4 Fuzzy tree automaty

4.1 Zavedení

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

Fuzzy tree automaty vznikly fuzzyfikací „klasických" tree automatů. O „klasických" tree automatech je možné se dočíst více informací např. v [**?**], popř. [8] a [**?**]. Problematice fuzzy tree automatů se věnuje například [**?**], [14], [15], [**?**] a [**?**]. V této kapitole bude vycházeno z [**?**].

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

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

4.2 Stromy a pseudotermy

Definice 4.1 (Doména stromu)**.** *Mějme abecedu Σ uspořádanou pomocí $\leq$. Pak konečnou množinu $U \subseteq \Sigma^*$ nazvěme doména konečného stromu, pokud splňuje následující podmínky:*

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

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

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

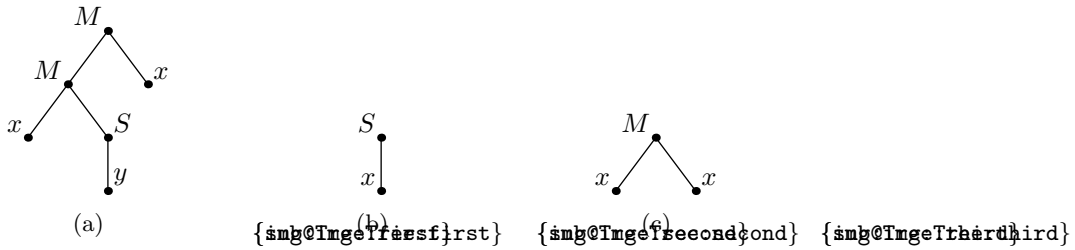
a množninu $U \setminus \overline{U}$ vnitřních uzlů.

Definice 4.2 (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 4.3 (Strom)**.** *Strom t nad částečně spořádnou abecedou (N, T) je zobrazení z domény U stromu do $(N \cup T)$ (psáno $t : U \to (N, T)$) takové, že*

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



Obrázek 5: Stromy k příkladu 4.1

Místo $t(w)$ budeme psát jen t.

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

Příklad 4.1. *Označme $T = \{x, y\}$ a $N = \{S, M\}$. Definujme doménu U_1 stromu pro abecedu $\Sigma = \mathbb{N}$ jako množinu řetězců $U_1 = \{\epsilon, 1, 11, 12, 121, 2\}$. Pak $\overline{U_1} = \{11, 121, 2\}$. Strom $t_1 : U_1 \to (T, N)$ 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 5a. Další ukázky stromů jsou na zbylých podobrázcích obrázku 5.

Stromy nám přirozeně reprezentují stromovou hierarchii. Pro nás bude ale občas vhodné mít lineární strukturu pro zápis téhož. Nadefinujeme si proto pseudotermy, protějšky termů predikátové logiky[1].

Definice 4.4 (Pseudoterm)**.** *Označme $D^p_{(N,T)}$ nejmenší podmnožinu $(N \cup T \cup \{(,)\})^*$ splňující následující podmínky[2]:*

- $T \subset D^p_{(N,T)}$
- *pokud $n > 0$, $A \in N$ a $t_1, \ldots, t_n \in D^p_{(N,T)}$, pak $A(t_1 \ldots t_n) \in D^p_{(N,T)}$*

Prvky množiny $t^p \in D^p_{(N,T)}$ nazývejme pseudotermy.

Poznámka 4.1. *Definice pseudotermu lze snadno přepsat do gramatiky. Vzhledem k tomu, že taková gramatika bude jistě bezkontextová, bude jazyk $D_{(N,T)}$ bezkontextový. Tento fakt bude mít důsledek na konstrukci fuzzy tree automatu.*

[1] Oproti termům predikátové logiky mají však jiný pohled na nulární funktory, které u pseudotermu neexistují

[2] Předpokládáme, že symboly závorek, (a) nejsou součástí $N \cup T$

{ex:PseTerms}

Příklad 4.2. *Pro částečně spořádanou abecedu (N, T) s předchozího příkladu můžeme za termy označit například: $t_1^p = y$, $t_2^p = S(x)$, $t_3^p = M(xx)$, $t_4^p = M(M(xS(y))x)$.*

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

Věta 4.1. *Pro každý strom $t \in D_{(N,T)}$ nad částečně spořádanou abecedou (N, T) existuje odpovídající pseudoterm $p(t)$.*

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

Věta 4.2. *Ke každému pseudotermu $p(t) \in D^p_{(N,T)}$ existuje odpovídající strom t.*

Důkaz. Opět dokážeme strukturálně:

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

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

□

Příklad 4.3. *Pseudoterm t_4^p z předchozího příkladu odpovídá stromu na obrázku 5a a pseudoterm t_3^p stromu 5c (a naopak).*

Máme tedy prokázano, 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 4.5 (Fuzzy jazyk stromů)**.** *Fuzzy množinu τ nad $D_{(N,T)}$ nazvěme fuzzy jazyk stromů.*

4.3 Fuzzy tree automat a jazyk jím rozpoznávaný

Začněme definicí fuzzy tree automatu.

Definice 4.6 (Fuzzy tree automat)**.** *Fuzzy tree automat A je pětice (Q, T, N, μ, F), kde:*

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

μ_x	q_1	q_2
ϵ	1	0

μ_y	q_1	q_2
ϵ	1	0

μ_S	q_1	q_2
q_1	0	1
q_2	0	0.4
q_1q_1	0	0,3

μ_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 μ fuzzy tree automatu {tab:MuOfFuzTreAut}

- *N je konečná množina neterminálích symbolů uzlů taková, že $N \cap T = \emptyset$*
- *$\mu : (N \cup T) \to \{f | f : (\mathcal{Q} \cup \{\epsilon\}) \times Q \to [0,1]\}$ je fuzzy přechodová funkce, kde $\mathcal{Q}$ je konečná podmnožina Q^+. Pro $X \in N$ je $\mu(X) = \mu_X$, kde μ_X je zobrazení z $\mathcal{Q} \times Q$ do $[0,1]$. Pro $a \in T$ je $\mu(a) = \mu_a$, kde μ_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 μ. Pro terminální symbol $a \in T$ nám μ_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 4.4. *Uvažujme množiny N a T stejné, jako v předchozích příkladech. Stanovme $Q = \{q_1, q_2\}$. Fuzzy množinu F položme rovnu $\{q_2\}$ a zobrazení μ je zaznačeno v tabulce 2. Pak $A = (Q, T, N, \mu, F)$ je fuzzy tree automatem.*

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

Definice 4.7 (Fuzzy přechodová funkce stromů)**.** *Pro strom $t \in D_{(N,T)}$ definujme fuzzy přechodovou funkci stromů jako zobrazení $\mu_t : Q \to [0,1]$ následovně:*

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

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

- *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 μ_a. Pokud vstupní strom není atomický, je přechod realizován ve stupni, který je dán stupněm přechodu neterminálního symbolu a stupňů příslušných podstromů.

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

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

Definice 4.8 (Jazyk rozpoznávaný)**.** *(zde bude doplněno: značení fuzzy jazyka) Fuzzy množinu $L(A)$ nad $D_{(N,T)}$ danou předpisem*

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

nazvěme fuzzy jazyk rozpoznávaný fuzzy tree automatem.

Příklad 4.5. *Uvažujme automat A z předchozího příkladu. Pak pro strom t_1 (kde $t_{1,1}$ značí levý podstrom kořene a $t_{1,2}$ pravý podstrom) z obrázku 5c platí $\mu_{t_1}(q_2)$:*

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

Pak tedy $A(t_1) = 0,6$. Pro stromy t_2 a t_3 z obrázku 5a a 5a platí $A(t_2) = 0.7$ a $A(t_3) = 1$.

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

Věta 4.3. *Pro jaždý fuzzy jazyk stromů existuje fuzzy tree automat, který ho rozpoznává.*

Důkaz. K dispozici v [**?**]. □

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

Definice 4.9 (Automat rozpoznávající strom)**.** *Mějme strom $t \in D_{(N,T)}$ („vzor“) kde $sub(t)$ značí množinu všech jeho podstromů. Pak jako fuzzy tree automat rozpoznávající strom t označme fuzzy tree automat $A = (Q, N, T, \mu, F)$, kde:*

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

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

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

4.4 Použití fuzy tree automatů

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

Konečnost množiny $\mathcal{Q}$ však teoreticky nemusí být nutná. Teoreticky by šlo jako vzory funkce μ_X namísto konkrétních řetězců nad Q použít například regulérní výraz popisující celou třídu takových řetězců (například `(q0 | q1)+`). To by však zkomplikovalo implementaci takového automatu a proto toto rozšíření nebude uvažováno.

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

4.5 Detekce úplných m-árních stromů

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

Definice 4.10 (Úplný strom)**.** *Úplný m-ární strom je takový strom, jehož každý uzel má buďto právě m potomků (vnitřní uzly) a nebo* 0 *(listové uzly).*

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

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

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

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

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

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

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

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

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

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

Věta 4.4. *Fuzzy tree automat $A = (Q, N, T, \mu, F)$, kde Q, N, T, μ a F jsou popsány výše, rozpoznává fuzzy jazyk úplných m-árních stromů.*

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

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

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

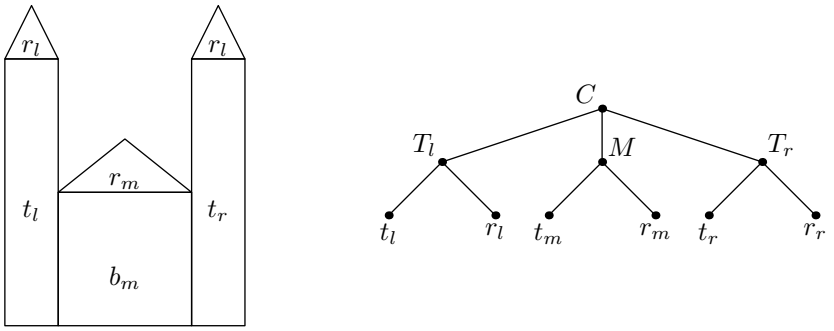
Předpokládejme, že platí $\mu_t(q_1) = \alpha_t$ pro všechny atomické stromy $t = a$, kde $a \in T$. Pak $\mu_t(q_1) = \alpha_v \wedge \bigwedge_{t' \in t} \mu_{t'}(q_1) = \alpha_v \wedge \bigwedge_{t' \in t} \alpha_{t'} = \alpha_t$ pro všechny stromy t tvořené vnitřními uzly. Pak tedy $\mu_t(q_1) =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

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

4.6 Složené geometrické útvary

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

Příklad 4.6. *Na obrázku 6 je vyobrazen složený geometrický útvar vyobrazující „budovu kostela“ a jemu odpovídající strom.*



Obrázek 6: Příklad složeného geometrického tvaru a jeho stromu {img:Geoms}

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

Pro rozpozánávání 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.

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

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

Definice 4.11 (Obdélník)**.** *Mějme čtyřúhelník $g = ABCD$. Pak tento čtyřúhelník je obdélník, pokud platí*

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

Definice 4.12 (Čtverec)**.** *Mějme obdélník $g = ABCD$. Pak tento obdélník je čtverec, pokud*

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

Definice 4.13 (Rovnoramenný trojúhelník)**.** *Mějme trojúhelník $g = ABC$. Pak tento trojůhelník je rovnoramenný, pokud*

$$\alpha_A = \alpha_B$$

Stranu AB nazývejme základna, strany AC a BC ramena.

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

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

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

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

S využitím fuzzy ekvivalence ε tka můžeme předchozí tři definice „fuzzyfikovat“:

Definice 4.14 („Fuzzy“ obdélník)**.** *Mějme čtyřúhelník $g = ABCD$. Pak tento čtyřúhelník je obdélníkem ve stupni $\gamma_r(g)$, kde*

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

Definice 4.15 („Fuzzy“ čtverec)**.** *Mějme geometrický útvar $g = ABCD$, který je obdélníkem ve stupni $\gamma_r(g)$. Pak tento obdélník je čtvercem ve stupni $\gamma_s(g)$, kde*

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

Definice 4.16 („Fuzzy“ rovnoramenný trojúhelník)**.** *Mějme trojúhelník $g = ABC$. Pak tento trojůhelník 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 fuzzyfikace jejich vzájemných vztahů. Pro vztah „být nad“ máme například:

Definice 4.17 (Vztah „být nad“[3])**.** *Mějme obdélník $r = ABCD$ a trojúhelník $q_t = EFG$. Pak „trojúhleník 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 „trojúhleník 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 geme-
750 trických 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 popsaný 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 μ' vznikla z μ nahrazením všech 1 (pro všechna $X \in (N \cup T)$) výrazem γ_X.

Příklad 4.7. *Uvažujme složený geometrický útvar C z obrázku 6. Pak automat A' bude nějak vypadat. (zde bude doplněno: domyslet to nějak!) .*

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

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

5 Buněčné fuzzy automaty

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

[3]Zde si dovolujeme značné zjednodušení. Vztah „být nad“ by měl být popsán například s využitím porovnávání y-ových souřadnic bodů.

5.1 „Bivalentní“ buněčný automat

Buněčné automaty a fuzzy buněčné automaty jsou výpočetní modely, které se koncepčně značně liší od klasických automatů. Dle [**?**] je jejich studium dokonce označováno za naprosto samostatné matematické paradigma. I přesto je v určitém smyslu 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 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 nadefinovanán 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 popsiuje přechodová funkce μ. Formálně tedy

Definice 5.1 (Bivalentní buněčný automat)**.** *Pro přirozené číslo m a konečnou množinu Q označme $A_m = (Q, \mu)$ jako dvoudimenzionální buněčný automat o rozměrech $m \times m$, kde Q je konečná množina stavů a μ přechodová funkce: $\mu : Q \times Q^k \to Q$ pro nějaké $1 \leq k \leq m^2$.*

Poznámka 5.1. *Buněčný automat se obvykle definuje jen předpisem pro přechodovou funkci, resp. výpisem přechodových pravidel. My se však budeme držet konceptu klasických automatů a budeme buněčný automat definovat jako strukturu.*

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

Přechodová funkce μ 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}:

$$\begin{aligned} round(c_{i,j}) = (&c_{i-1,j-1},\ c_{i,j-1},\ c_{i+1,j-1},\\ &c_{i-1,j},\qquad\quad c_{i+1,j},\\ &c_{i-1,j+1},\ c_{i,j+1},\ c_{i+1,j+1}) \end{aligned}$$

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

Přechodová funkce μ 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.

{ex:GameOfLife}

Příklad 5.1. *Typickým příkladem buněčného automatu je tzv. Hra života (Game of Life) (např. [?]). Jedná se o jednoduchý simulátor živého organizmu.*

Hra života uvažuje dvoustavovou možinu 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* μ *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 bunky, buňka umírá (ve smyslu „vyčerpání zdrojů“)*
3. *Je-li buňka* c *mrtvá, a v jejím okolí jsou přesně* 4 *živé bunky, je buňka oživena*
4. *Ve všech ostatních případech zůstává buňka buďto mrtvá nebo živá*

Označme

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

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

Pak je přechodová funkce μ *definována následovně:*

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

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

Definice 5.2 (Konfigurace buněčného automatu)**.** *Zobrazení* $C : [1, m] \times [1, m] \to Q$ *se nazývá konfigurace buněčného automatu.*

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

Definice 5.3 (Krok výpočtu a výpočet buněčného automatu)**.** *Binární relaci* $\vdash$ *na množině konfigurací buněčného automatu nazvěme krok výpočtu, pokud* $C \vdash C'$ *(*$(C, C') \in \vdash$*) a pro všechny* $c_{i,j} \in A$ *platí:*

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

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

Definice 5.4 (Konfigurace dosažitelná)**.** *Mějme konfiguraci C buněčného automatu. Pak o konfiguraci C' říkáme, že je dosažitelná z konfigurace C, pokud existuje výpočet z C k C', tj.*

$$C \vdash^* C'$$

V opačném případě říkáme, že konfigurace je nedosažitelná.

Vzhledem k tomu, že přechodová funkce buněčného automatu je deterministická, je zřejmě determinstický 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:

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

Nazývejme 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.

Příklad 5.2. *Ukázka výpočtu automatu $A_1 00$ realizující Hru života je na obrázku 7. Černá barva symbolizuje mrtvou buňku, bílá pak živou.*

5.2 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 determinstická 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“[4] (přechodová funkce by nebyla deterministická) nebo nepřešla do žádného (přechodová funkce by nebyla úplná).

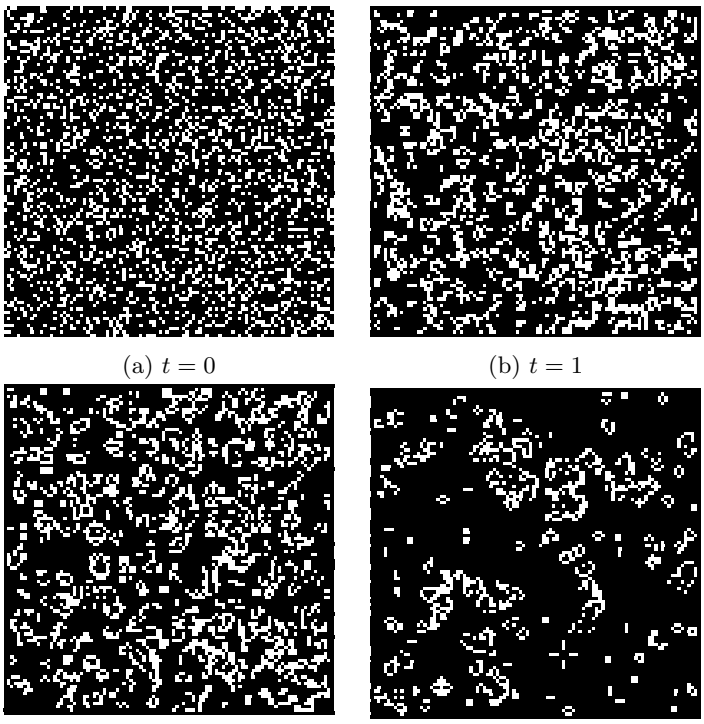
Vzhledem k tomu, že stejné chování budeme vyžadovat i u buněčného fuzzy automatu, „odstupňování“ přechodové funkce (zavední fuzzy přechodové funkce) by nemělo smysl[5].

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

Definice 5.5 ($[0, 1]$-buněčný fuzzy automat)**.** *Jako $[0, 1]$-buněčný fuzzy automat velikosti m budeme označovat bivalentní buněčný automat $A = (Q, \mu)$, kde $Q = [0, 1]$.*

[4] Jak bude ukázáno, tento předpoklad lze obejít

[5] 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.



(a) $t = 0$ (b) $t = 1$

(c) $t = 3$ (d) $t = 44$

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

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

$$\text{„buňka } c \text{ se nachází ve stavu 1“ ve stupni } q$$
$$\text{„buňka } c \text{ se nachází ve stavu 0“ ve stupni } 1 - q$$

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

Druhý způsob, jak definovat buněčný fuzzy automat, může pracovat s libovolnou množinou stavů. Fuzzy přístup se zde závádí dodatečně pomocí fuzzy množin (resp. výroků se stupni pravdivosti). Množině stavů je přiřazena jedna nebo více fuzzy množin. Každá tato fuzzy množina určitým způsobem charakterizuje stavy v ní obsažené. Přechodová funkce pak nepracuje přímo se stavy, ale pouze z těmito fuzzy množinami. Stav, do kterého má automat přejít je pak určen z fuzzy množiny (množin) popisující tento stav.

Takovýto buněčný fuzzy automat budeme nazývat buněčný automat s fuzzy logikou. Přechodová funkce μ tohoto automatu je určena množinou „If – Then“ pravidel. Zatímco u konečného buněčného automatu byla transformace množiny pravidel na přechodovou funkci triviální (viz příklad 5.1), při použití fuzzy logiky je situace složitější. Bude proto ještě před samotným nadefinováním automatu uveden postup pro konstrukci jeho přechodové funkce.

Mějme libovolnou množinu Q stavů, systém $\mathcal{D} \subseteq \mathcal{F}(Q)$ fuzzy množin nad touto množinou a zobrazení $\gamma : \mathcal{D} \to Q$ (defuzzyfikační funkce). Dále uvažujme množinu G fuzzy přechodových pravidel, tj. pravidel ve tvaru „If – Then“. Uvažujme pravidlo $g \in G$. Pravidlo g je v následujícím tvaru:

$$\text{Jestliže } f(c_1, \ldots, c_x) \text{ pak } c' = \gamma(\varsigma_y)$$

kde $c_1, \ldots, c_x \in Q$, $\varsigma_y \in D$ a f je nějaké zobrazení $f : Q^x \to [0, 1]$.

Označme předpoklad pravidla g jako formuli $pre(g) = g_x = f(c_1, \ldots, c_x)$. Budeme-li konkrétní hodnoty $e = (c_1, \ldots, c_x)$ považovat za pravdivostní ohodnocení, pak můžeme určit pravdivostní hodnotu (stupeň pravdivosti) $E_e(g_x)$ podmínky pravidla g při ohodnocení e.

Pro vektor e hodnot pak bude vybráno (a použito) pravidlo $g_i \in G$, které maximalizuje své ohodnocení, tj. pro které je $E_e(g_{i_x})$ největší.[6] Ohodnocení $E_e(g)$ pravidla g je pak rovno hodnotě $\gamma(\varsigma_y)$. Ohodnocení $E_e(G)$ množiny přechodových pravidel, tj. určení nového stavu na základě vektoru e stavů se tedy určí tímto předpisem

$$E_e(G) = con(g_i) \text{ kde } g_i \in G \text{ je takové, že } E_e(g_i) = \max_{g \in G} E_e(g)$$

(zde bude doplněno: ohodnocení pravidla vrací stav, ne stupeň pravdivosti. Chce to jiný termín, např. evaluace.) Tímto máme stanoveno, jak se aplikují fuzzy přechodová pravidla, a můžeme tedy přistoupit k definici buněčného automatu s fuzzy logikou.

Definice 5.6 (Buněčný automat s fuzzy logikou)**.** *Označme strukturu $A = (Q, G, \mathcal{D})$, kde Q je množina stavů, G množina „If – Then“ pravidel a $\mathcal{D}$ je*

[6] Je vhodné navrhnout pravidla tak, aby takové bylo vždy jen jedno. V opačném případě bude jedno vybráno.

fuzzy množina nad Q. Dále pak (Q, μ) formuje bivalentní buněčný automat s přechodovou funkcí μ:

$$\mu(c, round(c) = E_e(G),\ kde\ e = (c, round(c))$$

Značení. *V zápise pravidel se často pro vyšší názornost namísto zápisu $\varsigma_X(c)$ (kde $\varsigma_X \in \mathcal{D}$) používá zápis $c = X$.*

Příklad 5.3. *Příkladem buněčného automatu s fuzzy logikou může být například následující automat. Množina stavů bude obsahovat přirozená čísla z intervalu $[0, 150)$. Stanovíme čtveřici fuzzy množin ς_L, ς_M, a ς_H ve významu „hodnota q je nízká“, „hodnota q je střední“ a „hodnota q je vysoká“. (Přesnou definici těchto fuzzy množin vynecháme, není pro tento příklad důležitá.) Pravidla automatu pak mohou vypadat následovně:*

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

Provedme-li srovnání obou definic buněčných fuzzy automatů, zjistíme, že jsme vlastně obdrželi dva značně rozdílné výpočetní modely. To se projevuje i na jejich uplatnění. $[0, 1]$-Buněčný automat je vhodný více tam, kde pro popis přechodové funkce nevyžadujeme „If – Then“ pravidla ale jen funkční předpis. Oproti tomu buněčný automat s fuzzy logikou najde uplatnění spíše tam, kde máma data popsána logicky. Často se však oba přístupy kombinují (typicky se používá buněčný automat s fuzzy logikou spolu s nepodmíněnými pravidly).

5.3 Obecně k aplikacím

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

Co se buněčných fuzzy automatů týče, jejich aplikace nejsou tak rozšířené. Jedním z důvodů, proč tomu tak je, je velká podobnost bivalentních buněčných automatů a buněčných fuzzy automatů. Jak bylo uvedeno v předchozí podkapitole, $[0, 1]$-buněčný fuzzy automat je jen speciálním případem klasického byvalentního. Obdobně, reprezentovat přechodovou funkci „If – Then“ pravidly lze i bez použití fuzzy množin. Přechodová fuzzy pravidla jsou tak jen jiné pojmenování pro speciální třídu pravidel.

Například [**?**] používá čísla z intervalu $[0, 1]$ jako vstupní informaci. [**?**] používá automat podobný buněčnému automatu s fuzzy logikou, který navíc pracuje se stavy na intervalu $[0, 1]$. Obdobně, [3] používá buněčné automaty v kombinaci s určitou formou fuzzy logiky (avšak automat napovažují za buněčný fuzzy automat). V [**?**] je použit automat pracující na intervalu $[0, 255]$, který by šel snadnou modifikací konvertován na $[0, 1]$-buněčný fuzzy automat. Automat v [**?**] vykazuje určitou náhodnost a bylo by tak možné považovat jej za pravděpodobnostní.

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

Aplikace fuzzy automatů se dají rozdělit do dvou kategorií. Do první z nich spadají urbanistické simulace. Pomocí buněčných fuzzy automatů tak byly řešeny simulace hustoty dopravy [?], růstu mořských řas u pobřeží [?] či plánování elektrické rozvodné sítě [?]. Zdaleka nezčastější však bylo nasazení buněčných fuzzy automatů na řešení problému městského růstu. Z tohoto důvodu bude tomuto problému věnována samostatná podkapitola.

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

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

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

5.4 Problém městského růstu (urban growt problem)

Problém městského růstu je problém z oblasti urbanistiky. Řešením tohoto problému je co nejpřesnější predikce rozvoje městské zástavby na základě historických záznamů a současné situace. Ve zobecněné podobě se nemusí jednat jen o růst městské zástavby, ale například nárust 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 Krabiském moři [?], oblasti North Vancover 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 problém městského růstu je k dispozici např. zde: [2], [?] a [?].

Pojďme se nyní podívat, jak se buněčné fuzzy automaty pro řešení tohoto problému používají. Základní idea pro nasazení fuzzy automatů je následující: Stav zástavby reprezentujme 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, ...)
- 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:

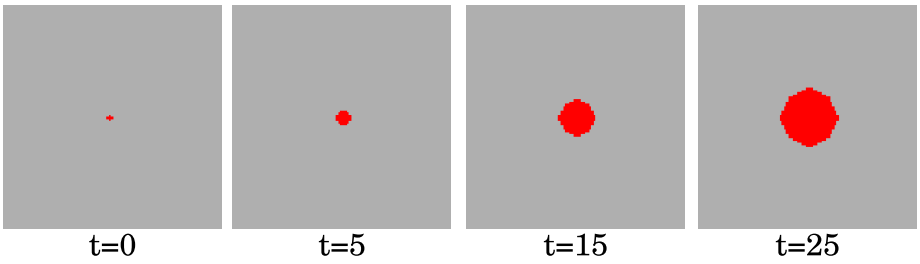
- Je-li vzdálenost parcely od centra města malá, pak růst zástavby bude velký
- Je-li vzdálenost od hlavní silnice velmi malá, pak růst zástavby bude malý a množství hluku bude velké
- Je-li vzdálenost od hlavní silnice velmi velká, pak růst zástavby bude malý
- Není-li lokalita atraktivní, pak růst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

Další ukázky pravidel jsou např. v [**?**], [12] a[**?**]. Ukázku toho, jak se chová automat při různých počátečních konfiguracích lze spatřit na obrázku 8.

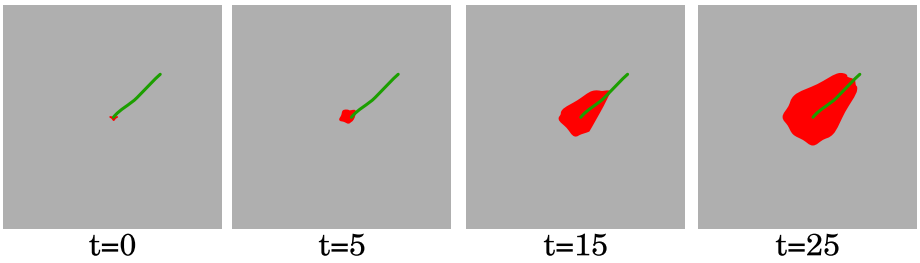
Na obrázku 9 je k vidění konkrétní ukázka městského růstu. Na obrázku jsou pro porovnání zobrazeny jak vypočtené stavy zástavby, tak i skutečné (upravené satelitní snímky). Na snímcích je patrné, že simulace rozvoje mezi lety 1987 a 1997 dosáhla poměrně přesných výsledků. Stejně tak, v simulaci růstu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

5.5 Zpracování obrazu

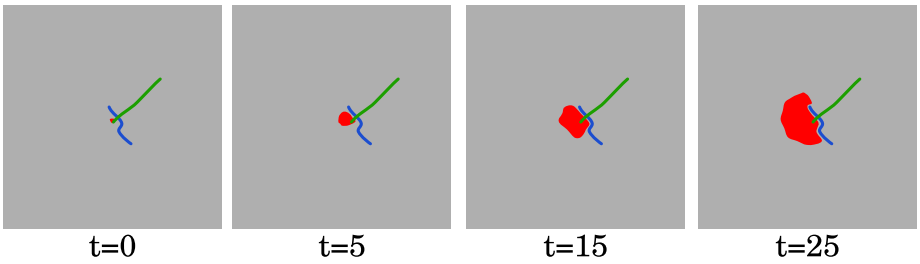
Zpracování obrazu je v dnešní době velmi populární informatická disciplína. Jak bude ukázáno, nasazení buněčných fuzzy automatů zde nachází značné uplatnění.



(a) Růst města bez dalších dodatečných podmínek



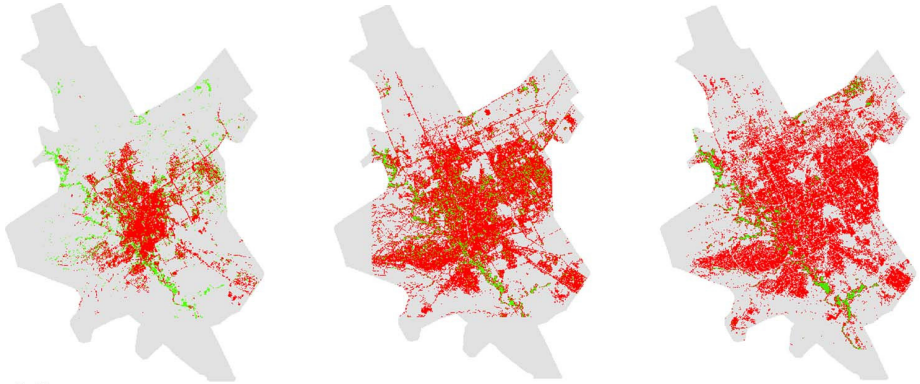
(b) Růst města podél silnice



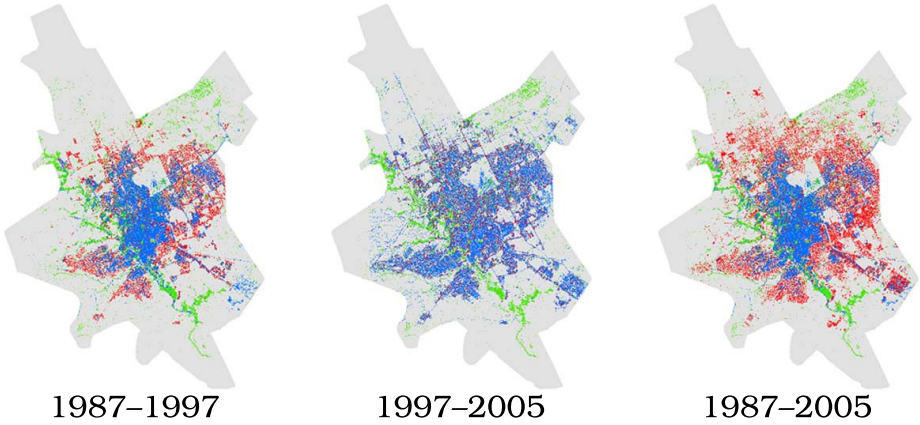
(c) Růst města brzděn řekou

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

{img-VarTransRuls}



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



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

Obrázek 9: (převzato z [**?**]) Ukázky simulace městského růstu {img-UrbGroProSample}

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

Uvažujme obraz jako mřížku $m \times m$ pixelů s odstíny šedi jako hodnotami od 0 do 1. Hodnota 0 značí černou, hodnota 1 bílou. Jinými slovy, barva (resp. stupeň šedi) pixelu odpovídá stupni pravdivosti tvrzení „pixel má bílou barvu“. Tato skutečnost nám umožňuje pracovat s obrazem pomocí fuzzy logiky.

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

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

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

je konvoluční maticí jednoduchého rozostření. Aplikuje se následujícím způsobem:

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

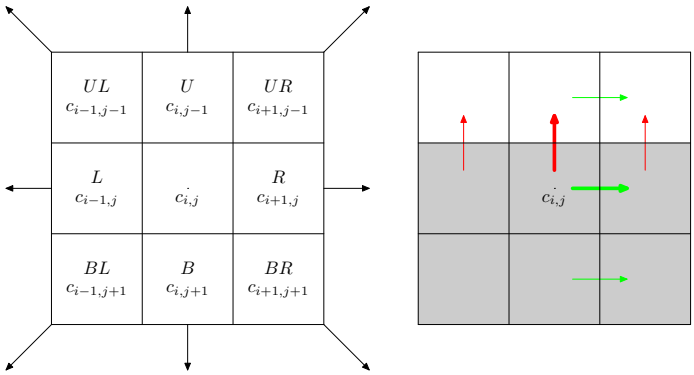
kde S je součet hodnot v matici B, tj. 16.

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

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

kde $\epsilon > 1$ je parametr udávají agresivitu zvýrazňování a $neighs_{i,j}$ je součet hodnot okolních buněk (viz příklad 5.1).

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



(a) 8 směrů

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

5.6 Hledání hran

Hledání hran je jednou ze základních technik zpracování obrazu. Hledání hran je často klíčové pro rozpoznávání vzorů v obrazech. V dnešní době existuje značné množství technik pro rozpoznávání hran [**?**]. V [**?**] je popsán poměrně elegantní způsob, jak hledání hran vyřešit pomocí buněčných fuzzy automatů.

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

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

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

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

Označme Δ' jako doplňek k Δ, tedy „buňky c a c' mají zcela shodnou barvu“. Pak můžeme stanovit fuzzy množiny ϵ_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 ϵ_X. Následně lze nadefinovat fuzzy množinu ϵ ve smyslu „buňkou c prochází hrana“ pomocí pravidel:

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

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

Obrázek 12: (převzato z [**?**]) Ukázka odstraňování šumu {img:Noises}

- 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 α tzv. Gas Diffusion Modelu, jednu z technik zaostřování obrazu.

5.7 Ostraňování šumu

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

V [**?**] je prezentována jednoduchá avšak efektivní technika, která kombinuje klasický bivalentní buněčný automat s buněčnýcm 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.

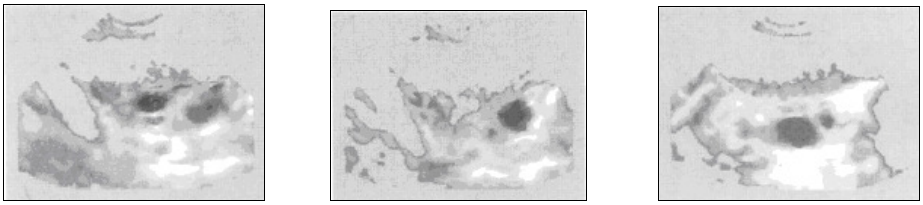
Jak autoři poukazují, tato technika je na odstraňování šumu velmi efektivní. Ukázky výsledků jsou na obrázku 12.

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

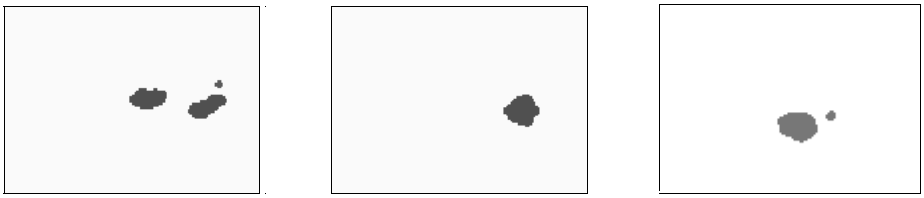
5.8 Rozpoznávání jednoduchých vzorů

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

Rozpoznávání vzorů v obrazech je však značně komplikované i pro buněčné fuzzy automaty. V [**?**] je popsán způsob, který popisuje rozpoznávání vzorů v



(a) Ultrazvukové snímky vaječníků



(b) Rozpoznané folikuly

Obrázek 13: (převzato z [**?**], upraveno) Ukázky rozpoznávání folikul {img:Follicles}

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 pužívají v [**?**]. Vzor nepovažují jako konkrétní kombinaci odstínů barev, ale jako část obrazu splňující určité vlastnosti.

Autoři metodu doporučují na vyhledávání vzorů v lékařských snímcích (např. ultrazvuk, Röntgen). Techniku demonstrují na ultrazvukových snímcích vaječníků. Poukazují na to, že folikula[7] je na snímku tvořena jednolitou svrnou stejné barvy, obklopena ostatní tkání (barevné přechody). Ukázka několika snímků je k dispozici na obrázku 13a. Obecně tak lze hovořit o „popředí“ (folikula) vystupující z „pozadí“.

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

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

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

[7] Folikula je dutinka ve vaječníku, v níž probíhá zrání vajíčka. *(zde bude doplněno: ocitovat: http://lekarske.slovniky.cz/pojem/folikul)*