

# Aplikace fuzzy a pravděpodobnostních automatů

Martin Jašek

12. září 2016 — ??

## Obsah

<b>1</b>	<b>Definice a značení</b>	<b>2</b>
<b>2</b>	<b>Fuzzy automaty, gramatiky a jazyky</b>	<b>4</b>
2.1	Jazyk rozpoznávaný fuzzy automatem . . . . .	5
2.2	Fuzzy a bivalentní regulární jazyky . . . . .	5
2.3	Fuzzy regulární výrazy . . . . .	5
<b>3</b>	<b>Rozpoznávání textových vzorů</b>	<b>5</b>
3.1	Formální zavedení problému . . . . .	5
3.2	Motivace k použití fuzzy automatů . . . . .	6
3.3	Automat rozpoznávající $\omega$ . . . . .	6
3.4	Podobnost symbolů . . . . .	8
3.5	Fuzzy symboly . . . . .	10
3.6	Editační operace . . . . .	11
3.7	Deformovaný automat . . . . .	16
3.8	Shrnutí . . . . .	17
<b>4</b>	<b>Fuzzy tree automaty</b>	<b>17</b>
4.1	Zavedení . . . . .	17
4.2	Stromy a pseudotermy . . . . .	18
4.3	Fuzzy tree automat a jazyk jím rozpoznávaný . . . . .	20
4.4	Použití fuzzy tree automatů . . . . .	22
4.5	Detekce úplných $m$ -árních stromů . . . . .	23
4.6	Složené geometrické útvary . . . . .	24
4.7	Jednoduchá klasifikace zvířat . . . . .	26
<b>5</b>	<b>Buněčné fuzzy automaty</b>	<b>26</b>
5.1	„Bivalentní“ buněčný automat . . . . .	27
5.2	Buněčné fuzzy automaty . . . . .	29
5.3	Obecně k aplikacím . . . . .	32
5.4	Problém městského růstu (urban growth problem) . . . . .	33
5.5	Zpracování obrazu . . . . .	35
<b>6</b>	<b>Konkrétní příklady</b>	<b>38</b>
6.1	Rozpoznání ručně psaného textu . . . . .	38
6.2	Detekce překlepů . . . . .	38

# 1 Definice a značení

Tato kapitola zatím poslouží jako „skladiště“ pro definice a zavedení značení pro ostatní kapitoly.

## Abecedy, řetězce, jazyky

Abecedy budou značeny standardně, tedy velkými řeckými písmeny (typicky  $\Sigma$ ). Řetězce pak malými písmeny ( $\omega, \alpha, \dots$ ). Jazyky velkými kaligrafickými písmeny. Jazyk přijímaný automatem  $A$  bude značen  $\mathcal{L}(A)$ .

## Fuzzy teorie

Fuzzy množiny a relace budou po vzoru [8] nejčastěji malými řeckými písmeny (obdobně jako jejich členské (angl. „membership“) funkce). Množinu všech fuzzy podmnožin množiny  $S$  budeme značit  $\mathcal{F}(S)$ .

## Deterministický bivalentní automat

(zde bude doplněno: zdroj: Eilenberg S.: Automata, Languages and Machines, Vol. A, Academic Press, New York, 1974. Pokud ji někde seženu (odkazuje se na ni Bel v [1])  
(zde bude doplněno: co citování definic? půlku jsem si vymyslel ...)

**Definice 1.1.** Konečný deterministický (bivalentní) automat je pětice  $A = (Q, \Sigma, \delta, q_0, F)$ , kde  $Q$  je konečná množina stavů,  $\Sigma$  je vstupní abeceda,  $\delta : Q \times \Sigma \rightarrow Q$  je přechodová funkce,  $q_0 \in Q$  je počáteční stav a  $F \subseteq Q$  je množina koncových stavů.

## Nedeterministický bivalentní automat

(zde bude doplněno: Značení převzato z FJAA, dohledat zdroj)

**Definice 1.2.** Konečný nedeterministický (bivalentní) automat je pětice  $A = (Q, \Sigma, \delta, I, F)$ , kde  $Q$  je konečná množina stavů,  $\Sigma$  je vstupní abeceda,  $\delta : Q \times \Sigma \rightarrow 2^Q$  je přechodová funkce,  $I \subseteq Q$  je množina počátečních stavů a  $F \subseteq Q$  je množina koncových stavů.

## Základní definice nedeterministického fuzzy automatu

Značení je převzato z [8] a lehce upraveno.

**Definice 1.3** (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.*

{def-ZaklDefNedFuzzAut}

**Definice 1.4** (Fuzzy stav). *Mějme nedeterministický fuzzy automat  $A$ . Pak jako fuzzy stav označujeme fuzzy podmnožinu jeho stavů, tj.  $V \in \mathcal{F}(Q)$ .*

{def-FuzzStav}

**Definice 1.5** (Aplikace fuzzy relace na fuzzy stav). *Mějme nedeterministický fuzzy automat  $A$  a fuzzy symbol  $V$ . Pak aplikací binární fuzzy relace  $R : Q \times Q \rightarrow [0, 1]$  na fuzzy stav  $V$  obdržíme fuzzy symbol  $V \circ R$  splňující pro každé  $p \in Q$ :  $(V \circ R)(p) = \max_{q \in Q} (V(q) \otimes R(q, p))$ .*

**Definice 1.6** (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  $V \in \mathcal{F}(Q)$  a symbol  $x \in \Sigma$  je  $\hat{\mu}(V, x) = V \circ \mu[x]$ .*

{def-PreFunFuzzStav}

**Poznámka 1.1.** *Označení  $\mu[x]$  je fuzzy relace, pro kterou platí:  $\mu[x](p, q) = \mu(p, x, q)$  pro všechna  $x \in \Sigma$  a  $p, q \in Q$ .*

**Definice 1.7** (Rozšířená přechodová funkce). *Mějme nedeterministický fuzzy automat  $A$ . Pak rozšířená přechodová funkce (fuzzy stavů) je fuzzy relace  $\mu^* : \mathcal{F}(F) \times \Sigma^* \rightarrow \mathcal{F}(F)$  ( zde bude doplněno: co je to  $F$ ? Nemá to být  $Q$ ?!) ) daná následujícím předpisem:*

{def-PreFunFuzzStav}

1.  $\mu^*(V, \epsilon) = V$  pro všechna  $V \in \mathcal{F}(Q)$
2.  $\mu^*(V, \alpha x) = \hat{\mu}(\mu^*(V, \alpha), x)$  pro všechna  $V \in \mathcal{F}(Q), \alpha \in \Sigma^*, x \in \Sigma$

{def-RetPriAut}

**Definice 1.8** (Řetězec přijímaný automatem). *Mějme nedeterministický fuzzy automat  $A$ . Pak řetězec  $\alpha \in \Sigma^*$  je automatem  $A$  přijat ve stupni*

$$A(\alpha) = \max_{q \in Q} (\mu^*(\sigma, \alpha)(q) \otimes \eta(q))$$

(zde bude doplněno: ověřit, dohledat, ozdrojovat)

{def-JazRozpAut}

**Definice 1.9** (Jazyk rozpoznávaný automatem). *Mějme nedeterministický fuzzy automat  $A$ . Pak fuzzy množinu  $\mathcal{L}(A)(\alpha) = A(\alpha)$  nad univerzem  $\Sigma^*$  nazýváme fuzzy jazyk rozpoznávaný automatem  $A$ .*

(zde bude doplněno: ověřit, dohledat, ozdrojovat)

## Nedeterministický fuzzy automat s $\epsilon$ přechody

{def-NedFuzzAutEpsPre}

**Definice 1.10** (Nedeterministický fuzzy automat s  $\epsilon$  přechody). *(zde bude doplněno: dohledat přesně, zkontrolovat a ozdrojovat) Nedeterministický fuzzy automat  $A$  je pětice  $(Q, \Sigma, \mu, \sigma, \eta)$ , kde  $Q$  je konečná množina stavů,  $\Sigma$  je abeceda,  $\mu$  je fuzzy přechodová funkce (fuzzy relace  $Q \times (\Sigma \cup \{\epsilon\}) \times Q \rightarrow [0, 1]$ ) a  $\sigma$  a  $\eta$  jsou po řadě fuzzy množiny nad  $Q$  počátečních, resp. koncových stavů.*

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

## Reprezentace fuzzy automatu

(zde bude doplněno: dohledat zdroje)

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

Přechodový diagram: Notace s lomítky např. zde: [10].

Tabulka: stav  $\times$  symbol nebo stav  $\times$  stav [12]?

## Konstrukce fuzzy automatu z konečného automatu

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

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

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

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

**Poznámka 1.2.** *Postup budeme provádět pro nedeterministické automaty. To jednak proto, že nedeterministické automaty jsou obecnější, než deterministické, a navíc, protože jsou v praxi využívány častěji. (zde bude doplněno: ozdrojovat, klidně někde, kde rozebírám determinismus vs. nedeterminismus)*

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

**Definice 1.11** (Fuzzy automat bivaletního automatu). *Mějme řetězec konečný nedeterministický automat  $A = (Q, \Sigma, \delta, I, F)$ . Pak nedeterministický fuzzy automat přijímající korespondující jazyk je automat  $A' = (Q, \Sigma, \mu, \sigma, \epsilon)$  kde pro všechna  $q_i, q_j \in Q$  a  $x \in \Sigma$ :*

{def-FuzzAutBivAut}

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

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

(zde bude doplněno: vymyslet nějaký fakt pěkný příklad)

## 2 Fuzzy automaty, gramatiky a jazyky

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

(zde bude doplněno: pojem „Lattice language“)

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

## 2.1 Jazyk rozpoznávaný fuzzy automatem

Věta 6.3 [16] (pro lattice monoid, není to někde jen pro  $[0, 1]$ ?).

Dle definice 4 [10] je fuzzy regulární jazyk fuzzy podmnožina bivalentního.

Automat s bivalentní  $\mu$  and  $\eta$  (a fuzzy  $\sigma$ ) taky rozpoznává fuzzy regulární jazyk [6]. Neřešil něco takového i Bel? Jinak řečeno, support konečný automat [10].

## 2.2 Fuzzy a bivalentní regulární jazyky

Univerzum fuzzy jazyka je regulární jazyk, pozorování 6.1 [16].

Stejně tak, zaříznutý jazyk ( $\alpha$ -řez jazyka) je také regulární, věta 2.2 [6].

Pumping lemma pro fuzzy regulární jazyky, lemma 4-7 pro různé typy automatů [10].

Uzávěrové vlastnosti fuzzy regulárních jazyků, např. [6].

## 2.3 Fuzzy regulární výrazy

LiPed-FuzzFinAutFuzzRegExMembValLattOrdMon, definice 5.1, 5.2 (+ opsat důkaz, že  $[0, 1]$  je lattice monoid) [16].

Algoritmus převodu reg na aut, [15]. Ale zdá se mi to až moc složité.

# 3 Rozpoznávání textových vzorů

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

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

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

**Poznámka 3.1.** *Pokud nebude uvedeno jinak, pojem „rozpoznávání textových vzorů“ bude v této kapitole zkracován jen na „rozpoznávání vzorů“.*

## 3.1 Formální zavedení problému

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

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

V této kapitole se však budeme zabývat pouze určováním podobnosti vzorového a pozorovaného řetězce. U každé instance problému budeme znát abecedu se kterou pracujeme a také vzor. Vzorem bude libovolný řetězec nad touto abecedou. Řešením tohoto problému pro nějaký, tzv. pozorovaný, vstupní řetězec bude úroveň podobnosti tohoto řetězce s vzorovým. Jako podobnost zde budeme uvažovat reálné číslo z intervalu  $[0, 1]$ , kde 0 znamená úplnou rozdílnost a 1 úplnou shodu.

**Poznámka 3.2.** *Vzorový řetězec budeme v této kapitole vždy značit  $\omega$ , pozorovaný pak  $\alpha$ .*

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

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

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

## 3.2 Motivace k použití fuzzy automatů

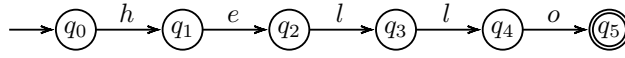
Klasická teorie automatů vznikla jako nástroj pro zpracování textových řetězců. Z tohoto důvodu je rozpoznávání textových vzorů jejím základním výsledkem. Automaty obecně jsou nástroje sloužící pro rozhodování, zda-li řetězec odpovídá vzoru automatem reprezentovaném. Použití pro rozpoznávání řetězcového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

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

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

## 3.3 Automat rozpoznávající $\omega$

Klíčovým pro rozpoznávání vzorů (chceme-li využívat fuzzy automaty) je bivalentní automat rozpoznávající vzorový řetězec. Tedy automat takový, který



Obrázek 1: Automat rozpoznávající **hello**

{diag-AutRozpHell}

přijímá jediný řetězec  $\omega$  a všechny ostatní zamítá. Nyní si takovýto automat zkonstruujeme.

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

Automat bude v každém kroku konzumovat symboly ze vstupního řetězce a porovnávat je se symboly vzorového řetězce na odpovídajících pozicích. Pokud dojde ke shodě na všech pozicích, automat dojde do koncového stavu a sledovaný řetězec přijme. Pokud se symboly shodovat nebudou, automat nebude mít definován žádný odpovídající přechod, kterým by pokračoval ve výpočtu, a řetězec tak zamítne.

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

**Definice 3.1** (Automat rozpoznávající  $\omega$  (deterministický)). *Mějme řetězec  $\omega$  délky  $n$  nad abecedou  $\Sigma$ . Automat rozpoznávající  $\omega$  je pak konečný automat  $A(\omega) = (Q, \Sigma, \delta, q_0, F)$  takový, že jeho množina stavů  $Q$  se sestává z právě  $n$  stavů  $q_0, \dots, q_n$ ,  $q_0$  je počáteční stav,  $F = \{q_n\}$  množina koncových stavů a  $\delta$  je přechodová funkce definována pro všechna  $0 \leq k < n$  následovně:*

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

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

**Příklad 3.1.** *Příklad automatu rozpoznávající řetězec  $\omega = \text{hello}$  se nachází na obrázku 1.*

My však budeme potřebovat fuzzy automat rozpoznávající  $\omega$ . To znamená, že musíme nejdříve automat z předchozí definice převést na nedeterministický a poté dle definice 1.11 k němu zkonstruovat odpovídající fuzzy automat.

{def-AutRozpOme}

**Definice 3.2** (Automat rozpoznávající  $\omega$  (nedeterministický)). *Mějme řetězec  $\omega$  nad abecedou  $\Sigma$  z předchozí definice. Nedeterministický automat rozpoznávající  $\omega$  je pak konečný automat  $A'(\omega) = (Q, \Sigma, \delta, I, F)$  takový, že jeho množina stavů  $Q$  je stejná jako v předchozí definici, dále  $I = \{q_0\}$  je množina počátečních a  $F = \{q_n\}$  množina koncových stavů a  $\delta$  je přechodová funkce definována pro všechna  $0 \leq k < n$  následovně:*

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

Následuje vytvoření fuzzy automatu.

{def-FuzzAutRozpOme}

**Definice 3.3** (Fuzzy automat rozpoznávající  $\omega$ ). *Mějme řetězec  $\omega$  nad abecedou  $\Sigma$  délky  $n$ . Fuzzy automat rozpoznávající  $\omega$  je pak automat  $A''(\omega)$  vytvořený z nedeterministického automatu rozpoznávající  $\omega$  (definice 3.2) dle definice 1.11. Bude to tedy automat  $A''(\omega) = (Q, \Sigma, \mu, \sigma, \epsilon)$  kde*

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

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

### 3.4 Podobnost symbolů

Nezákladnější technika pro zanesení neostrého (stupňovitého) rozpoznávání je s využitím podobnostní relace symbolů. Tato technika byla přejata z [3]. Myšlenkou této techniky je, že symbol v pozorovaném řetězci může být snadno zaměněn za jiný, podobný, jemu odpovídající v řetězci vzorovém.

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

**Příklad 3.2.** Jako příklad podobnostní relace (nad abecedou písmen anglické abecedy) může posloužit například vzdálenost patřičných kláves na klávesnici. V takovém případě by určitě platilo kupříkladu  $s(a, s) > s(a, d) > s(a, l)$ . Protože klávesy  $A$  a  $S$  jsou si blíže (a tudíž symboly  $a$  a  $s$  jsou si „podobnější“) než například  $A$  a  $D$  či  $A$  a  $L$ .

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

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

{def-AutPracGS}

**Definice 3.4** (Automat pracující s  $s$ ). Uvažujme, že máme nedeterministický automat  $A$  a relaci podobnosti symbolů  $s$ . Pak k automatu  $A$  můžeme zkonstruovat fuzzy automat  $A'$ , který navíc pracuje s  $s$ . Takový automat bude zkonstruován dle definice 1.11 s tím rozdílem, že přechodová funkce  $\mu$  bude definována pro všechna  $q_i, q_j \in Q$  a  $x \in \Sigma$  následovně:

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

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



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

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

**Definice 3.5** (Automat rozpoznávající  $\omega$  pracující s  $s$ ). *Mějme abecedu  $\Sigma$ , řetězec  $\omega$  nad touto abecedou a fuzzy relaci  $s$  nad touto abecedou. Dle definice 3.2 můžeme zkonstruovat nedeterministický bivalentní automat  $A(\omega)$  rozpoznávající  $\omega$ . Jako automat rozpoznávající  $\omega$  pracující s  $s$  označme nedeterministický fuzzy automat  $A'(\omega)$ , který byl z automatu  $A(\omega)$  vytvořen podle definice 3.4.*

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

Následuje jednoduchý příklad takového automatu.

{ex-AutRozp0mePodSym}

**Příklad 3.3.** *Mějme abecedu  $\Sigma = \{a, b, c, d\}$ . Dále uvažujme relaci podobnosti symbolů  $s$  takovou, že*

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

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

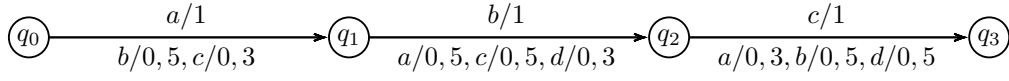
$$s = \begin{pmatrix} 1,0 & 0,5 & 0,3 & 0,0 \\ 0,5 & 1,0 & 0,5 & 0,3 \\ 0,3 & 0,5 & 1,0 & 0,5 \\ 0,0 & 0,3 & 0,5 & 1,0 \end{pmatrix}$$

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

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

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

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



Obrázek 2: Automat rozpoznávající abc pracující s

{diag-AutRozpABCPracGS}

### 3.5 Fuzzy symboly

Fuzzy symbol je technika využívající podobnosti symbolů. Ve své podstatě se jedná o téže techniku jak v předchozí podkapitole, jen je na ni nahlíženo jinak. Oproti podobnosti symbolů je použití fuzzy symbolů komplikovanější, avšak umožňuje jednoduše tuto techniku kombinovat s jinými. Princip fuzzy symbolů byl přejat z [9].

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

**Definice 3.6** (Fuzzy symbol). *Mějme abecedu  $\Sigma$  a fuzzy relaci  $p \subseteq \Sigma \times \Sigma$ . Pak pro každý symbol  $y \in \Sigma$  definujeme fuzzy symbol  $\tilde{y}$  symbolu  $y$  jako fuzzy množinu nad  $\Sigma$  takovou, že pro všechna  $x \in \Sigma$  platí*

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

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

**Definice 3.7** (Abeceda fuzzy symbolů). *Mějme abecedu  $\Sigma$  a fuzzy symboly  $\tilde{y}$  pro všechna  $y \in \Sigma$ . Pak množinu všech těchto fuzzy symbolů nazvěme abeceda fuzzy symbolů abecedy  $\Sigma$  a označme  $\tilde{\Sigma}$ . Tedy  $\tilde{\Sigma} = \{\tilde{y} \mid y \in \Sigma\}$ .*

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

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

**Definice 3.8** (Řetězec fuzzy symbolů). *Mějme abecedu  $\Sigma$  a nějaký řetězec  $a_1 \dots a_n = \alpha \in \Sigma^*$ . Pak definujeme  $\tilde{\alpha} = \tilde{a}_1 \dots \tilde{a}_n$  jako řetězec fuzzy symbolů řetězce  $\alpha$ .*

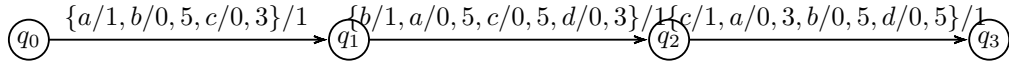
V této fázi jsme schopni plnohodnotně pracovat s řetězci fuzzy symbolů a konstruovat je z řetězců nad abecedou  $\Sigma$ . Nyní přejdeme k návrhu fuzzy automatu, který bude s fuzzy symboly pracovat. Stejně jako u podobnosti symbolů i fuzzy symboly mohou být aplikovány na libovolný typ automatu. Vytvoříme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

{def-AutPracFuzzSym}

**Definice 3.9** (Automat pracující s fuzzy symboly). *Mějme nedeterministický fuzzy automat  $A$ . Pak fuzzy automat  $\tilde{A}$  pracující s fuzzy symboly vytvoříme tak, že v definici automatu  $A$  nahradíme  $\Sigma$  za  $\tilde{\Sigma}$ .*

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

Formální zavedení automatu pracujícího s fuzzy symboly je intuitivní, jedná se jen o formalitu. Abychom však využili potenciál fuzzy symbolů, je třeba



Obrázek 3: Automat rozpoznávající abc pracující s s

{diag-AutRozpOmePraFuzSym

pozměnit výpočet automatu. Proces jeho výpočtu se změní ve fázi výpočtu přechodové funkce fuzzy stavů. Připomeňme, že ta je definována (definice 1.7) jako fuzzy relace  $\hat{\mu}$  přiřazující každému fuzzy stavu  $V$  a fuzzy symbolu  $x$  fuzzy stav dle předpisu

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

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

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

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

**Definice 3.10** (Automat rozpoznávající  $\omega$  pracující s fuzzy symboly). *Mějme abecedu  $\Sigma$ , řetězec  $\omega$  nad touto abecedou a abecedu fuzzy symbolů  $\tilde{\Sigma}$ . Dle definice 3.3 můžeme zkonstruovat fuzzy automat  $A(\omega)$  rozpoznávající  $\omega$ . Následně pak podle definice 3.9 automat  $\tilde{A}(\omega)$  rozpoznávající  $\omega$  pracující s fuzzy symboly.*

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

**Příklad 3.4.** *Mějme abecedu  $\Sigma$ , vzorový řetězec a podobnostní relaci  $p = s$  stejné jako v příkladu 3.3. Na obrázku 3 je zobrazen diagram automatu  $\tilde{A}(\omega)$  rozpoznávající  $\omega$  pracující s fuzzy symboly.*

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

### 3.6 Editační operace

Další technikou pro podobnostní porovnávání pozorovaného a vzorového řetězce je s využitím editačních operací. Tato technika byla přejata z [8]. Základní idea této techniky spočívá v trojici jednoduchých editačních operací, jejíž složením jsme schopni popsat transformaci pozorovaného řetězce na vzorový. Množství transformace pak udává podobnost pozorovaného a vzorového řetězce.

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

**Definice 3.11** (Editační operace). *Mějme abecedu  $\Sigma$ , uvažujme množinu  $E = (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\}) \setminus \{(\epsilon, \epsilon)\}$ . Pak každou dvojici  $(x, y) = z \in E$  nazvěme editační operace. Speciálně pak, pro všechna  $x, y \in \Sigma$ ,  $(x, y) \in E$  znamená nahrazení symbolu  $x$  symbolem  $y$ ,  $(x, \epsilon) \in E$  znamená odebrání symbolu  $x$  a*

naopak  $(\epsilon, y) \in E$  pak vložení symbolu  $y$ . Navíc jako editační operaci uvažujeme i všechny dvojice  $(x, x) \in E$  (pro každé  $x \in \Sigma$ ) symbolizující „žádnou editaci“.

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

Editační operace jsou tedy tři a to náhrada symbolu, vložení symbolu a odebrání symbolu. Například řetězec **hallo** vznikl záměnou **e** za **a** v řetězci **hello**. Obdobně, řetězec **hellow** vznikl přidáním **w** na konec a řetězec **helo** odebráním (prvního nebo druhého) symbolu **l**.

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

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

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

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

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

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

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

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

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

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

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

**Definice 3.12** (Vyrovnání řetězců [8]). *Jako množinu všech vyrovnání  $G(\alpha, \omega)$  řetězce  $\alpha$  na řetězec  $\omega$  (kde  $(\epsilon, \epsilon) \neq \alpha, \omega \in \Sigma^*$ ) označme takovou množinu  $\{\zeta \in E^+ \mid \zeta \text{ splňuje vlastnosti 1., 2. i 3.}\}$*

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

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

Pro určení podobnosti na základě vyrovnání řetězců budeme potřebovat znát míry pravdivosti editačních operací. Vstupem pro výpočet podobnosti řetězců tak bude navíc binární fuzzy relace  $R$  nad množinou všech editačních operací ( $E$ ), udávající stupeň akceptovatelnosti každé z možných editačních operací. S touto znalostí můžeme nadefinovat relaci podobnosti řetězců, tzv. fuzzy míru dvojice řetězců  $\alpha$  a  $\omega$ .

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

{def-FuzzMir}

**Definice 3.13** (Fuzzy míra[8]). Mějme binární fuzzy relaci  $R$  nad  $\Sigma \cup \{\epsilon\}$ . Pak jako fuzzy míru mezi řetězci  $\alpha, \omega \in \Sigma^*$  (značenou  $S_{\Sigma, R, \otimes}$ ) označme fuzzy relaci danou následujícím předpisem:

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

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

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

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

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

{def-AutRozpCaLL}

**Definice 3.14** (Automat rozpoznávající  $\mathcal{L}$  [8]). Mějme binární fuzzy relaci  $R$  nad  $\Sigma \cup \{\epsilon\}$  (stejná jako v definici 3.13). Pak pro vzorový řetězec  $a_1 a_2 \dots a_n = \omega \in \Sigma^*$  označme  $M_{\Sigma, R, \otimes}(\omega)$  automat rozpoznávající jazyk  $\mathcal{L}(\omega)$  dle definice 1.10, takový, že

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

- (f)  $\mu(q, q', \epsilon) = 0$  pro všechny  $q, q' \in Q$  nesplňující předchozí dva body
3. množina počátečních stavů  $\sigma: \sigma(q_0) = 1$  a pro všechny ostatní  $q_0 \neq q' \in Q$ :  $\sigma(q') = 0$
4. množina koncových stavů  $\eta: \eta(q_n) = 1$  a pro všechny ostatní  $q_n \neq q' \in Q$ :  $\eta(q') = 0$

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

**Věta 3.1.** *Mějme binární fuzzy relaci  $R$  nad  $\Sigma \cup \{\epsilon\}$  (stejná jako v definici 3.13) a vzorový řetězec  $\omega \in \Sigma^*$ . Pak pro automat  $M_{\Sigma, R, \otimes}(\omega)$  sestavený dle předcházející definice a fuzzy míru  $S_{\Sigma, R, \otimes}$  platí následující rovnost*

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

*Důkaz.* Kompletní důkaz je k nalezení v [8]. □

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

**Příklad 3.5.** *Uvažujme abecedu  $\Sigma = \{a, b, c\}$  a vzorový řetězec  $\omega = abc$ . Zkonstruujeme automat, který bude akceptovat ve stupni 0.5 náhradu symbolu  $x$  symbolem  $s$  ním v abecedě sousedícím. Navíc uvažujme vložení symbolu  $a$  ve stupni 0.2 a odebrání symbolu  $c$  ve stupni 0.1. Tedy, relace  $R$  bude vypadat následovně:*

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

Dle definice 3.14 můžeme sestavit automat  $M_{\Sigma, R, \otimes}$ . Jako  $\otimes$  použijme produktovou  $t$ -normu. Získáme tak automat  $M_{\Sigma, R, \otimes} = (Q, \Sigma, \mu, \sigma, \epsilon)$  takový, že:

1.  $Q = \{q_0, q_1, q_2, q_3\}$
2.  $\mu = \{$ 
  - (a)  $(q_0, q_0, a)/0.2, (q_1, q_1, a)/0.2, (q_2, q_2, a)/0.2, (q_3, q_3, a)/0.2,$
  - (b)  $(q_0, q_1, a)/1, (q_1, q_2, a)/0.5,$   
 $(q_0, q_1, b)/0.5, (q_1, q_2, b)/1, (q_2, q_3, b)/0.5,$   
 $(q_1, q_2, c)/0.5, (q_2, q_3, c)/1,$
  - (d)  $(q_0, q_0, \epsilon)/1, (q_1, q_1, \epsilon)/1, (q_2, q_2, \epsilon)/1, (q_3, q_3, \epsilon)/1,$
  - (e)  $(q_2, q_3, \epsilon)/0.1$

$\}$  (přechody dle bodů (b) a (d) v definici jsou s nulovým stupněm a ve výpisu jsou vynechány)
3.  $\sigma = \{q_0/1, q_1/0, q_2/0, q_3/0\}$

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

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

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

Nyní si na pár řetězcích zkuzme ukázat platnost věty 3.1. Dle definice 1.8 spočítáme stupeň, v jakém automat náš testovací rozpoznává řetězec  $\alpha$ :

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

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

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

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

- řetězec  $\alpha = ab$ :

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

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

- řetězec  $\alpha = bbb$ :

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

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

- řetězec  $\alpha = abaca$ :

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

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

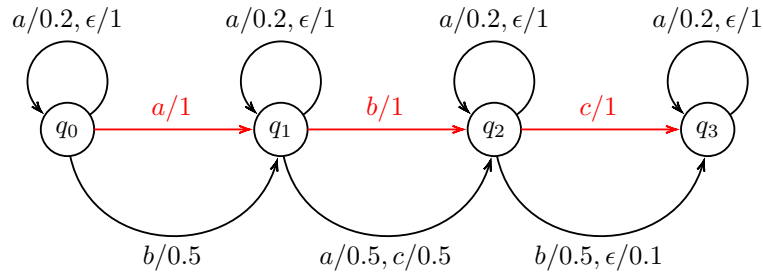
- řetězec  $\alpha = cba$ :

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

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

...

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



Obrázek 4: Přechodový diagram automatu z příkladu 3.5

{img-AutRozpCaLL}

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

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

### 3.7 Deformovaný automat

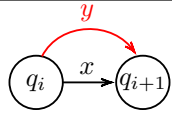
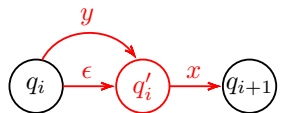
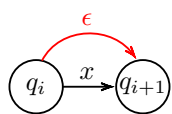
Dalším ze způsobů, jak přijímat řetězec podobný vzorovému je s využitím deformovaného (fuzzy) automatu. Tato technika využívá tzv. deformovaného automatu, neboli automatu který byl stanoveným způsobem upraven, neboli deformován. Tato technika byla přejata z [4].

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

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

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



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

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

{tbl-DefAutDef}

### 3.8 Shrnutí

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

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

## 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. [7] a [?]. Problematicke fuzzy tree automatů se věnuje například [?], [13], [14], [?] 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 je 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 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

(zde bude doplněno: Pokud se budu potřebovat zbavit stránky, nebo problémů s Stromy vs. pseudotermy, tak ty pojmy spojit do pojmu třeba „pseudostrom“ (= pseudoterm, u kterého budeme „uvažovat“ operace nad stromy) a vše patřičně překopat.)

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

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

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

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

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

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

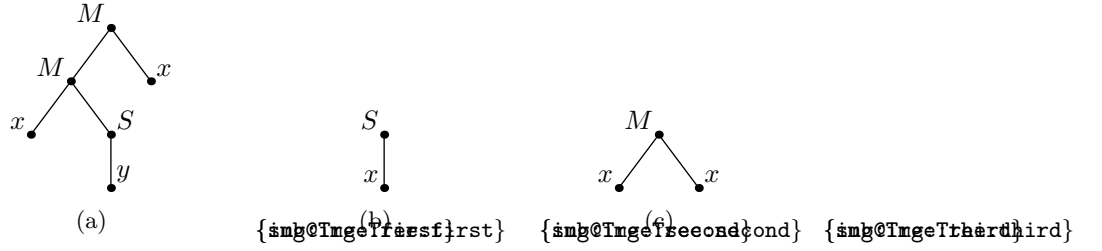
{def:Tree}

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

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

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

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



Obrázek 5: Stromy k příkladu 4.1

{img:Tree}

{ex:Trees}

**Příklad 4.1.** Označme  $T = \{x, y\}$  a  $N = \{S, M\}$ . Definujme doménu  $U_1$  stromu pro abecedu  $\Sigma = \mathbb{N}$  jako množinu řetězců  $U_1 = \{\epsilon, 1, 11, 12, 121, 2\}$ . Pak  $\bar{U}_1 = \{11, 121, 2\}$ . Strom  $t_1 : U_1 \rightarrow (T, N)$  nad  $(T, N)$  pak může vypadat například takto:

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

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

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

**Definice 4.4** (Pseudoterm). Označme  $D_{(N,T)}^p$  nejmenší podmnožinu  $(N \cup T \cup \{(\cdot, \cdot)\})^*$  splňující následující podmínky<sup>2</sup>:

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

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

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

{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  $p(t)$ .

<sup>1</sup>Oproti termům predikátové logiky mají však jiný pohled na nulární funktory, které u pseudotermu neexistují

<sup>2</sup>Předpokládáme, že symboly závorek,  $( \ )$  nejsou součástí  $N \cup T$

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

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

*Důkaz.* Opět dokážeme strukturálně:

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

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

$\square$

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

### 4.3 Fuzzy tree automat a jazyk jím rozpoznávaný

Začneme definicí fuzzy tree automatu.

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

- $Q$  je konečná množina symbolů stavů
- $T$  je konečná množina terminálních symbolů uzlů
- $N$  je konečná množina neterminálních symbolů uzlů taková, že  $N \cap T = \emptyset$
- $\mu : (N \cup T) \rightarrow \{f | f : (\mathcal{Q} \cup \{\epsilon\}) \times Q \rightarrow [0, 1]\}$  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.

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

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

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

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

{tab:MuOfFuzTreAut}

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

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

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

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

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

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

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

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

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

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

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

nazvěme fuzzy jazyk rozpoznávaný fuzzy tree automatem.

**Příklad 4.5.** Uvažujme automat  $A$  z předchozího příkladu. Pak pro strom  $t_1$  (kde  $t_{1,1}$  značí levý podstrom kořene a  $t_{1,2}$  pravý podstrom) z obrázku 5c 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 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 každý fuzzy jazyk stromů existuje fuzzy tree automat, který ho rozpoznává.

*Důkaz.* K dispozici v [?]. □

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

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

- $Q = \{q_{t'} \mid t' \in \text{sub}(t)\}$  (každému podstromu odpovídá jeden stav)
- $\mu_a(q_a) = 1$  pro všechna  $a \in T$
- $\mu_X(w, q_{t'}) = 1$  pro všechny  $t' \in \text{sub}(t)$ , 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  $Q$ , pro kterou je definována přechodová funkce  $\mu_X$  neterminálů  $X \in N$  však přináší omezení na aritu každého uzlu, která tak vždy musí být konečným číslem.

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

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 určíme míru jeho úplnosti. Jinými slovy stupeň pravdivosti výroku „uzel  $v$  má  $m$  potomků“. Tuto míru označme  $\alpha_v$  a můžeme ji určit následujícím vztahem

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

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

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

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

Ný 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  $w \in \{q_1^i \mid 1 \leq i \leq m\}$ . Množina koncových stavů  $F$  bude rovna  $\{q_1\}$ .

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

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

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

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

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

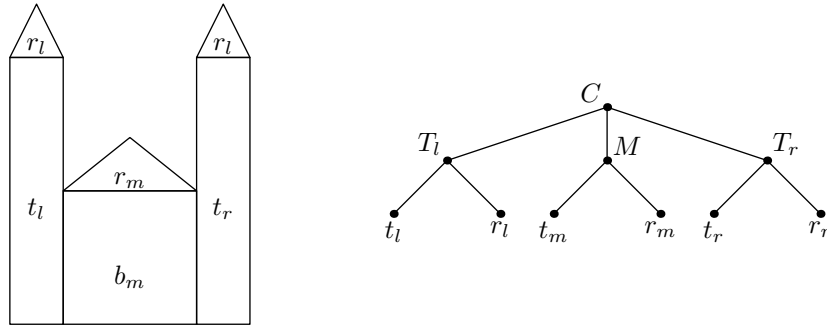
Předpokládejme, že platí  $\mu_t(q_1) = \alpha_t$  pro všechny atomické stromy  $t = a$ , kde  $a \in T$ . Pak  $\mu_t(q_1) = \alpha_v \wedge \bigwedge_{t' \in t} \mu_{t'}(q_1) = \alpha_v \wedge \bigwedge_{t' \in t} \alpha_{t'} = \alpha_t$  pro všechny stromy  $t$  tvořené vnitřními uzly. Pak tedy  $\mu_t(q_1)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

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

## 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 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.

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

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

**Definice 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 trojúhelník je rovnoramenný, pokud*

$$\alpha_A = \alpha_B$$

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

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

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

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

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 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 trojúhelník je rovnoramenný ve stupni  $\gamma_i(g)$ , kde*

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

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

**Definice 4.17** (Vztah „být nad“<sup>3</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  $t$ .*

<sup>3</sup>Zde si dovoluujeme značné zjednodušení. Vztah „být nad“ by měl být popsán například s využitím porovnávání y-ových souřadnic bodů.

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

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

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 popsáný výše.

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

**Příklad 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“ (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 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.

## 4.7 Jednoduchá klasifikace zvířat

Autor práce přichází s využitím fuzzy tree automatů pro jednoduchou klasifikaci zvířat. Každé zvíře je charakterizováno prostředím, kde obvykle žije nebo je možné jej spatřit (voda, souš, vzduch), povrchem těla (kůže, šupiny, srst, peří), seznamem končetin (nohy, ruce, křídla, ploutve) a ocasem (žádný, krátký, dlouhý).

Jmenovaným vlastnostem byly přiřazeny neterminální symboly *Env, Surf, Ext, Tail* a terminální symboly po řadě *water, land, air, leather, scales, hair, feather, leg, hand, wing, fin, short, long*. Zvíře jako celek je pak reprezentováno terminálem *Animal*, který má čtveřici potomků, a to právě *Env, Surf, Ext* a volitelně *Tail* (zde bude doplněno: další volitelné „atributy“? *umí plavat? ve slané/sladké vodě? loví? létá? ...*). Ukázka stromů reprezentující zvířata pes, pták, netopýr, tyranosaurus, ryba, žába, krokodýl, pavouk, (chlupatá tarantule/čmelák?), vydra/bobr, velryba/delfín, létající ryba je na obrázku (zde bude doplněno: obrázek!!!).

## 5 Buněčné fuzzy automaty

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

## 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 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 konečnou se simulovanou nekonečností (odpovídající hrany jsou logicky propojeny (*zde bude doplněno: popsáno to nějak lépe*)) či konečné (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 5.1** (Bivalentní buněčný automat). *Pro přirozené číslo  $m$  a konečnou množinu  $Q$  označme  $A_m = (Q, \mu)$  jako dvoudimenzionální buněčný automat o rozměrech  $m \times m$ , kde  $Q$  je konečná množina stavů a  $\mu$  přechodová funkce:  $\mu : Q \times Q^k \rightarrow Q$  pro nějaké  $1 \leq k \leq m^2$ .*

**Poznámka 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. (zde bude doplněno: Tady [?] je definice, ale je pro nás moc zbytečná a tak jsme si vymysleli svoji).*

Fakt, že nějaká buňka  $c$  se nachází ve stavu  $q$  budeme značit prostým  $c = q$ . Přechodová funkce  $\mu$  přiřazuje buňce  $c$  stav  $q'$  na základě aktuálního stavu  $q$  a stavu dalších  $k$  buněk. V této práci se budeme vždy za tyto buňky uvažovat okolní buňky. Označme  $round(c_{ij})$  jako okolí buňky  $c_{ij}$ . Nejpoužívanějším okolím, které se používá, je Mooreovo okolí, které je definováno jako sousedních 8 buněk buňky  $c_{ij}$ :

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

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  $\mu$  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.

{ex:GameOfLife}

**Příklad 5.1.** Typickým příkladem buněčného automatu je tzv. Hra života (Game of Life) (např. [?]). Jedná se o jednoduchý simulátor živé 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“. Přejchodová funkce  $\mu$  je dána následujícími pravidly:

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

Označme

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

jako počet živých buněk sousedících s  $c_{i,j}$ . Pak předchozí pravidla mohou být zapsána (kde  $c' = \mu(c, round(c))$ )

1. Je-li  $c_{i,j} = 1$  a  $neighs_{i,j} < 2$  pak  $c'_{i,j} = 0$
2. Je-li  $c_{i,j} = 1$  a  $neighs_{i,j} > 3$  pak  $c'_{i,j} = 0$
3. Je-li  $c_{i,j} = 0$  a  $neighs_{i,j} = 4$  pak  $c'_{i,j} = 1$
4. Jinak  $c'_{i,j} = c_{i,j}$

Přejchodová funkce  $\mu$  je tedy definována následovně:

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

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

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

Podobně jako u klasických automatů můžeme hovořit o počáteční konfiguraci. Ta – na rozdíl od klasických automatů – může být libovolná. Koncová konfigurace však pro buněčné automaty neexistuje. Výpočet buněčného automatu je totiž z definice nekonečný. Můžeme však uvažovat konfiguraci dosažitelnou. Pro její zavedení 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  $c \in A$  platí:*

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

*Tranzitivní uzávěr  $\vdash^*$  (zde bude doplněno: fakt?) 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 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. Označíme-li si počáteční konfiguraci  $C^{(0)}$ , pak můžeme výpočet automatu zapsat:

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

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

Výpočet buněčného automatu, resp. jeho konfigurace lze znázornit 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 buněčného automatu  $A$  realizující Hru života je na obrázku 7. Černá barva symbolizuje mrtvou buňku, bílá pak živou.*

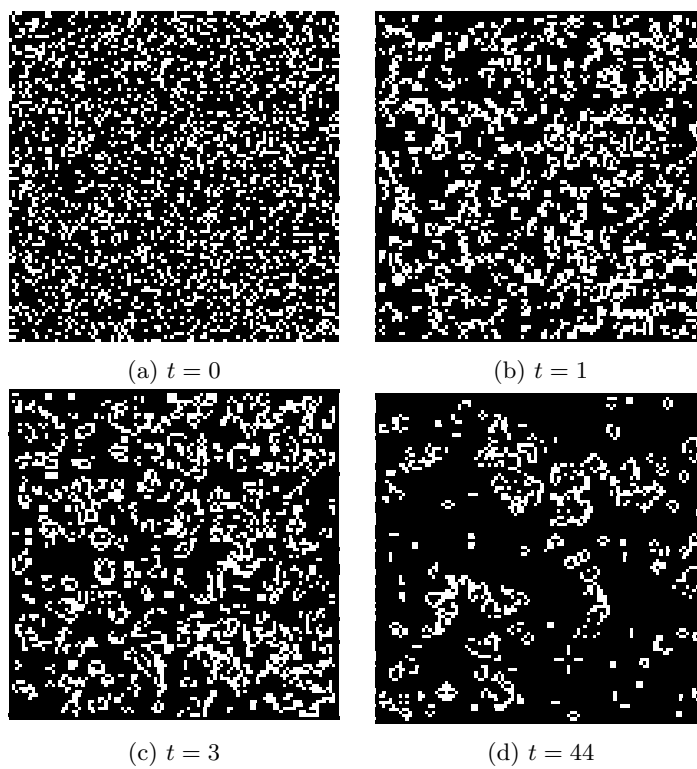
## 5.2 Buněčné fuzzy automaty

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 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>4</sup> (přechodová funkce by nebyla deterministická) nebo nepřešla do žádného (přechodová funkce by nebyla úplná).

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

<sup>4</sup>Jak bude ukázáno, tento předpoklad lze obejít

<sup>5</sup>Teoreticky by bylo možné nahradit stav buňky fuzzy stavem buňky. Takováto úprava by však výrazně zvýšila výpočetní složitost výpočtu automatu a například také znesnadnila grafické znázornění jeho konfigurací a proto zde nebude uvažován.



Obrázek 7: Výpočet buněčného automatu „Hra života“ s náhodnou počáteční konfigurací

{img:GameOfLife}

V praxi se lze setkat se dvěma způsoby, jak je buněčný fuzzy automat definován. První, jednodušší z nich, je prostý 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  $Q = [0, 1]$ .*

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

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

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

Druhý způsob, jak definovat buněčný fuzzy automat, používá libovolnou množinu stavů. Fuzzy přístup se zde závadí 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. Přejížděcí 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. Přejížděcí funkce  $\mu$  tohoto automatu je určena množinou „If – Then“ pravidel. Zatímco u konečného buněčného automatu byla transformace množiny pravidel na přejížděcí 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 přejížděcí funkce.

(zde bude doplněno: V CEL0 PRÁCI: ověřit konečné a neprázdné množiny !!!)  
(zde bude doplněno: „If – Then“ pravidla, odkud jsou, zdůraznit, že je to vlastně implikace. Jo a používají se všude možně (mj. např. v knowledge base a obecně data mining?), takže to nadefinovat asi někde nahoře v úvodních epizodách. Budu značit pre (předpoklad) a con (důsledek).)

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

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

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

Označme předpoklad pravidla  $g$  jako formulí  $pre(g) = g_x = f(c_1, \dots, c_x)$ . Budeme-li konkrétní hodnoty  $e = (c_1, \dots, c_x)$  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.<sup>6</sup> Ohodnocení  $E_e(g)$

<sup>6</sup>Je vhodné navrhnout pravidla tak, aby takové bylo vždy jen jedno. V opačném případě bude jedno vybráno.

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

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

(zde bude doplněno: ohodnocení pravidla vrací stav, ne stupeň pravdivosti. Chce to jiný termín, např. evaluace.) Tímto máme stanoveny, 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 fuzzy množina nad  $Q$ . Dále pak  $(Q, \mu)$  formuje bivalentní buněčný automat s přechodovou funkcí  $\mu$ :

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

**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  $\varsigma_L, \varsigma_M, \text{ a } \varsigma_H$  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  $q_{i-1,j-1}^{(t)} = L$ , 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  $q_{i,j}^{(t+1)} = L$

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 je třeba mít přesnější kontrolu nad přechodovou funkcí. Tento typ automatu nám totiž dovoluje použít libovolnou matematickou funkci. Oproti tomu buněčný automat s fuzzy logikou najde uplatnění spíše tam, kde máme data popsána logicky. Přechodovou funkci totiž tvoří sada pravidel a formulí, které přechody popisují.

### 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, environmentalistickým 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. Fuzzy množiny v tomto případě jen pravidla činí více srozumitelná.

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

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

Aplikace fuzzy automatů se dají rozdělit do dvou kategorií. Do první z nich spadají urbanistické simulace. Pomocí buněčných fuzzy automatů tak byly řešeny problémy hustoty dopravy [?], růstu mořských řas u pobřeží [?] či plánování elektrické rozvodné sítě [?]. Zdaleka nejběžnější 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 [?][?]. Technikám zpracování obrazu pomocí buněčných fuzzy automatů bude věnována samostatná podkapitola.

Co se pravděpodobnostních buněčných automatů týče, ty nacházejí uplatnění všude tam, kde je třeba dodat náhodnost. Kromě toho také pro městský růst [?] [?]. Dále pak například pro náhodný generátor, generátor šumu, pohyb částic, difuze částic, 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 precizní kalibraci a přizpůsobení na míru konkrétní instanci problému. Většina zdrojů proto automaticky kombinuje několik technik do hromady. Například [?] [?] využívají neuronové sítě a [?] [?] učící automat. Většinou je také nutná přítomnost experta na danou problematiku.

## 5.4 Problém městského růstu (urban growt problem)

Problém městského růstu je problém z oblasti geoinformatiky. Řešením tohoto problému je co nejpresnější predikce rozvoje městské zástavby na základě historických záznamů a současné situace. Ve obecněné 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 městský růst.

Základní idea pro nasazení (fuzzy) automatů je následující: Bude-li stav zástavby reprezentovat stav (resp. obecně konfiguraci) automatu, pak růst zástavby bude odpovídat přechodům mezi těmito stavy (resp. konfiguracemi). Najdeme-li vhodný způsob, jak stav zástavby reprezentovat pomocí konfigurací automatu a budeme-li schopni navrhnout odpovídající přechodovou funkci, pak budeme moci modelovat rozvoj zástavby do budoucnosti.

Vzhledem k tomu, že města bývají často rozlehlá, bylo by nepraktické pro problém městského růstu klasický (nedeterministický) fuzzy automat. Bude proto výhodné využít buněčný fuzzy automat, který dokáže modelovat téměř neomezené množství buněk. To je v souladu s tím, že rozvoj městské zástavby probíhá (v ideálním případě) rovnoměrně.

Praxe ukazuje, že (fuzzy) buněčné automaty mohou být skutečně použity pro modelování městského růstu. 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 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 problém městského růstu je k dispozici např. zde: [2], [?] a [?].

Pojďme se nyní podrobněji podívat, jak se konstruuje fuzzy buněčný automat pro problém městského růstu. Důležité je, že návrh buněčného automatu je v praxi značně empirická záležitost a neobejde se bez experta na urbanizaci a studovanou lokalitu. V první fázi je nutné získat urbanistická data o studované lokalitě. Taková data lze získat například pomocí satelitních snímků či z katastrálních záznamů. Data o zástavbě je posléze potřeba zakódovat do mřížek, které budou reprezentovat konfigurace buněčného automatu. Je tedy třeba stanovit jak měřítko (velikost parcely odpovídající jedné buňce automatu), tak i zobrazení, přiřazující zástavbě stupeň z intervalu  $[0, 1]$ . Zobrazení přiřazující zástavbě stupeň může být stanoveno například jako

- poměr zástavby a zeleně na parcele
- (normalizovaný) počet obyvatel žijících na parcele
- (normalizovanou) výšku nejvyšší budovy
- (normalizované) množství produkovaného hluku/světelného znečištění/výfukových plynů/...

Máme-li sestaveny konfigurace automatu reprezentující stavy zástavby v jednotlivých časových okamžicích, je třeba navrhnout přechodovou funkci. U buněčného automatu reprezentující městský růst je obvykle standardní přechodová funkce (*zde bude doplněno: jak se tomu říká oficiálně?*) nedostačující. Růst zástavby obvykle závisí na mnohem více parametrech, než je aktuální stav zástavby na okolních parcelách. Mezi další ukazatele, které ovlivňují růst zástavby patří například:

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

Za posvícení stojí, že některé ukazatele jsou neměnné v čase (např. vzdálenost od centra města nebo podloží). To značně usnadňuje V závislosti na rozhodnutí experta a následném experimentálním ověření je třeba i tyto ukazatele „fuzziﬁkovat“. Následně je možné sestavit pravidla definující přechodovou funkci automatu.

**Příklad 5.4.** Označme  $v(t, c)$ ,  $\rho(t, c)$ ,  $\tau(t, c)$ ,  $\nu(r, c)$ ,  $\sigma(t, c)$  jako „stupeň zástavby“, „stupeň vzdálenosti k hlavní silnici“, „stupeň vzdálenosti k nejbližší zastávce autobusu“, „stupeň obydlivosti sousedních 8 buněk“, „stupeň znečištění ovzduší“ buňky  $c$  v čase  $t$ . Pak pravidla automatu by mohla vypadat například takto:

- $v(t + 1, c) = 1 - \sigma(t, c)$
- pokud  $\tau(t, c) \leq 0,2$  a  $\rho(t, c) \leq 0,2$  pak  $v(t + 1, c) = 0$ , jinak  $v(t + 1, c) = \tau(t, c) \vee \rho(t, c)$
- $$v(t + 1, c) = \begin{cases} 4\mu(t, c) & \text{pokud } 0 \leq \mu(t, c) < 1/4 \\ 1 & \text{pokud } 1/4 \leq \mu(t, c) < 3/4 \\ 4(1 - \mu(t, c)) & \text{pokud } 3/4 \leq \mu(t, c) \leq 1 \end{cases}$$
- $\sigma(t + 1, c) = \nu(t, c) + (1 - \rho(t, c))$

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

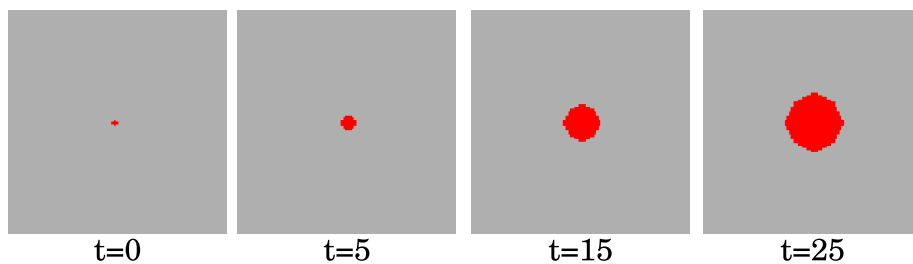
Na obrázku 9 je k vidění konkrétní ukázka městského růstu. Na obrázku jsou pro porovnání zobrazeny jak vypočtené stavy zástavby, tak i skutečné (upravené satelitní snímky). Na snímcích je patrné, že simulace rozvoje mezi lety 1987 a 1997 dosáhla poměrně přesných výsledků. Stejně tak, v simulaci růstu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

Je potřeba zdůraznit, že obyčejná sada přechodových pravidel bývá silně nedostačující. Pro obdržení optimálního výsledku je nutné přechodová pravidla rozšířit o další, sotsifikovanější nástroj nebo nástroje. V praxi se tak nejvíce používají umělé neuronové sítě [?], [?], genetické algoritmy [?], často však jejich kombinace. V [?] používají algoritmus fuzzy  $k$ -means pro určení okolních buněk pomocí shlukování. Návrh automatu tak vyžaduje značné množství experimentů a konzultací s expertem.

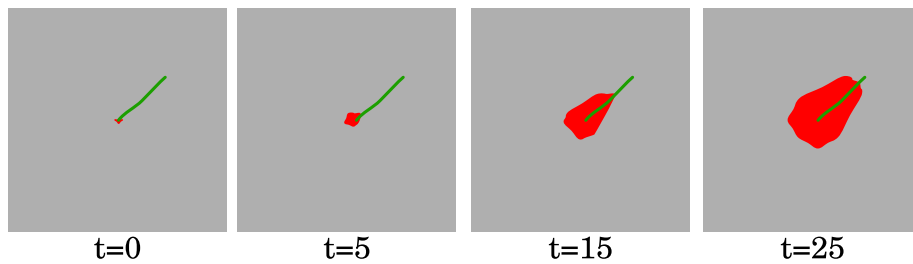
## 5.5 Zpracování obrazu

Hledání hran je jednou ze základních technik zpracování obrazu. Rozpoznávání hran je například klíčové pro rozpoznávání vzorů v obrazech. V dnešní době existuje velké množství algoritmů, pro nalezení hran. *(zde bude doplněno: najít nějaký článek je porovnávající)* Použití fuzzy buněčných automatů však přichází s elegantním řešením problému.

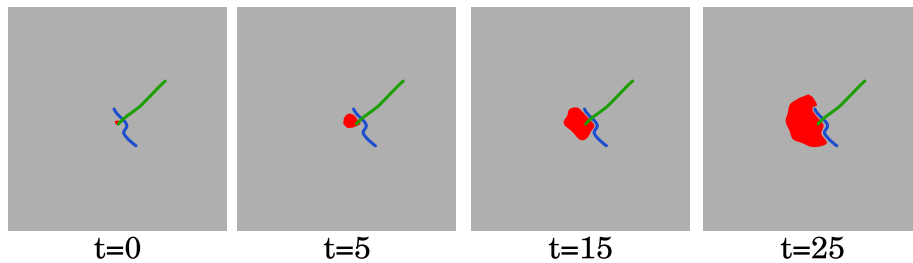
Zaostřování obrázku: [?], resp. [?] (tam to je vylepšeno pro zašumné obrázky).



(a) Růst města bez dalších dodatečných podmínek



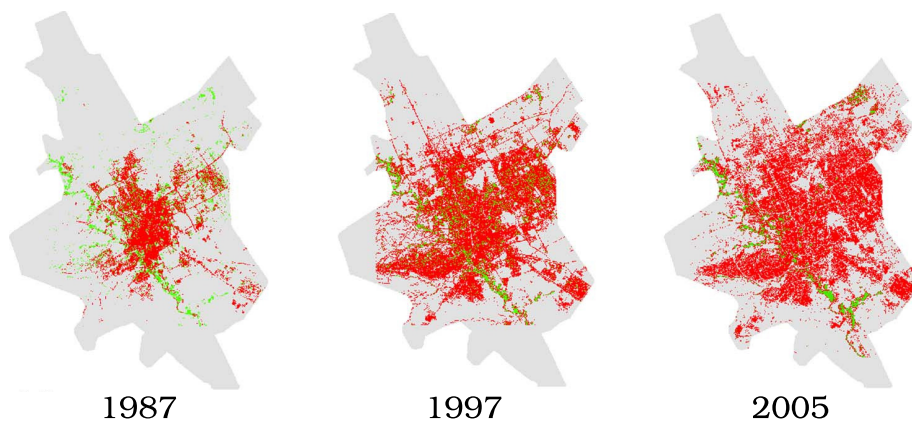
(b) Růst města podél silnice



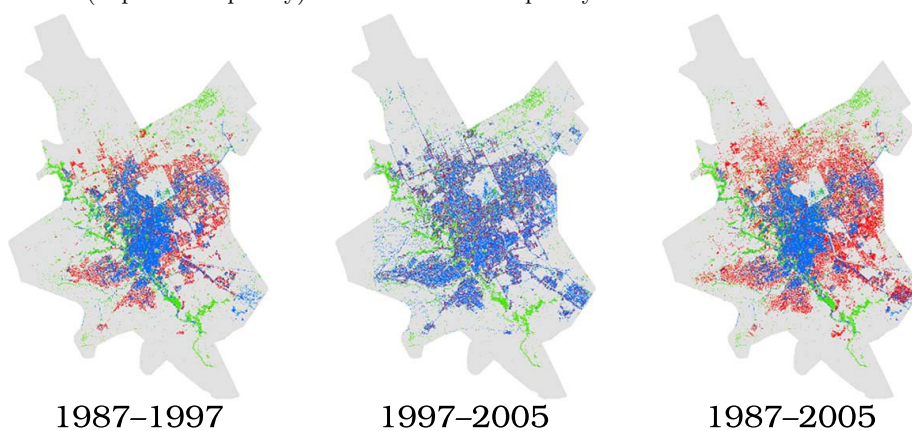
(c) Růst města bržděn řekou

Obrázek 8: (převzato z [?]) 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}

## **6 Konkrétní příklady**

### **6.1 Rozpoznávání ručně psaného textu**

...

### **6.2 Detekce překlepů**

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

## Reference

- [1] Radim Bělohlávek. *Fuzzy Relational Systems: Foundations and Principles*. Kluwer, New York, 2002.
- [2] Roger White Conrad Power, Alvin Simms. Hierarchical fuzzy pattern matching for the regional comparison of land use maps. *International Journal of Geographical Information Science*, 2010.
- [3] H. A. Girijamma Dr. V. Ramaswamy. Conversion of finite automata to fuzzy automata for string comparison. *International Journal of Computer Applications*, 2012.
- [4] J. J. Astrain; J. R. Garitagoitia; J. R. Gonzalez De Mendivil; J. Villadangos; F. Farina. Approximate string matching using deformed fuzzy automata: A learning experience. *Springer Science & Business Media*, 2004.
- [5] Abdulwahed Almarimi Gabriela Andrejková and Asmaa Mahmoud. Approximate pattern matching using fuzzy logic. *CEUR Workshop Proceedings*, 2003.
- [6] S.S. Yau G.F. DePalma. Fractionally fuzzy grammars with application to pattern recognition. *US–Japan Seminar on Fuzzy Sets and their Applications*, 1974. článek jako e-book: <http://bit.ly/2cumxjz>, výcuc v MorMal-FuzzAutAndLangs 10.7.
- [7] Kou-Yuan Huang. *Syntactic Pattern Recognition for Seismic Oil Exploration*. World Scientific, 2002.
- [8] J.R. Garitagoitia J. Astrain, J.R. González de Mendivil. Fuzzy automata with  $\epsilon$ -moves compute fuzzy measures between strings. *Fuzzy Sets and Systems 157*, 2005.
- [9] José R. Garitagoitia Carlos F. Alastrueya Javier Echanobe, José R. Gonzáles Mendivil.
- [10] José R. Garitagoitia José R. González de Mendivil. Fuzzy languages with infinite range accepted by fuzzy automata: Pumping lemma and determination procedure. *Fuzzy sets and Systems*, 2014.
- [11] Thomas Hatzichristos Lefteris A. Mantelas and Poulicos Prastacos. A fuzzy cellular automata modeling approach – accessing urban growth dynamics in linguistic terms. *Computational Science and Its Applications*, 2010.
- [12] S. C. Kremer M. Doostfateme. New directions in fuzzy automata. *International Journal of Approximate Reasoning*, 2004.
- [13] M. M. ZAHEDI S. MOGHARI and R. AMERI. New direction in fuzzy tree automata. *Iranian Journal of Fuzzy Systems*, 2011.
- [14] Mukta N. Joshi S. R. Chaudhari. A note on fuzzy tree automata. *International Journal of Computer Applications*, 2012.
- [15] Miroslav Stamenkovic, Aleksandar; Ciric. Construction of fuzzy automata from fuzzy regular expressions. *Fuzzy Sets and Systems*, 2012.

- [16] Witold Pedrycz Yongming Li. Fuzzy finite automata and fuzzy regular expressions with membership values in lattice-ordered monoids. *Fuzzy Sets and Systems*, 2005.