Aplikace fuzzy a pravděpodobnostních automatů

Martin Jašek

12. září 2016 — ???

Obsah

1 Úvod

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

Teorie automatů, fuzzy množiny i pravděpodobnostní počet jsou v současné době poměrně hluboce prozkoumány. Je proto přirozené, že se současný výzkum zabývá také jejich vzájemným propojováním, tj. fuzzy a pravděpodobnostními automaty. Teoretický výzkum těchto modelů přinesl mnohé výsledky. Otázkou však je, jaké je uplatnění těchto modelů v praxi.

Fuzzy množiny a pravděpodobnostní počet slouží k popisu různých forem neurčitosti, nepřesnosti či nejistoty. To jsou jevy, se kterými se setkáváme v běžném životě prakticky denně. Fuzzy a pravděpodobnostní automaty by tak mohly mít velký potenciál pro uplatnění v praxi. Úkolem této práce je tuto hypotézu ověřit.

V první části této práce jsou uvedeny základní teoretické poznatky studia fuzzy a pravděpodobnosních automatů. Větší část práce je poté věnována popisu samotných aplikací. Kromě teoretického popisu možností uplatnění tato práce prezentuje reálné ukázky uplatnění a konkrétní příklady. Závěr práce je věnován popisu softwaru, který některé z aplikací přímo implementuje.

2 Úvodní pojmy

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

2.1 Fuzzy množiny a fuzzy logiky

Pomocí matematiky můžeme formalizovat svět kolem nás. Zatímco matematika je založena na přesnosti, většina okolního světa lze přesně popsat jen s těží. Při popisu okolního světa pravujeme mnohem častěji s vágními pojmy jako „málo“, „téměř“ či „trochu“ než s jednoznačným „ano“ a „ne“. Například, tvrzení „student učivo umí“ lze těžko ohodnotit pravdou či nepravdou, pokud student z testu získal 50% bodů. Naopak, říci, že „student učivo umí z 1/2“ je mnohem přirozenější. V případech jako je tento mohou fuzzy množiny přinést značné zpřehlednění.

Podobně jako například predikátová logika tvoří matematický nástroj pro popis logického uvažování, fuzzy množiny (případně fuzzy logiky od nich odvozené) jsou matematický nástroj pro práci s nepřesnými pojmy[1]. Oproti „klasické“ množině, která prvek buďto obsahuje nebo neobsahuje, fuzzy množině může prvek nálažet v určitém stupni, který se nachází někde mezi „nenáleží vůbec“ a „náleží plně“. Obdobně, pravdivost tvrzení (tedy formule vyjádřené v přirozeném jazyce) ve fuzzy logice nemusí být pouze pravdivé či nepravdivé, ale může nabývat libovolného stupně pravdivosti.

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

[1]Slovo „fuzzy“ v angličtině znamená „nejasný“, „rozmazaný“, „neostrý“

Definice 2.1 (Fuzzy množina)**.** *Mějme libovolnou neprázdnou množinu U (tzv. univerzum). Pak zobrazení $\rho_A : U \to [0, 1]$ nazvěme členská funkce. Množinu A nazýváme fuzzy množina nad U. Pro libovolné $x \in U$ říkáme, že prvek x náleží do množiny A ve stupni $\rho_A(x)$.*

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

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

Nyní se podíváme na základní množinové operace nad fuzzy množinami.

Definice 2.2 (Operace nad fuzzy množinami)**.** *Mějme dvě fuzzy množiny π a ρ nad univerzem U. Pak definujme jejich průnik, sjednocení, rovnost a inkluzi (relaci „býti podmnožinou") jako:*

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

pro všechna $x \in U$.

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

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

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

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

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

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

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

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

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

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

Tabulka 1: Tři nejpoužívajší t-normy a jim odpovídající t-konormy {tbl:Norms}

Příklad 2.1. *Příkladem fuzzy množiny může být fuzzy množina reálných čísel, která jsou blízko nule. Taková fuzzy množina bude mít univerzum* $U = \mathbb{R}$ *a předpis například* $\zeta(x) = 2^{-|x|}$. *Pak platí* $\zeta(0) = 1$ *(tj. číslo* 0 *je blízko nule v jednotkovém stupni),* $\zeta(1) = 1/2$ *(tj. pravdivost tvrzení „číslo* 1 *je blízko nule“ je* 0,5*),* $\zeta(2) = 1/4$ *a podob.*

Příklad fuzzy relace může být podobnost čísel. Tedy fuzzy relace popsiující vztah „čísla x *a* y *jsou si podobná ve stupni* $\sigma(x, y)$*“. Jedná se tedy o fuzzy relaci na* $\mathbb{R} \times \mathbb{R}$ *a dána může být například předpisem:* $\sigma(x, y) = 1 - |x - y|$. *Přirozeně, pak platí například* $\sigma(0, 0) = 1$, $\sigma(0, 1) = 0$ *a* $\sigma(0{,}1, 0{,}9) = 0{,}2$.

2.2 Pravděpodobnostní počet

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

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

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

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

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

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

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

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

6. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

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

Definice 2.8 (Podmíněná pravděpodobnost)**.** *Mějme pravděpodobností prostor Ω a dva jevy $A, B \subseteq \Omega$ z tohoto prostoru tak, že $P(B) > 0$. Podmíněná pravděpodobnost jevu A za podmínky, že nastal jev B je pravděpodobnost*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Příklad 2.2. *Uvažujme příklad s klasickou šestistěnnou hrací kostkou. Prostor jevů je $\Omega = \{1, 2, 3, 4, 5, 6\}$. Označme $A = \{2, 4, 6\}$ jako jev „padlo sudé číslo“ a $B = \{5, 6\}$ jako „padlo číslo větší než 4“.*

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

2.3 Fuzzy vs. pravděpodobnostní přístup

{subsec:FuzzyVsProb}

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

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

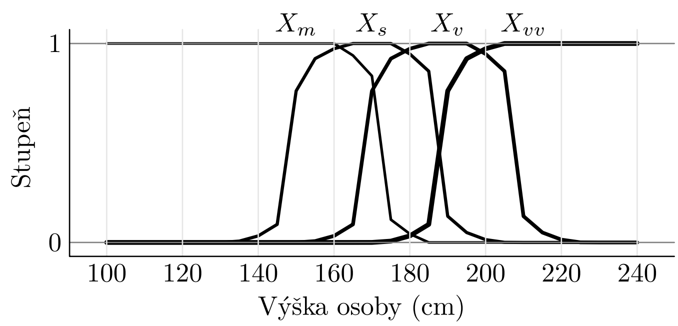
Naopak, tvrzení „zítra mi ujede autobus“ je popsáno přesně (nepoužívá žádné vágní pojmy), avšak nejistota je zde skryta v pozorovaném jevu. Zda-li tento jev nastane (a tvrzení je tudíž pravdivé) nebo nenastane nám není známo. Můžeme se však bavit o pravděpodobnosti, s jakou nastane.

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

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

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

Poznámka 2.1. *Existují snahy fuzzy a pravděpodobnostní přístup sjednotit. Takovýmto „univerzálním stupňům neurčitosti“ se obvykle říká „váhy“. Systém pracující s vahami však nemá oporu v logickém popisu okolního světa a tak není vhodné jej používat v reálných aplikacích. Sloužit může jen jako teoretický model.*



Obrázek 1: Graf významových funkcí štítků X_m („malá“), X_s („střední“), X_v („velká“) a X_{vv} („velmi velká“) lingvistické porměnné „výška osoby“. {fig:lingVarsMeansChart}

2.4 Popis v přirozeném jazyce

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

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

Pro popis stupňů pravdivosti by bylo možné používat „Tvrzení t je pravdivé ve stupni d.“. Taková formulace však zní poněkud kostrbatě. Mnohem elegantněji lze podobná tvrzení vyjádřit pomocí tzv. lingvistikých proměnných [**?**]. Každá lingvistická proměnná α je tvořena univerzem U hodnot a množinou lingvistických štítků T. Pro každý štítek $X \in T$ pak α definuje jeho význam $M(X)$ (významovou funkci) jako fuzzy množinu nad U (tj. funkci, která hodnotě z univerza přiřadí stupeň pravdivosti). Fakt, že lingvistická proměnná α nabývá „hodnoty“ X budeme zapisovat $\alpha = X$.

Příklad 2.3. *Mějme lingvistickou proměnnou α s významem „výška osoby“. Univerzem hodnot jsou všechna nezáporná reálná čísla. Lingvistické štítky jsou X_m („malá“), X_s („střední“), X_v („velká“) a X_{vv} („velmi velká“). Významové funkce jednotlivých štítků jsou vyobrazeny na obrázku 1.*

Mějme výšku h osoby. Namísto „Platnost tvrzení „výška h osoby je střední“ je ve stupni $M(X_s)(h)$“ můžeme říkat „výška h osoby je střední“. Symbolicky zapsáno: $\alpha = X_s$.

Poznámka 2.2. *Někdy se můžeme setkat s ligvistickými proměnnými, které mají jen jeden štítek. Například, ligvistická proměnná „závodník se nachází“ se štítkem „v cíli“. V takovém případě můžeme psát rovnou „závodník se nachází v cíli“.*

Pro popis složitějších zákonitostí je potřeba pokročilejčí nástroj. Vztahy tak budeme popisovat pomocí tzv. fuzzy IF–THEN pravidel[**?**]. Klasické bivalentní IF–THEN pravidlo je výrok ve tvaru:

$$\text{Jestliže } x_1, \ldots, x_n, \text{ pak } y$$

kde $x_1, \dots x_n, y$ jsou podvýroky[2].

Příklad 2.4. *Následující výroky jsou* If–Then *pravidla:*

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

Fuzzy If–Then pravidla pak budou vypadat následovně:

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

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

Příklad 2.5. *Uvažujme systém, u kterého známe tlak (lingvistická proměnná* α_t*) a teplotu (lingvistická proměnná* α_p*) a popisujeme úroveň otevření ventilu (lingvistická proměnná* α_v*). Označme štítky* p_L *(„tlak je nízký“),* p_N *(„tlak je normální“),* p_H *(„tlak je vysoký“) a dále* t_N *(„teplota je v normě“) a* t_H *(„teplota je zvýšená“). Konečně zaveďme štítky* v_O *(„ventil je uzavřen“) a* v_C *(„ventil je otevřen“).*

Pak pravidla

Pokud je tlak nízký a teplota zvýšená, pak ventil je uzavřen
Pokud je tlak vysoký a teplota v normě, pak ventil je otevřen

mohou být přepsána jako

$$\textit{Jestliže } \alpha_p = p_L, \alpha_t = t_H \textit{ pak } \alpha_v = v_C$$
$$\textit{Jestliže } \alpha_p = p_H, \alpha_t = t_N \textit{ pak } \alpha_v = v_O$$

2.5 Základní pojmy z teorie automatů

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

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

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

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

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

Symbolem Σ^* se značí množina všech řetězců nad abecedou Σ (včetně ε). Symbol Σ^+ pak značí všechny řetězce nad abecedou Σ vyjma ε.

[2] Zapíšeme-li If–Then pravidlo jako formuli, obdržíme Hornovskou klauzuli. Můžeme tak určit pravdivost jejího důsledku y na základě pravdivosti předpokladů $x_1, \dots, x_n$.

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

Nad jazyky L, L_1 a L_2 nad abecedami Σ, Σ_1 a Σ_2 se zavádí:

$$
\begin{array}{ll}
L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\} & \text{zřetězení (produkt)} \\[2ex]
L^n = \begin{cases} \{\varepsilon\} & \text{pokud } n = 0 \\ L L^{n-1} & \text{jinak} \end{cases} & n\text{-tá mocnina} \\[2ex]
L^* = \bigcup_{i=0}^{\infty} L^i & \text{Kleeneho uzávěr} \\[2ex]
L^+ = \bigcup_{i=1}^{\infty} L^i & \text{pozitivní uzávěr}
\end{array}
$$

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

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

{note:RegLangsExprsGrams}

Poznámka 2.3. *V teorii automatů se dále pracuje s pojmy regulérní jazyk, regulérní gramatika a regulérní výraz (a jim odpovídající regulérní fuzzy jazyk, regulérní fuzzy gramatika a regulérní fuzzy výraz). S těmito pojmy tato práce pracuje jen okrajově, takže se spokojíme jen s velmi zjednodušeným popisem těchto pojmů. Regulérní jazyk je jazyk rozpoznávaný konečným „bivalentním“ automatem (deterministickým nebo nedeterministickým). Regulérní gramatika je předpis popisující konstrukci řetězců regulérního jazyka. Regulérní výraz je způsob, jak poměrně přehledně zapsat regulérní jazyk.*

Dalším třídou jazyků jsou bezkontextové jazyky. Zatímco regulérní jazyk si lze představit jako množinu lineárních sekvencí symbolů, bezkontextový jazyk umožňuje u řetězců rekurenci. Typickým regulérním jazykem je například jazyk sudých čísel zapsaných v binárním tvaru (tj. řetězce 0 *a* 1 *končící vždy* 0*), typickým regulérním jazykem je jazyk aritmetických výrazů.*

Důležitou vlastností regulérních jazyků, se kterou se v této práci opakovaně setkáme, je, že každý konečný jazyk je regulérním. Pro každý konečný jazyk tak existuje konečný „bivalentní“ automat, který tento jazyk rozpoznává.

3 Definice fuzzy automatu

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

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

*

3.1 Koncept automatu

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

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

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

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

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

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

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

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

3.2 Nedeterminstický bivalentní automat

Nedeterministický bivalentní automat je definován následovně:

{def-NedBivAut}

Definice 3.1 (Nedeterministický bivalentní automat)**.** *Nedeterministický bivalentní automat* $\mathbf{A}$ *je pětice* (Q, Σ, μ, I, F)*, kde* Q *je konečná množina stavů,* Σ *je abeceda,* $\mu : Q \times \Sigma \to 2^Q$ *je přechodová funkce a* $I \subseteq Q$ *a* $F \subseteq Q$ *je množina počátačních a koncových stavů.*

Výpočet automatu, tedy zpracování vstupního řetězce, je definován jako posloupnost konfigurací. Konfigurace je přesný popis aktuálního stavu výpočtu (tzn. nezpracovaná část vstupního řetězce a množina stavů, ve kterých se automat nachází). Na počátku se výpočet nachází v tzv. počáteční konfiguraci, tj.

nezpracovanou částí celého řetězce je celý vstupní řetězec a množinou stavů, ve kterých se automat nachází je celá množina I.

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

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

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

Příklad 3.1. *Mějme* $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $I = \{q_0\}$, $F = \{q_1\}$ *a přechodovou funkci* $\mu(q_0, a) = \{q_0, q_1\}$, $\mu(q_0, b) = \{q_0\}$, $\mu(q_1, a) = \{q_0\}$ *a* $\mu(q_1, b) = \emptyset$. *Pak* $\mathbf{A} = (Q, \Sigma, \mu, I, F)$ *je nedeterministický bivalentní automat.*

Uvažujme řetězec `a b a`*. Z počáteční konfigurace (vstup* `a b a` *a aktuální stavy* $\{q_0\}$*) přejde zpracováním* `a` *do množiny stavů* $\{q_0, q_1\}$*. Dále pak zpracováním* `b` *přejde do* $\{q_0\}$ *a následně zpracováním* `a` *do* $\{q_0, q_1\}$*. Automat se ocitá v koncové konfiguraci a protože se nachází ve stavech* $\{q_0, q_1\}$ *z nichž stav* q_1 *je koncový, je řetězec* `a b a` *automat přijat. Stejnětak je přijat například řetězec* `b a` *nebo* `a a b a`*. Řetězec* `a b` *je automatem zamítán.*

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

3.3 Nedeterministický fuzzy automat

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

Vzhledem k tomu, že automat je definován jako struktura, je zprvu třeba stanovit, které její části má smysl zobecňovat na fuzzy množiny (relace, funkce). Abeceda symbolů i množina stavů zcela určitě můsí zůstat zachovány jako konečné bivalentní množiny. U přechodové funkce naopak očekáváme ostupňovanost (očekáváme možnost říkat „automat přejde do stavu q ve stupni d“). Co se počátečních a koncových stavů týče, někde se lze setkat s reprezentací bivalentní množinou (např. v. [?], [?]), jinde zase s fuzzy množinami ([?], [?], [?] [?])[3]. Vzhledem k tomu, že druhý jmenovaný, tj. automat s fuzzy množinou vstupních i výstupních stavů, je zřejmě obecnější, bude nadále uvažován pouze tento.

{def-ZaklDefNedFuzzAut}

Definice 3.2 (Nedeterministický fuzzy automat)**.** *Nedeterministický fuzzy automat* $\mathbf{A}$ *je pětice* $(Q, \Sigma, \mu, \sigma, \eta)$*, kde* Q *je konečná množina stavů,* Σ *je abeceda,*

[3]Mezi oběma druhy automatů existuje ekvivalence.

q	$\mu(q,q')$			$\sigma(q)$	$\eta(q)$
	q_0	q_1	q_2		
q_0	$(a,1)$, $(b,0{,}9)$	$(a,0{,}8)$	$\emptyset$	1	0
q_1	$\emptyset$	$(a,1)$	$(b,1)$	0,1	0,9
q_2	$(a,0{,}5)$	$(b,0{,}7)$	$\emptyset$	0	0,6

Tabulka 2: Fuzzy automat reprezentovaný tabulkou {tab:FuzAutTab}

μ je fuzzy přechodová funkce (fuzzy relace $Q \times \Sigma \times Q \to [0,1]$) a σ a η jsou po řadě fuzzy množiny nad Q počátačních, resp. koncových stavů.

Značení 1. ?? *Označme $\mu_x(p,q) = \mu(p,x,q)$, tj. přechody realizované přes symbol x.*

Před tím, než bude popsán výpočet automatu budou ukázány možnosti, jak fuzzy automat reprezentovat.

3.4 Reprezentace nedeterministického fuzzy automatu

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

Jedná se o tabulku, která v řádcích obsahuje aktuální stavy a ve sloupcích následující stavy. Poté buňka na řádku q a ve sloupci q' obsahuje $\mu(q,q')$, tj. výčet symbolů a stupňů pravdivosti těchto přechodů. Dále tabulka obsahuje dva dodatečné sloupce pro určení stupně počátečního a koncového stavu každého stavu.

Tabulka však často může být rozsáhlá, proto se reprezentace tabulkou často nahrazuje maticovým způsobem (např. [**?**], [**?**]). Očíslujme si stavy $Q = \{q_0, \dots q_n\}$. Poté pro každý symbol $x \in \Sigma$ sestavíme matici μ_x takovou, že $\mu_x, i, j = \mu(q_i, x, q_j)$ pro $0 \leq i, j \leq n$. Dále přiložíme vektor počátečních (koncových) stavů takový, že i-tá složka vektoru je rovna hodnotě $\sigma(q_i)$ $(eta(q_i))$.

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

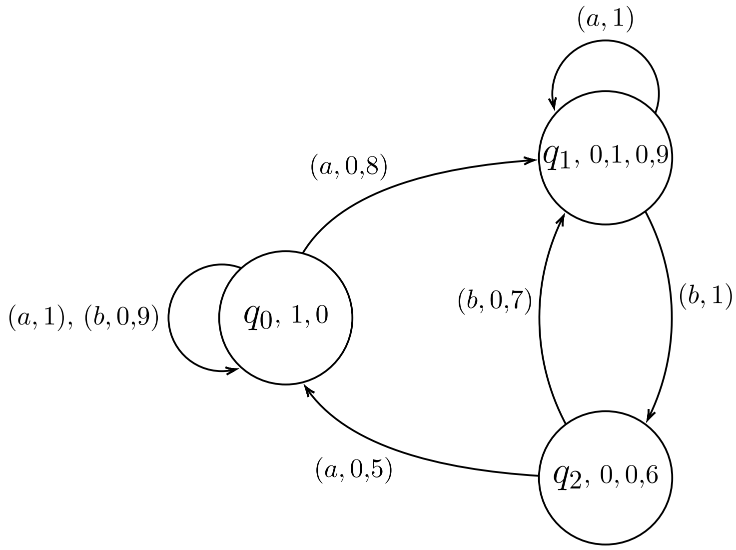
- stavy automatu tvoří uzly grafu
- každý uzel stavu q je označen $(q, \sigma(q), \eta(q))$
- hrana od uzlu q k uzlu q' je ohodnocena seznamem takových $(x,d) \in \Sigma \times [0,1]$, pro které $\mu(q,x,q') = d$

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

{ex:FuzAut}

Příklad 3.2. *V tabulce 2 je uveden příklad reprezentace fuzzy automatu pomocí tabulky. Na obrázku 2 je téže automat reprezentován grafem. Maticově reprezentován by tento automat vypadal následovně:*

$$\mu_a = \begin{pmatrix} 1 & 0{,}8 & 0 \\ 0 & 1 & 0 \\ 0{,}5 & 0 & 0 \end{pmatrix}, \mu_b = \begin{pmatrix} 0{,}9 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0{,}7 & 0 \end{pmatrix}, \sigma' = \begin{pmatrix} 1 \\ 0{,}1 \\ 0 \end{pmatrix}, \eta' = \begin{pmatrix} 0 \\ 0, \\ 0{,}6 \end{pmatrix}$$



Obrázek 2: Fuzzy automat reprezentován grafem {img:FuzAutDiag}

3.5 Výpočet nedeterministického fuzzy automatu

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

{def-FuzzStav}

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

Poznámka 3.1. *Fuzzy množiny počátečních a koncových stavů jsou ve své podstatě také fuzzy stavy.*

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

Definice 3.4 (Konfigurace nedeterministického fuzzy automatu)**.** *Mějme nedeterminsitický fuzzy automat* $\mathbf{A}$*. Pak každý prvek* $(w, \widehat{Q})$ *relace* $\Sigma^* \times \mathcal{F}(Q)$ *nazýváme konfigurace automatu* $\mathbf{A}$*.*

Definice 3.5 (Aplikace fuzzy relace na fuzzy stav (tzv. t-kompozice))**.** *Mějme nedeterministický fuzzy automat* $\mathbf{A}$ *a fuzzy stav* $\widehat{Q}$*. Pak aplikací binární fuzzy relace* $R : Q \times Q \to [0, 1]$ *na fuzzy stav* $\widehat{Q}$ *obdržíme fuzzy stav* $\widehat{Q} \circ R$ *splňující pro každé* $p \in Q$*:* $(\widehat{Q} \circ R)(p) = \bigoplus_{q \in Q}(\widehat{Q}(q) \otimes R(q, p))$*.*

{def-PreFunFuzzStav}

Definice 3.6 (Přechodová funkce fuzzy stavů)**.** *Mějme nedeterministický fuzzy automat* $\mathbf{A}$. *Pak přechodová funkce fuzzy stavů je fuzzy relace* $\widehat{\mu} : \mathcal{F}(F) \times \Sigma \to \mathcal{F}(F)$ *taková, že pro každý fuzzy stav* $\widehat{Q} \in \mathcal{F}(Q)$ *a symbol* $x \in \Sigma$ *je* $\widehat{\mu}(\widehat{Q}, x) = \widehat{Q} \circ \mu_x$.

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

1. $w_i = aw_{i+1}$ *kde* $a \in \Sigma$
2. $\widehat{Q}_{i+1} = \widehat{Q}_i \circ \widehat{\mu}(\widehat{Q}_i, a)$

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

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

{def-RozPreFunFuzzStav}

Definice 3.8 (Rozšířená přechodová funkce)**.** *Mějme nedeterministický fuzzy automat* $\mathbf{A}$. *Pak rozšířená přechodová funkce je fuzzy relace* $\mu^* : Q \times \Sigma^* \times Q \to [0, 1]$ *daná následujícím předpisem:*

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

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

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

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

Můžeme tedy zapsat:

{def-RetPriAut}

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

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

Poznámka 3.2. *V literatuře (např. [**?**] [**?**] [**?**]) se obvykle lze setkat s „techničtějším“ zápisem ať už jen rozšířené přechodové funkce, tak* $\mathbf{A}(w)$. *Pro řetězec* $w =$

$a_0 \dots a_n$ rozvojem rekurence μ^* můžeme napsat (poznamenejme, že $\mu^*(q, \epsilon, p_0) = 1$ pokud $q = p_0$, jinak 0):

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

Poté může být (1) zapsána jako:

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

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

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

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

{def-JazRozpAut}

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

Příklad 3.3. *Uvažujme automat* $\mathbf{A}$ *v příkladu 3.2 a řetězec* `a b a`*. Použijeme-li Gödelovu t-normu, pak automat postupně projde při svém výpočtu následujícími fuzzy stavy:*

$$\{(q_0, 1), (q_1, 0{,}1)\}, \{(q_0, 1), (q_1, 0{,}8)\}, \{(q_0, 0{,}9), (q_2, 0{,}8)\}, \{(q_0, 0{,}9), (q_1, 0{,}8)\}$$

Řetězec je tak přijímán ve stupni $0{,}8$*.*[5]

Dále pak řetězec `a a b b b` *je příjímán ve stupni* $0{,}6$ *a řetězec* `b a` *ve stupni* $0{,}8$.

∗

[4]Výpočet však může být optimalizován. Pokud některý přechod není definován (tj. automat by jej realizoval v nulovém stupni) můžou být cesty, procházející tímto přechodem při vypočtu vynechány.

[5]Pokud namísto Gödelovy t-normy použijeme Łukasiewiczovu, je řětězec přijímán ve stupni $0,7$. Pokud použijeme produktovou, je příjímán ve stupni 0,72.

3.6 Pravděpodobností automat

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

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

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

Definice 3.11 (Pravděpodobnostní automat)**.** *Jako pravděpodobnostní automat označujeme strukturu (Q, Σ, μ, q_0, F), kde Σ je abeceda, Q je neprázdná množina stavů, $q_0 \in Q$ je počáteční stav, $F \subseteq Q$ je množina koncových stavů a $\mu : Q \times \Sigma \to (Q \to [0, 1])$ je pravděpodobnostní přechodová funkce*[6]*, která každému $q \in Q$ a $a \in \Sigma$ přiřazuje pravděpodobnost $P_{q,a} : Q \to [0, 1]$ takovou, že:*

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

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

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

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

$$B(w) = B(a_1) \times \cdots \times B(a_m)$$

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

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

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

kde $B(w)$ je matice popsána výše, je pravděpodobnost přijetí řetězce w automatem $\mathbf{A}$.

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

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

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

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

Příklad 3.4. *Mějme* $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_1\}$ *a* $\mu(q_0, a) = \mu(q_0, b) = \{(q_0, 1/2), (q_1, 1/2)\}$, $\mu(q_1, a) = \{(q_0, 1)\}$, $\mu(q_1, b) = \{(q_1, 1)\}$. *Pak* $\mathbf{A} = (Q, \Sigma, \mu, q_0, F)$ *je pravděpodobnostní automat.*

Tento automat řetězec `a b a` *přijímá s pravděpodobností* $1/8$ *a řetězec* `a a b` *s pravděpodobností* $5/8$.

*

4 Varianty fuzzy automatů

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

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

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

4.1 Deterministický fuzzy automat

V teorii „klasických“ bivaletních automatů se rozlišuje automat deterministický a nedeterministický. Deterministcký automat je charakterizován tím, že každý krok jeho výpočtu je jednoznačně určen. U nedeterministického tento předpoklad platit nemusí.

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

Podobně jako u nedeterministického automatu existuje několik různých definic. V [**?**] je definován jako deterministický bivalentní automat, jen koncové stavy jsou fuzzy množinou. Dle [**?**] je to deterministický bivalentní automat, ale s fuzzy počátečními stavy, koncovými stavy i přechodovou funkcí.

V [**?**] definují determinsitický fuzzy automat jako nedeterministický fuzzy automat s tím, že je jeho přechodová funkce omezene tak, že z každého stavu při každém vstupním symbolu automat může přejít do nejvýše jednoho stavu.

Deterministické fuzzy automaty mají význam především pro implementace. Jak bylo zmíněno v předchozí kapitole, časová složitost výpočtu nedeterminstického fuzzy automatu je exponenciální vzhledem k délce vstupu. U deterministického automatu je tato složitost lineární. Obvykle však bývá vykoupena mnohonásobně vyšším počtem stavů automatu.

U všech tří citovaných definic deterministického fuzzy automatu jsou uvedeny algoritmy pro tzv. determinizaci automatu, tedy převod nedeterministického fuzzy automatu na jemu odpovídající deterministický.

4.2 Nedeterminstický fuzzy automat s ϵ-přechody

Nedeterminstický fuzzy automat s ϵ-přechody je rozšíření nedeterminsitického fuzzy automatu. Toto rozšíření je realizováno pomocí tzv. ϵ-přechodů. „Klasický“ přechod je automatem realizován, je-li na vstupu symbol odpovídající symbolu pravidla. Oproti tomu ϵ-přechod však automat může realizovat kdykoliv, bez ohledu na symbol na vstupu.

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

4.3 Zobecněné automaty

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

4.4 Fuzzy automat s výstupem

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

4.5 Stavový stroj

V určitém smyslu opakem fuzzy automatu s výstupem je stavový stroj. Oproti němu totiž výstupní schopnosti běžného fuzzy automatu nerozšiřuje, ale naopak ubírá. Stavový stroj je totiž výpočetní model, který nedisponuje žádnou formou výstupu[7]. U stavových strojů není totiž důležitý výsledek výpočtu, ale jeho průběh.

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

4.6 Událostmi řízený fuzzy automat

{subsec:FuzzEvMach}

Jako událostmi řízený fuzzy automat se obvykle označuje další třída výpočetních modelů, které zobecňují fuzzy automaty. Vycházejí z myšlenky, že přechodová funkce může být zapsána jako fuzzy If–Then pravidla. Slovní popis činnosti

[7] V určitém smyslu však může být za výstup považován stav (resp. fuzzy stav), ve kterém výpočet zkončil.

přechodu $(q, a, q', d) \in \mu$ je totiž následující:

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

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

Poznámka 4.2. *V souvislosti s automaty budeme často používat pojem „abecda událostí“. Tímto pojmem budeme souhrnně nazývat všechny „události“, tj. ligvistické proměnné a jejich štítky, které se v přechodové funkci vyskytují.*

4.7 Zásobníkový fuzzy automat

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

5 Další modely podobné fuzzy automatům

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

5.1 Fuzzy tree automaty

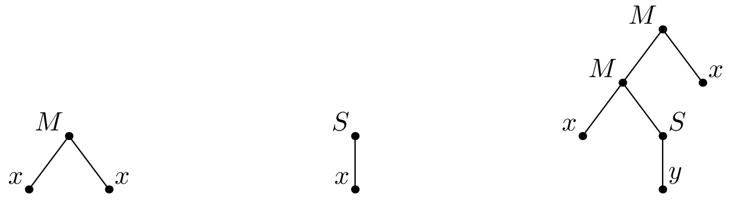
Fuzzy tree automaty jsou speciální třídou automatů, které jsou navrženy pro rozpoznávání dat, které mají v sobě obsaženu určitou stromovou strukturu.

Fuzzy tree automaty byly odvozeny od „klasických“ bivalentních tree automatů. O „klasických“ tree automatech je možné se dočíst více informací např. v [**?**], popř. [**?**] a [**?**]. Problematice fuzzy tree automatů se věnuje například [**?**], [**?**], [**?**], [**?**] a [**?**].

Zatímco běžné konečné (fuzzy) automaty pracují s řetězci symbolů, (fuzzy) tree automaty pracují se speciálními strukturami symbolů, tzv. S-termy[8]. S-term je řetězec tvořený symboly z $(N \cup T \cup \{(,)\})$ (za předpokladu, že $(N \cap T \cap \{(,)\}) = \emptyset$) takové, že:

1. S-term je každé $t \in T$
2. S-term je každý řetězec $X(t_1 \dots t_m)$ takový, že $X \in N$, $m > 0$ a $t_1, \dots t_m$ jsou S-termy

[8] S-termy jsou podobné termům známým například z predikátové logiky.



Obrázek 3: Stromy S-termů $M(xx)$ (vlevo), $S(x)$ (uprostřed), $M(M(xS(y))x)$ (vpravo)

{img:Tree}

S-termem je tedy buďto symbol z abecedy T (tzv. abecedy terminálních symbolů) nebo řetězec tvořený symbolem z abecedy N (tzv. abecedy neterminálních symbolů) následovaný symbolem otevírající závorky, neprázdnou posloupností S-termů a zakončen symbolem uzavírajcí závorky.

V tomto textu budeme jako abecedu terminálních symbolů uvažovat malá písmena abecedy (minusky) a jako množinu neterminálních symbolů velká písmena abecedy (verzálky). Pak S-termem jsou například následující řetězce: x, $P(xy)$, $S(P(xK(zz)yz))$.

Je patrné, že S-termy mají rekurentní strukturu, tj. že S-term může obsahovat další S-term(y). S-term má tedy stromovou strukturu. Je-li S-term ve tvaru $X(t_1 \dots t_m)$, pak S-termy $t_1, \dots, t_m$ jsou jeho potomky. Takový S-term budeme nazývat vnitřní S-term. S-Term $t \in T$ žádné potomky nemá a jedná se proto o listový S-term. Grafické znázornění stromu S-termu je vyobrazeno na obrázku 3.

Vzhledem k tomu, že S-termy jsou řetězce, můžeme hovořit o jazycích S-termů, případně fuzzy jazycích S-termů.

(Fuzzy) Tree automaty jsou nástrojem, který rozpoznává S-termy, resp. (fuzzy) jazyky S-termů. Pro účely této práce se spokojíme pouze s popisem činnosti fuzzy tree automatu.

Automat má na svém vstupu S-term x. Automat rekurzivně prochází podtermy vstupního S-termu, při výpčtu si tedy zásobník zpracovávaných S-termů. Automat se při svém výpočtu může nacházet ve více stavech současně. Toho je dosaženo tak, že automat se nachází v řetězci stavů.

Je-li x listový S-term (tj. je tvořen symbolem $t \in T$), pak na základě symbolu t přechodová funkce určí do jakého stavu a v jakém stupni má automat přejít. Je-li x vnitřní S-term (tj. je ve tvaru $X(t_1 \dots t_m)$), pak automat rekurzivně zpracuje podtermy $t_1, \dots, t_m$ (čímž přejde do řetězce stavů $q_1 \dots q_m$) a následně přejde do stavu a v určeném stupni dle symbolu S-termu X a řetězce stavů, v jakých se nachází.

Pokud automat zkončí svůj výpočet v některém z koncových stavů je přijímán ve stupni, který je roven t-normě stupňů přechodů jeho podtermů. Pokud výpočet neskončí v koncovém stavu, je řetězec přijímán v nulovém stupni. Jazyk S-termů rozpoznávaný fuzzy tree automatem pak fuzzy množina všech S-termů ve stupni, v jakém jsou automatem přijímány.

Příklad 5.1. *Uvažujme fuzzy tree automat o stavech q_1, q_2. V tabulce 3 je uveden příklad jeho možných přechodových pravidel.*

zpracovávaný S-term	ze stavů	do stavu q_1	q_2
x	ϵ	1	1
y	ϵ	1	0
$S(t_1)$	q_1	0	1
$S(t_1)$	q_2	0	0,4
$S(t_1t_2)$	q_1q_1	0	0,3

zpracovávaný S-term	ze stavů	do stavu q_1	q_2
$M(t_1)$	q_1	0,1	0,8
$M(t_1)$	q_2	0	0,5
$M(t_1t_2)$	q_1q_1	0	0,6
$M(t_1t_2)$	q_1q_2	0	1
$M(t_1t_2)$	q_2q_1	0	0,7

Tabulka 3: Příklad přechodů fuzzy tree automatu. x, y jsou terminální symboly, M, S neterminální a t_1, t_2 jsou S-termy. {tab:FuzTreAutTrans}

5.2 Buněčné fuzzy automaty

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

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

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

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

Přechodová pravidla obvykle určují nový stav buňky na základě okolních buněk. Okolí buňky $c_{i,j}$ je osmice jejich sousedních buněk[9]:

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

{ex:GameOfLife}

Příklad 5.2. *Typickým příkladem buněčného automatu je tzv. Hra života (Game of Life) (např. [?]). Jedná se o jednoduchý simulátor živého organizmu.*

Hra života uvažuje dvoustavovou možinu stavů, tj. $Q = \{0, 1\}$. Je-li hodnota buňky c rovna 1 hovoříme, že je buňka „živá“, je-li rovna 0 nazýváme buňku „mrtvou“. Pravidla popisující její chod jsou následující:

1. *Je-li buňka c živá a je v jejím okolí méně, než 2 živé buňky, buňka umírá (ve smyslu „samoty“)*

[9] U krajních buněk mřížky se okolí zmenšuje jen na pětici, popř. trojici sousedicích buněk

2. *Je-li buňka c živá a je v jejím okolí více, než* 3 *živé bunky, buňka umírá (ve smyslu „vyčerpání zdrojů“)*

3. *Je-li buňka c mrtvá, a v jejím okolí jsou přesně* 4 *živé bunky, je buňka oživena*

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

Soubor všech stavů všech buněk nazvěme konfigurací buněčného automatu. Dvojice konfigurací tvoří krok výpočtu, pokud stav každé z buněk přešla ze svého stavu v první konfiguraci do stavu v druhé konfiguraci podle pravidel automatu. Posloupnost konfigurací, jejiž po sobě jdoucí dvojice konfigurací tvoří kroky konfigurace, se nazývá výpočet buněčného automatu.

Vzhledem k tomu, že pro každou konfiguraci existuje vždy právě jedna taková, která by s ní tvořila krok výpočtu, výpočet je tak jednoznačně svou určen první konfigurací. Tuto konfiguraci nazýváme počáteční konfigurace. Konfigurace pak číslujeme nultá, první, ..., případně konfigurace nulté generace, konfigurace první generace a podob.

Je zjevné, že výpočet buněčného automatu je obecně nekonečný. Nemá tedy smysl uvažovat konfiguraci koncovou.

Konfigurace buněčného automatu se často zobrazuje graficky. Zakresluje se jako bitmapa, kde jednotlivé pixely reprezentují stavy buněk na odpovídajících souřadnicích. Každému stavu je přiřazena barva, kterou je tento stav znázorněn. Pro zobrazení výpočtu se obvykle používá obrazová animace či videosekvence.

Příklad 5.3. *Ukázka výpočtu automatu* $\mathbf{A}$ *se čtvercovou mřížkou velikosti* 100 *realizující Hru života je na obrázku 4. Černá barva symbolizuje mrtvou buňku, bílá pak živou.*

Buněčné fuzzy automaty se poté definují značně různě. Např. [?] a [?] jej definuje jako buněčný automat, jehož množina stavů je interval $[0, 1]$. Častěji však bývá chápán jako buněčný automat, jehož přechodová pravidla jsou IF–THEN pravidla. Takovéto automaty často bývají nazývány buněčné automaty s fuzzy logikou. Takto bude buněčný fuzzy automat chápan i v této práci.

Příklad 5.4. *Příkladem buněčného fuzzy automatu může být například následující automat. Množina stavů bude obsahovat přirozená čísla z intervalu* $[0, 150)$. *Mějme lingvistickou proměnnou* ς *a její čtyři štítky* ς_L, ς_M, *a* ς_H.

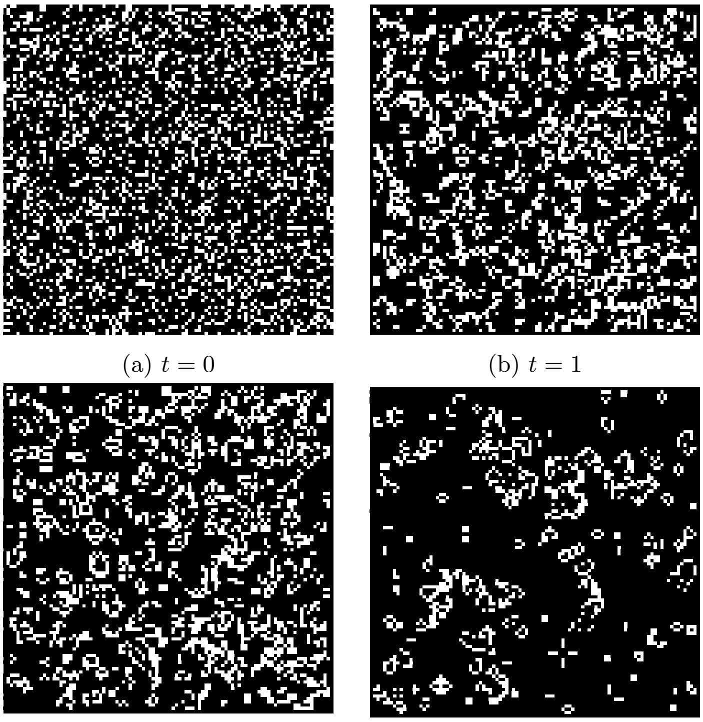
Pravidla automatu pak mohou vypadat následovně:

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

$*$

6 Užitečné techniky pro práci s fuzzy automaty

Fuzzy automaty či jim podobné modely zavedené v předchozích kapitolách jsou z pohledu pracktických aplikací pouze teoretické modely. V reálných aplikacích



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

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

Obrázek 4: Několik generací výpočtu buněčného automatu „Hra života“ s náhodnou počáteční konfigurací {img:GameOfLife} je často nutné pracovat s těmito modely spíše jen jako s kostrou, od které je poté odvozen výsledný model nebo algoritmus.

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

6.1 Fuzzy automat bivalentního automatu

V některých situacích máme k dispozici „klasický“ bivalentní automat[10] a potřebujeme k němu sestavit odpovídající fuzzy automat. Odpovídající ve smyslu, že řetězce, které bivalentní automat přijímá (zamítá) musí fuzzy automat přijímat ve stupni 1 (0).

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

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

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

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

6.2 Fuzzy automat rozpoznávající w

{sec:FuzAutRozpOme}

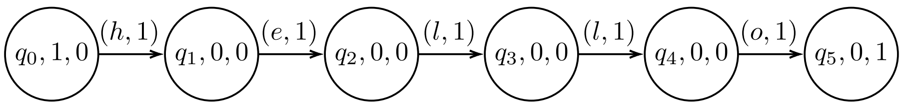
Další ze základních pomůcek pro práci s fuzzy automaty je automat, který v jednotkovém stupni rozpoznává pouze jeden konkrétní řetězec. Tento řetězec budeme v této a některých dalších podkapitolách nazývat „vzorový“ a značit w.

Chceme-li sestavit fuzzy automat rozpoznávající jeden řetězec, mohli bychom využít následující postup:

1. Označme $L(w)$ jako jednoprvkový jazyk obsahující řetězec w ve stupni 1
2. Jazyk $L(w)$ je konečný a tudíž i regulérní (viz poznámka 2.3).
3. Můžeme proto sestavit konečný bivalentní automat $\mathbf{A}'$, který jazyk $L(w)$ rozpoznává.
4. Pomocí postupu uvedeného v předchozí kapitole můžeme automatu $\mathbf{A}'$ sestavit fuzzy automat rozpoznávající řětězec w.

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

[10]Případně regulérní výraz či regulérní gramatiku



Obrázek 5: Automat rozpoznávající `hello` {`diag-AutRozpHell`}

{`def-FuzzAutRozpOme`}

Definice 6.1 (Fuzzy automat rozpoznávající w)**.** *Mějme řetězec w nad abecedou Σ délky n. Fuzzy automat rozpoznávající w je pak automat $\mathbf{A}(w) = (Q, \Sigma, \mu, \sigma, \eta)$ kde*

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

Příklad 6.1. *Příklad fuzzy automatu rozpoznávající řetězec w =* ***`hello`*** *se nachází na obrázku 5.*

6.3 Podobnost symbolů

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

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

Formálně se tedy jedná o fuzzy relaci $\simeq : \Sigma \times \Sigma \to [0, 1]$. Tato relace zjevně musí splňovat symetrii ($\simeq(x, x) = 1$) a reflexivitu ($\simeq(x, y) = \simeq(y, x)$) (pro všechny $x, y \in \Sigma$). Fuzzy relace podobnosti symbolů pak může být součástí automatu. V takovém případě dojde ke modifikaci výpočtu automatu takovým způsobem, že ve značení **??** a definici 3.8 bude nahrazen výraz $\mu(q, x, q')$ následujícím výrazem

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

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

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

6.4 Editační operace, deformovaný automat

{sec:DefAut}

Další technikou pro modifikaci chování automatu je využití editačních operací. Základní myšlenka této techniky spočívá v trojici jednoduchých editačních operací, jejiž složením jsme schopni popsat požadovanou transformaci vstupu na automatem reprezentovaný vzor. „Množství“ transformace pak udává podobnost pozorovaného a vzorového řetězce.

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

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

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

Pracujme nyní s abecedou Σ. Editační operace budeme zapisovat jako uspořádané dvojice $(a, b) \in (\Sigma \cup \{\epsilon\})^2 \setminus (\epsilon, \epsilon)$. Vložení symbolu x značme (ϵ, x), odstranění symbolu pak (x, ϵ) a náhradu symbolu x symbolem y pak (x, y). Pak aplikací každé z těchto posloupností editačních operací na řetězec `ahoj` obdržeíme řetězec `hello`:

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

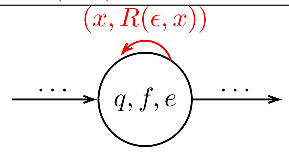
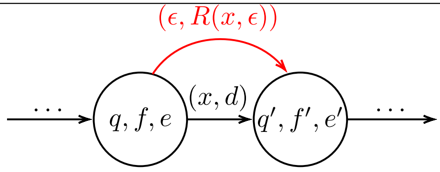
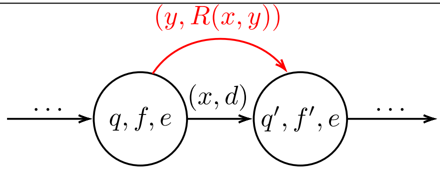
Označme E jako množinu všech editačních operací, tj. $E = (\Sigma \cup \{\epsilon\})^2 \setminus (\epsilon, \epsilon)$. Budeme-li chtít pro každou editační operaci stanovit stupeň, v jakém má být provedena, použijeme fuzzy relaci $P : E \to [0, 1]$. Následovat bude popis tzv. „deformace“ automatu, tedy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

{def-AutRozpCalL}

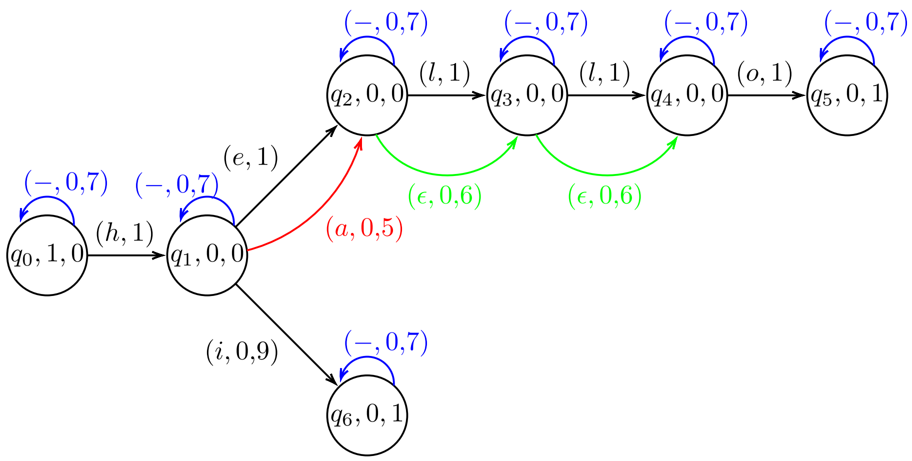
Definice 6.2 (Deformovaný automat)**.** *Mějme binární fuzzy relaci $R : E \to [0, 1]$ a nedeterministický fuzzy automat $\mathbf{A} = (Q, \Sigma, \mu, \sigma, \eta)$. Pak jako deformovaný automat označme nedeterministický fuzzy automat $\mathbf{A}' = (Q, \Sigma, \mu', \sigma, \eta)$ s ϵ-přechody, kde Q, Σ, σ a η zůstávají stejné a fuzzy přechodová funkce μ' je dána:*

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

Význam jednotlivých složek přechodové funkce (deformací) je názorně zobrazen v tabulce 4. Příklad deformovaného automatu je na obrázku 6.

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

Tabulka 4: Deformace automatu. Černě jsou znázorněny přechody původního automatu, červeně pak nově přidané {tbl:DefAutDef}



Obrázek 6: Ukázka deformovaného automatu. Černé přechody jsou přechody původního automatu, červený přechod realizuje náhradu symbolu e za a, zelené přechody odebrání symbolu l a modré přechody vložení symbolu symbolu $-$. {img:DefAut}

Je vidět, že pomocí editačních operací jsme schopni popsat libovolnou transformaci. Deformovaný automat tak může být schopen rozpoznat v nenulovém stupni jakoukoliv transformaci řetězce, který původní automat přijímal. Je také vidět, že deformovaný automat nedokáže ošetřit pokročilejší transformace, jako je vložení právě jednoho symbolu, vložení symbolu na určité pozici či nahrazení podřetězce jiným podřetězcem. Takovéto deformace automatu je zpravidla potřeba provést ručně. Další informace ohledně editačních operací a deformovaných automatů lze nalést např. v [?].

6.5 Učící se fuzzy automaty

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

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

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

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

6.6 Fuzzy automaty a neuronové sítě

Propojení fuzzy automatů s neuronovými sítěmi může přinést značně lepší výsledky, než učící se automaty. Budou tedy ve stručnosti uvedeny, spolu s popisem jejich vzájemného vztahu.

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

Způsoby, jak sestavit neuronovou síť existují dva (v praxi se však obvykle tyto přístupy kombinují). A to pomocí trénovacích dat nebo pomocí znalostní báze. Konstrukce pomocí trénovacích dat je podobná, jako v předchozí podkapitole. Tedy, že vstupní data se postupně nechávají vyhodnocovat neuronovou sítí a ta je posléze datům přizpůsobována. Konstrukce pomocí znalostní báze je pak založena na kódování znalostní báze ve formě If–Then pravidel do vah neuronové sítě.

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

V jednodušších případech se jedná pouze o neuronovou síť, která realizuje jen přechodovou funkci automatu. Tedy neurnovou síť, jejimž vstupem je fuzzy stav, ve kterém se automat nachází, a symbol na vstupu, a výstupem fuzzy stav, do kterého automat přejde. Existují však i způsoby, jak pomocí neuronové sítě reprezentovat celý automat.*

Možnosti, jak takovou neuronovou síť sestavit, jsou stejné, jako při návrhu obecné neuronové sítě. Lze tedy využít jak konstrukci pomocí trénovacích dat, tak pomocí znalostní báze. Použití trénovacích dat spočívá ve vygenerování dostatečného množství kombinací fuzzy stavů a vstupních symbolů. Pomocí automatu jsou určeny „výstupní“ fuzzy stavy, které jsou spolu předány neuronové síti k učení. Při použití znalostní báze jsou přechodová pravidla přepsána do fuzzy If–Then pravidel, jak bylo uvedeno v kapitole 4.6.

Stejně tak existují způsoby, jak konvertovat neuronovou síť zpět na fuzzy automat. Postup je popsán např. v [?], [?]. Je tedy možné převést fuzzy automat na neuronovou síť, provést její přeučení trénovacími daty a poté ji převést zpět na automat.

7 Rozpoznávání, klasifikace a korekce textových dat

{sec:Rec}

Automaty jsou nástroje, který v základu slouží k rozpoznávání řetězců. Fuzzy a pravděpodobnostní automaty pak rozpoznávání v určitém smyslu zobecňují z dvoustavového „odpovídá vzoru – neodpovídá vzoru“ na stupeň shody, resp. pravděpodobnost shody.

Můžeme tak určovat podobnost řetězce se vzorem. Tato myšlenka však může být dále rozvinuta. Máme-li vzorů více, můžeme určit, kterému z nich je vstup nejpodobnější. V takovém případě hovoříme o klasifikaci. Případně se můžeme pokusit odchylku od nejpodobnějšího vzoru odstranit. V takovém případě hovoříme o korekci.

V této kapitole budou prezentovány některé reálné aplikace fuzzy a pravděpodobnostních pro rozpoznávání, klasifikaci a korekci a to textových dat. Využití fuzzy automatu pro rozpoznávání (a případně klasifikace a korekci) netextových dat se věnuje následující kapitola. Speciálně pak, rozpoznávání obrazových dat je popsáno v kapitole 10.

7.1 Výpočet podobnosti řetězců

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

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

Pro určení podobnosti řetězců lze poměrně elegantně použít fuzzy automaty. Sestavíme-li automat $\mathbf{A}_w$ rozpoznávající řetězec w (viz kapitola 6.2) a poté automat upravíme (např. pomocí deformací dle kapitoly 6.4), obdržíme automat, který rozpoznává kromě řetězce w také řetězce jemu podobné. Jinými slovy, automat realizuje relaci podobnosti řetězců w a w'. Podobnostní relace pak bude vypadat následovně:

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

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

7.2 Klasifikace a korekce textových řetězců

Jak již bylo uvedeno, fuzzy a pravděpodobnostní automaty lze využít pro klasifikaci a následnou korekci (obecných) textových řetězců. V [?], [?], [?], [?] a [?] řeší tento problém pomocí pravděpodobnostních automatů.

V [?] automat učí texty v přirozeném jazyce a poté s jeho pomocí opravují pozměněný text Bible. Techniku používají také pro segementaci DNA (která je podrobněji popsána v kapitole 11.1).

Pozměněný automat[11] používají v [?] a [?]. Automat tak dokáže počítat (pravděpodobný) počet výskytů vzoru ve vstupním řetězci.

Toho pak využívají pro rozpoznávání proteinů. Sledovaný vzorek proteinu nejdříve chemicky rozštěpí na jednotlivé peptidy. Ty jsou hmnotnostním spektrometrem analyzovány a naměřené hodnoty jsou kódovány do symbolů. Z naměřených symbolů jsou posléze sestaveny řetězce, které jsou vyhodnocovány automatem.

V [?] a [?] pomocí pravděpodobnostních automatů snaží překládat slova fonetické abecedy na „běžnou“ (angličtinu). Dá se totiž předpokládat, že vyřčené slovo nemuselo být řečeno zcela zprávně a navíc mohlo být upraveno určitým akcentem. Učením sestavené automaty pro jednotlivá slova pak určijí nejpravděpodobněji vyřčené slovo.

7.3 Detekce překlepů

{subs:DetTyp}

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

7.4 Fuzzy lexikální analyzátor

Lexikální analyzátor je základní součást všech překladačů (nejen) programovacích jazků. Jeho úkolem je v sekvenci vstupních znaků rozpoznat jednotlivé

[11] Zde se jedná o tzv. aritmetický pravděpodobnostní automat. Automat při přechodech provádí elementární algebraické operace, v tomto případě případě sčítání.

elementy jazyka, tokeny. Tokenem může být například „číslo 42“, „identifikátor `foo`“, „operátor +“ či „symbol konce příkazu `;`“.

Pro rozpoznávání (resp. klasifikaci) jednotlivých tokenů se používají „klasické“ automaty. Jejich nevýhodou je, že nijak nereflektuje malé, často bezvýznamné chyby ve vstupu. Například vstup `iif` by zřejmě měl být reprezentován jako token „klíčové slovo `if`“ než jako „identifikátor `iif`“. Obdobně, středník by mohl být nahrazen čárkou nebo uzavírající kulatá závorka složenou.

Dle [?] může tyto problém vyřešit fuzzy lexikální analyzátor. Fuzzy lexikální analyzátor používá fuzzy automaty namísto „klasických“ automatů.

7.5 Pravděpodobnostní rozpoznávání přirozeného textu

Pravděpodobnostní automaty lze uplatnit při zpracování přirozeného jazyka. Jako abeceda je zde chápána množina všech slov daného přirozeného jazyka.

Věty obvykle mívají složitější strukturu, než jakou dokáže pokrýt „klasický“ (pravděpodobnostní) automat, je proto nutné použít buď pravděpodobnostní tree automaty [?] nebo zásobníkové pravděpodobnostní automaty [?].

7.6 Dvouúrovňové vyhledávání v dlouhém textu

V [?] používají dvouúrovňové vyhledávání vzorů v dlouhém textu. Nejdřív sestaví malý, jednoduchý (avšak nepřesný) automat pro přibližné vyhledání vzoru. Tento malý (a tudíž rychlý automat) je aplikován na každý větší blok textu, např. na každou stránku. Stránky, které tímto testem prošly v nenulovém stupni (tj. je pravděpodobné, že obsahují vzor) jsou pak prohledány přesným automatem.

7.7 Parsování referencí

V [?] pomocí pravděpodobnostního automatu parsují reference z textu a rozpoznávají v nich strukturu. Reference v textu mají totiž pevnou strukturu. Např. dle normy ČSN ISO 690:2017[?] má reference vypadat následovně:

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

Automat nejdříve rozpoznává části oddělené tečkou a posléze jim doplňuje pravděpodobný význam. Některá pole totiž mohou být vynechána, nebo naopak obsahovat tečku. Automat by tak měl určit nejpravděpodobnější variantu.

8 Další oblasti pro rozpoznávání

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

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

8.1 Rozpoznávání signálů

{subs:RozpSign}

Rozpoznávání signálů je jedna ze základních možností, jak využít rozpoznávání netextových dat. Věnují se tomu např. v [?], [?], [?] a [?].

Signál, jakožto spojitý průběh jedné proměnné (času) nejprve kvantujeme v čase. Obdržíme tak posloupnost po sobě jdoucích vzorků (typicky číselných hodnot). Každý takový vzorek můžeme snadno zakódovat do odpovídajícího symbolu. Takovými symboly může být například $s_{0 \leq x < 10}$, $s_{10 \leq x < 20}$ a $s_{30 \leq x < 30}$[12]. Zakódujeme-li tímto způsobem nakvantovanou posloupnost vzorků, obdržíme řetězec.

*

V [?] tuto techniku testují na detekci závady na elektromotoru. Vstupem je úhel natočení rotoru, jehož sinus by měl kopírovat sinusoidu. Při poruše se však průběh zdeformuje, což automat ropozná. Stejně tak používají tuto techniku pro rozpoznávání vzorů v datech z elektrokardiogramu (EKG). Problematice analýzy signálu EKG je věnována kapitola 11.5. V [?] je rozpoznávání signálů demonstrováno na rozpoznávání gest rukou. To je podrobněji rozebráno v kapitole 8.4.

V [?] monitorují teplotu taženého profilu v ocelárně. Specifická kombinace změn teploty taženého profilu může způsobit jeho zlomení. Monitorováním teploty, jejím přepočtem na řetězec a následnou analýzou pomocí automatu tak lze předvídat hrozící zlom.

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

8.2 Rozpoznávání dvourozměrného signálu

V [?] je popisován způsob, jak rozpoznávat dvojrozměrný signál (mřížku signálů), např. data z termokamery.

Používá se k tomu model podobný pravděpodobnostním buněčným automatům.

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

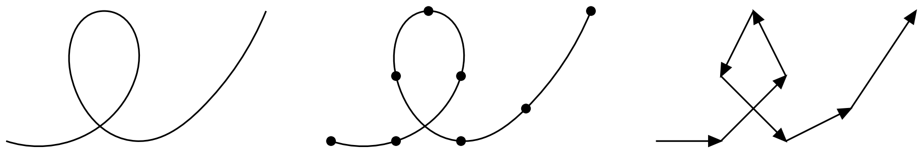
8.3 Rozpoznávání ručně psaného textu

{subs:RecHandWrit}

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

Uvažujme tedy, že máme k dispozici již křivku, a to křivku, kterou tvoří jedno písmeno. V první fázi křivku podle určitého kritéria rozdělíme na podkřivky, segmenty. Kritériem pro rozklad křivky na segmenty může být například změna směru. Každému segmentu je posléze přiřazen symbol (např. na základě jeho

[12] Alternativně lze použít ligvistickou proměnnou se štítky například „hodnota x je nízká“, „hodnota x je střední“ a „hodnota x“ je vysoká



Obrázek 7: Ukázka postup při rozpoznávání ručně psaného textu. Vlevo vstupní křivka, uprostřed rozdělena na segmenty, vpravo pak jednotlivým segmentům přiřazeny směry. Přepis na řetězec by pak mohl vypadat např.: `right upright upleft downleft downright right upright`. {img:HandWritEx}

směru). Celé křivce je tak přiřazen řetězec, který může být následně rozpoznán automatem. Tento proces je znázorněn na obrázku 7.

Této technice se věnují např. v [?] nebo [?]. V obou případech poukazují, že pro řádově přesnější výsledky rozpoznávání je vhodné systém vylepšit strojovým učením.

8.4 Rozpoznávání gest {subs:GestRec}

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

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

8.5 Práce se zvukem

Vzhledem k tomu, že zvuk je ve své podstatě analogový signál, může s ním být pracováno jako se signálem. To znamená, že může být rozpoznáván tak, jak bylo popsáno v kapitole 8.1. Věnují se tomu v [?] a [?]. V [?] využívají tuto techniku pro konverzi zvukového záznamu do formátu MIDI.

8.6 Fuzzy programy

V [?] popisují fuzzy programy. Fuzzy program je fuzzy regulérní výraz. Fuzzy program si lze představit jako deklarativní program, tj. program, který popisuje (fuzzy) jazyk přípustných řešení. Fuzzy automat tak může sloužit jako interpret takovýchto programů.

8.7 Metoda lisování dat {subs:DataPresTech}

Autor práce prezentuje novou, jednoduchou techniku pro tzv. klasifikaci dat. Klasifikace dat je problém, který spočívá v určení hodnoty výstupního attributu testovacích dat na základě trénovacích dat. Tato technika bude označována jako lisování dat.

Vstupem této techniky je (konečná) trénovací množina a parametr δ. Trénovací množinou je tabulka binárních hodnot. Každý záznam v tabulce má několik vstupních atributů a jeden výstupní. Parametr δ je číslo z intervalu $[0, 1]$.

Myšlenka metody lisování dat je založena na zakódování vstupních dat do řetězců a jejich následném zpracování fuzzy automatem. V první fázi jsou z trénovací množiny vybrány pouze záznamy, jejiž výstupní atribut je roven 1. Každý záznam trénovací množiny je zapsán jako řetězec nad abecedou $\Sigma = \{0, 1\}$. Celá trénovací množina tak může být konvertována na jazyk nad abecedou Σ. Vzhledem k tomu, že trénovací množina je konečná, bude i tento jazyk konečný. Může tak být sestaven konečný bivalentní automat a následně fuzzy automat, který tento jazyk rozpoznává.

Takový automat však obykle obsahuje velké množství stavů. Je proto třeba provést jeho determinizaci a minimalizaci. Zde je vhodné použít fuzzy minimalizaci s parametrem δ. Fuzzy minimalizace v takovém případě „vyhladí“ ostrost mezi řetězci, které automat přijímal, a řetězci, které zamítal. Lze říci, že automat se „naučí“ rozpoznávat také řetězce, které jsou podobné trénovacím, avšak nebyly součástí trénovací množiny.

Je-li proveden výpočet automatu s řetězcem, který byl sestaven ze záznamu z testovacích dat (stejným způsobem jako u trénovací množiny), automat určí stupeň v jakém je pravdivé trvzení „záznam má hodnotu výstupního atributu rovnu 1“.

V kapitole 12.7 je popsána implementace této metody a některé experimentální výsledky.

Poznámka 8.1. *Podobnou techniku, založenou na pravděpodobnostních automatech, používají v [**?**].*

8.8 Detekce úplných m-árních stromů

{subs:DetComTrees}

Fuzzy tree automaty mohou být použity pro poměrně elegentní určení úplnnosti stromu. Úplný m-ární strom je takový strom, jehož každý uzel má buďto právě m potomků (vnitřní uzly) a nebo 0 (listové uzly).

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

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

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

Uvažujme, že S-term t je tvořen vnitřními uzly I a listovými uzly o. Automat bude mít jeden stav q_1, který bude koncovým stavem. Přechodová funkce bude mít následující pravidla:

1. Při vstupu $t = o$ přejdi do stavu q_1 ve stupni 1
2. Při vstupu $t = I(t_1)$ přejdi ze stavu q_1 do stavu q_1 ve stupni $1/m$

   ...

$(m+1)$. Při vstupu $t = I(t_1 \dots t_m)$ přejdi ze stavu $q_1 \dots q_1$ do stavu q_1 ve stupni $m/m = 1$

Takto navržený automat zřejmě rozpoznává m-ární úplné stromy.

9 Modelování a simulace

Modelování a simulace je odvětví jehož účelem je získat přehled o chování určitého systému. Jakmile toto chování známe, můžeme například simulovat různé jevy, které by mohly hypoteticky nastat.

Fuzzy a pravděpodobnostní automaty v určitých situacích mohou posloužit jako počítačové modely. Jedinou podmínkou obvykle je, aby pozorovaný systém bylo možné popsat pomocí diskrétních stavů, mezi kterými se překlápí.

Pro modelování a simulace se obvykle používají událostmi řízené automaty.

9.1 Skrytý Markovův model

Skrytý Markovův je pravděpodobnostní model, se kterým se lze setkat v mnoha oblastech. Jedná se o diskrétní systém stavový systém, který na základě podnětů z vnějšku náhodně přechází mezi svými stavy. Jedná se tak vlastně o pravděpodobnostní stavový stroj. Více informací o vzájemném vztahu skrytých Markovových modelů a pravděpodobnostních automatů je k nalezení v [**?**].

9.2 Automobilismus

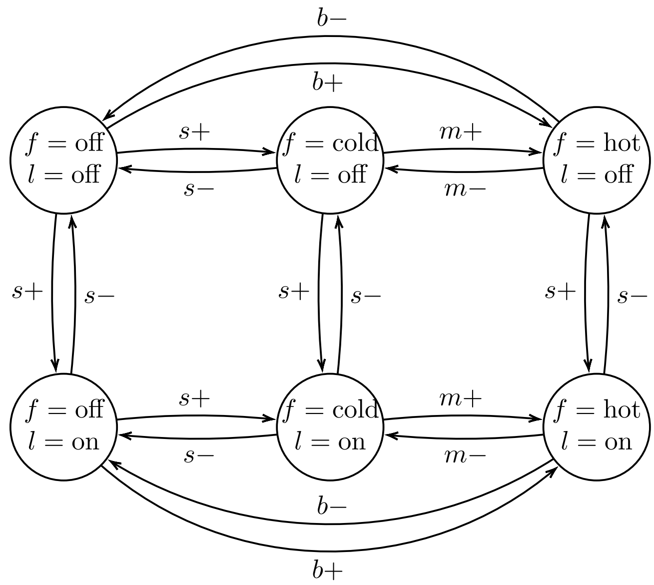
Automobily jsou v dnešní době zařízení vybavené značným množstvím palubní elektroniky. Je výhodné, aby elektronika měla přehled o tom, co se s automobilem děje, případně, co se v nejbližších sekundách bude dít.

V [**?**] a [**?**] jsou prezentován způsoby, jak jednoduše modelovat manévry řidiče. Mezi takové manévry patří např. předjíždění, zatáčení doleva v křižovatce nebo pouštění chodce na přechodu. Model využívá sadu lingvistických proměnných jako např. „aktuální rychlost“, „zrychlení“, „natočení volantu“ nebo „směrová světla“. Následně mohou být stanoveny vzory jednotlivých manévrů. Například při odbočování vlevo se jedná o následující posloupnost událostí:

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

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

Podobný způsob je popsán v [**?**]. V tomto případě však jako vstupní data slouží měření z trojice akcelerometrů (pro všechny tři osy, x, y a z). Lingvistické proměnné popisují, zda-li je zrychlení ve směru patřičné osy kladné, neutrální či záporné. Manévry, které tento systém dokáže rozpoznávat jsou např. „dlouhá táhlá zatáčka vlevo“ nebo „přejezd přes horizont“.



Obrázek 8: Automat pro monitorování elektrické sítě tvořené vysoušečem vlasů a světlem. Obsahuje-li stav systému f = off, znamená to, že je vysoušeč vlasů vypnut, pokud obsahuje f = cold pak vysoušeč vlasů fouká studený vzduch a pokud f = hot pak fouká teplý vzduch. Obdobně, l = off značí vypnuté světlo a l = on zapnuté. Přechody jsou poté označeny štítky $b-$ („velký pokles spotřeby"), $m-$ („střední pokles spotřeby"), $s-$ („malý pokles spostřeby"), $s+$ („malý nárust spotřeby"), $m+$ („střední nárust spotřeby"), $b+$ („velký nárust spotřeby"). {img:PowConsAut}

9.3 Monitorování elektrických a počítačových sítí

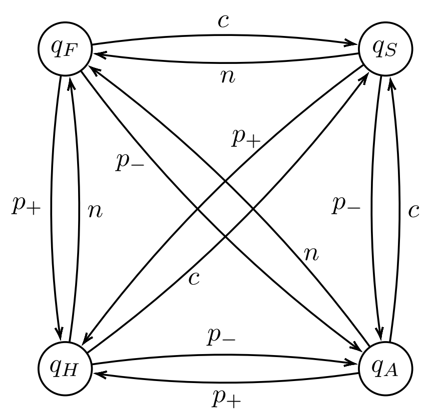
{subs:MonElComNet}

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

Technika využívá faktu, že každý spotřebič se může nacházet v různých stavech. Například vysoušeč vlasů se může nacházet v jednom z těchto tří stavů: „vypnut", „fouká studený vzduch" a „fouká teplý vzduch". Předpokládejme, že v každou časovou jednotku dojde vždy pouze k jedné změně stavu jednoho spotřebiče. Každému stavu je asociována konkrétní hodnota odběru energie.

Označme změny odběru elektrické energie lingvistickou proměnnou „spotřeba el. proudu ..." a se štítky např. „výrazně klesla", „je konstantní" a „lehce stoupla" (čím více štítků použijeme, tím přesnější model bude). Nastane-li v síti událost x (např. značný pokles spotřeby), automat bude schopen určit, který ze spotřebičů změnil svůj stav. Tím způsobem tak můžeme neinvazivně sledovat, které spotřebiče, kdy a jak zatěžují síť. Ukázka automatu, který tuto techniku realizuje, je na obrázku 8.

V [?] modelují pomocí pravděpodobnostního automatu počítačovou síť. Model je tvořen automaty pro každý z uzlů sítě a dále pro každý kanál. Zasílání zpráv mezi uzly vede ke změnám stavů jednotlivých automatů. Výsledkem je



Obrázek 9: Automat pro modelování nálad hráče ve hře šachy. Stavy a události jsou popsány v textu. Převzato z [?], upraveno. {img:ChessEmosFA}

návrh optimální směrovací tabulky pro celou síť.

9.4 Teorie her, počítačové hry

V [?] je použit fuzzy automat pro modelování nálady hráče ve hře šachy. Automat je tvořen čtveřicí stavů reprezentující jednotlivé „elementární" nálady, konkrétně „zamyšlený" (označme q_F), „překvapený" (q_S), „radostný" (q_H) a „rozzuřený" (q_A). Přechodová abeceda je tvořena následnujícími nastanuvšími tahy (ty tvoří abecedu událostí):

- tah, který hráče dostal do výhody (označme p_+)
- tah, který dostal hráče do nevýhody (p_-)
- tah, po kterém se hráč octl v šachu (c)
- tah, kdy se nestalo nic z předchozích (n

Přechodová pravidla pak mohou obsahovat například: dostane-li se hráč v rozzuřeném stavu do výhodné pozice, přejde do radostného stavu. Pokud se v dalším tahu nestane nic zajímavého, přechází do zamyšleného stavu. Kompletní automat je k vidění na obrázku 9.

Na základě aktuálního fuzzy stavu (kombinace různých stupňů elementárních nálad) automatu je upravována mimika hráče a jeho obličej je zobrazován v počítačové hře. Ukázka některých herních situací je na obr. 10. Pro lepší uživatelský dojem autoři navíc implementovali plynulé přechody mezi stavy.

Podobná technika je popsána v [?]. Zde ovšem neslouží k popisu emocí hráče, ale predikci možného dalšího vývoje hry.

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

9.5 Interakce s člověkem

V [?] je prezentován prototyp systému pro modelování lidských emocí. Systém je založen na faktu, že velký vliv na emoce člověka má počasí. V jejich modelu

Obrázek 10: Ukázka výrazů ve tváři hráče při ztrátě pěšce (vlevo) a při sebrání dámy (vpravo). Převzato z [?]. {img:ChessEmosScreens}

pak konkrétně teplota vzduchu a množství světla. Oba tyto ukazatele popisují pomocí dvojice lingvistických proměnných, každá se třemi štítky. Navržený fuzzy automat je tvořen devíti stavy (odpovídající kombinacím hodnot těchto lingvistických proměnných).

Přechody mezi jednotlivými stavy prostředí poté modelují fuzzy automatem na základě změn počasí. Na základě aktuálního stavu automatu pak pomocí tzv. Plutchikova kola emocí určijí emoční stav pozorované osoby.

V [?] pomocí pravděpodobnostních automatů analyzují tvrzení. U tvrzení sledují jeho tzv. polaritu, tedy zda je tvrzení míněno pozitivně, negativně nebo neutrálně). Každému slovu tvrzení je přiřazena hodnota lingvistické proměnné „polarita slova“ (na základě slovníku). Automat pak obsahuje stavy popisující aktuální polaritu. Při zpracování pozitivního slova se přesouvá do stavu „více“ pozitivního, při negativném naopak a při neutrálním setrvává.

V [?] používají pravděpodobnostní automat pro řízení robotického pracovníka banky. Robot se ptá uživale v závislosti na tom, v jakém stavu se nachází (a na základě odpovědí svůj stav mění). Příkladem stavu může být „Uživatel si chce vybrat peníze“. Dotazy mohou být „Přejete si vybrat v Korunách?“, „Přejete si vybrat v Eurech?“ nebo „Přejete si vrátit se k předchozí volbě?“.

9.6 Sledování pohybu a aktivit osoby

V [?] je prezentován postup, jak pomocí sledování různých parametrů (pohyb získaný akcelerometrem a elektrická vodivost kůže) určit pravděpodobnou činnost (nicnedělání, chůze, práce, odpočinek) osoby. Ty jsou popsány lingvistickými proměnnými, které tvoří události automatu. Stavy automatu odpovídají aktivitám (např. odpočinek, pomalá chůze, běh). Pravidla jsou stanovena empiricky – např. je pravděpodobnější, že člověk ze stavu nicnedělání přejde do stavu práce než do stavu běh.

Stejnou techniku využívají v [?]. Zde však detekují činnosti: chůze, práce u sebe, hovor s kolegy, dávání si kávy, míting, a to pomocí polohy zařízení (chytrý telefon) (určené triangulací Wifi sítě) osoby a polohy těla. Poloha těla pak může být buď, že osoba stojí, sedí nebo jde. Tyto hodnoty jsou sledovány akcelerometrem, pohybovým čidlem, a gyroskopem.

Podobnou techniku popisují v [?]. Zde však namísto fuzzy automatů používají pravděpodobnostní automat. Uvádějí však další aplikace pro tuto techniku, například identifikace pravděpodobného lupiče v bance.

9.7 Průmyslové řídící systémy, fuzzy kontroléry

Vztah fuzzy automatů a řídících systémů jsou popsány např. zde [?], [?], [?] nebo [?].

Systémy k modelování bývají často tak složité, že je obvykle nemožné popsat je zcela kompletně. Mohlo by tak například z důvodu chyby nebo neočekávaného vnějšího podnětu dojít k tomu, že model se octne v nekonzistením stavu, ze kterého nebude schopen se zotavit. Použití fuzzy automatů umožňuje do modelů určitou formu zotavené zanést.

*

V [?] používají učící se fuzzy událostmi řízené automaty, pro řízení výkonu při obloukovém svařování. Vstup automatu je realizován pomocí lingvistické proměnné „naposledy bylo provedeno“ se štítky „zvýšení výkonu“ a „snížení výkonu“. Automat je sestaven tak, aby správně rozpoznával vhodné posloupnosti zvýšení a snížení výkonu za účelem zvýšení kvality sváru.

V téže publikaci demonstrují obdobným způsobem uplatnění pro řízení robota pohybujícího se v neznámém prostředí. Systém sleduje změny rychlosti pohybu robota a na jejich základě řídí přidávání a ubírání výkonu tak, aby se robot v libovolném terénu pohyboval konstantní rychlostí.

V [?] a [?] je fuzzy automat použit jako model pro návrh řídících systémů. Řídící systém je stavový stroj, který je překlápěn mezi diskrétními stavy pomocí událostí. Řídící tak systém může být reprezentován automatem.

9.8 Problém městského růstu {subs:UrbGrow}

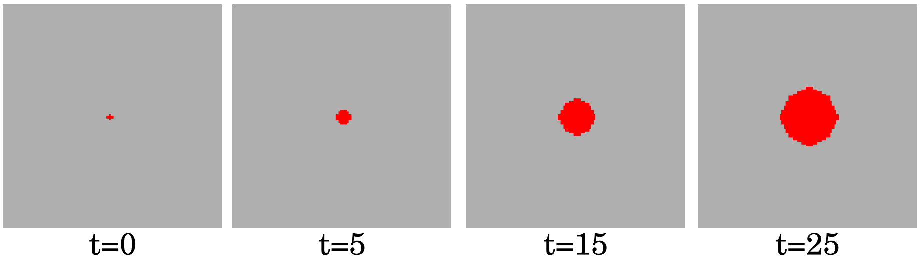
Problém městského růstu je problém z oblasti urbanistiky. Řešením tohoto problému je co nejpřesnější predikce rozvoje městské zástavby na základě historických záznamů a současné situace. Ve zobecněné podobě se nemusí jednat jen o růst městské zástavby, ale například nárust vytíženosti silnic, kácení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

Tento problém bývá často řešen pomocí buněčných fuzzy automatů. Věnují se jim např. v [?], [?] [?] [?] [?] [?], [?] [?] či[?]. Další literatura věnující se uplatnění (fuzzy) buněčných automatů při řešení problém městského růstu je k dispozici např. zde: [?], [?], [?] a [?]. V [?] podobnou technikou simulují šíření lesního požáru.

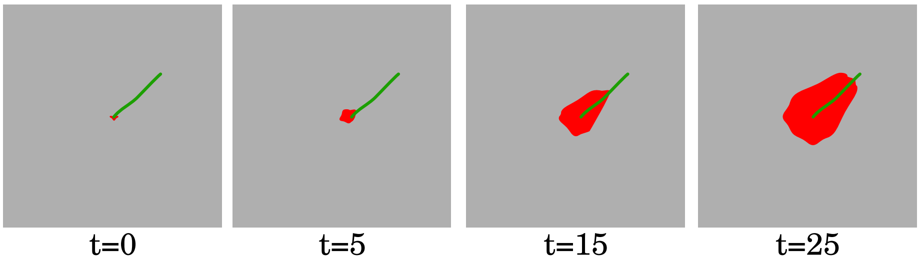
Problém bývá řešen obvykle tak, že sledovaná oblast (město) je rozděleno na parcely. Každá parcela je popsána obvykle několika ukazateli, např. jak moc je zastavěna či jak moc hluku parcela produkuje. Tyto ukazatele tvoří stav buňky bunečného automatu. Konfigurace automatu pak tedy popisuje celé město v určitý časový okamžik.

Přechodová pravidla pak mohou vypadat například (ukázka vlivu různých pravidel na růst města je vyobrazen na obrázku 11):

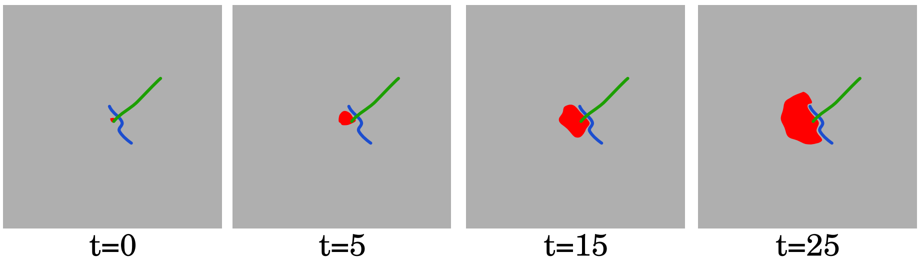
- Je-li vzdálenost parcely od centra města malá, pak růst zástavby bude velký



(a) Růst města bez dalších dodatečných podmínek



(b) Růst města podél silnice



(c) Růst města bržděn řekou

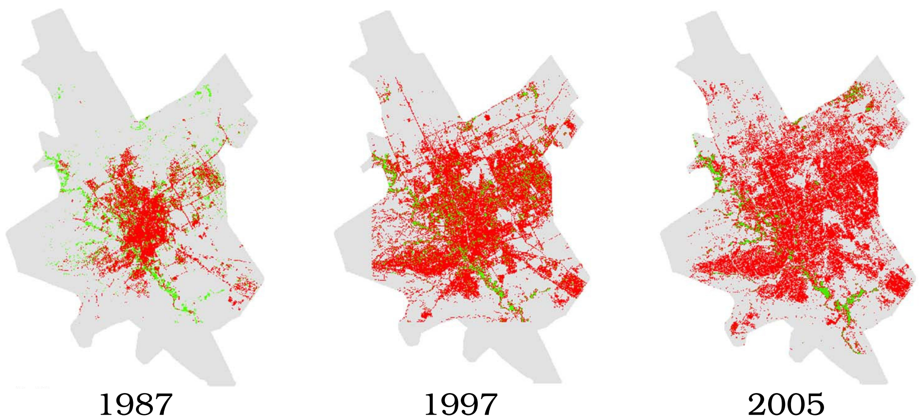
Obrázek 11: (převzato z [?], upraveno) Ukázky chování automatu při různých počátečních konfiguracích. Šedě jsou znázorněny prázdné parcely, červeně zástavba, zeleně hlavní silnice a modře řeky. {img:VarTransRuls}

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

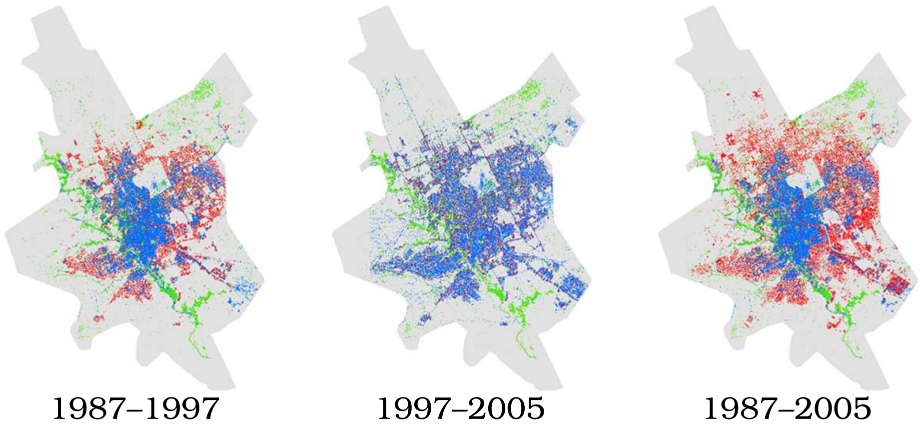
Na obrázku 12 je k vidění konkrétní ukázka městského růstu.

9.9 Další uplatnění

V [?] prezentují nástroj založený na pravděpodobnostních automatech, jehož úkolem je určovat nejpravděpodobnější aktivitu počítačového programu. Autoři mají k dispozici softwarový nástroj pro výpis nízkoúrovňového chování počítačového programu (např. systémová volání, změny na programovém zásobníku). Sledováním těchto informací při známém chování programu mohou stanovit vzory chování programu. Pravděpodobnostní automaty sestavené na základě těchto



(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 12: (převzato z [**?**]) Ukázky simulace městského růstu {img:UrbGroProSample} vzorů tak mohou modelovat chování programu. Automat tak například může předpovídat, že pokud program otevřel soubor, pak následující akcí bude s nejvyšší pravděpodobností alokace paměti na haldě.

V [**?**] navrhují používat učení fuzzy automatů pro konstrukci logických klopných obvodů. Tj. obvodů, které se na základě vstupů překlápí mezi různými diskrétními stavy. Autoři doporučují navrhovat elektronické takové obvody tak, že se dle očekávaného chování obvodu sestaví fuzzy automat (pomocí učení). Ten se poté „zaokrouhlí", tj. převede na dvojstavový a převede na elektronický obvod.

10 Zpracování obrazu

{cha:ImgProc}

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

10.1 Konvoluce

{subs:Convol}

Uvažujme obraz jako mřížku $m \times m$ pixelů s odstíny šedi jako hodnotami od 0 do 1. Hodnota 0 značí černou, hodnota 1 bílou. Jinými slovy, o pixelu můžeme hovořit jako o ligvistické proměnné „pixel má bílou barvu".

Každý obraz tak můžeme považovat za konfiguraci buněčného fuzzy automatu. Návrhem vhodné přechodové funkce tak můžeme vytvořit automat, který provádí určitou operaci pro úpravu obrazu. Typickou operací je tzv. obrazový filtr, což je zobrazení které obrazu přiřazuje jiný obraz stejných rozměrů.

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

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

je konvoluční maticí jednoduchého rozostření. Konvoluční matice se následujícím způsobem:

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

kde S je součet hodnot v matici B.

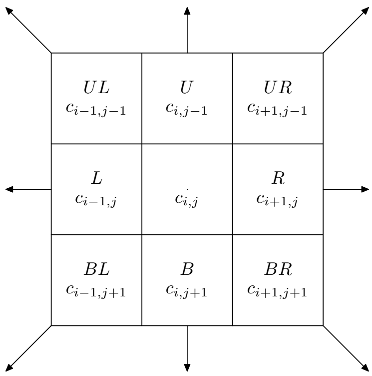
Další ukázkou grafického filtru pracujícího s využitím buněčného fuzzy automatu je například filtr pro zvýraznění tvarů. Je daný následujícím předpisem

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

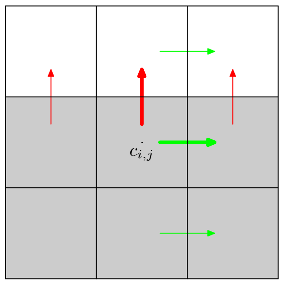
kde $\epsilon > 1$ je parametr udávají agresivitu zvýrazňování a $neighs_{i,j}$ je součet hodnot okolních buňek.

Ukázky aplikací obou filtrů jsou k nalezení na obrázku 13. V následujících podkapitolách budou prezentovány některé další (pokročilejší) techniky zpracování obrazu využívající buněčné fuzzy automaty.

Obrázek 13: Ukázky jednoduchých filtrů. Vlevo původní obrázek, uprostřed obrázek po 5 generacích jednoduchého rozostřovacího filtru a vpravo obrázek po 8 generacích filtru pro zvýraznění tvarů ($\epsilon = 1{,}1$). {img:Filters}



(a) 8 směrů



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

10.2 Hledání hran

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

Označme osmici směrů dle obrázku 14a jako U, UR, R, BR, B, BL L a UL. Množinu těchto směrů označme dim. Dále označme c_X (kde $X \in dim$) jako sousední buňku buňky c ve směru $X \in dim$.

Rozpoznávání hran vychází z následující úvahy: Má-li buňka c_X výrazně jinou barvu, než buňka c, pak buňkou c prochází hrana ve směru X (červené šipky v obrázku 14b). Má-li buňka c_X velmi podobnou barvu než buňka c, pak buňkou c neprochází hrana ve směru X (zelené šipky v obrázku 14b).

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

Dle [?] může být hodnota lingvistické proměnné „buňkou c prochází hrana" dále použita pro zaostřování obrazu.

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

Obrázek 15: (převzato z [?]) Ukázka odstraňování šumu {img:Noises}

10.3 Ostraňování šumu

{subs:NoisRem}

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

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

V první fázi je klasickým buněčným automatem šum detekován. Buňka obsahuje šum, pokud rozdíl její barvy od průměrné barvy jejich sousedů překračuje stanovenou mez. Tato mez může být stanovena statisticky, například na základě směrodatné odchylky barev pixelů celého obrazu. V druhé fázi je aplikován buněčný fuzzy automat, který buňky obsahující šum nahradí hodnotami spočtenými z jejich okolí. Ukázky výsledků jsou na obrázku 15.

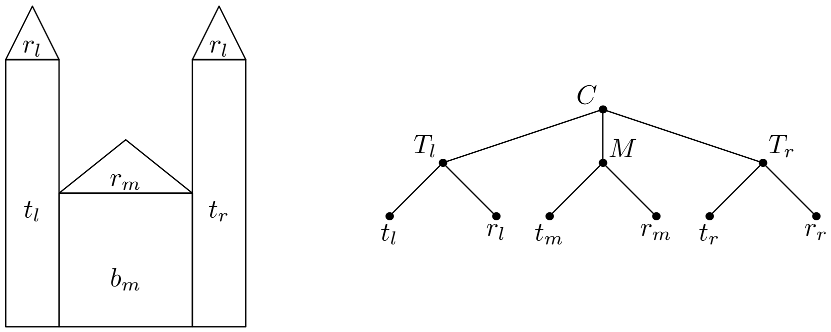
Velmi podobný způsob ostranění šumu je popsán v [?]. Zde však operace detekce šumu a jeho odstranění provádějí v jednom kroku.

10.4 Rozpoznávání jednoduchých vzorů

Ropoznávání vzorů je další z častých způsobů práce s obrazy. Obecně je problém definován (obdobně, jako rozpoznávání textových vzorů v kapitole 7) jako problém určení, zda-li obraz obsahuje předem stanovený vzor či ne. Obvykle nás také zajímá, kde přesně se vzor v obraze vyskytuje.

V [?] je popsán způsob, který popisuje rozpoznávání vzorů v obrazu velikosti $1 \times m$ pomocí (jednodimenzionálního) buněčného automatu. Automat pracuje s lingvistickou proměnnou „pixel má šedou barvu“, která má několik štítků (reprezentující různé stupně šedi). Přechodová pravidla jsou navžena tak, aby rozpoznávaný (a případně jemu podobný) vzor zvýrazňovala. Přechodová pravidla však obvykle bývají tvořena všemi kombinacemi odstínů pro všechny sousední buňky, čímž značně roste její mohutnost. Tato technika je proto prakticky nepoužitelná.

Zcela jiný přístup pužívají v [?]. Vzor nepovažují za konkrétní kombinaci odstínů barev, ale jako část obrazu splňující určité vlastnosti. Konkrétně, skvrny jednolité barvy, které jsou obklopeny různorodě zabarvenou plochou. Podrobnější popis (včetně ukázky výstupů) je uveden v kapitole 11.4.



Obrázek 16: Příklad složeného geometrického tvaru a stromová reprezenace jemu odpovídajícímu S-termu {img:Geoms}

10.5 Složené geometrické útvary {subs:CompGeoms}

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

Příklad 10.1. *Na obrázku 16 je vyobrazena ukázka složeného geometrického útvaru vyobrazující „budovu kostela“ a stromová reprezentace jemu odpovídajícího S-termu.*

Mějme primitivní geometrický tvar x. Pak tento tvar je čtvercem, pokud je tvořen čtveřicí bodů a všechny jeho vnitřní úhly jsou pravé. Vlastnost „být pravý úhel“ můžeme snadno vyjádřit stupni pravdivosti. Tedy, že úhel $\pi/2$ je pravým ve stupni 1 a například úhel π v nulovém stupni. Díky tomu můžeme i vlastnost „být čtvercem“ uvažovat se stupni pravdivosti.

Obdobným způsobem můžeme popsat různé další primitivní geometrické tvary. Stejně tak můžeme stanovit stupně pravdivosti pro vlastnosti např. „geometrický tvar x má neprázdný průnik s geometrickým tvarem y“ či „geometrické tvary x a y jsou přímky a jsou navzájem rovnoběžné“.

Na základě těchto popisů může být navržen automat, který S-termy reprezentující geometrické tvary rozpoznává a to včetně geometrických tvarů „podobných“ vzoru.

10.6 Detekce požárů

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

V první fázi se snímek rozdělí na několik regionů a na základě barvy (teplé světlé barvy) se určí, zda-li jednotlivé regiony mohou být plamenem (tzv. kandidáti). Pro každého kandidáta je pak sestaven fuzzy automat o čtyřech stavech q_{VL}, q_L, q_H, q_{VH}. Pokud se automat regionu nachází ve stavu $q_V L$, znamená to, že „region je velmi málo pravděpodobně tvořen plamenem“. Obdobně pro q_L („málo pravděpodobně“), q_H („hodně pravděpodobně“) a q_{VH} („velmi pravděpodobně“). Abecedou událostí jsou pak kombinace dalších atributů (svit, pohyb v určitém směru, vlnění) spočtených z předchozích snímků. Přechodové pravidlo pak může být např. „pokud je region ve stavu q_H a došlo k velkému posunu směrem nahoru a malému snížení svitu, pak přejdi do stavu q_{VH}“. Přesně byla přechodová funkce navržena statistickým pozorováním známých videosekvencí s požáry.

11 Biologie a medicína

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

Vzhledem k tomu, že většina biologických a medicínských uplatnění vyžaduje znalost dané problematiky, budou tyto uplatnění rozebrána jen zevrubně.

11.1 Rozpoznávání řetězců DNA

{subs:DNA}

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

Řětězce DNA často kódují základní informace o živých organizmech a je proto snaha pochopit její strukturu.

Popis využití fuzzy a pravděpodobnostních automatů na rozpoznáváním rětězců DNA je uevend v [?], [?], [?] a [?].

11.2 Biologické simulace

V [?] je prezentován model založený na buněčných fuzzy automatech modelujících růst mořských řas. Funguje na velmi podobném principu jako problém městského růstu (viz kapitola 9.8).

V [?] používají pravděpodobnostní automaty na simulaci živého ogranizmu. Princip je podobný, jak u buněčných automatů, jen model není založen na pevné mřížce, ale buňky vznikají a mizí. Každá buňka se nachází v některém stavu z množiny stavů (jako je např. „buňka se zrodila“, „buňka je přípravena k dělení“, „buňka je rodičovskou buňkou jiné buňky“). Přechodová pravidla mohou vypadat například následovně: je-li buňka „připravena k dělení“, pak v dalším kroku provede dělení, tj. vytvoří novou buňku ve stavu „buňka se zrodila“ a buňka samotná přechází do stavu „buňka je rodičovskou buňkou jiné buňky“. Díky využití pravděpdobnostních automatů se takovéto modely mohou značně více přiblížit reálným buněčným sítím.

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

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

11.3 Analýza zdravotního stavu pacienta

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

Technika pracuje s událostmi řízeným fuzzy automatem. Stavy automatu v tomto případě reprezentují choroby (popř. stádia jedné choroby), přechodová funkce pak přechody mezi nimi. Přechodová pravidla pak popisují přechody při různých událostech či akcích (např. medikace).

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

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

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

11.4 Analýza lékařských snímků

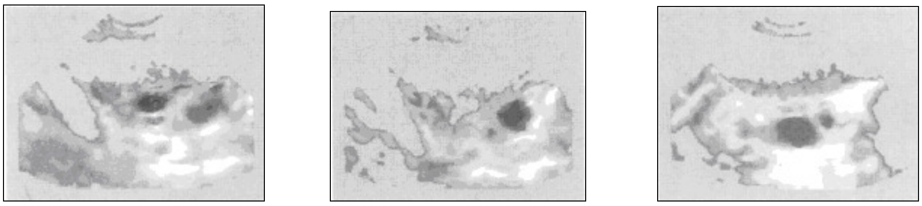
{subs:MedImgs}

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

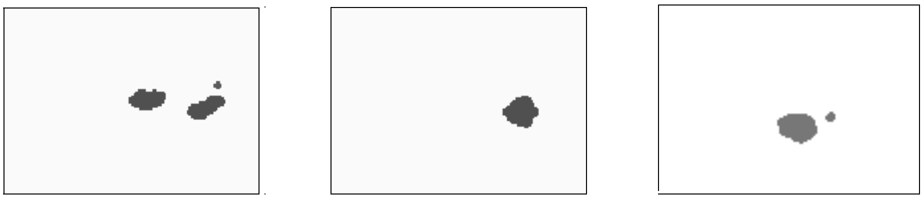
Proces rozpoznávání probíhá následujícím způsobem:

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

Stromy jsou do řetězců kódovány jako posloupnost čísel, kde každé číslo odpovídá počtu potomků jednotlivých uzlů stromu. Automat, který tyto řetězce rozpoznává, byl sestaven učením z veřejné databáze snímků.



(a) Ultrazvukové snímky vaječníků



(b) Rozpoznané folikuly

Obrázek 17: (převzato z [?], upraveno) Ukázky rozpoznávání folikul {img:Follicles}

V [?] popisují, jak pomocí buněčných fuzzy automatů rozpoznávat folikuly[13] v ultrazvukových snímcích vaječníků. Poukazují na to, že folikula je na snímku tvořena jednolitou svrnou stejné barvy, obklopena ostatní tkání (různorodě zabarvená plocha).

Automat v první fázi vyhledává folikuly. Ty posléze automat zvýrazňuje. Na obrázku 17 jsou k nahlédnutí ukázky rozpoznaných folikul pomocí této techniky.

V [?] popisují způsob, jak pomocí fuzzy automatů sledovat vývoj kostí ruky dítěte. Z Röntgenového snímku ruky pomocí detekce hran obrdrží obrysy kostí. Ty pak pomocí techniky podobné rozpoznávání složených geometrických tvarů (kapitola 10.5) používají k určování stádia vývinu patřičné kosti.

11.5 Další aplikace

{subs:BioMedRest}

V [?], popř. i [?] analyzují signál z elektrokardiogramu (EKG). Využivají při tom techniky uvedené v kapitole 8.1.

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

12 Implementace vybraných problémů

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

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

[13] Folikula je dutinka ve vaječníku, v níž probíhá zrání vajíčka. [?]

12.1 Základní informace

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

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

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

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

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

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

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

12.2 Korekce překlepů

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

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

1. výstupem je slovo x, jestliže je vstupní slovo x ve slovníku

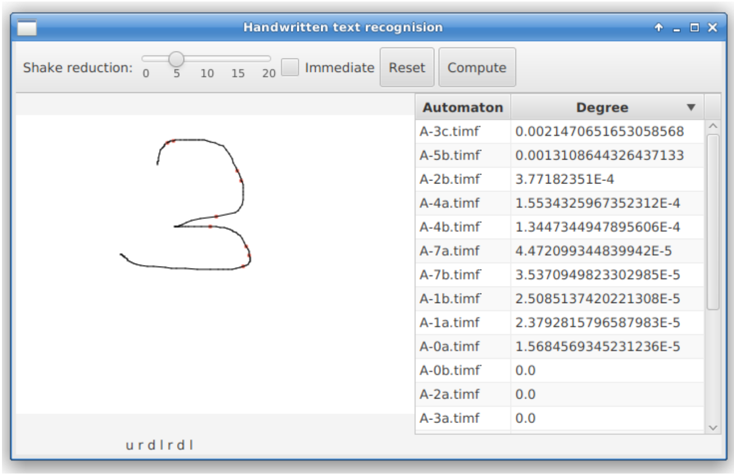
2. výstupem je slovo y, jestliže je vstupní slovo x nejpodobnější slovu y, které je ve slovníku

3. výstupem je slovo x, pokud žádné slovo ze slovníku mu není podobné

Podobnost řetězců je počítána pomocí deformovaných automatů rozpoznávající slova ze slovníku. Ukázka korekce překlepů (při výchozím nastavení) je uvedena v tabulce 5. Jak je v tabulce vidět, program má problém s krátkými slovy, avšak u dlouhých slov dokáže obnovit i velmi „poškozené“ slovo.

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

Tabulka 5: Ukázka korekce překlepů při slovníku anglických názvů měsíců {tab:TyposOut}



Obrázek 18: Ukázka aplikace pro rozpoznávání ručně psaného textu {img:HandWritScreen}

12.3 Rozpoznávání ručně psaného textu

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

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

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

Ukázka okna aplikace je na obrázku 18.

12.4 Detekce úplných m-árních stromů

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

12.5 Simulace spotřeby elektrického produdu

Simulace spotřeby elektrického proudu byla naimplementována dle kapitoly 9.3. Výpočet realizuje třída `PowerConsumptionComputer`. Jejím vstupem jsou informace o spotřebičích v síti, jejich možných stavech a spotřebách. Dále pak seznam naměřených spotřeb elektrického proudu a dodatečné parametry.

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

12.6 Zpracování obrazu

Naimplementovány byly také vybrané techniky pro zpracování obrazu. Realizovány byly pomocí buněčných fuzzy automatů. Pro import a export (převod mezi bitmapou a souborem konfigurace buněčného automatu) je slouží spustitelná třída `ImageConfigConverter`.

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

Třída `NoiseReductionAutomaton` implementuje metodu pro odstranění šumu z obrazu popsanou v kapitole 10.3. Pro přidání šumu do obrázku je možné použít spustitelnou třídu `ConfigNoiserTool`.

12.7 Metoda lisování dat {subs:DataPresImpl}

Metoda lisování dat byla popsána v kapitole 8.7. Metoda byla naimplementována tak, aby bylo možné ji použít nejen na binární data. Implementace je vybavena jednoduchým binárním škálováním, tedy nahrazením hodnoty každého atributu 0 resp. 1 pokud překračuje stanovenou prahovou hodnotu. Pokud prahové hodnoty nejsou uvedeny, jsou spočítány jako medián hodnot daného atributu.

Samotnou metodu implementuje třída `DataPressurePerformer`. Data pro tuto třídu, tj. prahové hodnoty, trénovací a testovací množina a parametr δ jsou preprezentovány instancemi třídy `DataPressureDataset`. Program se spouští třídou `DataPressureApp`. Pro minimalizaci byl použit algoritmus minimalizace bivalentních automatů z [**?**] upravený tak, aby pracoval s fuzzy koncovými stavy a fuzzy přechodovou funkcí.

Metoda byla otestována na datasetu Iris [**?**]. Dataset je tvořen 150 záznamy spadající do tří tříd (setosa, versicolor a virginica). KAždý záznam je tvořen čtyřmi vstupními reálněčíselnými atributy.

Bylo provedeno několik experimentů. Ukázalo se, že nejlepších výsledků se podařilo dosáhnout při $\delta = 0$, produktové t-normě a rozpoznávání třídy (tj. určování zda-li záznam patří či nepatří do třídy) setosa. Při testování byly dva náhodné záznamy z každé třídy odebrány z trénovací množiny a vloženy do testovací. Při této konfiguraci model fungoval korektně (tj. všechny testovací záznamy z třídy setosa označil vyšším stupněm pravdivosti, než záznamy z jiných tříd) přibližně ve 3/4 takovýchto pokusů.

Metoda byla také otestována na datasetu flag data [?] ve snaze určit, zdali stát (záznam v tabulce) se nachází či nenachází v Evropě na základě barev (červená, zelená, modrá, zlatá, bílá, černá, oranžová), které obsahuje. Zde však metoda nepřinesla žádné výsledky, tj. tuto třídu nedokázala rozpoznávat při žádné konfiguraci.

Zde je vhodné upozornit, že tato metoda je přímo závislá jednak na kódování vstupních dat do symbolů a řetězců, ale také na použitém algoritmu minimalizace fuzzy automatu. Oba tyto kroky byly implementovány velmi jednoduchým způsobem. Pokud by bylo kódování hodnot na symboly řešeno vícestupňovým škálováním, automat by tak snáze mohl reflektovat[14] podobnost blízkých hodnot. Stejně tak, návrh účinnějšího minimalizačního algoritmu by mohl přispět pro přesnější rozpoznávání tříd.

*

13 Závěr

Úkolem této práce bylo ukázat, že fuzzy a pravděpodobnostní automaty mohou najít uplatnění v praxi. V rámci této práce bylo nastudováno více než 200 kusů literatury (článků a knih), ve kterých bylo vyhledáno značné množství praktických aplikací těchto modelů.

Tato práce tedy prokázala, že fuzzy a pravděpodobnostní automaty nacházejí uplatnění pro řešení různých problémů. V mnoha případech fuzzy či pravděpodobnostní automaty značně zjednodušují či zpřehledňují řešení patřičného problému.

Na druhou stranu, použití těchto modelů je spjato s některými dalšími problémy. V drivé většině případů je nutné provést vhodné přezdpracování dat, které výrazně ovlivňuje podobu použitého automatu. Stejnětak, samotný automat obvykle nedokáže dostatečně pružně pracovat s reálnými vstupy a je třeba upravit např. deformováním nebo strojovým učením.

Dalším faktorem, který výrazně komplikuje nasazení fuzzy automatů v praxi, je časová složitost jejich výpočtu. Již při prováděných experimentech na relativně malých datech se ukázalo, že 1 výpočet na běžném počítači mnohdy překročí jednu sekundu. Pro použití v praxi je tak obvykle nutné provést značné optimalizace. To často vede ke vzniku kompletně nového modelu, jehož je fuzzy automat pouze základní myšlenkou.

Zatímco při delších vstupních řetězcích automat trpí dlouhým časem výpočtu, při krátých pak naopak automat obvykle bojuje s korektností výsledků. Stává se, že automat chybně krátký řetězec rozpozná, protože automat „nestihne“ přejít do stavu, který by jej zamítal.

Přes to všechno je patrný rostoucí zájem o uplatnění fuzzy a pravděpodobnostních automatů. Jen za poslední tři roky, kdy tato práce vznikala, byla vydána další publikace věnující se aplikaci fuzzy automatů. Dle autorova názoru však využití fuzzy a pravděpodobnostních automatů bude v budoucnu utlačováno stále vyšší popularitou umělé inteligence. Ta většinu z představených problémů dokáže řešit mnohdy i jednouššeji, ovšem na úkor elegance řešení.

[14] Toho by šlo dosáhnout například deformací „náhrada symbolu“

Další z poznatků, který dle autorova názoru práce přinesla, je, že značné množství autorů si nedostatečně uvědomuje rozdíl, mezi fuzzy a pravděpodobnostním přístupem. V práci bylo opakovaně naráženo na problémy, které bylo vhodné řešeit spíše pomocí fuzzy automatů namísto pravděpodobnostních (a naopak). Stejnětak, některá literatura popisující aplikace fuzzy automatů na problematiku nahlížela z pohledu gramatik, přestože to není zcela přesné.