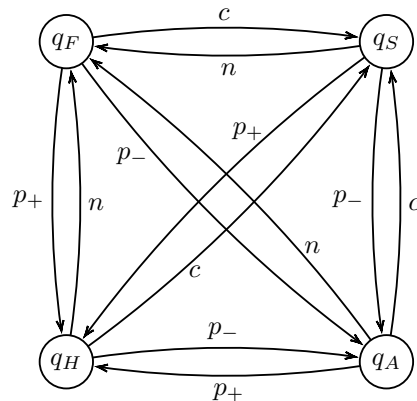
1230 V [?] používají fuzzy automaty pro sledování vytížení el. rozvodné sítě. Technika předpokládá znalost spotřebičů v síti a nástroj pro monitorování odběru el. proudu (elektroměr). Výstupem je pak rozpis, kdy byl jaký spotřebič (pravděpodobně) zapnut a kolik odebíral energie.

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“ a „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. Zavedme lingvistickou proměnnou „spotřeba el. proudu ...“ a její štičky „výrazně klesla“, „je konstantní“ a „lehce stoupla“.

1240 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), 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.

Obdobná technika je uvedena v [?]. V článku modelují pomocí pravděpodobnostního automatu počítačovou síť (zvláště 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.

1250



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}

### 9.3 Teorie her, počítačové hry

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

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

1260

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 výženy 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.

1270

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.

\* \* \*

Podobná technika je popsána v [?]. Zde ovšem neslouží k popisu emocí hráče, ale predikci možnému dalšímu vývoji hry.





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

{img:ChessEmosScreens}

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.

## 9.4 Interakce s člověkem

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

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 ohodnocujících funkcí pro štítky lingvistických proměnných) lze chování systému přizpůsobit povaze člověka (např. klidná povaha, pesimista a podob.), což může výsledky značně zpřesnit.

V [?] pomocí pravděpodobnostních automatů analyzují aspektová tvrzení (tj. zda-li je tvrzení míněno pozitivně, negativně, neutrálně nebo s rozporem). Stavby automatu jsou všechna slova z celé databáze, vstupní abeceda taktéž. Počáteční

1310 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.

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). \*

## 9.5 Sledování pohybu a aktivit osoby

1320 V [?] je prezentován postup, jak pomocí sledování různých parametrů (pohyb, získaný akcelerometrem, a el. vodivost kůže) určit pravděpodobnou činnost (nicnedělání, chůze, práce, odpočinek) osoby. Činnosti odpovídají stavům automatu a na základě parametrů se určí přechody. Navíc, je pravděpodobnější, že člověk ze stavu nicnedělání přejde do stavu práce než do stavu relaxace.

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

1330 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).

Podobnou techniku popisují v [?]. Zde však namísto fuzzy automatů používají pravděpodobnostní automat. Uvádějí však další aplikace pro tuto techniku (biometrika, sémantická analýza videa, bezpečnost, uživatelská rozhraní, syntéza animací). \*

## 9.6 Průmyslové řídicí systémy, fuzzy kontroléry

Vztah fuzzy automatů a řídicích systémů jsou popsány např. zde [?], [?], [?] nebo [?]. Systémy bývají často tak složité, že je nemožné zanést do znalostní báze, která má jejich chod popisovat, všechny možné předpoklady. Z tohoto důvodu může u systému dojít k tzv. porušení předpokladů (např. při chybě či 1340 nečekaném vnějším podnětu). Dalo by se říci, že systém se takto dostane do nekonzistentního stavu.

Je-li systém modelován klasickým, bivalentním automatem \* 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.

\*

1350 V [?] používají učící se fuzzy událostmi řízené automaty, pro řízení výkonu při obloukovém svařování. Vstup automatu je realizován pomocí jedné lingvistické proměnné se štitky „zvýšení výkonu“ a „snížení výkonu“. Automat je 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 lingvistické proměnné s dvojicí štitků pro zvýšení výkonu motorů a snížení výkonu motorů si kladou za cíl navrzení systému pro řízení robota. Tento systém má za cíl řídit přidávání

a ubírání výkonu tak, aby se robot, bez ohledu na terén, pohyboval konstantní rychlostí.

V [?] je fuzzy automat použit pro návrh řídicích systému. Řídicí systém je stavový stroj, který je překlápěn mezi diskrétními stavy pomocí událostí. Množina možných událostí tak tvoří abecedu a řídicí systém může být reprezentován automatem. Použitím nepřesností je tak možno získat fuzzy automat. \* S pomocí znalosti očekávaného chování systému je pak možné pomocí učícího automatu sestavit adekvátní řídicí systém.

## 9.7 Problém městského růstu

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 [?].

1380 \*

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 parcele pak bude odpovídat jedna buňka buněčného fuzzy automatu. U každé parcely je třeba zjistit rozličné ukazatele, např.:

- je-li parcela zastavěna (popř. jak moc)
- typ zástavby na parcele (např. rodinné domy, obytné domy, komerční prostory, prostory pro rekreaci, dopravní stavby, průmyslová zóna)
- jak moc žije na parcele obyvatel
- 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, ...)

- 1400
- 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:

- 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
- 1410 • 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 už jsou patrné větší odlišnosti (simulace nevyprodukovala tak výrazný růst, jaký 1420 doopravdy nastal).

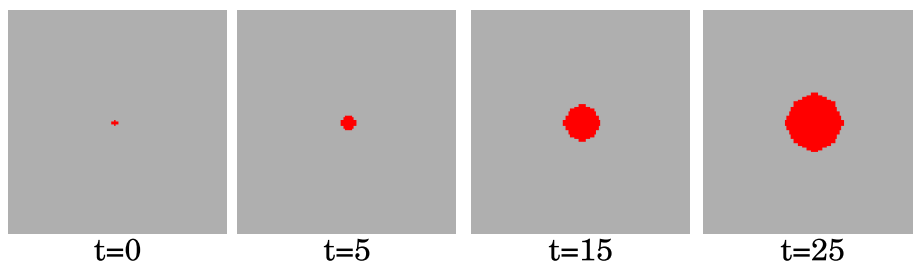
## 9.8 Další uplatnění

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 informací 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 1430 paměti na haldě.

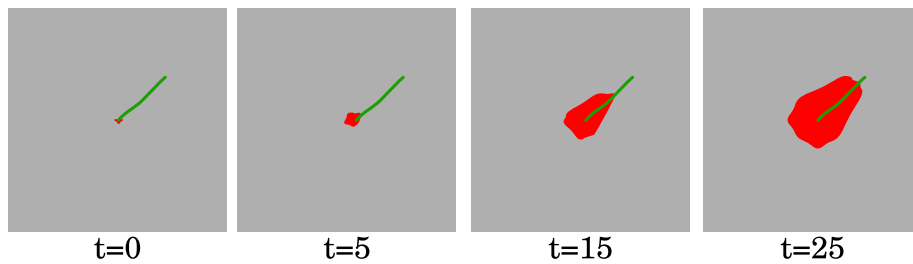
V [?] navrhuje používat učení fuzzy automatů pro konstrukci logických klopných obvodů. Tj. obvodů, které se na základě vstupů překlápí mezi různými diskrétními stavy. Sami autoři uvádějí, že bylo výhodné pracovat s učícím fuzzy automatem a teprve poté jej „diskretizovat“, tj. konvertovat na kýžený bivalentní.

## 10 Zpracování obrazu

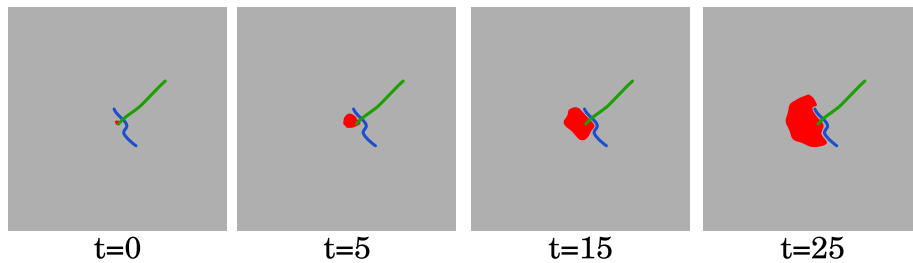
Zpracování obrazu je velmi populární informatická disciplína. V této kapitole budou prezentovány některé problémy, které lze řešit pomocí fuzzy automatů. Vzhledem k tomu, že obraz bývá obvykle reprezentován bitmapou, bývá obraz 1440 obvykle zpracováván pomocí buněčných fuzzy automatů.



(a) Růst města bez dalších dodatečných podmínek



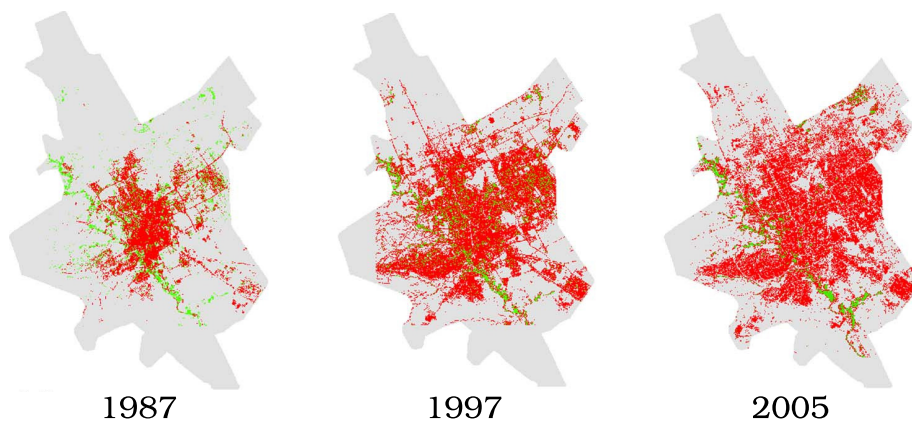
(b) Růst města podél silnice



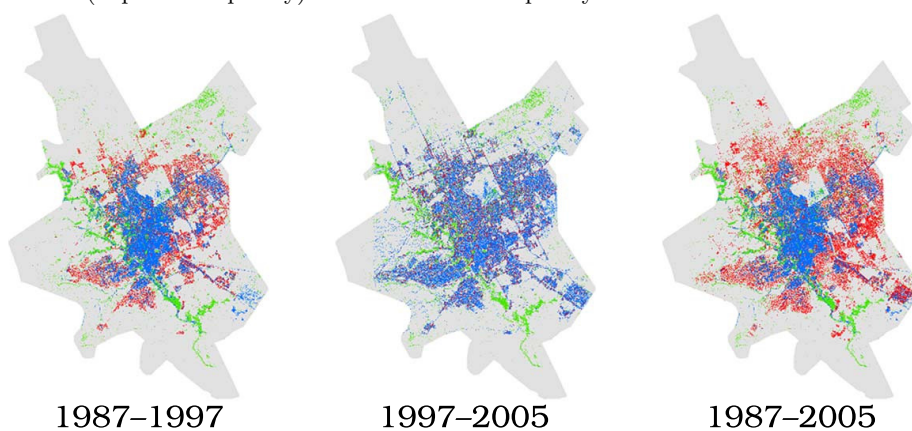
(c) Růst města bržděn řekou

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

{img-VarTransRuls}



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



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

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

{img-UrbGroProSample}

## 10.1 Konvoluce

{subs:Convol}

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 je ohodnocující funkce lingvistické proměnné se štitkem „pixel má bílou barvu“. Tato skutečnost nám umožňuje pracovat s obrazem pomocí fuzzy logiky.

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

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

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

je konvoluční maticí jednoduchého rozostření. Aplikuje se následujícím způsobem:

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

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

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

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

kde  $\epsilon > 1$  je parametr udávající agresivitu zvýrazňování a  $\text{neighs}_{i,j}$  je součet hodnot okolních buněk (viz příklad 5.6).

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.

## 10.2 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. V [?] a [?] pak tuto techniku vylepšují pomocí strojového učení.

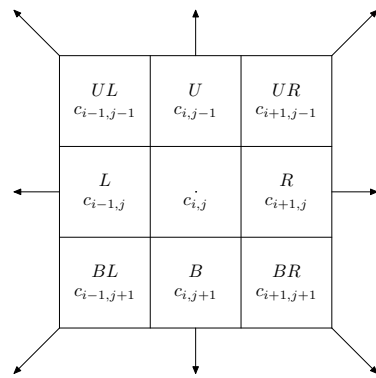
Označme osmici směrů dle obrázku 11a. Množinu těchto směrů nazvěme  $\text{dim}$ . Dále označme  $c_X$  (kde  $X \in \text{dim}$ ) jako sousední buňku buňky  $c$  ve směru  $X$ .

Autoři vycházejí z následující úvahy: Prochází-li buňkou  $c$  hrana ve směru  $X \in \text{dim}$ , pak má buňka  $c_X$  výrazně jinou barvu, než buňka  $c$  (viz obrázek 11b).

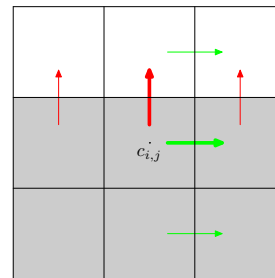


Obrázek 10: Ukázky jednoduchých filtrů. Vlevo původní obrázek, uprostřed obrázek po 5 generacích jednoduchého rozostřovacího filtru a v pravo obrázek po 8 generacích filtru pro zvýraznění tvarů ( $\epsilon = 1, 1$ ).

{img:Filters}



(a) 8 směrů



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

{img@imgDimensionsEdges}



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  $\epsilon_U(c) = 0$
- 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.

### 10.3 Ostraňování šumu

{subs:NoisRem}

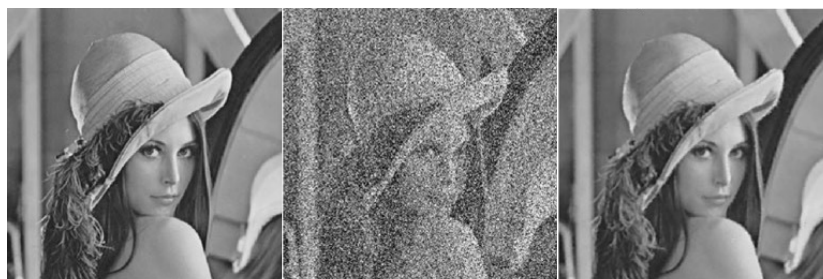
Odstraňování šumu je další častý problém, který je třeba při zpracování obrazů řešit. Pro studium technik odstraňování šumu se používá zašumnění tzv. impulzním šumem popř. šumem „sůl a pepř“. Zašumnění impulzním šumem nahradí stanovený počet pixelů náhodnými barvami. Šumění „sůl a pepř“ pak narazuje 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.

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

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



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

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

{img:Noises}

## 10.4 Rozpoznávání jednoduchých vzorů

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

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

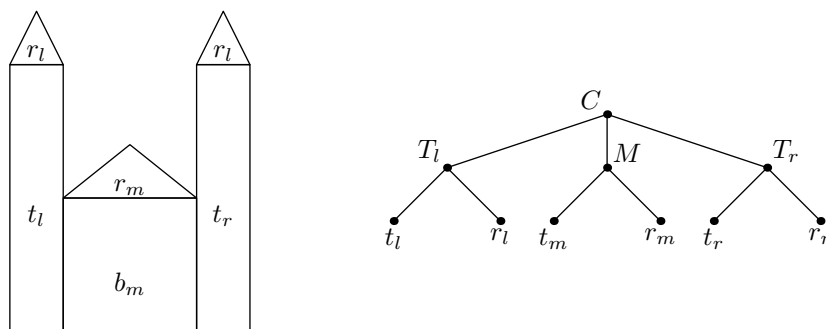
Zcela jiný přístup používají v [?]. Vzor nepovažují jako konkrétní kombinaci odstínů barev, ale jako část obrazu splňující určité vlastnosti. Konkrétně, skvrny jednolitě barvy, které jsou obklopeny různorodě zabarvenou plochou. Podrobnější popis (včetně ukázky výstupů) je uveden v kapitole \*.

## 10.5 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.

**Příklad 10.1.** Na obrázku 13 je vyobrazen složený geometrický útvar vyobrazující „budovu kostela“ a jemu odpovídající strom.

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



Obrázek 13: Příklad složeného geometrického tvaru a jeho stromu

{img:Geoms}

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.

1540

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. \*

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

**Definice 10.1** (Obdélník). Mějme čtyřúhelník  $g = ABCD$ . Pak tento čtyřúhelník je obdélník, pokud platí

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

**Definice 10.2** (Čtverec). Mějme obdélník  $g = ABCD$ . Pak tento obdélník je čtverec, pokud

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

**Definice 10.3** (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  $c$ “.

1550

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

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

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 10.4** („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 10.5** („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 10.6** („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 10.7** (Vztah „být nad“<sup>15</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.

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 popsany 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 10.2.** Uvažujme složený geometrický útvar  $C$  z obrázku 13. Pak automat  $A'$  bude nějak vypadat. \*.

<sup>15</sup>Zde si dovoluujeme značné zjednodušení. Vztah „být nad“ by měl být popsán například s využitím porovnávání y-ových souřadnic bodů.

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 smýšlejí, 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.

\*

## 1580 10.6 Detekce požárů

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

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 (světlo, pohyb v určitém směru, vlnění) spočtených z předchozích snímků. Přechodové pravidlo pak může být např. „pokud jsi ve stavu  $q_H$  a došlo k velkému posunu směrem nahoru a malému snížení světla, pak přejdi do stavu  $q_{VH}$ “. Přechodovou funkci pak navrhli statistickým pozorováním známých videosekvencí s požáry<sup>16</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. \*

## 11 Biologie a medicína

1600 Biologie a medicína jsou odvětví, které zpravidla disponují velikým množstvím dat, které je třeba zpracovat. Typicky, v datech získaných nějakým sledováním či měřeními 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.

### 11.1 Rozpoznávání řetězců DNA

Řetězec DNA, neboli deoxyribonukleová kyselina je organická makromolekula. Je tvořena sekvencí tzv. nukleotidů. Každý nukleotid obsahuje jednu ze čtyř nukleových bází, adeninu, guaninu, cytosinu a nebo thyminu. Nukleové báze se často značí po řadě A, G, C a T a celá DNA tak může být zapsána jako posloupnost těchto symbolů.

1610 Řetězce DNA kódují základní informace o živých organizmech a je proto snaha pochopit její strukturu.

<sup>16</sup>Dá se říci, že automat vznikl pomocí učení s učitelem

Použití fuzzy a pravděpodobnostních automatů s rozpoznáváním vzorců DNA je k dispozici v [?], [?], [?] a [?].

## 11.2 Biologické simulace

V [?] je prezentován model založený na buněčných fuzzy automatech modelující růst mořských řas. Funguje na velmi podobném principu jako problém městského růstu (viz kapitola \*).

1620 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“). Přechodová 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.

1630 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.

## 11.3 Analýza zdravotního stavu pacienta

1640 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).

Důležitým předpokladem, který automat musí splňovat, je tzv. vlastnost pozdržení vrcholu\*. Tato vlastnost říká, že výpočet automatu pro libovolný vstup nikdy nezkončí v prázdném fuzzy stavu, tj. výpočet se vždy bude muset 1650 nacházet v nenulovém stupni v alespoň některém stavu.

\*

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

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

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

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

\* \*  
\*

## 11.4 Analýza lékařských snímků

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

Proces rozpoznávání probíhá následujícím způsobem:

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

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

\*

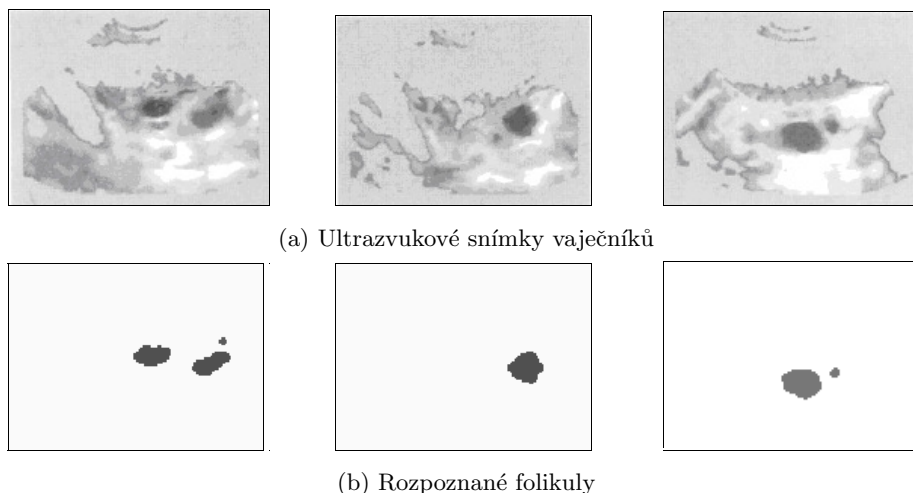
V [?] popisují, jak pomocí buněčných automatů rozpoznávat folikuly<sup>17</sup> v ultrazvukových snímcích vaječníků. Poukazují na to, že folikula je na snímku tvořena jednolitou svrnou stejné barvy, obklopena ostatní tkání (různorodě zbarvená plocha). Ukázka několika snímků je k dispozici na obrázku 14a. 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).

Přechodová funkce prvního automatů 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 14 jsou k nahlédnutí ukázky rozpoznávaných folikul pomocí této techniky.

<sup>17</sup>Folikula je dutinka ve vaječniku, v níž probíhá zrání vajíčka. \*



Obrázek 14: (převzato z [?], upraveno) Ukázky rozpoznávání folikul

{img:Follicles}

V [?] popisují způsob, jak pomocí fuzzy automatů sledovat vývoj kostí ruky dítěte. Z Röntgenového snímku ruky pomocí detekce hran obrází obrysy kostí. Ty pak pomocí techniky podobné rozpoznávání složených geometrických tvarů (kapitola \*) používají k určování stádia vývinu patřičné kosti.

## 11.5 Další aplikace

V [?], popř. i [?] analyzují signál z elektrokardiogramu (EKG). Využívají při tom techniky uvedené v kapitole \*.

V [?] (popř. i [?] \*) používají fuzzy automaty pro modelování fungování 1710 protézy dolní končetiny. Pomocí akcelerometrů sledují pohyb patřičné náhrady a pomocí známých vzorů spouštějí automat, který určuje, v jaké fázi pohybu protéza je.

## 12 Implementace vybraných problémů

V rámci této práce byly vybrané aplikace naimplementovány. Tato kapitola popisuje základní informace o způsobu, jak byly tyto aplikace naimplementovány.

Studium této kapitoly vyžaduje alespoň základní znalost objektového programování, nejlépe pak znalost programovacího jazyka Java.

### 12.1 Základní informace

Aplikace byly implementovány v programovacím jazyce Java na platformě Java 1720 Standart Edition. Projekt byl strukturován tak, aby bylo možné jej automaticky sestavit pomocí nástroje Apache Maven. Pro jeho sestavení a spuštění nejsou potřebné žádné dodatečné nástroje či knihovny.

Samotný projekt je rozdělen na moduly (podprojekty) odpovídající použitému typu automatu, tj. `fuzzy-automata` (nedeterministický fuzzy automat), `event-driven-fuzzy-automata`



(událostmi řízený fuzzy automat), `fuzzy-tree-automata` (fuzzy tree automat) a `cellular-fuzzy-automata`.

1730 Každý z těchto modulů obsahuje vždy základní, abstraktní, třídu implementující definici patřičného automatu (tj. datovou strukturu). Každý modul poté obsahuje třídu, která je jejím potomkem a rozšiřuje ji o implementaci vybraných algoritmů (např. výpočet, determinizace a podob.).

Každý modul tak může být používán jako softwarová knihovna. Pro snazší použití každý modul obsahuje obvykle několik spustitelných tříd (tj. tříd s metodou `main`), které zaobalují vybranou funkcionalitu modulu do spustitelného programu. Každý modul spravidla obsahuje také adresář `data/test` obsahující testovací data.

Implementované aplikace jsou umístěny v modulech podle typu automatu, se kterým pracují. Každá aplikace je implementována (obvykle) samostatnou třídou a obsahuje patřičné spustitelné třídy a testovací data.

1740 Projekt dále obsahuje modul `core`, který implementuje základní funkcionalitu společnou pro všechny moduly. Tedy implementaci abeced, symbolů, řetězců, stupňů pravdivosti, fuzzy množin a relací a podob. Implementuje také tzv. `TIMFILE`, což je speciální formát souboru navržený pro vstup a výstup dat z aplikace.

Všeckeré podrobnější informace jsou k nalezení v javadoc dokumentaci v samotném kódu. Nyní budou ve stručnosti popsány jednotlivé aplikace, které byly implementovány.

## 12.2 Korekce překlepů

1750 Detekce a korekce překlepů byla implementována na základě popisu v kapitole 7.3. Realizuje ji třída `TyposCorrecter` a spustí se pomocí `TyposCorrecterApp`. Proces korekce je konfigurován slovníkem (seznam korektních slov), popisem klávesnice a stupni v jakých má být akceptován stisk právě jedné extra klávesy, stisk více extra kláves, stisk jiné klávesy či vynechání klávesy.

Instance této třídy bere na svém vstupu textový řetězec tvořený jedním nebo více slovy oddělených mezerou. Výstupem je posloupnost slov dle následujícího klíče:

1. výstupem je slovo  $x$ , jestliže je vstupní slovo  $x$  ve slovníku
2. výstupem je slovo  $y$ , jestliže je vstupní slovo  $x$  nejpodobnější slovu  $y$ , které je ve slovníku
3. výstupem je slovo  $x$ , pokud žádné slovo ze slovníku mu není podobné

1760 Podobnost řetězců je určována pomocí deformovaných automatů rozpoznávající slova ze slovníku. Ukázka korekce překlepů (při výchozím nastavení) je uvedena v tabulce 4. Jak je v tabulce vidět, program má problém s krátkými slovy, kde se často ztrácí, avšak u dlouhých slov dokáže obnovit i velmi „poškozené slovo“.

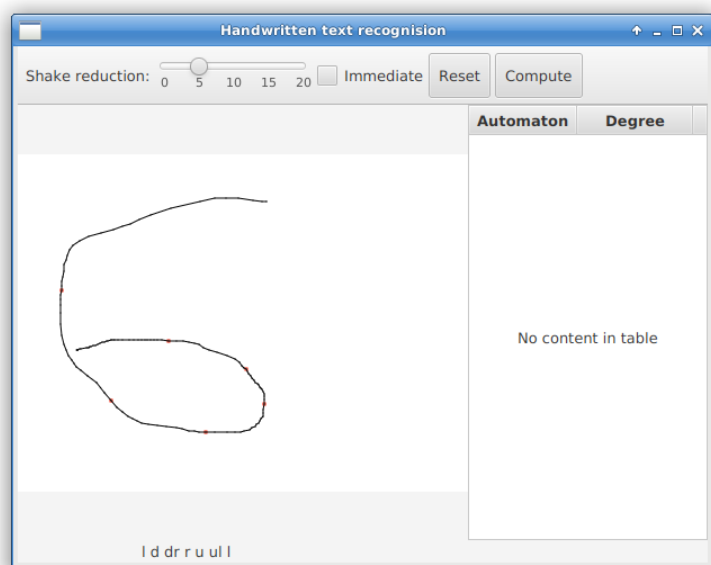
## 12.3 Rozpoznávání ručně psaného textu

Rozpoznávání ručně psaného textu bylo naimplementováno dle kapitoly 8.3. Tato aplikace je naimplementována jako interaktivní grafická aplikace spouštěná třídou `HandwrittenTextGuiApp`.

vstup	februacy	jaruanry	devmber	october	asdbril
výstup	february	january	december	october	april
vstup	maj	jana	poctober	asauguszt	mnobmvmert
výstup	march	may	october	august	november

Tabulka 4: Ukázka korekce překlepů při slovníku anglických názvů měsíců

{tab:TyposOut}



Obrázek 15: Ukázka aplikace pro rozpoznávání ručně psaného textu

{img:HandWritScreen}

V prostřední části okna aplikace se nachází plátno, kde je možno pomocí tahu myši psát. Na plátně se červeně zvýrazňují body, ve kterých se lámou segmenty. V stavovém řádku se zobrazuje řetězec, který napsanému tvaru odpovídá. V levé části okna se nachází tabulka s automaty a stupni pravdivosti, jak moc je jimi napsaný tvar přijímán. Nástrojová lišta pak obsahuje prvky pro konfiguraci a ovládání programu. Parametr *Shake reduction* udává minimální délku segmentu, zaškrťovací tlačítko *Immediate* zapíná automatické spuštění výpočtu při uvolnění tlačítka myši.

Seznam automatů se zadává při spuštění programu. Pro vygenerování automatu rozpoznávající vzor a jeho následnou deformaci lze použít spustitelné třídy `AutomatonOfWordApp` a `DeformAutomatonApp`. Pro testování bylo v adresáři s testovacími daty vytvořeno 10 vzorů odpovídajících číslicím 0 až 9 (každá ve více variantách).

Ukázka okna aplikace je na obrázku 15.

## 12.4 Detekce úplných $m$ -árních stromů

Detekce úplných  $m$ -árních stromů je naimplementována pomocí fuzzy tree automatů a to dle kapitoly 8.8. V adresáři s testovacími daty se nachází vzorové stromy pro úplný ternární a kvadrární strom. Testování úplnosti vstupu se

spouští třídou `TreeOnFTaRunner`.

## 12.5 Simulace spotřeby elektrického proudu

1790 Simulace spotřeby elektrického proudu byla naimplementována dle kapitoly 9.2. Výpočet realizuje třída `PowerConsumptionComputer`. Jejím vstupem jsou informace o spotřebičích v síti, jejich možných stavech a spotřebách. Dále pak seznam naměřených spotřeb elektrického proudu a dodatečné parametry.

Pomocí třídy `PowerConsumptionsToFEDA` jsou tato data transformována na fuzzy událostmi řízený automat a sekvenci fuzzy událostí. Automat je posléze spuštěn s těmito událostmi a výsledný stav je opět převeden na informace o spotřebičích a jejich stavech.

## 12.6 Zpracování obrazu

Naimplementovány byly také vybrané techniky pro zpracování obrazu. Realizovány byly pomocí buněčných fuzzy automatů.

1800 Byly implementovány některé grafické filtry uvedené v kapitole 10.1. Třída `ConvolutionalCFA` implementuje automat realizující obecnou konvoluci. Třída `SimpleBlurFilterAutomaton` pak pomocí konvoluce implementuje jednoduché rozostření. Třída `MyBEFilterAutomaton` implementuje BE filtr.

Třída `NoiseReductionAutomaton` implementuje metodu pro odstranění šumu z obrazu popsanou v kapitole 10.3. Pro přidání šumu je možné použít spustitelnou třídu `ConfigNoiserTool`. Pro import a export (převod mezi bitmapou a souborem konfigurace buněčného automatu) je slouží spustitelná třída `ImageConfigConverter`.

## 12.7 Metoda lisování dat

Bude doplněno, až budu mít nějaká data. \*