

Aplikace fuzzy a pravděpodobnostních automatů

Martin Jašek

12. září 2016 — ??

Obsah

1	Definice a značení	2
2	Fuzzy automaty, gramatiky a jazyky	4
2.1	Jazyk rozpoznávaný fuzzy automatem	5
2.2	Fuzzy a bivalentní regulérní jazyky	5
2.3	Fuzzy regulární výrazy	5
3	Rozpoznávání textových vzorů	5
3.1	Formální zavedení problému	5
3.2	Motivace k použití fuzzy automatů	6
3.3	Automat rozpoznávající ω	6
3.4	Podobnost symbolů	8
3.5	Fuzzy symboly	10
3.6	Editační operace	11
3.7	Deformovaný automat	16
3.8	Shrnutí	17
4	Fuzzy tree automaty	17
4.1	Zavedení	17
4.2	Stromy a termy	18
4.3	Fuzzy tree automat a jazyk jím rozpoznávaný	20
4.4	Použití fuzzy tree automatů	22
5	Buněčné fuzzy automaty	24
6	Konkrétní příklady	25
6.1	Rozpoznávání ručně psaného textu	25
6.2	Detekce překlepů	25

1 Definice a značení

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

Abecedy, řetězce, jazyky

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

Fuzzy teorie

Fuzzy množiny a relace budou po vzoru [7] nejčastěji malými řeckými písmeny (obdobně jako jejich členské (angl. „membership“) funkce). Množinu všech fuzzy podmnožin množiny S budeme značit $\mathcal{F}(S)$.

Deterministický bivalentní automat

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

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

Nedeterministický bivalentní automat

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

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

Základní definice nedeterministického fuzzy automatu

Značení je převzato z [7] a lehce upraveno.

Definice 1.3 (Nedeterministický fuzzy automat). Nedeterministický fuzzy automat A je pětice $(Q, \Sigma, \mu, \sigma, \eta)$, kde Q je konečná množina stavů, Σ je abeceda, μ je fuzzy přechodová funkce (fuzzy relace $Q \times \Sigma \times Q \rightarrow [0, 1]$) a σ a η jsou po řadě fuzzy množiny nad Q počátečních, resp. koncových stavů.

{def-ZaklDefNedFuzzAut}

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

{def-FuzzStav}

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

Definice 1.6 (Přechodová funkce fuzzy stavů). *Mějme nedeterministický fuzzy automat A . Pak přechodová funkce fuzzy stavů je fuzzy relace $\hat{\mu} : \mathcal{F}(F) \times \Sigma \rightarrow \mathcal{F}(F)$ taková, že pro každý fuzzy stav $V \in \mathcal{F}(Q)$ a symbol $x \in \Sigma$ je $\hat{\mu}(V, x) = V \circ \mu[x]$.*

{def-PreFunFuzzStav}

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

{def-PreFunFuzzStav}

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

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

{def-RetPriAut}

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

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

(zde bude doplněno: ověřit, dohledat, ozdrojovat)

{def-JazRozpAut}

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

(zde bude doplněno: ověřit, dohledat, ozdrojovat)

Nedeterministický fuzzy automat s ϵ přechody

{def-NedFuzzAutEpsPre}

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

(zde bude doplněno: Tady by asi bylo vhodné rozebrat ϵ -uzávěry)

Reprezentace fuzzy automatu

(zde bude doplněno: dohledat zdroje)

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

Přechodový diagram: Notace s lomítky např. zde: [9].

Tabulka: stav \times symbol nebo stav \times stav [10]?

Konstrukce fuzzy automatu z konečného automatu

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

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

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

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

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

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

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

{def-FuzzAutBivAut}

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

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

(zde bude doplněno: vymyslet nějaký fakt pěkný příklad)

2 Fuzzy automaty, gramatiky a jazyky

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

(zde bude doplněno: pojem „Lattice language“)

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

2.1 Jazyk rozpoznávaný fuzzy automatem

Věta 6.3 [14] (pro lattice monoid, není to někde jen pro $[0, 1]$?).

Dle definice 4 [9] je fuzzy regulární jazyk fuzzy podmnožina bivalentního.

Automat s bivalentní μ and η (a fuzzy σ) taky rozpoznává fuzzy regulární jazyk [5]. Neřešil něco takového i Bel? Jinak řečeno, support konečný automat [9].

2.2 Fuzzy a bivalentní regulární jazyky

Univerzum fuzzy jazyka je regulární jazyk, pozorování 6.1 [14].

Stejně tak, zaříznutý jazyk (α -řez jazyka) je také regulární, věta 2.2 [5].

Pumping lemma pro fuzzy regulární jazyky, lemma 4-7 pro různé typy automatů [9].

Uzávěrové vlastnosti fuzzy regulárních jazyků, např. [5].

2.3 Fuzzy regulární výrazy

LiPed-FuzzFinAutFuzzRegExMembValLattOrdMon, definice 5.1, 5.2 (+ opsat důkaz, že $[0, 1]$ je lattice monoid) [14].

Algoritmus převodu reg na aut, [13]. Ale zdá se mi to až moc složité.

3 Rozpoznávání textových vzorů

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

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

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

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

3.1 Formální zavedení problému

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

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

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

Poznámka 3.2. *Vzorový řetězec budeme v této kapitole vždy značit ω , pozorovaný pak α .*

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

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

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

3.2 Motivace k použití fuzzy automatů

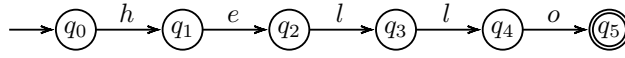
Klasická teorie automatů vznikla jako nástroj pro zpracování textových řetězců. Z tohoto důvodu je rozpoznávání textových vzorů jejím základním výsledkem. Automaty obecně jsou nástroje sloužící pro rozhodování, zda-li řetězec odpovídá vzoru automatem reprezentovaném. Použití pro rozpoznávání řetězcového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

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

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

3.3 Automat rozpoznávající ω

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



Obrázek 1: Automat rozpoznávající **hello**

{diag-AutRozpHell}

přijímá jediný řetězec ω a všechny ostatní zamítá. Nyní si takovýto automat zkonstruujeme.

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

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

Takovýto automat označme jako *automat rozpoznávající ω* .

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

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

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

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

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

{def-AutRozpOme}

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

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

Následuje vytvoření fuzzy automatu.

{def-FuzzAutRozpOme}

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

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

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

3.4 Podobnost symbolů

Nezákladnější technika pro zanesení neostrého (stupňovitého) rozpoznávání je s využitím podobnostní relace symbolů. Tato technika byla přejata z [2]. Myšlenkou této techniky je, že symbol v pozorovaném řetězci může být snadno zaměněn za jiný, podobný, jemu odpovídající v řetězci vzorovém.

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

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

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

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

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

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

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

{def-AutPracGS}

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

Nyní aplikujeme tento způsob konstrukce fuzzy automatu na automat rozpoznávající ω .

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

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

Následuje jednoduchý příklad takového automatu.

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

{ex-AutRozp0mePodSym}

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

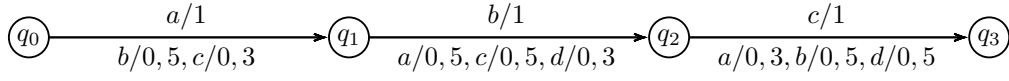
Tuto relaci můžeme zapsat do matice (sloupce i řádky odpovídají po řadě symbolům a, b, c, d):

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

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

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

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



Obrázek 2: Automat rozpoznávající abc pracující s g_s

{diag-AutRozpABCPracGS}

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

3.5 Fuzzy symboly

Fuzzy symbol je technika využívající podobnosti symbolů. Ve své podstatě se jedná o téže techniku jak v předchozí podkapitole, jen je na ni nahlíženo jinak. Oproti podobnosti symbolů je použití fuzzy symbolů komplikovanější, avšak umožňuje jednoduše tuto techniku kombinovat s jinými. Princip fuzzy symbolů byl přejat z [8].

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

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

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

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

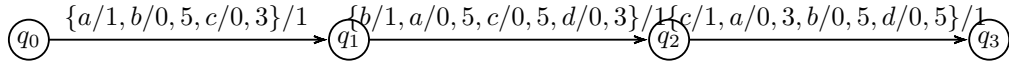
Definice 3.7 (Abeceda fuzzy symbolů). *Mějme abecedu Σ a fuzzy symboly \tilde{y} pro všechna $y \in \Sigma$. Pak množinu všech těchto fuzzy symbolů nazvěme abeceda fuzzy symbolů abecedy Σ a označme $\tilde{\Sigma}$. Tedy $\tilde{\Sigma} = \{\tilde{y} \mid y \in \Sigma\}$.*

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

(zde bude doplněno: sjednotit značení ω vs. α , když se používá jen jeden obecný řetězec)

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

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



Obrázek 3: Automat rozpoznávající abc pracující s g_s

{diag-AutRozpOmePraFuzSym}

{def-AutPracFuzzSym}

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

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

Formální zavedení automatu pracujícího s fuzzy symboly je intuitivní, jedná se jen o formalitu. Abychom však využili potenciál fuzzy symbolů, je třeba pozměnit výpočet automatu. Proces jeho výpočtu se změní ve fázi výpočtu přechodové funkce fuzzy stavů. Připomeňme, že ta je definována (definice 1.7) jako fuzzy relace $\hat{\mu}$ přiřazující každému fuzzy stavu V a fuzzy symbolu x fuzzy stav dle předpisu

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

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

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

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

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

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

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

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

3.6 Editační operace

Další technikou pro podobnostní porovnávání pozorovaného a vzorového řetězce je s využitím editačních operací. Tato technika byla přejata z [7]. Základní idea této techniky spočívá v trojici jednoduchých editačních operací, jejíž složením jsme schopni popsat transformaci pozorovaného řetězce na vzorový. Množství transformace pak udává podobnost pozorovaného a vzorového řetězce.

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

Definice 3.11 (Editační operace). *Mějme abecedu Σ , uvažujme množinu $E = (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\}) \setminus \{(\epsilon, \epsilon)\}$. Pak každou dvojici $(x, y) = z \in E$ nazvěme editační operace. Speciálně pak, pro všechna $x, y \in \Sigma$, $(x, y) \in E$ znamená nahrazení symbolu x symbolem y , $(x, \epsilon) \in E$ znamená odebrání symbolu x a naopak $(\epsilon, y) \in E$ pak vložení symbolu y . Navíc jako editační operaci uvažujeme i všechny dvojice $(x, x) \in E$ (pro každé $x \in \Sigma$) symbolizující „žádnou editaci“.*

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

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

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

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

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

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

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

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

Podíváme-li se nyní jen na levé části editačních operací ve vyrovnání ζ_1 z předchozího příkladu, zjistíme, že jejich zřetěžením získáme řetězec α :

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

Stejně tak, zřetěžením pravých částí editačních operací v ζ_1 získáme řetězec ω :

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

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

Definice 3.12 (Vyrovnání řetězců [7]). *Jako množinu všech vyrovnání $G(\alpha, \omega)$ řetězce α na řetězec ω (kde $(\epsilon, \epsilon) \neq \alpha, \omega \in \Sigma^*$) označme takovou množinu $\{\zeta \in E^+ \mid \zeta \text{ splňuje vlastnosti 1., 2. i 3.}\}$*

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

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

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

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

{def-FuzzMir}

Definice 3.13 (Fuzzy míra[7]). *Mějme binární fuzzy relaci R nad $\Sigma \cup \{\epsilon\}$. Pak jako fuzzy míru mezi řetězci $\alpha, \omega \in \Sigma^*$ (značenou $S_{\Sigma, R, \otimes}$) označme fuzzy relaci danou následujícím předpisem:*

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

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

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

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

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

{def-AutRozpCaLL}

Definice 3.14 (Automat rozpoznávající \mathcal{L} [7]). *Mějme binární fuzzy relaci R nad $\Sigma \cup \{\epsilon\}$ (stejná jako v definici 3.13). Pak pro vzorový řetězec $a_1 a_2 \dots a_n = \omega \in \Sigma^*$ označme $M_{\Sigma, R, \otimes}(\omega)$ automat rozpoznávající jazyk $\mathcal{L}(\omega)$ dle definice 1.10, takový, že*

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

- (c) $\mu(q, q', x) = 0$ pro všechny $q, q' \in Q$ nesplňující předchozí dva body
 - (d) $\mu(q_i, q_i, \epsilon) = 1$ pro všechny $q_i \in Q$ taková, že $i = 0, \dots, n$
 - (e) $\mu(q_i, q_{i+1}, \epsilon) = R(\epsilon, a_{i+1})$ pro všechny $q_i \in Q$ taková, že $i = 0, \dots, n-1$
 - (f) $\mu(q, q', \epsilon) = 0$ pro všechny $q, q' \in Q$ nesplňující předchozí dva body
3. množina počátečních stavů $\sigma: \sigma(q_0) = 1$ a pro všechny ostatní $q_0 \neq q' \in Q: \sigma(q') = 0$
 4. množina koncových stavů $\eta: \eta(q_n) = 1$ a pro všechny ostatní $q_n \neq q' \in Q: \eta(q') = 0$

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

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

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

Důkaz. Kompletní důkaz je k nalezení v [7]. □

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

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

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

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

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

} (přechody dle bodů (b) a (d) v definici jsou s nulovým stupněm a ve výpisu jsou vynechány)

$$3. \sigma = \{q_0/1, q_1/0, q_2/0, q_3/0\}$$

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

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

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

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

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

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

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

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

- řetězec $\alpha = ab$:

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

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

- řetězec $\alpha = bbb$:

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

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

- řetězec $\alpha = abaca$:

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

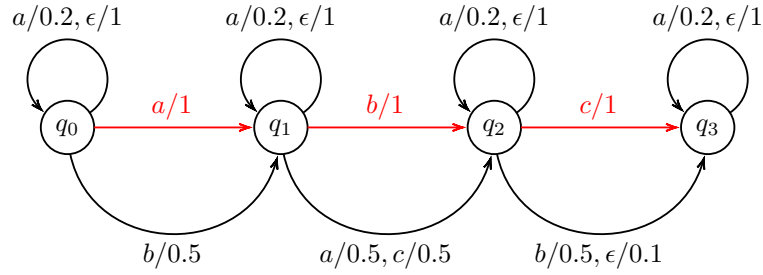
$$S_{\Sigma, R, \otimes}(\alpha) = R(a, a) \otimes R(b, b) \otimes R(a, \epsilon) \otimes R(c, c) \otimes R(a, \epsilon) = 1 \otimes 1 \otimes 0, 2 \otimes 1 \otimes 0, 2 = 0, 04$$

- řetězec $\alpha = cba$:

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

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

...



Obrázek 4: Přechodový diagram automatu z příkladu 3.5

{img-AutRozpCall}

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

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

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

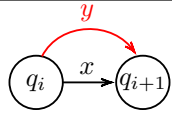
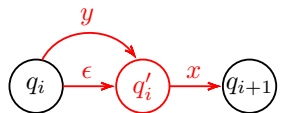
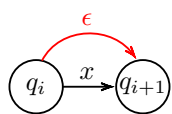
3.7 Deformovaný automat

Dalším ze způsobů, jak přijímat řetězec podobný vzorovému je s využitím deformovaného (fuzzy) automatu. Tato technika využívá tzv. deformovaného automatu, neboli automatu který byl stanoveným způsobem upraven, neboli deformován. Tato technika byla přejata z [3].

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

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

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

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

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

{tbl-DefAutDef}

3.8 Shrnutí

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

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

4 Fuzzy tree automaty

4.1 Zavedení

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

Fuzzy tree automaty vznikly fuzzyfikací „klasických“ tree automatů. O „klasických“ tree automatech je možné se dočíst více informací např. v [?], popř. [6] a [?]. Problematické fuzzy tree automatů se věnuje například [?], [11], [12], [?] a [?]. V této kapitole vycházeno z [?].

Fuzzy automaty pracují se speciálními strukturami symbolů, tzv. stromy. K těmto stromům je navrženo zakódování do řetězců, kterým se říká pseudotermy. Oba tyto pojmy, a jejich vzájemný vztah budou rozebrány v následující podka-

pitole. Dále bude nadefinován fuzzy jazyk stromů a automat, fuzzy tree automat, který fuzzy jazyk stromů rozpoznává. Na závěr bude předloženo několik konkrétních aplikací fuzzy automatů.

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

4.2 Stromy a termy

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

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

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

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

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

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

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

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

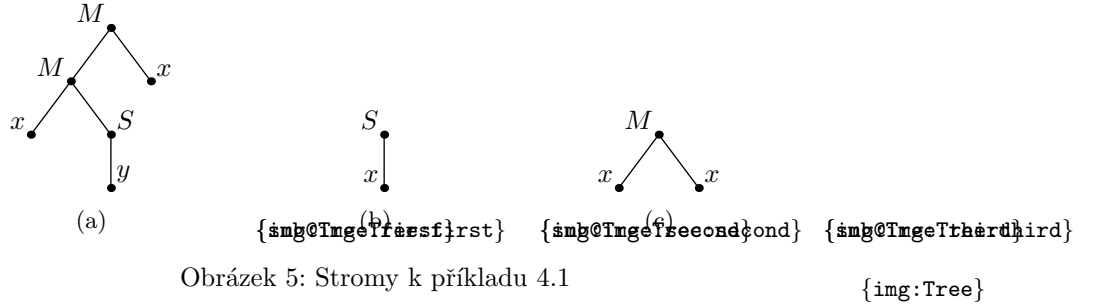
Místo $t(w)$ budeme psát jen t .

Strom t je tedy předpis pro „přejmenování“ uzlů prefixového stromu daného doménou U . To znamená, že strom dle definice 4.3 je v korespondenci s pojmem „strom“ (resp. „kořenový strom“) z teorie grafů. Z tohoto důvodu si pro jednoduchost můžeme odpustit definici souvisejících pojmů z teorie grafů pro strom z definice 4.3. Díky tomu můžeme stromy graficky zobrazovat, hovořit o jejich uzlech, potomcích, podstromech, vnitřních a listových uzlech bez nutnosti formálního nadefinování.

Příklad 4.1. *Označme $T = \{x, y\}$ a $N = \{S, M\}$. Definujme doménu U_1 stromu pro abecedu $\Sigma = \mathbb{N}$ jako množinu řetězců $U_1 = \{\epsilon, 1, 11, 12, 121, 2\}$.*

{def:Tree}

{ex:Trees}



Obrázek 5: Stromy k příkladu 4.1

Pak $\overline{U_1} = \{11, 121, 2\}$. Strom $t_1 : U_1 \rightarrow (T, N)$ nad (T, N) pak může vypadat například takto:

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

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

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

Definice 4.4 (Pseudoterm). Označme $D_{(N,T)}^p$ nejmenší podmnožinu¹ $(N \cup T \cup \{(\cdot, \cdot)\})^*$ splňující následující podmínky²:

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

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

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

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

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

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

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

¹pseudotermy mohou být zřejmě generovány bezkontextovou gramatikou

²Předpokládáme, že symboly závorek, (\cdot) nejsou součástí $N \cup T$

Důkaz. Opět dokážeme strukturálně:

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

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

□

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

Množinu stromů můžeme nazývat jazyk stromů. Fuzzy množinu stromů pak fuzzy jazyk stromů.

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

Stromy nám tedy reprezentují „stromovou“ strukturu tvořenou vnitřními a listovými uzly. Termy jsou jejich zakódováním do řetězců. Máme tak nadefinován fuzzy jazyk stromů. Nyní se podíváme na fuzzy tree automaty, mechanismy, které umožňují fuzzy jazyky stromů rozpoznávat.

4.3 Fuzzy tree automat a jazyk jím rozpoznávaný

Začneme definicí fuzzy tree automatu.

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

- Q je konečná množina symbolů stavů
- T je konečná množina terminálních symbolů uzlů
- N je konečná množina neterminálních symbolů uzlů taková, že $N \cap T = \emptyset$
- $\mu : (N \cup T) \rightarrow \{f | f : (\mathcal{Q} \cup \{\epsilon\}) \times Q \rightarrow [0, 1]\}$ je fuzzy přechodová funkce, kde \mathcal{Q} je konečná podmnožina Q^+ . Pro $X \in N$ je $\mu(X) = \mu_X$, kde μ_X je zobrazení z $\mathcal{Q} \times Q$ do $[0, 1]$. Pro $a \in T$ je $\mu(a) = \mu_a$, kde μ_a je zobrazení z $\{\epsilon\} \times Q$ do $[0, 1]$.
- $F \subseteq Q$ je množina koncových stavů.

Podívejme se nyní podrobněji na fuzzy přechodovou funkci μ . Pro terminální symbol $a \in T$ nám μ_a definuje fuzzy stav, do kterého automat přejde při vstupu a . Pro neterminál $X \in N$ nám definuje přechodovou funkci $\mu_X(q_1 \dots q_k, q') = c$ s významem „pokud je na vstupu X a automat se nachází ve stavech q_1, \dots, q_k , pak automat přejde do stavu q' ve stupni c “. Vzhledem k tomu, že automat se může nacházet ve více stavech současně, jeho výpočet bude zcela určitě neterministický.

μ_x	q_1	q_2
ϵ	1	0
μ_y	q_1	q_2
ϵ	1	0

μ_S	q_1	q_2
q_1	0	1
q_2	0	0,4
$q_1 q_1$	0	0,3

μ_M	q_1	q_2
q_1	0,1	0,8
q_2	0	0,5
$q_1 q_1$	0	0,6
$q_1 q_2$	0	1
$q_2 q_1$	0	0,7

Tabulka 2: Příklad přechodové funkce μ fuzzy tree automatu

{tab:MuOfFuzTreAut}

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

Nyní se podíváme na stromy a jazyk stromů, které fuzzy tree automat rozpoznává.

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

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

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

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

Fuzzy přechodová funkce stromů je obdobou fuzzy rozšířené přechodové funkce. Pokud je vstupní strom atomický je přechod realizován pomocí fuzzy přechodové funkce μ . Pokud vstupní strom není atomický, je přechod realizován ve stupni, který je dán stupněm přechodu neterminálního symbolu a stupňů příslušných podstromů.

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

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

Definice 4.8 (Jazyk rozpoznávaný). Fuzzy množinu $D(A)$ nad $D_{(N,T)}$ danou předpisem

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

nazvěme fuzzy jazyk rozpoznávaný fuzzy tree automatem.

Příklad 4.5. Uvažujme automat A z předchozího příkladu. Pak pro strom t_1 (s levým podstromem $t_{1,1}$ a pravým podstromem $t_{1,2}$ kořene) z obrázku 5c platí $\mu_{t_1}(q_2)$:

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

(zde bude doplněno: dalších pár stromů)

Podobně jako u klasických automatů, i u tree automatů platí, že vždy existuje automat, který rozpoznává daný (fuzzy) jazyk.

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

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

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

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

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

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

4.4 Použití fuzy tree automatů

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

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

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

Geometrické tvary

V [?] byl popsán způsob, jak pomocí fuzzy tree automatů rozpoznávat složené geometrické tvary. Techniku demonstrují na problému rozpoznávání jednoduchého nákresu domu a kostela.

Dům totiž popisují jako „obdélník, jehož horní strana splývá se základnou rovnoramenného trojúhelníku“. Vlastnosti „být obdélník“, „být rovnoramenný trojúhelník“ a „mít splývající hrany“ se dají poměrně snadno fuzzyfikovat. Například, určení stupně rovnoramennosti trojúhelníku lze spočítat jako poměr obou úhlů svíranými základnou a oběma rameny.

Autoři pak navrhnou strom reprezentující takovýto objekt domu a konstruují fuzzy tree automat, který tento strom rozpoznává. Automat správně rozpoznává objekt domu sestavený dle požadavků ve stupni 1. Strom, který reprezentuje objekt se „střechou“ ve tvaru pravoúhlého trojúhelníku je pak rozpoznáván ve stupni menším než 1 (v závislosti na sklonu střechy).

Obdobným způsobem autoři nadefinovali strom reprezentující jednoduchou budovu kostela jako dům, který má na špičku střechy připevněn kříž.

Detekce úplných m -árních stromů

Dalším způsobem, jak lze používat fuzzy tree automaty je rozpoznávání úplných m -árních stromů. Úplný m -ární strom je takový, jehož každý uzel má buďto právě m potomků (vnitřní uzly) a nebo 0 (listové uzly). My však můžeme můžeme vlastnost „být úplný m -ární“ snadno fuzzifikovat.

Uvažujme strom t , který je tvořen vnitřními uzly I (konkrétní uzly značme I_1, \dots, I_n) a listovými uzly o (konkrétní uzly značme o_1, \dots, o_k). Označme $N = \{I\}$ a $T = \{o\}$. Strom t je pak prvkem množiny $D_{(N,T)}$.

Zavedme fuzzy množinu $\alpha_m : I_1, \dots, I_n \rightarrow [0, 1]$ úplných m -árních uzlů předpisem:

$$\alpha_m(I_i) = \frac{|children(I_i)|}{m}$$

kde $children(I_i)$ značí počet potomků uzlu I_i . Stupeň, ve kterém je strom úplný m -ární pak určíme jako infimum z $\tau_m(I_i)$ všech jeho vnitřní uzlů I_i . Pak můžeme definovat fuzzy množinu τ_m úplných m -árních stromů pro každý $t \in D_{(N,T)}$:

$$\tau_m(t) = \bigwedge_{i=1}^n \alpha_m(I_i)$$

Nyní zkonstruujeme automat A rozpoznávající fuzzy množinu τ_m . Automat bude mít jeden stav q_1 a přechodovou funkci $\mu_o(q_1) = 1$ a

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

pro $w \in \{q_1, q_1 q_1, \dots, q_1^m\}$. Množina koncových stavů F bude rovna $\{q_1\}$.

Jednoduchá klasifikace zvířat

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

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

5 Buněčné fuzzy automaty

6 Konkrétní příklady

6.1 Rozpoznávání ručně psaného textu

...

6.2 Detekce překlepů

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

Reference

- [1] Radim Bělohlávek. *Fuzzy Relational Systems: Foundations and Principles*. Kluwer, New York, 2002.
- [2] H. A. Girijamma Dr. V. Ramaswamy. Conversion of finite automata to fuzzy automata for string comparison. *International Journal of Computer Applications*, 2012.
- [3] J. J. Astrain; J. R. Garitagoitia; J. R. Gonzalez De Mendivil; J. Villadangos; F. Farina. Approximate string matching using deformed fuzzy automata: A learning experience. *Springer Science & Business Media*, 2004.
- [4] Abdulwahed Almarimi Gabriela Andrejková and Asmaa Mahmoud. Approximate pattern matching using fuzzy logic. *CEUR Workshop Proceedings*, 2003.
- [5] S.S. Yau G.F. DePalma. Fractionally fuzzy grammars with application to pattern recognition. *US–Japan Seminar on Fuzzy Sets and their Applications*, 1974. článek jako e-book: <http://bit.ly/2cumxjz>, výcuc v MorMalFuzzAutAndLangs 10.7.
- [6] Kou-Yuan Huang. *Syntactic Pattern Recognition for Seismic Oil Exploration*. World Scientific, 2002.
- [7] J.R. Garitagoitia J. Astrain, J.R. González de Mendivil. Fuzzy automata with ϵ -moves compute fuzzy measures between strings. *Fuzzy Sets and Systems* 157, 2005.
- [8] José R. Garitagoitia Carlos F. Alastrueya Javier Echanobe, José R. Gonzáles Mendivil.
- [9] José R. Garitagoitia José R. González de Mendivil. Fuzzy languages with infinite range accepted by fuzzy automata: Pumping lemma and determination procedure. *Fuzzy sets and Systems*, 2014.
- [10] S. C. Kremer M. Doostfateme. New directions in fuzzy automata. *International Journal of Approximate Reasoning*, 2004.
- [11] M. M. ZAHEDI S. MOGHARI and R. AMERI. New direction in fuzzy tree automata. *Iranian Journal of Fuzzy Systems*, 2011.
- [12] Mukta N. Joshi S. R. Chaudhari. A note on fuzzy tree automata. *International Journal of Computer Applications*, 2012.
- [13] Miroslav Stamenkovic, Aleksandar; Ciric. Construction of fuzzy automata from fuzzy regular expressions. *Fuzzy Sets and Systems*, 2012.
- [14] Witold Pedrycz Yongming Li. Fuzzy finite automata and fuzzy regular expressions with membership values in lattice-ordered monoids. *Fuzzy Sets and Systems*, 2005.