

# Aplikace fuzzy a pravděpodobnostních automatů

Martin Jašek

12. září 2016 — ??

## Obsah

<b>1</b>	<b>Definice a značení</b>	<b>2</b>
<b>2</b>	<b>Rozpoznávání textových vzorů</b>	<b>3</b>
2.1	Formální zavedení problému . . . . .	4
2.2	Motivace k použití fuzzy automatů . . . . .	4
2.3	Automat rozpoznávající $\omega$ . . . . .	5
2.4	Podobnost symbolů . . . . .	6
2.5	Fuzzy symboly . . . . .	7
2.6	Editační operace . . . . .	8
2.7	Deformovaný fuzzy automat . . . . .	13

# 1 Definice a značení

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

## Abecedy, řetězce, jazyky

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

## Fuzzy teorie

Fuzzy množiny a relace budou po vzoru [5] nejčastěji malými řeckými písmeny. 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])

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

## Nedeterministický bivalentní automat

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

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

## Základní definice nedeterministického fuzzy automatu

Značení je převzato z [5] a lehce upraveno.

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

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

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

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

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

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  $V \in \mathcal{F}(Q), \alpha \in \Sigma^*, x \in \Sigma$

**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)

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

(zde bude doplněno: ověřit, dohledat, ozdrojovat)

## Nedeterministický fuzzy automat s $\epsilon$ přechody

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

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

## 2 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 výsledcích 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 2.1.** *Pokud nebude uvedeno jinak, pojem „rozpoznávání textových vzorů“ bude v této kapitole zkracován jen na „rozpoznávání vzorů“.*

## 2.1 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 2.2.** *Vzorový řetězec budeme v této kapitole vždy značit  $\omega$ , pozorovaný pak  $\alpha$ .*

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

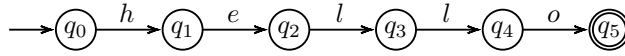
Například, budeme-li porovnávat vstup zadaný z klávesnice počítače oproti nějakému vzoru, je možné, že uživatel udělá překlep. V takovém případě bude vzorovému řetězci určitě více podobný řetězec obsahující dva překlepy (zámena symbolu za některý sousedící na klávesnici) než jiný, který se sice bude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

## 2.2 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



Obrázek 1: Automat rozpoznávající **hello**

texty, které jsou nepřesné a nedokonalé. Fuzzy přístup by nám tak mohl pomoci na tyto nepřesnosti adekvátně reagovat.

### 2.3 Automat rozpoznávající $\omega$

Klíčovým pro rozpoznávání vzorů (chceme-li využívat fuzzy automaty) je bivalentní automat rozpoznávající vzorový řetězec. Tedy automat takový, který přijímá jediný řetězec  $\omega$  a všechny ostatní zamítá. Nyní si takovýto automat zkonstruujeme.

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

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

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

**Definice 2.1** (Automat rozpoznávající  $\omega$  (deterministický)). *Mějme řetězec  $\omega$  délky  $n$  nad abecedou  $\Sigma$ . Automat rozpoznávající  $\omega$  je pak konečný automat  $A(\omega) = (Q, \Sigma, \delta, q_0, F)$  takový, že jeho množina stavů  $Q$  se sestává z právě  $n$  stavů  $q_0, \dots, q_n$ ,  $q_0$  je počáteční stav,  $F = \{q_n\}$  množina koncových stavů a  $\delta$  je přechodová funkce definována pro všechna  $0 \leq k < n$  následovně:*

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

Tato definice automatu je vcelku intuitivní. K stejnému výsledku bychom došli, pokud bychom automat zkonstruovali konverzí gramatiky nebo regulérního výrazu. Příklad automatu rozpoznávající řetězec **hello** naleznete na obrázku 1.

Nyní tento automat převedeme na ekvivalentní fuzzy automat. Tedy na automat rozpoznávající řetězec  $\omega$  ve stupni 1 a všechny ostatní ve stupni 0. Vzhledem k tomu, že budeme konstruovat fuzzy automat nedeterministický, bude vhodné si nejdříve přetransformovat deterministický automat z předchozí definice na nedeterministický.

**Definice 2.2** (Automat rozpoznávající  $\omega$  (nedeterministický)). *Mějme řetězec  $\omega$  nad abecedou  $\Sigma$  z předchozí definice. Nedeterministický automat rozpoznávající  $\omega$  je pak konečný automat  $A'(\omega) = (Q, \Sigma, \delta, I, F)$  takový, že jeho množina stavů  $Q$  je stejná jako v předchozí definici, dále  $I = \{q_0\}$  je množina počátečních a  $F = \{q_n\}$  množina koncových stavů a  $\delta$  je přechodová funkce definována pro*

všechna  $0 \leq k < n$  následovně:

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

**Definice 2.3** (Fuzzy automat rozpoznávající  $\omega$ ). Mějme řetězec  $\omega$  nad abecedou  $\Sigma$  délky  $n$  z předchozí definice. Automat rozpoznávající  $\omega$  je pak fuzzy automat  $A''(\omega) = (Q, \Sigma, \mu, \sigma, \epsilon)$  takový, že jeho množina stavů  $Q$  je stejná jako v předchozí definici, a dále

- $\sigma(q_0) = 1$  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 odstraňují a nahrazují podobností.

## 2.4 Podobnost symbolů

nejzákladnější technika pro zanesení podobnostního rozpoznávání je s pomocí 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 zjevně 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 nepíší, ale měla by to být relace ekvivalence (*Sym*, *Ref*, *Tra*). Existuje něco, jako fuzzy relace ekvivalence?)

Máme-li k dispozici relaci  $g_s$ , je nutné ji zakomponovat do automatu. Jak autoři uvádějí, automat pracující s relací  $g_s$  je možné zkonstruovat pro libovolný (konečný) automat. Vzhledem k tomu, že v této kapitole pracujeme s rozpoznáváním textového vzoru, budeme uvažovat konečný automat rozpoznávající  $\omega$  (dle definice 2.2). Nyní k němu sestavíme fuzzy automat pracující s  $g_s$ .

**Definice 2.4** (Automat rozpoznávající  $\omega$  pracující s  $g_s$ ). Uvažujme, že máme nedeterministický automat  $A$  rozpoznávající  $\omega$  a relaci podobnosti symbolů  $g_s$ . Pak k automatu  $A$  můžeme zkonstruovat automat  $A'$ , který navíc pracuje s  $g_s$ . Takový automat bude zkonstruován dle definice 2.3 s tím rozdílem, že přechodová funkce  $\mu'$  bude definována pro všechna  $y \in \Sigma$  následovně:

$$\mu'(q_i, x, q_j)(y) = (g_s(x, y) \wedge \delta_y(q_i, q_j))$$

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

(zde bude doplněno: mám tu definici  $\mu'$  dobře?)

(zde bude doplněno: provádí se fuzzyfikace KNA. Nezavést ji v teoretické části jako pojem a pak ho jen všude používat?)

Definice je vcelku přímočará. U každého přechodu, který je v původním automatu  $A$  definován, je navíc kontrolována podobnost symbolu  $y$  (aktuální symbol v pozorovaném řetězci) se symbolem  $x$  (aktuální očekávaný symbol dle vzorového řetězce). ...

## 2.5 Fuzzy symboly

Nejzákladnějším způsobem, jak zanést neurčitost do rozpoznávání vzorů, je použití fuzzy symbolů. Idea pro použití fuzzy symbolů byla přejata z [3] a vychází z [2].

Fuzzy symbol je nástroj, který akceptuje nepřesnost na úrovni jednotlivých symbolů. Fuzzy symbol reprezentuje relaci podobnosti symbolů.

Mějme abecedu  $\Sigma$  a pro každé dva symboly z této abecedy číslo z intervalu  $[0, 1]$  udávající jejich podobnost. Například, symbol reprezentující malé psací  $i$  bude mít vyšší podobnost s malým psacím  $e$  než s malým psacím  $m$ . Tuto podobnost můžeme realizovat jako fuzzy relaci nad  $\Sigma \times \Sigma$ . Tento způsob je použit v [2]. Vzhledem k tomu, že očekáváme použití automatů, bude však vhodnější držet se symbolů a abeced. Zavádí se proto speciální třída symbolů, tzv. fuzzy symboly, a s nimi související pojmy.

**Definice 2.5** (Fuzzy symbol [4]). *Mějme abecedu  $\Sigma$  a nějaký symbol  $y$  z této abecedy. Fuzzy symbol  $\tilde{y}$  symbolu  $y$  je fuzzy množina nad abecedou  $\Sigma$  popisující podobnost symbolů. Tedy, pro každé  $x, y \in \Sigma$  je  $\tilde{y}(x)$  rovna stupni podobnosti  $x$  a  $y$ . Mělo by platit, že  $\tilde{y}(y) = 1$ .*

Máme-li definován fuzzy symbol pro všechny symboly  $y$  z abecedy  $\Sigma$ , můžeme tak nadefinovat abecedu fuzzy symbolů.

**Definice 2.6** (Abeceda fuzzy symbolů). *Mějme abecedu  $\Sigma$  a fuzzy symboly  $\tilde{y}$  pro všechna  $y \in \Sigma$ . Pak množinu všech těchto fuzzy symbolů nazvěme abeceda fuzzy symbolů a označme  $\tilde{\Sigma}$ . Tedy  $\tilde{\Sigma} = \{\tilde{y} \mid y \in \Sigma\}$ .*

Nyní máme k dispozici abecedu pracující s fuzzy symboly. S touto abecedou můžeme pracovat jak jsme zvyklí, tedy používat operace definované nad abecedami. Máme tak také možnost sestavit řetězec fuzzy symbolů nad touto abecedou. Pro naše účely bude výhodné, abychom dokázali převést řetězec nad abecedou  $\Sigma$  na odpovídající řetězec nad abecedou  $\tilde{\Sigma}$ .

**Definice 2.7** (Řetězec fuzzy symbolů). *Mějme abecedu  $\Sigma$  a nějaký řetězec  $\alpha = a_1 \dots a_n$  nad touto abecedou (tj.  $\alpha \in \Sigma^*$ ). Pak definujeme  $\tilde{\alpha} = \tilde{a}_1 \dots \tilde{a}_n$  jako řetězec fuzzy symbolů řetězce  $\alpha$ .*

Nyní máme nadefinováno vše potřebné a můžeme tedy zkonstruovat automat, který rozpoznává vzorový řetězec, a to s ohledem na podobnost symbolů. Idea je jednoduchá – vezmeme fuzzy automat rozpoznávající  $\omega$ , jeho abecedu nahradíme abecedou fuzzy symbolů a patřičně předefinujeme některé další pojmy.

**Definice 2.8** (Automat pracující s fuzzy symboly). *Mějme abecedu  $\Sigma$ . Pak na základě podobností symbolů v  $\Sigma$  sestavíme abecedu  $\tilde{\Sigma}$  fuzzy symbolů. Pak automat pracující s fuzzy symboly je fuzzy automat definovaný v definici 1.3, ve které je  $\Sigma$  nahrazena  $\tilde{\Sigma}$ .*

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

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

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

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

Nyní můžeme zkonstruovat automat rozpoznávající  $\omega$  pracující s fuzzy symboly.

**Definice 2.9** (Automat rozpoznávající  $\omega$  pracující s fuzzy symboly). *Mějme abecedu  $\Sigma$  a vzorový řetězec  $\omega$ . Pak na základě podobnosti symbolů v  $\Sigma$  sestavíme abecedu  $\tilde{\Sigma}$  fuzzy symbolů. Pak automat pracující s fuzzy symboly je fuzzy automat rozpoznávající  $\omega$  definovaný v definici 2.3, ve které je  $\Sigma$  nahrazena  $\tilde{\Sigma}$ .*

Nyní máme zkonstruován požadovaný automat. Podíváme-li se na jeho rozpoznávací charakteristiku, je zjevné, jaké řetězce tento automat rozpoznává. Tento automat rozpoznává vzorový řetězec ve stupni 1 (za předpokladu, že byla dodržena vlastnost  $\tilde{y}(y) = 1$  pro všechna  $y \in \alpha$ ). Při zpracování jakéhokoli jiného řetězce automat nezkončí v koncovém stavu. Nicméně, díky pozměněnému výpočtu přechodové funkce fuzzy stavů je i v takové situaci schopen řetězec přijmout v nenulovém stupni. K tomu dojde právě v případě, kdy se vzorový a pozorovaný řetězec na odpovídajících pozicích liší, ale tyto symboly jsou si podobné. Připomeňme, že tato podobnost je dána hodnotou stupně fuzzy symbolu.

Automat tak dobře zvládá rozpoznat řetězce, u kterých došlo k záměně podobným symbolem. Na druhou stranu, tato technika se nehodí na situace, kdy by mohl být do řetězce vložen, nebo naopak odebrán symbol (nebo více symbolů).

## 2.6 Editační operace

Další technikou pro podobnostní porovnávání pozorovaného a vzorového řetězce je s využitím editačních operací. Tato technika byla přejata z [5]. 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 2.10** (Editační operace). *Mějme abecedu  $\Sigma$ , uvažujme množinu  $E = (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\}) \setminus \{(\epsilon, \epsilon)\}$ . Pak každou dvojici  $(x, y) = z \in E$  nazvěme editační operace. Speciálně pak, pro všechna  $x, y \in \Sigma$ ,  $(x, y) \in E$  znamená nahrazení symbolu  $x$  symbolem  $y$ ,  $(x, \epsilon) \in E$  znamená odebrání symbolu  $x$  a naopak  $(\epsilon, y) \in E$  pak vložení symbolu  $y$ . Navíc jako editační operaci uvažujme i všechny dvojice  $(x, x) \in E$  (pro každé  $x \in \Sigma$ ) symbolizující „žádnou editaci“.*

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



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

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

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

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

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

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

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

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

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

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

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

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

**Definice 2.11** (Vyrovnání řetězců [5]). *Jako množinu všech vyrovnání  $G(\alpha, \omega)$  řetězce  $\alpha$  na řetězec  $\omega$  (kde  $(\epsilon, \epsilon) \neq \alpha, \omega \in \Sigma^*$ ) označme takovou množinu  $\{\zeta \in E^+ \mid \zeta \text{ splňuje vlastnosti 1., 2. i 3.}\}$*

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

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

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

(zde bude doplněno: Musí být  $R$  reflexivní a symetrická (def. automatu to vyžaduje, ale je to nutné?). A co  $T$ -tranzitivita?)

**Definice 2.12** (Fuzzy míra[5]). *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 dedefinována jako 1.

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

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

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

**Definice 2.13** (Automat rozpoznávající  $\mathcal{L}$  [5]). *Mějme binární fuzzy relaci  $R$  nad  $\Sigma \cup \{\epsilon\}$  (stejná jako v definici 2.12). Pak pro vzorový řetězec  $a_1 a_2 \dots a_n = \omega \in \Sigma^*$  označme  $M_{\Sigma, R, \otimes}(\omega)$  automat rozpoznávající jazyk  $\mathcal{L}(\omega)$  dle definice 1.10, takový, že*

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

4. množina koncových stavů  $\eta$ :  $\eta(q_n) = 1$  a pro všechny ostatní  $q_n \neq q' \in Q$ :  $\eta(q') = 0$

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

**Věta 2.1.** *Mějme binární fuzzy relaci  $R$  nad  $\Sigma \cup \{\epsilon\}$  (stejná jako v definici 2.12) 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 [5]. □

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

**Příklad 2.1.** *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 2.13 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 2. V přechodovém diagramu jsou červeně zvýrazněny pravidla pro rozpoznávání  $\omega$ , ostatní pravidla (doplněna dle definice) jsou černá.

Nyní si na pár řetězcích zkoušíme ukázat platnost věty 2.1. Dle definice 1.8 spočítáme stupeň, v jakém automat náš testovací rozpoznává řetězec  $\alpha$ :

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

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

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

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

- řetězec  $\alpha = ab$ :

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

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

- řetězec  $\alpha = bbb$ :

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

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

- řetězec  $\alpha = abaca$ :

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

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

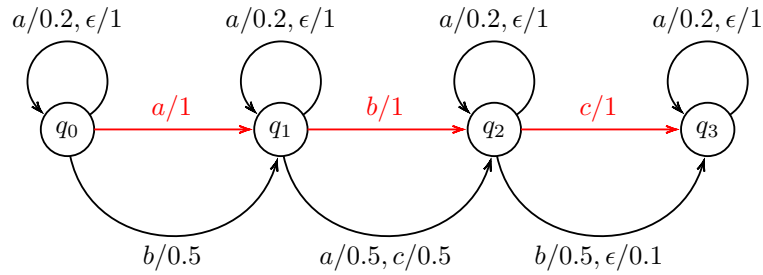
- řetězec  $\alpha = cba$ :

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

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

...

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



Obrázek 2: Přechodový diagram automatu z příkladu 2.1

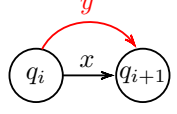
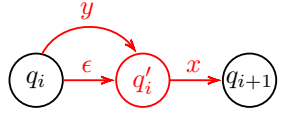
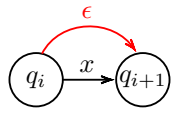
Je vidět, že automat zkonstruován dle editačních operací je značně silný nástroj. Umožňuje nám velmi pohodlně popsat, jak moc mohou být konkrétní editační operace akceptovány. Editací operace vložení symbolu, náhrada symbolu a odebrání symbolu jsou pro popis modifikace vzorového řetězce přirozené.

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.

## 2.7 Deformovaný fuzzy automat

Další ze způsobů, jak přijímat řetězec podobný vzorovému je s využitím deformovaného (fuzzy) automatu. Deformovaný automat je automat, který umožňuje přijímání i jiných řetězců, než vzorových. A to takových, které byly předem stanoveným způsobem upraveny, neboli zdeformovány.

Uvažujme tedy nyní, že máme vzorový řetězec  $\omega$ . Sestrojme k němu automat  $A(\omega)$  rozpoznávající  $\omega$  (dle definice 2.2). Nyní bude následovat postup, jak k němu zkonstruovat deformovaný automat  $A'(\omega)$ . Pro zjednodušení nyní předpokládejme, že automat bude obsahovat akceptovat pouze jednu z výše jmenovaných deformací. Má-li k deformaci dojít na  $i$ -té pozici vzorového řetězce, pak je třeba zkonstruovat  $Q'$  a  $\delta'$  automatu  $A'(\omega)$  pomocí pravidel uvedených v tabulce 1.

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

Tabulka 1: Konstrukce deformovaného automatu

## 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] J R G; Echanobe J; Astrain J J; Farina F. Garitagoitia, J R; de Mendivil. Deformed fuzzy automata for correcting imperfect strings of fuzzy symbols. *IEEE Transactions on Fuzzy Systems*, 2003. K dispozici na IEEEExplore.
- [5] J.R. Garitagoitia J. Astrain, J.R. González de Mendivil. Fuzzy automata with  $\epsilon$ -moves compute fuzzy measures between strings. *Fuzzy Sets and Systems* 157, 2005.