Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Fuzzy regulátor

2017

Michal Sladkovský

Vedoucí práce: doc. RNDr. Miroslav Kolařík, Ph.D.

Studijní obor: Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Michal Sladkovský
Název práce:	Fuzzy regulátor
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2017
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	doc. RNDr. Miroslav Kolařík, Ph.D.
Počet stran:	50
Přílohy:	1 DVD
Jazyk práce:	český

Bibliograhic info

Author:	Michal Sladkovský
Title:	Fuzzy controller
Thesis title:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2017
Study field:	Applied Computer Science, combined form
Supervisor:	doc. RNDr. Miroslav Kolařík, Ph.D.
Page count:	50
Supplements:	1 DVD
Thesis language:	Czech

Anotace

Tato práce se zabývá problematikou fuzzy regulace. Praktickou část tvoří aplikace pro karetní hru Texas Hold'em poker pro 2 - 9 hráčů (max. 1 lidský hráč), ve které je počítačový hráč implementován jako fuzzy regulátor. Hra je vhodná pro začínající hráče pokeru. Průběh hry lze sledovat i jako demo (pouze s počítačovými hráči). Aplikace je vyvinuta v jazyce Java s použitím platformy JavaFX pro tvorbu grafického uživatelského rozhraní (GUI). Fuzzy systém je vytvořen a testován v programu Fuzzy Logic Toolbox v programovém prostředí MATLAB.

Synopsis

This thesis deals with the issue of fuzzy regulation. The practical part consists of a Texas Hold'em poker card game application for 2-9 players (max. 1 human player) in which the computer player is implemented as a fuzzy controller. The game is suitable for novice poker players. You can watch the game as a demo (only with computer players). The application is developed in Java using JavaFX to create a graphical user interface (GUI). The Fuzzy system is created and tested in the Fuzzy Logic Toolbox in the MATLAB program environment.

Klíčová slova: fuzzy logika; fuzzy regulátor; fuzzy regulace; fuzzy systém

Keywords: fuzzy logic; fuzzy controller; fuzzy control; fuzzy system

Tímto bych rád poděkoval svému vedoucímu práce doc. RNDr. Miroslavu Kolaříkovi, Ph.D. za konzultace a inspiraci nezbytně nutnou k vývoji aplikace.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Už na začátku svého studia jsem měl představu, že jako svou bakalářskou práci naprogramuji nějakou počítačovou hru, nejlépe karetní. Zadáním práce bylo pojednat o problematice fuzzy regulátorů a implementace vlastního fuzzy regulátoru do vlastní aplikace. Vybral jsem si toto téma, které mně připadalo mimo jiné velmi zajímavé díky svému uplatnění v mnoha odvětvích a vytvořil jsem aplikaci pro karetní hru *Texas Hold'em poker*, kde je počítačový hráč implementován jako fuzzy regulátor.

Hlavní motivací pro mě byla zvědavost, zda-li je vůbec možné fuzzy regulátor v této karetní hře smysluplně použít. Byl jsem nadšen představou spojit příjemné s užitečným, tedy zabývat se mou oblíbenou hrou a ještě se dozvědět něco, o čem jsem toho moc nevěděl.

V první fázi jsem se snažil zjistit, jestli tuhle techniku už někdo na programování pokeru použil. Našel jsem nějaké pokusy a pro mě nejasné dokumentace na toto téma na zahraničních webech, které mi příliš nepomohly. Po důkladném nastudování problematiky fuzzy regulátorů jsem tedy začal s vývojem vlastní aplikace bez jakéhokoliv výhledu úspěšného konce. K vývoji jsem použil jazyk Java, s kterým mám dobré zkušenosti.

Jako výsledek mého snažení jsem měl představu hratelné hry pro minimálně začínající pokerové hráče, kteří si rádi zahrají s 1 - 8 počítačovými hráči a rozhodně se alespoň pár hodin nudit nebudou.

2 Základy teorie fuzzy modelování

2.1 Úvod do fuzzy myšlení

Ze všeho nejdříve uvedu, co to vlastně znamená pojem *fuzzy* (čti fazi). Výraz pochází z angličtiny a ve slovnících můžeme narazit na pojmy jako: neurčitý, mlhavý, nejasný, neostrý, či vágní. V češtině nepoužíváme spojení *mlhavé množiny, neurčitá logika*, ale zůstáváme u anglického *fuzzy množiny, fuzzy logika* apod.

Nováčka v této oblasti by mohlo zarazit, jak je vůbec možné v matematice, či v logice hovořit o nějaké nepřesnosti, nejasnosti. Vždyť je to v přímém rozporu s učením, které trvá již několik staletí. Můžeme říci, že teorie fuzzy množin a fuzzy logiky, což je základem pro fuzzy regulaci, jsou přesné matematické disciplíny, které nabízí možnost modelovat velmi složité procesy, při nichž je nepřesnost tolerována. [1] Fuzzy logika se stala populární od konce osmdesátých let především díky Japonsku. Nejrozšířenější jsou aplikace v řízení a regulaci. Dnes je principů fuzzy regulace využito např. v automobilovém průmyslu (ABS, řízení motoru, řízení volnoběhu, klimatizace), v ekonomice (investice na kapitálovém trhu), při řízení výtahu, řízení metra, ve fotoaparátech (při zaostřování), kamerách atd. [1, 3]

Jedna z největších předností fuzzy modelování je možnost pracovat s výrazy používanými v běžném přirozeném jazyce. V určitých případech, kde běžně funguje teorie pravděpodobnosti, nebo je použito jiné matematické disciplíny (např. statistiky), je možné použít vhodnější řešení. Existují úlohy, kde přesnost nehraje podstatnou roli. Např. člověk při řízení automobilu uvažuje asi takto „silnice se začíná stáčet trochu doleva, je potřeba trochu ubrat plyn a volant mírně otočit doleva“. Fuzzy logika (někdy nazývána *vícehodnotová logika*) resp. fuzzy logická dedukce se podobá lidskému uvažování. Na základě získaných poznatků provedeme nějakou akci. Člověk nepotřebuje při řízení auta znát přesně na milimetr vzdálenost zatáčky, kroutící moment, úhel zatočení nebo jiné přesné informace. Ve fuzzy modelování si vystačíme s něčím podobným jako např. „je-li zatáčka velmi prudká, o trochu více uber plyn a volant otoč hodně“. [1]

První článek o fuzzy množinách vydal Max Black roku 1937. Za průkopníka fuzzy množin je považován Lotfi Asker Zadeh, který roku 1965 úspěšně publikoval svůj článek „*Fuzzy sets.*“ a právě on zformuloval tzv. *princip inkompatibility (neslučitelnosti)*: „*S rostoucí složitostí sytému klesá naše schopnost formulovat přesné a významné vlastnosti o jeho chování, až je dosáhnutá hranice, za kterou jsou přesnost a relevantnost prakticky vzájemně se vylučující jevy.*“ [3] Prakticky použil fuzzy logiku až E. H. Mamdani, který v sedmdesátých letech 20. století vyvinul řídící systém parního generátoru.

2.2 Fuzzy množiny a modelování sémantiky přirozeného jazyka

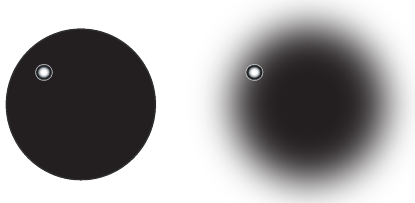
Zopakujme si nejdříve obecně pojem množina. Množina je soubor prvků, které chápeme jako celek. Výčet prvků zapisujeme např. takto: $M = \{1, 3, 5, 9\}$. Je jed-

noznačně určena svými prvky bez ohledu na pořadí. Existuje i prázdná množina neobsahující žádné prvky. Množina je dále charakterizována pravidlem, kterému její prvky musí vyhovovat a charakteristickou funkcí $M(x)$, pro kterou platí: [1]

$$M(x) = \begin{cases} 1, & x \in M \\ 0, & x \notin M \end{cases}$$

V tomto smyslu, pokud prvek do množiny buď patří nebo nepatří, chápeme množiny jako ostré.

Dostáváme se k případu, kdy charakteristická funkce udává stupeň příslušnosti do množiny v celém intervalu $\langle 0, 1\rangle$. V tomto případě nemůžeme pouze říci, že prvek do množiny patří, či nepatří. Takové množiny nazýváme neostré - fuzzy množiny, viz obr. 2.1. Každému prvku fuzzy množiny tedy náleží tzv. míra (stupeň) příslušnosti do této množiny. Je potřeba určit, jak moc prvek do množiny patří. V teorii fuzzy množin vycházíme z množiny všech možných prvků tzv. *univerzum*. Například budeme mít množinu vzdáleností od 0 km do 50 km a ptáme se u každé z těchto vzdáleností např.: „Je tato vzdálenost velká?“ V klasické teorii množin odpovíme buď ano nebo ne. Problémem je stanovit hranici, která vzdálenost je velká a která už není: např. 30 km je velká, 29 km není velká. Ve fuzzy množině je přiřazen každé této vzdálenosti stupeň příslušnosti z intervalu $\langle 0, 1\rangle$ a následná interpretace se už jeví jako přijatelnější: 30 km je velká 0.6, 29 km je velká 0.55 (čísla 0.6 resp. 0.55 vnímáme jako stupně příslušnosti vzdáleností 30 km resp. 29 km do množiny velkých vzdáleností).



Obrázek 2.1: Příklad příslušnosti prvku do obecné množiny a do fuzzy množiny.

2.2.1 Stručný popis a definice fuzzy množin

Při definici fuzzy množin budeme vycházet z následujícího popisu: [1, 2]

$$\begin{aligned} U &\quad ... \quad \text{univerzální množina (univerzum),} \\ A &\quad ... \quad \text{fuzzy množina v univerzu } U, \\ A : U \to \langle 0, 1\rangle &\quad ... \quad \text{funkce určující jednoznačně nějakou fuzzy množinu.} \end{aligned}$$

Funkce $U \to \langle 0, 1\rangle$ se nazývá funkce příslušnosti. V principu můžeme ztotožňovat funkci příslušnosti (charakteristickou funkci) s fuzzy množinou. Už víme, že pro každý prvek $x \in U$ hodnota $A(x) \in \langle 0, 1\rangle$ říká, do jaké míry je x prvkem fuzzy množiny A. Fuzzy množiny lze zapisovat např. tímto způsobem: [1]

$$A = \{a_1/x_1, ..., a_n/x_n\},$$

kde $x_1, ..., x_n \in U$ jsou prvky, kterým jsou přiřazeny stupně příslušnosti $a_1, ..., a_n \in (0, 1\rangle$. Prvky se stupněm příslušnosti 0 sem nepatří. Fuzzy množina neobsahující žádné prvky se nazývá *prázdná fuzzy množina*: $\emptyset = \{0/x \mid x \in U\}$.

V teorii fuzzy množin existují 3 speciální případy klasických množin. Jsou to: [1]

(1) Nosič

$$Supp(A) = \{x \mid A(x) > 0\},$$

je nosič fuzzy množiny A, tj. množina všech prvků z univerza, jejichž stupeň příslušnosti do A je nenulový.

(2) Jádro

$$Ker(A) = \{x \mid A(x) = 1\},$$

je množina prvků, které mají stupeň příslušnosti roven 1, tj. určitě patří do fuzzy množiny A. Jsou to typické prvky (prototypy) této fuzzy množiny.

(3) a-řez (používá se i označení α-řez)

$$A_a = \{x \mid A(x) \geq a\},$$

je množina prvků, které mají stupeň příslušnosti větší nebo roven stupni a. Množina vznikne odřezáním všech prvků z fuzzy množiny A, které mají stupeň příslušnosti menší než a.

2.2.2 Operace s fuzzy množinami

Podobně jako u klasických množin definujeme **inkluzi fuzzy množin** $A \subseteq B$. Inkluzí dvou fuzzy množin na univerzu U rozumíme schopnost být podmnožinou (A je podmnožinou B) tak, že platí: $\forall x \in U : A(x) \leq B(x)$.

V teorii fuzzy množin lze definovat mnoho operací a možnosti jsou narozdíl od klasických množin značně rozšířené. Za základní operace považujeme sjednocení, průnik a doplněk. [1, 2]

Sjednocení

$$C = A \cup B \qquad \text{právě když} \qquad C(x) = max\{A(x), B(x)\}$$

Do sjednocení fuzzy množin $A \cup B$ patří takový prvek $x \in U$, který má z obou stupňů příslušnosti $A(x)$ i $B(x)$ ten větší. Sjednocení je definováno pomocí operace maxima (suprema) a odpovídá logické disjunkci $A(x) \vee B(x)$.

Průnik

$$C = A \cap B \qquad \text{právě když} \qquad C(x) = min\{A(x), B(x)\}$$

Do průniku fuzzy množin $A \cap B$ patří takový prvek $x \in U$, který má z obou stupňů příslušnosti $A(x)$ i $B(x)$ ten menší. Průnik je definován pomocí operace

minima (infima) a odpovídá logické konjunkci $A(x) \wedge B(x)$.

Doplněk (komplement)

$$\overline{A}(x) = 1 - A(x)$$

Doplněk fuzzy množiny $\overline{A}$ je definován podle uvedeného vztahu a platí pro všechna $x \in U$.

T-normy a rozšíření fuzzy množinových operací

V pokročilejší teorii fuzzy množin lze definovat další operace a ty lze realizovat pomocí tzv. *t-norem (trojúhelníková norma, angl. triangular norm).* [1, 4]

T-norma je binární funkce $T : \langle 0,1\rangle \times \langle 0,1\rangle \to \langle 0,1\rangle$, která pro všechna $x, y, z \in \langle 0,1\rangle$ splňuje následující axiomy:

$$\begin{array}{ll}
T(x,y) = T(y,x), & \textit{komutativnost} \\
T(x,T(y,z)) = T(T(x,y),z), & \textit{asociativnost} \\
\textit{Jestliže } x \le y, \textit{ pak } T(x,z) \le T(y,z), & \textit{monotónnost} \\
T(0,x) = 0 \textit{ a } T(1,x) = x. & \textit{omezenost}
\end{array}$$

Základní t-normy jsou:

$$\begin{array}{rcll}
M(x,y) & = & x \wedge y = min(x,y), & \textit{operace minima (konjunkce)} \\
P(x,y) & = & x \cdot y, & \textit{součinová (produktová) konjunkce} \\
T_\infty(x,y) & = & 0 \vee (x + y - 1), & \textit{Lukasiewiczova konjunkce} \\
W(x,y) & = & \begin{cases} x \wedge y, & \textit{jestliže } x \vee y = 1, \\ 0 & \textit{jinak.} \end{cases} & \textit{drastický součin}
\end{array}$$

Pro libovolné t-normy T platí vztah uspořádání:

$$W(x,y) \le T(x,y) \le M(x,y)$$

pro všechna $x, y \in \langle 0,1\rangle$. Z toho vyplývá, že drastický součin je nejmenší a operace minima největší mezi všemi t-normami.

Jestliže je pomocí t-norem definována operace minima (průniku), pak je možné definovat i operaci maxima (sjednocení) pomocí tzv. *t-konorem.* Máme-li definovanou t-normu T, pak odpovídající t-konormu $S : \langle 0,1\rangle \times \langle 0,1\rangle \to \langle 0,1\rangle$ lze vyjádřit vztahem:

$$S(x,y) = 1 - T(1-x, 1-y).$$

Základní t-konormy odpovídající základním t-normám jsou: [1, 4]

$$\begin{array}{rcll}
N(x,y) & = & x \vee y = max(x,y), & \textit{operace maxima (disjunkce)} \\
Q(x,y) & = & x + y - x \cdot y, & \textit{součinová (produktová) disjunkce} \\
S_\infty(x,y) & = & 1 \wedge (x + y), & \textit{Lukasiewiczova disjunkce} \\
V(x,y) & = & \begin{cases} x \vee y, & \textit{jestliže } x \wedge y = 0, \\ 1 & \textit{jinak.} \end{cases} & \textit{drastický součet}
\end{array}$$

Pro libovolné t-konormy S platí vztah uspořádání:

$$N(x,y) \leq S(x,y) \leq V(x,y)$$

pro všechna $x, y \in \langle 0,1\rangle$. Z toho vyplývá, že drastický součet je největší a operace maxima nejmenší mezi všemi t-konormami.

Operace s fuzzy množinami na více univerzech

Kartézský součin

$$(A \times B)(\langle x,y\rangle) = min\{A(x), B(y)\}$$

Předešlým vztahem je pomocí operace minima definován kartézský součin fuzzy množin $A \times B \subseteq U \times V$, kde platí $x \in U$ a $y \in V$, přičemž $A \subseteq U,\ B \subseteq V$ a $\langle x,y\rangle$ je uspořádaná dvojice prvků. Výsledek kartézského součinu je pochopitelně opět fuzzy množina. [1]

Fuzzy relace

$$R : U_1 \times U_2 \times ... \times U_n \to \langle 0,1\rangle$$

Obecně n-ární fuzzy relace je fuzzy množina v kartézském součinu univerz $U_1, ..., U_n$. Stupeň příslušnosti $R(\langle x_1, ..., x_n\rangle)$ vyjadřuje stupeň příslušnosti prvků z jednotlivých univerz $U_1, ..., U_n$, v němž prvky patří do relace R. [1]

2.2.3 Fuzzy čísla

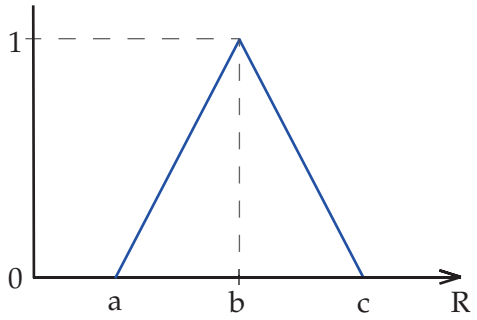
Fuzzy čísla jsou speciální případ konvexních fuzzy množin v množině reálných čísel $\mathbb{R} = (-\infty, +\infty)$ se spojitou funkcí příslušnosti a jednoprvkovým jádrem. Fuzzy množina A na univerzu U je konvexní, právě když pro libovolné prvky $x, y \in U$ a pro libovolné $0 \leq \lambda \leq 1$ platí: [4]

$$A(\lambda x + (1-\lambda)y) \geq min\{A(x), A(y)\}.$$

Fuzzy číslo se skládá z hodnoty a rozptylu. Pokud je rozptyl nulový, fuzzy číslo se stává jednoprvkovou množinou reálných čísel, tj. původním reálným číslem. V praxi se nejvíce používá speciální typ fuzzy čísel tzv. *trojúhelníková fuzzy čísla* mající funkci příslušnosti ve tvaru trojúhelníka. Nechť Z je fuzzy číslo (fuzzy množina) na univerzu reálných čísel $\mathbb{R}$. Pak platí následující předpis: [1]

$$Z(x,a,b,c) = \begin{cases} 0, & x < a \text{ nebo } x > c \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 1, & x = b, \end{cases}$$

kde a, b, c jsou parametry znázorněné na obr. 2.2, jednoprvkové jádro v tomto případě znázorňuje parametr $b \in \mathbb{R}$, tj. $Ker(Z) = \{b\}$. Všechny *a-řezy* Z_a fuzzy čísla Z, kde $a \in \langle 0,1\rangle$ jsou souvislé intervaly v $\mathbb{R}$.



Obrázek 2.2: Funkce příslušnosti trojúhelníkového fuzzy čísla.

V teorii fuzzy množin existuje **základní aritmetika** s fuzzy čísly (sčítání, odčítání, násobení, dělení). Máme-li ostré intervaly $\langle a, b\rangle$ a $\langle c, d\rangle$, pak platí: [1, 4]

$$\begin{aligned}
\langle a, b\rangle + \langle c, d\rangle &= \langle a + c, b + d\rangle, \\
\langle a, b\rangle - \langle c, d\rangle &= \langle a - d, b - c\rangle, \\
\langle a, b\rangle \cdot \langle c, d\rangle &= \langle min(ac, ad, bc, bd), max(ac, ad, bc, bd)\rangle, \\
\langle a, b\rangle / \langle c, d\rangle &= \langle min(a/c, a/d, b/c, b/d), max(a/c, a/d, b/c, b/d)\rangle.
\end{aligned}$$

Poslední operace dělení platí pouze tehdy, jestliže $0 \notin \langle c, d\rangle$. V opačném případě výsledkem není interval.

Obecně se dá říci, že aritmetika fuzzy čísel vychází z tzv. **principu rozšíření**. Pokud označíme $\bullet$ aritmetickou operaci (sčítání, odčítání, násobení) a Z_1, Z_2 jsou fuzzy čísla na univerzu reálných čísel $\mathbb{R}$, pak princip rozšíření umožňuje rozšířit tuto operaci na operaci $\odot$ s fuzzy čísly takto:

$$(Z_1 \odot Z_2)(z) = \bigvee_{z=x\bullet y} (Z_1(x) \wedge Z_2(y)).$$

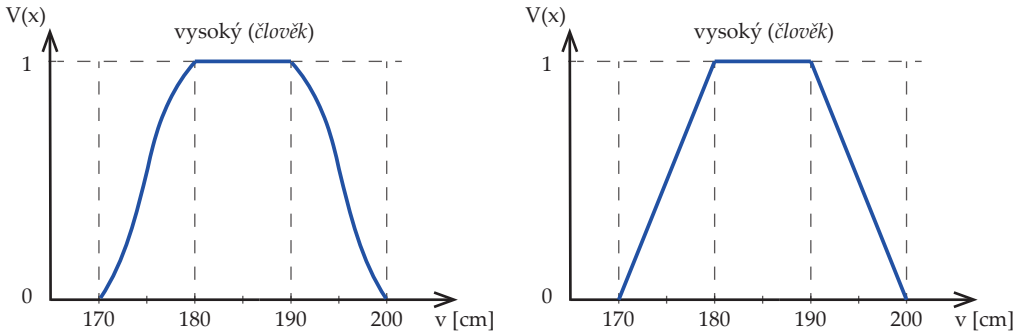
Výsledkem nově vzniklé operace $\odot$ je fuzzy číslo $Z_1 \odot Z_2$, které obsahuje prvky $z = x \bullet y$ se stupněm příslušnosti, který je roven minimu stupňů příslušnosti prvků x v Z_1 a y v Z_2. [1]

2.2.4 Formalizace jazykové vágnosti ve fuzzy modelování

Největší předností, či přímo zbraní fuzzy modelování je na rozdíl od klasické matematiky schopnost velmi dobře pracovat s jazykovou vágností. Nepřesné pojmy jako hodně, málo, velký, nízký jsou velmi dobře pochopitelné pro běžného člověka z hlediska významu.

Budeme-li se snažit pomocí fuzzy množiny vyjádřit např. vágní pojem „vysoký člověk“, každé výšce z nějaké stanovené množiny výšek lidí přiřadíme stupeň příslušnosti, který bude vyjadřovat míru našeho přesvědčení o velké výšce nějakého člověka. K tomu, abychom byli schopni vytvořit takovou množinu, je zapotřebí určitá znalost výšek lidí. Přiřazení míry příslušnosti nějaké konkrétní hodnoty do určité fuzzy množiny závisí tedy na subjektivním pozorování a na kontextu.

Na obr. 2.3 vlevo je zobrazena fuzzy množina formalizující pojem „vysoký člověk“ jako spojitá křivka zvonového tvaru. V pravo na obrázku je pak tato křivka zjednodušena a aproximována lomenými přímkovými úseky. Svislá osa představuje odpovídající stupně příslušnosti jednotlivým výškám z univerza výšek na vodorovné ose. Z grafického znázornění této fuzzy množiny lze vyčíst, že sem zcela určitě patří (tj. mající stupeň příslušnosti 1) člověk s výškou od 180 cm - 190 cm a naopak sem zcela určitě nepatří (tj. mající stupeň příslušnosti 0) člověk mající výšku menší nebo rovnu 170 cm a vyšší nebo rovnu 200 cm (výška 200 cm by mohla být součástí fuzzy množiny např. „velmi vysoký člověk“). V intervalech $(170, 180)$ a $(190, 200)$ je stupeň příslušnosti do fuzzy množiny „vysoký člověk“ větší než 0 a zároveň menší než 1, tj. vyjadřuje částečnou příslušnost do této množiny.



Obrázek 2.3: Funkce příslušnosti fuzzy množiny „vysoký člověk“.

Jazyková proměnná a její původní definice

Na tomto místě je vhodné popsat pojem *jazyková proměnná* (v některé literatuře bývá uváděn termín *lingvistická proměnná*), jejíž hodnoty jsou *jazykové výrazy (tzv. termy)* přirozeného jazyka. Původní definice jazykové proměnné byla zformulována L. A. Zadehem, tj. pětice $\langle X, T(\mathcal{X}), U, G, M\rangle$: [3]

$\mathcal{X}$	...	název jazykové proměnné,
$T(\mathcal{X})$	...	množina možných hodnot - jazykových výrazů,
U	...	univerzum,
G	...	syntaktické pravidlo, pomocí kterého jsou tvořeny jazykové výrazy $\mathcal{A}, \mathcal{B}$...z množiny $T(\mathcal{X})$,
M	...	sémantické pravidlo, pomocí kterého je každému jazykovému výrazu $\mathcal{A} \in T(\mathcal{X})$ přiřazen jeho význam, tzn. $A = M(\mathcal{A})$ je fuzzy množina na univerzu U.

Nejčastěji uváděný příklad jazykové proměnné v literatuře je $\mathcal{X} = výška$, jejíž množinu jazykových výrazů $T(\mathcal{X})$ tvoří např. výrazy *malý, velký, střední, velmi velký, ne malý*, apod. [1]

Sémantika přirozeného jazyka

Nejdůležitější pojmy k pochopení významu propojení přirozeného jazyka s fuzzy množinami si zde vysvětlíme. *Intenze* reprezentuje vlastnost jazykového výrazu, kterou daný výraz vyjadřuje. Vlastnost zůstává stejná bez ohledu na čas, místo, kontext. Např. intenze pojmu „vysoká výška“ je vlastnost „být vysoký“ uspořádaných objektů. Intenzí tedy myslíme vlastnost jako takovou. Zatím neznáme žádný konkrétní objekt, kterého se daná vlastnost týká. Uvažujeme-li konkrétní kontext např. výšku lidí, budov, stromů apod., dostáváme se k pojmu *možný svět*. Pojem možný svět chápeme zjednodušeně jako stav všech věcí kolem nás. V tomto možném světě potom uvažujeme konkrétní objekty. Třída všech možných objektů mající danou vlastnost (intenzi) v možném světě se nazývá *extenze*. Takže např. extenze pojmu „vysoká výška“ v uvažovaném možném světě výšek dospělých osob představuje všechny hodnoty výšek lidí, kteří splňují danou vlastnost „být vysoký“ (např. výšky od 180 cm - 190 cm). [1]

Obr. 2.3 zobrazuje fuzzy množinu, která představuje výšky lidí obecně. Pokud bychom chtěli znázornit např. průměrnou výšku budov ve městech nebo průměrnou výšku osob nad 65 let, hodnoty výšek na vodorovné ose i tvar fuzzy množiny by mohl být jiný. Proto je ve fuzzy modelování důležité pochopit konkrétní kontext resp. možný svět, kterého se dané objekty týkají.

Z výše uvedených poznatků vyplývá, že původní Zadehova definice jazykové proměnné je nedostačující a je potřeba nějakým způsobem formálně definovat intenzi a její odpovídající extenze ve všech možných světech. Důležité je nejprve definovat abstraktní množinu *kanonických objektů*: [1]

$$M = \{t_a \mid a \in \langle 0, 1\rangle\}.$$

Dále definujeme jazykový výraz $\mathcal{A}$, který je jménem vlastnosti objektů. Vlastnost objektů x formálně označíme $A(x)$ (označuje formuli přiřazenou jazykovému výrazu $\mathcal{A}$, formuli chápeme jako jméno vlastnosti). Pak *intenzí* jazykového výrazu $\mathcal{A}$ rozumíme fuzzy množinu: [1]

$$Int(\mathcal{A}) = \{\alpha(a)/A[t_a] \mid a \in \langle 0, 1\rangle\},$$

kde prvek fuzzy množiny $\alpha(a)/A[t_a]$ vyjadřuje, že kanonický objekt t_a má vlastnost A se stupněm příslušnosti $\alpha(a) \in \langle 0, 1\rangle$ a $\alpha(x)$ je nějaká funkce.

Možný svět si definujeme jako dvojici: [1]

$$\mathcal{V} = (V, h),$$

kde $V = \langle v_l, v_r\rangle$, přičemž $v_l < v_r,\ v_l, v_r \in \mathbb{R}$ a h je izomorfismus $h : M \to V$ (bijektivní zobrazení množiny kanonických objektů M do uzavřeného intervalu V), pro který platí $h(t_0) = v_l,\ h(t_1) = v_r$ a $h(t_a) \leq h(t_b) \Leftrightarrow a \leq b,\ a, b \in \langle 0, 1\rangle$.

Uvažujeme-li jazykový výraz $\mathcal{A}$ a určitý možný svět $\mathcal{V}$, pak *extenzí* $\mathcal{A}$ ve $\mathcal{V}$ rozumíme fuzzy množinu: [1]

$$Ext_{\mathcal{V}}(\mathcal{A}) = \{\beta(v)/v \mid v \in V\},$$

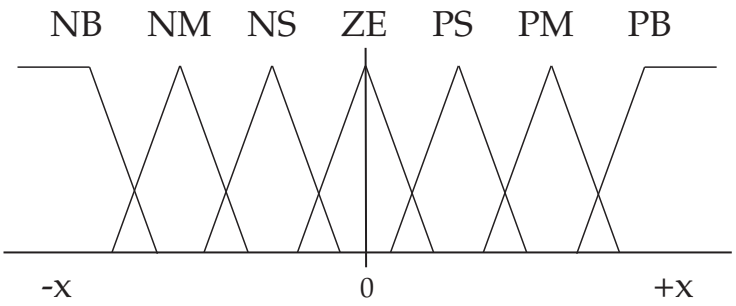
kde $\beta(h(t_a)) = \alpha(a)$.

Pokud nyní upravíme původní Zadehovu definici jazykové proměnné a místo univerza použijeme množinu kanonických objektů, původní pětice se rozšíří o třídu možných světů

$$\mathcal{P} = \{\mathcal{V} \mid \mathcal{V} \; je \; možný \; svět\}.$$

Množina jazykových výrazů $T(\mathcal{X})$ jazykové proměnné $\mathcal{X}$ generovaná nějakou formální gramatikou by teoreticky mohla být nekonečná. V praxi se pracuje s omezenou množinou jazykových výrazů, většinou je to lichý počet - 3, 5 nebo 7 prvků. Nejčastěji se ve fuzzy regulaci setkáme s těmito 7 výrazy (schéma zvané „pila“ ukazuje obr. 2.4): [1]

NB ... *negatively big* (záporná velká),
NM ... *negatively medium* (záporná střední),
NS ... *negatively small* (záporná malá),
ZE ... *zero* (nulová),
PS ... *positively small* (kladná malá),
PM ... *positively medium* (kladná střední),
PB ... *positively big* (kladná velká).



Obrázek 2.4: Zjednodušený tvar funkcí příslušnosti fuzzy množinové škály odpovídající jazykovým výrazům nejčastěji používaných ve fuzzy regulaci.

Evaluační jazykové výrazy

Evaluační jazykové výrazy reprezentují buď přímo nějakou konkrétní hodnotu na uspořádané škále (většinou číslo), nebo charakterizují pozici na této škále (vlevo, vpravo, více vpravo). Představují je zejména *atomické jazykové výrazy* (např. malý, střední, velký) a *fuzzy čísla*, která bývají doplněna *jazykovými operátory*. Jsou to typické hodnoty jazykových proměnných používaných ve fuzzy logice a ve fuzzy regulaci. [1]

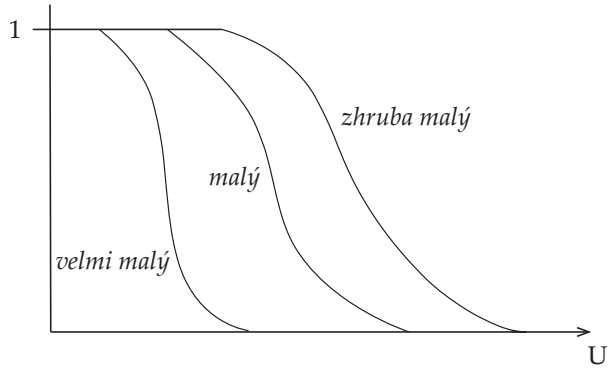
Mezi základní atomické jazykové výrazy patří *malý, střední, velký* a základní pozice na uspořádané škále jsou *vlevo, uprostřed, vpravo*. Nahradíme-li výraz „malý“ výrazem „slabý“ nebo „velký“ výrazem „silný“ apod., z hlediska principu (tj. uspořádané škály) se význam nezmění. Proto můžeme říci, že uvedené 3 atomické jazykové výrazy tvoří tzv. *základní evaluační trichotomii*. Atomický jazykový výraz může být upřesněn nebo ho naopak může udělat „hrubším“ *jazykový*

operátor (modifikátor). Tyto operátory se nacházejí před atomickými jazykovými výrazy. [1]

Jednoduchý evaluační jazykový výraz vypadá obecně takto:

$$[\langle jazykový\ operátor\rangle]\langle atomický\ jazykový\ výraz\rangle,$$

kde jazykový operátor nemusí být přítomný. Evaluační jazykové výrazy lze spojovat i pomocí logických spojek *ne, a, nebo* (např. „malý nebo spíše střední“). Jak se změní křivka reprezentující atomický jazykový výraz, použijeme-li jazykový operátor, je znázorněno na obr. 2.5. [1]



Obrázek 2.5: Zužující a rozšiřující efekt při použití jazykových operátorů před atomickým jazykovým výrazem „malý“.

Vezmeme si atomický jazykový výraz „malý“, pak různé modifikace, či korekce jeho významu pomocí jazykových operátorů mohou vypadat následovně: [1]

výrazně malý,
značně malý,
velmi malý,
malý,
více méně malý,
zhruba malý,
dosti zhruba malý,
velmi zhruba malý.

2.2.5 Tvary fuzzy množin a jejich funkce příslušnosti

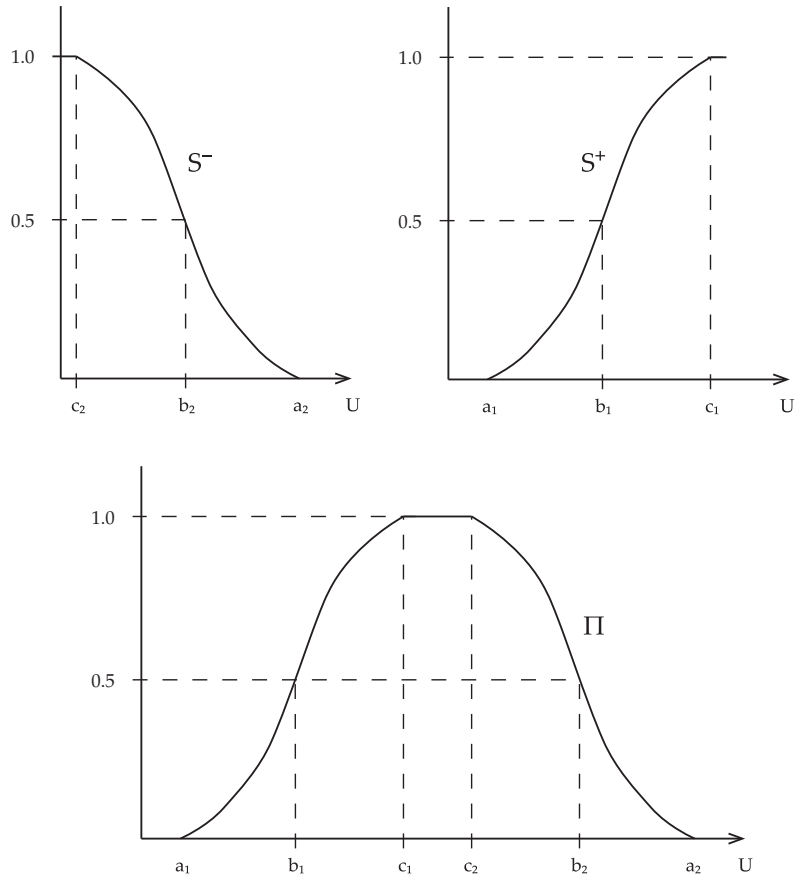
Už jsme si řekli, že fuzzy množinu v praxi ztotožňujeme s její funkcí příslušnosti a z předchozích poznatků také plyne, že fuzzy množiny jsou základní prostředek pro modelování sémantiky evaluačních jazykových výrazů. Běžně ve fuzzy regulaci jsou skutečné tvary fuzzy množin aproximovány lomenými přímkami (např. trojúhelníkový nebo trapezoidní tvar funkce příslušnosti), což má za následek ne úplně přesný význam jazykového výrazu.

Nejprve si uvedeme pár triviálních pojmů. Fuzzy množina A na univerzu U se nazývá *normální*, pokud má výšku rovnu 1, tzn.:

$$\exists x \in U \mid A(x) = 1.$$

V kapitole o fuzzy číslech bylo řečeno, že ostré číslo lze také reprezentovat fuzzy množinou. Můžeme tedy říci, že ostré číslo je normální fuzzy množina definovaná na jediném prvku univerza (tzv. *singleton*). [2]

Obr. 2.6 reprezentuje funkce příslušnosti fuzzy množin odpovídající atomickým jazykovým výrazům „malý“, „střední“, „velký“. Je zřejmé, že první dvě fuzzy množiny jsou speciálním případem třetí. Položíme-li $a_1 = b_1 = c_1$, vznikne fuzzy množina S^-. V případě $c_2 = b_2 = a_2$ dostaneme fuzzy množinu S^+. Parametry a_1, a_2 označují meze, za kterými zcela určitě nelze objekt x přiřadit k danému jazykovému výrazu (otevřený interval (a_1, a_2) tvoří nosič dané fuzzy množiny). Parametry b_1, b_2 jsou tzv. přechodové body, jejichž stupeň příslušnosti vyjadřující, že objekt x lze označit daným jazykovým výrazem, je roven 0.5. Parametry c_1, c_2 vymezují skutečnost, že všechny objekty x v tomto intervalu lze zcela určitě označit daným jazykovým výrazem (uzavřený interval $\langle c_1, c_2 \rangle$ tvoří jádro dané fuzzy množiny). [1]



Obrázek 2.6: Funkce příslušnosti fuzzy množin S^-, S^+ a Π s jejich parametry.

Pomocí níže uvedeného předpisu lze sestrojit všechny tři typy fuzzy množin z obr. 2.6: [1]

$$F(x, a_1, b_1, c_1, c_2, b_2, a_2) = \begin{cases} 0, & x \leq a_1 \text{ nebo } x \geq a_2 \\ \frac{1}{2}(\frac{x-a_1}{b_1-a_1})^2, & a_1 < x < b_1 \\ 1 - \frac{1}{2}(\frac{c_1-x}{c_1-b_1})^2, & b_1 \leq x < c_1 \\ 1 - \frac{1}{2}(\frac{x-c_2}{b_2-c_2})^2, & c_2 < x < b_2 \\ \frac{1}{2}(\frac{a_2-x}{a_2-b_2})^2, & b_2 \leq x < a_2 \\ 1, & c_1 \leq x \leq c_2. \end{cases}$$

2.3 Fuzzy logika a přibližná dedukce

Fuzzy logika je obecně řečeno nadstavbou klasické dvouhodnotové logiky. Rozlišujeme fuzzy logiku *v úzkém slova smyslu (FLn)* tzv. vícehodnotová logika, jejíž prostředky slouží k modelování fenoménu vágnosti pomocí stupňů příslušnosti a fuzzy logiku *v širším slova smyslu (FLb)*, která představuje především teorii přibližné dedukce. Fuzzy logika v širším smyslu je odvozena od vícehodnotové fuzzy logiky, zabývá se lidským usuzováním, pracuje s přirozeným jazykem a její aparát je využíván ve fuzzy regulaci. [1] Proto ji částečně rozeberu v této kapitole.

2.3.1 Fuzzy pravidla typu JESTLIŽE-PAK

Spojení JESTLIŽE-PAK je přeloženo z anglického IF-THEN. Na první pohled se zdá, že je zde určitá podobnost s podmíněným příkazem *IF*, který je známý z většiny programovacích jazyků. Opak je pravdou. V programování konstrukce *IF* představuje podmínku, která je buď splněna nebo nesplněna (klasická dvouhodnotová logika). Fuzzy pravidla typu JESTLIŽE-PAK pracují k našemu překvapení s vágními výrazy a představují sofistikovanější nástroj. Jejich význam lze matematicky modelovat a také naprogramovat. Jsou to v podstatě implikace dvou výroků, které se skládají z jazykových výrazů ve tvaru: [1, 2]

$$\mathcal{R} := \textbf{JESTLIŽE } X_1 \text{ je } \mathcal{A}_1 \textbf{ A}... \textbf{ A } X_m \text{ je } \mathcal{A}_m \textbf{ PAK } Y \text{ je } \mathcal{B},$$

kde $X_1, ..., X_m, Y$ jsou *fuzzy proměnné*, které mohou nabývat fuzzy hodnot reprezentujících jazykové výrazy $\mathcal{A}_1, ..., \mathcal{A}_m, \mathcal{B}$. Pro ilustraci si uvedeme příklad fuzzy regulace s fuzzy proměnnými:

$$\begin{aligned} X_1 &:= \text{ vzdálenost,} \\ X_2 &:= \text{ brzdná dráha,} \\ Y &:= \text{ akční zásah (sešlápnutí brzdového pedálu),} \end{aligned}$$

pak jedno fuzzy pravidlo $\mathcal{R}$ může vypadat např.:

$$\begin{aligned} \mathcal{R} := \ & \textbf{JESTLIŽE } \text{vzdálenost je } \textit{malá} \textbf{ A } \text{brzdná dráha je } \textit{velká} \\ & \textbf{PAK } \text{sešlápnutí brzdy je } \textit{spíše velké}. \end{aligned}$$

První část pravidla za částí JESTLIŽE se nazývá *antecedent* a většinou je to složený výraz, jehož jednotlivé části jsou spojeny pomocí logických spojek. Druhá část pravidla za částí PAK se nazývá *konsekvent (sukcedent)*. Vstupní fuzzy proměnné $X_1, \ldots, X_m$ se nazývají *nezávislé proměnné*. Y je výstupní fuzzy proměnná (akční zásah), která závisí na hodnotách vstupních fuzzy proměnných a nazýváme ji *závislou proměnnou*. [1, 2]

Ve fuzzy regulaci je většinou zapotřebí charakterizovat určitou situaci pomocí více pravidel (podmíněných výroků). Množinu fuzzy pravidel $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ nazýváme *jazykovým popisem*. [1]

Báze fuzzy pravidel

Na soubor pravidel typu JESTLIŽE-PAK (jazykový popis) lze nahlížet jako na funkci, která vyjadřuje vztah mezi vstupními proměnnými $X_1, \ldots, X_m$ a výstupní proměnnou Y. Tuto funkci nazveme *bází fuzzy pravidel*: [1, 6]

$$\begin{aligned}
\mathcal{R}_1 &:= \textbf{JESTLIŽE } X_1 \text{ je } \mathcal{A}_{1,1} \textbf{ A}\ldots \textbf{ A } X_m \text{ je } \mathcal{A}_{1,m} \textbf{ PAK } Y \text{ je } \mathcal{B}_1, \\
\mathcal{R}_2 &:= \textbf{JESTLIŽE } X_1 \text{ je } \mathcal{A}_{2,1} \textbf{ A}\ldots \textbf{ A } X_m \text{ je } \mathcal{A}_{2,m} \textbf{ PAK } Y \text{ je } \mathcal{B}_2, \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \\
\mathcal{R}_n &:= \textbf{JESTLIŽE } X_1 \text{ je } \mathcal{A}_{n,1} \textbf{ A}\ldots \textbf{ A } X_m \text{ je } \mathcal{A}_{n,m} \textbf{ PAK } Y \text{ je } \mathcal{B}_n.
\end{aligned}$$

Jedna z možností vytvoření báze fuzzy pravidel je expertní zadání, kde expert na základě znalostí vztahů mezi vstupními a výstupními fuzzy proměnnými vytvoří všechny možné kombinace termů (jazykových výrazů) fuzzy proměnných $X_1, \ldots, X_m$ a přiřadí jim odpovídající termy fuzzy proměnné Y. Při expertním vytvoření báze fuzzy pravidel se nejdříve provede návrh, pak se dále ještě testuje a případně modifikuje. [6] Další možností vytvoření báze fuzzy pravidel je generování z dat pomocí shlukové analýzy nebo např. pomocí neuronových sítí (v mé práci se těmito metodami nezabývám).

2.3.2 Přibližná dedukce

Metody přibližné dedukce se využívají ke stanovení závěru na základě zadaných vstupů a fuzzy pravidel typu JESTLIŽE-PAK. Dedukce je obecně proces, ve kterém odvozujeme nějaká fakta (závěr) z dosud známých fakt pomocí dedukčních pravidel (usuzování). Obecně lze zapsat tuto situaci takto: [1]

$$\begin{aligned}
\textit{Podmínka}: &\quad \mathcal{R} := \textbf{JESTLIŽE } X_1 \text{ je } \mathcal{A}_1 \textbf{ A}\ldots \textbf{ A } X_m \text{ je } \mathcal{A}_m \textbf{ PAK } Y \text{ je } \mathcal{B} \\
\textit{Pozorování}: &\quad X_1 \text{ je } \mathcal{A}_1' \textbf{ A}\ldots \textbf{ A } X_m \text{ je } \mathcal{A}_m' \\
&\quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
\textit{Závěr}: &\quad Y \text{ je } \mathcal{B}',
\end{aligned}$$

kde symboly $\mathcal{A}_1', \ldots, \mathcal{A}_m'$ představují termy, které se mohou mírně lišit od původních termů $\mathcal{A}_1, \ldots, \mathcal{A}_m$. Z tohoto faktu vyplývá, že i závěr $\mathcal{B}'$ se může mírně lišit od $\mathcal{B}$. Základní princip je nalezení významu termu $\mathcal{B}'$, tj. nalezení fuzzy množiny. Schéma popsané výše není nic jiného než zobecněné známé pravidlo *modus*

ponens, které formálně můžeme zapsat: [1]

$$\frac{A, A \Rightarrow B}{B}.$$

Toto pravidlo říká, že pokud platí formule A a zároveň platí, že z formule A vyplývá formule B, platí i formule B.

Přibližná dedukce je tedy nástroj fuzzy regulace, kde hlavním cílem je dostat řízený systém do požadovaného stavu. Pro výpočet se využívá několik *fuzzy inferenčních mechanismů*. Nejprve si stručně popíšeme základní kroky fuzzy regulace:

Fuzzifikace, tj. přiřazení ke každému vstupu (většinou ostrá reálná hodnota) stupeň příslušnosti do jedné nebo více fuzzy množin.

Fuzzy inference je proces zpracování množiny (báze) pravidel a na základě zvolené inferenční implikace (inferenčního mechanismu) určení výstupní fuzzy množiny. Při zpracování změřených či vypozorovaných vstupních ostrých hodnot požadujeme i ostrou výstupní hodnotu, proto přichází na řadu proces defuzzifikace.

Defuzzifikace je proces aproximace neostrých termů výstupní fuzzy proměnné, při kterém dostaneme jedinou ostrou hodnotu, tj. akční zásah.

Fuzzy inferenčních mechanismů se v praxi používá několik: např. *Larsenův, Sugenův, Takagi-Sugenův*. Podrobněji zde popíšu nejčastěji používaný *Mamdaniho inferenční mechanismus*.

Mamdaniho inferenční mechanismus

Tento mechanismus je také někdy nazýván Mamdaniho-Assilianův inferenční mechanismus. Patří k nejznámějším a nejpoužívanějším ve fuzzy regulátorech.

Nechť $\mathcal{A}_{i,1}, \dots, \mathcal{A}_{i,m}, \ \mathcal{B}_i$ jsou termy báze fuzzy pravidel $\mathcal{R}$ tvořeny fuzzy čísly $A_{i,1}, \dots, A_{i,m}, \ B_i$, kde $i = 1, \dots, n$ a termy $\mathcal{A}_1^{'}, \dots, \mathcal{A}_m^{'}$ vzniklé pozorováním jsou tvořeny fuzzy čísly $A_1^{'}, \dots, A_m^{'}$ (pokud je pozorování tvořeno m-ticí reálných čísel, značíme $a_1, \dots, a_m$), pak Mamdaniho inferenční mechanismus popíšeme ve 3 krocích: [6]

1. Vypočítáme stupeň zasažení h_i i-tého pravidla fuzzy vstupem, kde $i = 1, \dots, n$:

$$h_i = min\{hgt(A_1^{'} \cap A_{i,1}), \dots, hgt(A_m^{'} \cap A_{i,m})\},$$

kde hgt je výška průniku významu pozorování a významu levé strany pravidla. V případě vstupů tvořenými reálnými čísly lze použít vztah:

$$h_i = (A_{i,1} \times \ \dots \ \times A_{i,m})(a_1, \dots, a_m).$$

2. Uřízneme fuzzy výstupní hodnoty B_i^M na pravé straně jednotlivých pravidel ve výšce h_i:

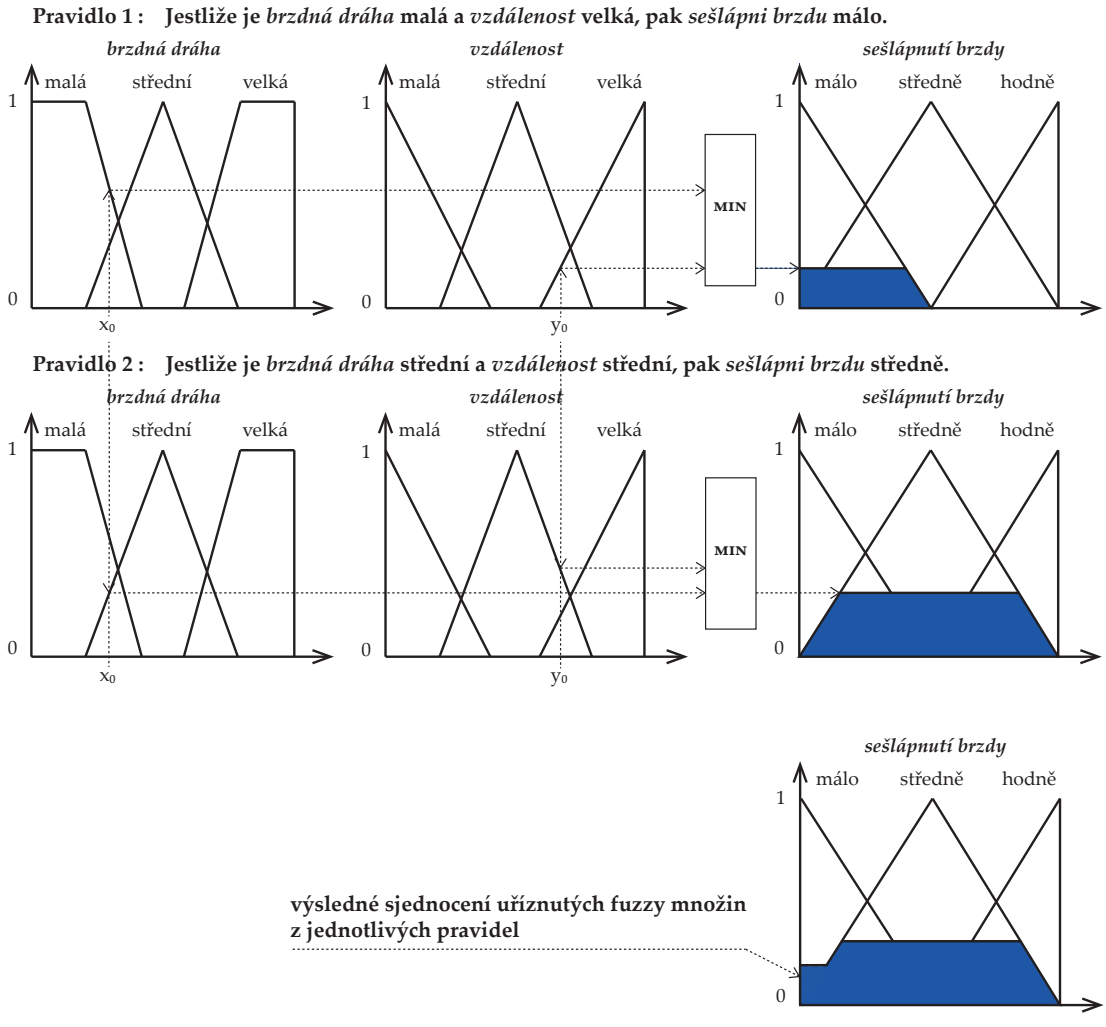
$$\forall y \in V : B_i^M(y) = min\{h_i, B_i(y)\},$$

kde M označuje Mamdaniho inferenční mechanismus.

3. Posledním krokem je stanovení výstupní fuzzy množiny B^M, která vznikne sjednocením všech fuzzy výstupních hodnot B_i^M:

$$B^M = \bigcup_{i=1}^{n} B_i^M.$$

Výsledkem Mamdaniho inferenčního mechanismu je fuzzy množina, která nemusí být fuzzy číslem (viz obr. 2.7). V případě fuzzy regulátorů je většinou potřeba stanovení akčního zásahu, tj. převedení výstupní fuzzy množiny na ostrou hodnotu (reálné číslo). Tento proces se nazývá defuzzifikace a opět existuje několik metod, z nichž nejpoužívanější si stručně rozebereme níže.



Obrázek 2.7: Příklad Mamdaniho inferenčního mechanismu pro 2 pravidla a 2 nezávislé fuzzy proměnné.

2.3.3 Defuzzifikace

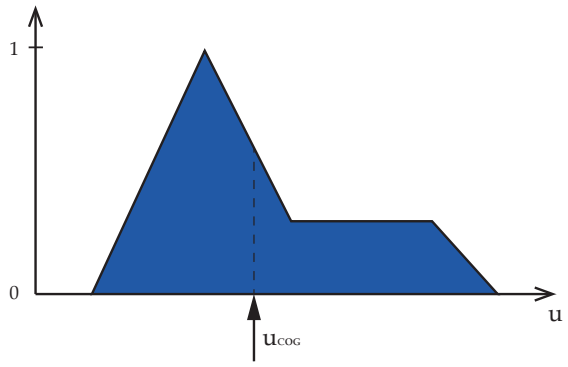
Defuzzifikace je proces, při kterém požadujeme z výstupní fuzzy množiny dostat pouze jednu reálnou hodnotu. Otázkou zůstává, jaká ostrá hodnota nejlépe reprezentuje různé tvary vzniklých fuzzy množin. Intuitivně dojdeme k poznatku, že pravděpodobně neexistuje univerzální algoritmus, který by vyhovoval vždy v každé situaci. Proto existuje celá řada defuzzifikačních metod využívaných ve fuzzy regulaci a je na expertovi, kterou vybere.

Pro následující popis defuzzifikačních metod budeme uvažovat fuzzy množinu $A = \{a_1/u_1, \dots, a_n/u_n\}$ na univerzu reálných čísel $\mathbb{R}$, která má konečný nosič: [1]

Metoda COG/COA (Center of Gravity/Area)

$$u_{COG} = \frac{\sum_{i=1}^{n} A(u_i) \cdot u_i}{\sum_{i=1}^{n} A(u_i)}$$

je nejpoužívanější metoda při fuzzy aproximaci. Někdy je nazývána metodou těžiště plochy. Výsledná hodnota představuje těžiště fuzzy množiny A (viz obr. 2.8). Nevýhodou je velká výpočetní složitost.



Obrázek 2.8: Metoda defuzzifikace COG.

Metoda MOM (Mean/Middle of Maxima)

$$u_{MOM} = \frac{1}{n_{max}} \sum_{j=1}^{n_{max}} u_j^{max},$$

kde $u_j^{max} = u_j$, pokud $A(u_j) = max\{A(u_i) \mid i = 1, \dots, n\}$, tj. $\{u_j^{max} \mid j = 1, \dots, n^{max}\}$ jsou všechny prvky nosiče fuzzy množiny s maximálním stupněm příslušnosti $A(u_j^{max})$. Tato metoda je nazývána metodou nejvýznamnějšího maxima nebo někdy také metoda středu maxim. Výsledná hodnota je tedy střed maximálních hodnot resp. střed hodnot s maximálním stupněm příslušnosti (na obr. 2.9 je výsledná hodnota metody MOM vyznačena šipkou uprostřed). V případě, že je fuzzy množina symetrická a má pouze jedno maximum (např. fuzzy číslo), je výsledek shodný s metodou COG. Metoda MOM je výpočetně méně

náročnější než COG.

Metody FOM (First of Maxima) a LOM (Last of Maxima)

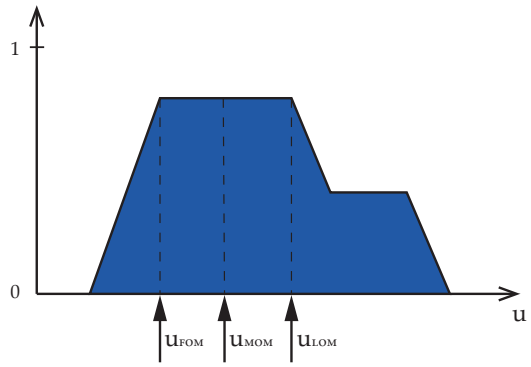
$$u_{FOM} = \bigwedge \{u_j^{max} \mid j = 1, \dots, n_{max}\}$$

je metoda prvního maxima, která spočívá v tom, že se vybere první prvek u_j^{max} s maximálním stupněm příslušnosti $A(x_j^{max})$ (na obr. 2.9 je výsledná hodnota metody FOM vyznačena šipkou vlevo).

$$u_{LOM} = \bigvee \{u_j^{max} \mid j = 1, \dots, n_{max}\}$$

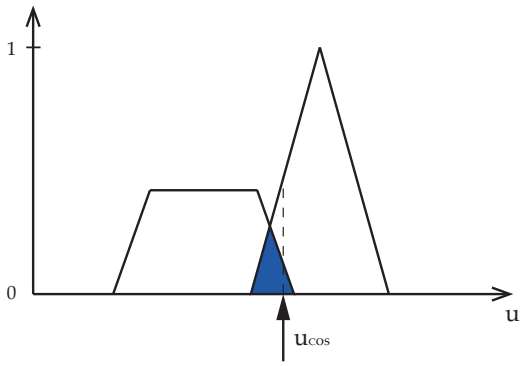
je metoda posledního maxima, kde se analogicky vybere poslední prvek u_j^{max} s maximálním stupněm příslušnosti $A(x_j^{max})$ (na obr. 2.9 je výsledná hodnota metody LOM vyznačena šipkou vpravo).

Obě metody FOM i LOM se v praxi příliš nepoužívají a patří mezi nejjednodušší defuzzifikační metody.



Obrázek 2.9: Metody defuzzifikace FOM, MOM, LOM.

Metoda COS (Center of Sums)



Obrázek 2.10: Metoda defuzzifikace COS.

$$u_{COS} = \frac{\sum_{i=1}^{n}(u_i \cdot \sum_{j}^{k} B_j(u_i))}{\sum_{i=1}^{n}\sum_{j=1}^{k} B_j(u_i)}$$

je volně přeloženo metoda středu součtů, kde výsledná fuzzy množina A je dána sjednocením fuzzy množin $A = B_1 \cup ... \cup B_k$. Fuzzy množiny B_i jsou výsledkem přibližné dedukce v rámci uplatněných pravidel. Výstup této defuzzifikační metody je těžiště sjednocení fuzzy množin A, přičemž průniky fuzzy množin B_i jsou na rozdíl od metody COG zde zohledněny vícekrát (viz obr. 2.10).

Tato metoda má oproti COG lepší výpočetní složitost a je také poměrně často používána.

3 Fuzzy regulace a Texas Hold'em poker

3.1 Princip fuzzy regulace

V následujícím textu shrnu předchozí poznatky a stručně vysvětlím princip fuzzy regulace. Aplikujeme-li přibližnou dedukci v praxi, tj. většinou v řízení technologických procesů (v mém případě jako počítačového hráče v pokeru), hovoříme o fuzzy regulaci. Fuzzy regulátor podobně jako klasický regulátor může být zapojen do uzavřené zpětnovazební smyčky nebo do složitějšího řídícího systému. Klasický regulátor můžeme vnímat jako funkci odvozenou od určitého matematického popisu. U fuzzy regulátoru předpokládáme, že matematický model není znám, ale existuje způsob, tj. určitá znalost, jak systém řídit. Fuzzy regulace je tedy vhodná všude tam, kde daný řídící systém dokáže ovládat člověk (*expert*) na základě pravidel typu JESTLIŽE-PAK („jestliže ručička na tlakoměru rychle stoupá, tlakový ventil pootoč docela vlevo"). [1]

Pro stanovení akčního zásahu není třeba znát složitý algoritmus mezi vstupem a výstupem. Charakteristickým znakem fuzzy regulace je tedy možnost použití empirických znalostí člověka (experta daného řídícího procesu). Tuto znalost regulační strategie označujeme jako *bázi znalostí*.
Báze znalostí je tvořena: [5]

a) bází dat, což jsou data tvořena intervaly obsahující hodnoty vstupních a výstupních veličin včetně jejich mezních hodnot (informace o stacionárních stavech) a funkcemi příslušnosti všech vstupních a výstupních fuzzy množin.

b) bází pravidel, což jsou kvantitativně formulované zkušenosti a slovně charakterizované strategie řízení daných procesů, pomocí nichž lze stanovit akční zásah (výstupní veličinu).

Shrnutí obecného postupu při aplikaci fuzzy regulace

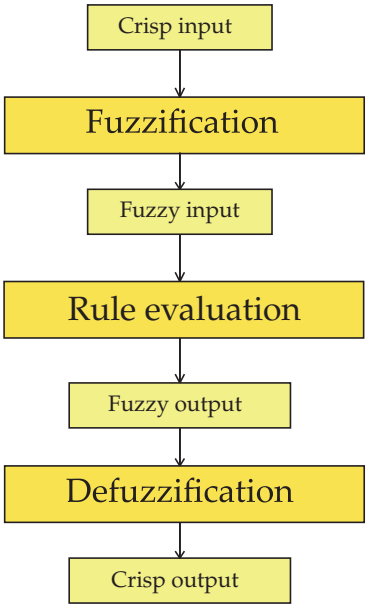
Na obr. 3.1 je znázorněno zjednodušené schéma fuzzy systému, který tvoří 3 základní bloky - *fuzzifikace, inference (vyhodnocení pravidel), defuzzifikace.*

Ve fázi fuzzifikace se transformují ostrá data (naměřená, zadaná, popř. získaná jiným způsobem) na fuzzy data pomocí přiřazení stupňů příslušnosti do jedné nebo více fuzzy množin (přeměna na jazykové termy).

Ústřední část fuzzy systému tvoří inference, která je realizována vhodně zvoleným inferenčním mechanismem (v mém případě Mamdaniho inferenční mechanismus). Do inferenčního mechanismu vstupuje báze znalostí, přičemž výsledkem tohoto procesu je určení výstupní fuzzy množiny.

V poslední fázi defuzzifikace z výstupní fuzzy množiny dostaneme vhodnou defuzzifikační metodou ostrou výstupní hodnotu, tj. požadovaný akční zásah.

Úspěšný běh každého fuzzy systému závisí na mnoha faktorech. Z předchozího textu je zřejmé, že mezi nejdůležitější faktory patří vhodně definovaná báze pravidel, tvar funkcí příslušnosti (i definice jejich mezních hodnot) a výběr metody defuzzifikace. Bázi pravidel můžeme vytvořit buď na základě expertních znalostí člověka, tj. obsluhy daného fuzzy systému nebo existuje ještě jeden způ-



Obrázek 3.1: Schéma fuzzy systému.

sob vytvoření báze pravidel, a sice na základě obecně platných metapravidel, podle kterých se příslušná pravidla dají odvodit: [5]

- $MP1$ – je-li regulační odchylka $e(k)$ a její změna $\Delta e(k)$ nulová nebo blízká nule, přírůstek akční veličiny $\Delta u(k)$ by měl být nulový nebo blízký nule
- $MP2$ – klesá-li regulační odchylka $e(k)$ k nule nebo se blíží nule dostačující rychlostí, pak je vhodné akční veličinu neměnit
- $MP3$ – jestliže se regulační odchylka $e(k)$ nekoriguje sama, pak je třeba akční veličinu změnit a akční zásah $\Delta u(k)$ bude nenulový. Jeho velikost a znaménko závisí na znaménku a velikosti regulační odchylky $e(k)$ a její změny $\Delta e(k)$

V případech, kdy fuzzy regulátor nepracuje zcela podle našich představ, je potřeba přehodnotit tvorbu rozhodovacích pravidel, popř. upravit funkce příslušnosti nebo zvolit jinou metodu defuzzifikace.

3.2 Něco o Texas Hold'em pokeru

O pokeru obecně

V této fázi, kdy už víme něco o fuzzy regulátorech, nastal čas krátce pohovořit o hře *Texas Hold'em poker*. Ještě než popíšu princip a pravidla hry, něco málo povím o pokeru obecně. Texas Hold'em poker má poměrně jednoduchá pravidla, takže této hře může propadnout opravdu každý. Z ostatních variant pokeru jsou ještě rozšířené např. *Omaha High, Omaha Hi-Lo, 5 Card Omaha, 7 Card Stud*

nebo 2-7 Triple Draw. Všechny tyto varianty se hrají s balíčkem 52 karet. Existuje také několik variant pokeru z pohledu možnosti sázení - *fixed limit, pot limit a no limit*. [7]

V *Limit (fixed) Texas Hold'em* je pevně stanovena minimální i maximální sázka, stejně tak jako počet jednotlivých možných sázek, navýšení (*raisů*) a povinných sázek na začátku hry.

V *Pot limit Texas Hold'em* je velikost možné sázky omezena aktuální výší banku neboli *potu*. Hráč na řadě tak může vsadit či navýšit o libovolné množství žetonů do maximální velikosti aktuálního potu.

V případě mé aplikace je použit princip sázení *No limit Texas Hold'em*, kde počet sázek ani jejich výše není nijak omezena. Hráči mohou kdykoliv vsadit jakékoliv množství žetonů (nejvíce všechny svoje žetony) bez ohledu na výši potu.

Výherní kombinace (od nejslabší po nejsilnější)

Nejvyšší karta (high card)

Nejslabší výherní kombinací je nejvyšší karta. Pokud mají hráči tuto kartu stejnou, o výherní kombinaci rozhodne 2., popř. až 5. karta (tzv. *kicker*).

Jeden pár (one pair)

Jeden pár tvoří dvě karty stejné hodnoty. Vítězí vyšší hodnota páru. Je-li shodná, rozhodne další nejvyšší karta (případně 2. nebo 3. nejvyšší).

Dva páry (two pair)

Dva páry se skládají ze dvou dvojic karet o stejné hodnotě. Rozhoduje hodnota vyššího z párů. Pokud je vyšší pár shodný, rozhodne nižší. V případě shody i nižšího páru rozhoduje 5. karta.

Trojice (three of a kind)

Trojici tvoří tři karty stejné hodnoty. V případě shody rozhoduje 4., popř. až 5. karta.

Postupka (straight)

Postupka je pětikaretní kombinace a skládá se z po sobě jdoucích karet, přičemž nezáleží na barvě. Nesmí mezi nimi být žádná mezera. Eso se dá v postupce použít i jako jednička. Má-li více hráčů postupku, vyhrává nejvyšší. Nejsilnější postupka je *A, K, Q, J, T*, nejslabší je *A, 2, 3, 4, 5*.

Barva (flush)

Opět pětikaretní kombinace karet jedné barvy. Má-li více hráčů flush, vyhrává nejvyšší karta v barvě, popř. se v případě shody porovnávají karty další.

Fullhouse

Fullhouse je tvořen trojicí karet stejné hodnoty a dvojicí karet stejné hodnoty. Nejprve rozhoduje hodnota trojice, pak hodnota dvojice.

Čtveřice - poker (four of a kind)

Čtveřici neboli poker tvoří čtyři karty stejné hodnoty. Pokud je poker otočen ve společných kartách, rozhoduje kicker (5. karta).

Čistá postupka (straight flush)

Čistá postupka se skládá z pěti po sobě jdoucích karet jedné barvy.

Královská postupka (royal flush)

Královská postupka je nejvyšší čistá postupka *A, K, Q, J, T*, tj. nejvyšší možná kombinace v pokeru.

Texas Hold'em - pravidla [7]

V Texas Hold'emu se používají společné (komunitní) karty. Na začátku každé hry dostanou všichni hráči na ruku rozdané dvě karty (*hole cards*). Dalších pět karet je postupně během hry rozdáno na stůl a jsou společné pro všechny hráče. Cílem hry je pak složit z 5 společných a 2 vlastních karet co nejlepší výherní kombinaci, která se skládá vždy z 5 karet a získat tak pot. V případě, že ve hře zbývá jeden hráč (ostatní zahodili karty), tento hráč vyhrává celý pot bez nutnosti ukazování karet.

Blindy - povinné sázky

Před začátkem hry je nutné vylosovat dealera (rozdávajícího). Tato pozice se následně spolu s povinnými sázkami po ukončení vyhodnocení každé hry posouvá mezi hráči o jedno místo ve směru hodinových ručiček. Hráč nalevo od dealera vloží do hry tzv. *small blind*, což je malá povinná sázka. Druhý hráč nalevo od dealera (první hráč nalevo od small blindu) pak vloží do hry dvojnásobek small blindu (*big blind*), což je velká povinná sázka. Big blind také hodnotou odpovídá minimální sázce, kterou lze ve hře vsadit. Povinné sázky jsou prvními žetony vloženými do potu a jejich účel je zřejmý, tj. v každé hře je o co hrát i když všichni na začátku zahodí karty.

Kolo sázek před flopem

Po vložení povinných sázek rozdá dealer každému z hráčů u stolu 2 karty. Rozdává se po jedné kartě a první na řadě je hráč v levo od dealera (na small blindu). Tyto karty má každý hráč na ruce a nejsou pro ostatní viditelné. Prvním hráčem, který vstupuje do hry (musí provést nějakou akci), je hráč sedící nalevo od pozice big blindu. Pokud je hráč na řadě, může buď dorovnat sázku (*call*) v tomto případě ve výši velkého blindu, může zvýšit sázku (*raise*) a samozřejmě může své karty složit (*fold*) a hru vzdát. Vzhledem k tomu, že ve hře je velká povinná sázka, není možné zůstat ve hře bez vsazení. Pouze hráč na big blindu se může zdržet sázky (*check*), pokud nikdo jiný během tohoto kola sázku nenavýšil.

Kolo sázek po flopu

Poté, co je uzavřeno první kolo sázení, se rozdají na stůl 3 společné karty. Dealer sejme, tzv. *spálí* jednu kartu z vrchu balíčku a na stůl současně otočí tři karty lícem nahoru (*flop*). Každý hráč nyní už může společně se svými kartami na ruce pomalu vyhodnocovat karetní kombinace a svoje šance proti soupeřům v dalším průběhu hry. Prvním hráčem, který provádí nějakou akci v tomto kole, je hráč hned nalevo od dealera (samozřejmě se tím myslí první hráč, který ještě nesložil karty). Sázení se řídí stejnými pravidly jako v předchozím kole. Dokud nikdo nevsadí, je možné zůstat ve hře bez vsazení. V ostatních případech je nutné provést dorovnání sázky, navýšení sázky nebo složit karty.

Kolo sázek po turnu

V tomto kole se vykládá další společná karta. Po ukončení sázek po flopu

se jde do dalšího sázkového kola. Dealer opět spálí jednu kartu z vrchu balíčku a následně otočí další kartu na stůl (*turn*). Hráči si opět v duchu vyhodnocují možné karetní kombinace ze svých i společných karet. Sázky se řídí stejnými pravidly jako v předchozích kolech.

Kolo sázek po riveru

V tomto kole se vykládá poslední společná karta. Nejprve po spálení karty z vrchu balíčku otočí dealer poslední kartu (*river*) a potom proběhne rozhodující kolo sázení podobně jako v předchozích kolech. Pokud po ukončení všech sázek zůstali ve hře minimálně 2 hráči, přichází na řadu tzv. *showdown*, tj. ukazování karet. Pot získává hráč s nejsilnější karetní kombinací, kterou vybral ze svých 2 a z 5 společných karet (výsledná výherní kombinace bude mít vždy 5 karet). V případě tzv. *splitu*, tj. dva či více hráčů mají stejně silnou kombinaci, dochází k rovnoměrnému dělení potu. Následně se posune pozice dealera a hráčů na blindech, karty se zamíchají a pokračuje se v další hře.

Co je ALL-IN

Pokud hráč vsadí v No limit Texas Hold'emu všechny své žetony (*all-in*), nemůže zahodit karty. Mají-li ostatní hráči ve hře další žetony, mohou tito hráči pokračovat v sázení. Žetony navíc se sází do vedlejšího banku (*side pot*) o který hráč v all-inu už nehraje. Na konci se pak pot rozdělí spravedlivě podle toho, jak kdo na jaký pot dosahuje.

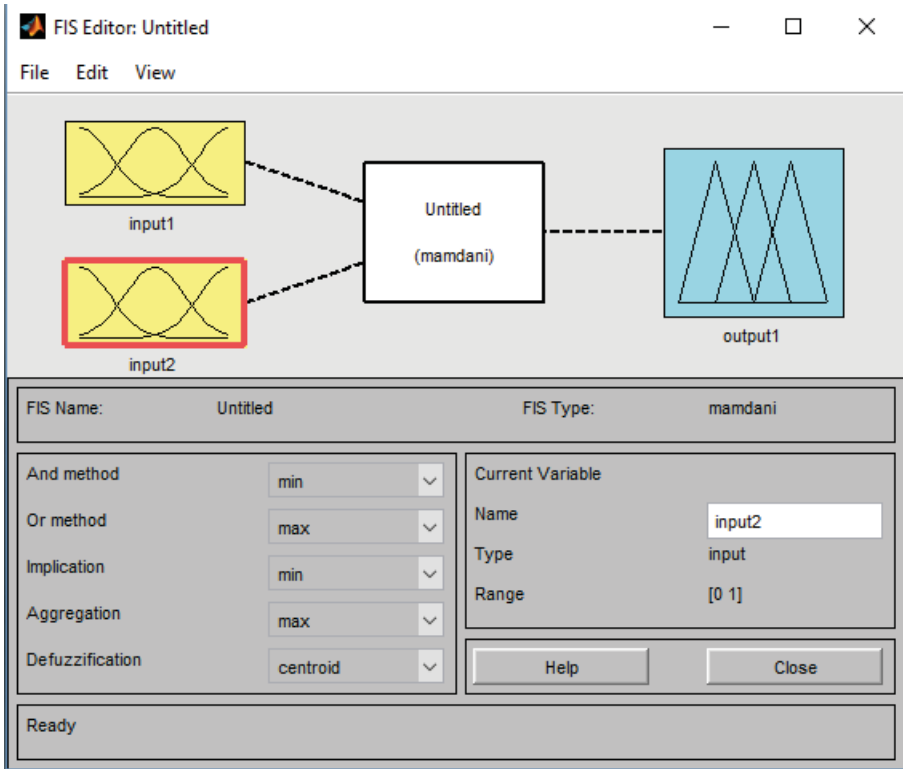
3.3 MATLAB a Fuzzy Logic Toolbox

Dříve než popíšu vlastní fuzzy regulátor, musím říci pár slov o programu, v kterém jsem fuzzy regulátor sestavil a následně pak implementoval do svého programu. Využil jsem vyzkoušený a osvědčený nástroj *MATLAB* (konkrétně *MATLAB R2014a*). Jedná se o výkonné interaktivní programové prostředí, které je vhodným nástrojem pro počítání matic, vizualizaci 2D i 3D grafů funkcí, numerickou analýzu, matematické výpočty, algoritmizaci, modelování, simulaci, zpracování signálů a mnoho dalšího. Hlavním přínosem je relativně snadná orientace v příjemném uživatelském prostředí. Další obrovská výhoda je jeho snadná rozšiřitelnost, která umožňuje program rozšířit o nové uživatelsky napsané funkce (*m-soubory*) i o celé programy. MATLAB obsahuje celou řadu nadstaveb (*toolboxů*), což jsou kolekce m-souborů orientované na určitou třídu problémů. V případě mého fuzzy regulátoru jsem využil program *Fuzzy Logic Toolbox*. [8]

Vlastní fuzzy regulátor jsem navrhl pomocí interaktivního grafického prostředí (GUI), kde je možno využít mnoho nástrojů pro fuzzy logickou dedukci, analýzu a implementaci. V GUI tohoto toolboxu lze jednoduše definovat vstupní a výstupní proměnné, jejich rozsahy, funkce příslušnosti a jejich parametry. Dále zde velmi snadno probíhá vytváření rozhodovacích pravidel a zadávání metod fuzzifikace a defuzzifikace. Fuzzy inferenční systém ve Fuzzy Logic Toolboxu lze vytvářet pomocí tří editorů: [5, 9]

FIS Editor (editor inferenčního systému fuzzy regulátoru),

Membership Function Editor (editor funkcí příslušnosti),



Obrázek 3.2: Editor inferenčního systému fuzzy regulátoru ve Fuzzy Logic Toolboxu.

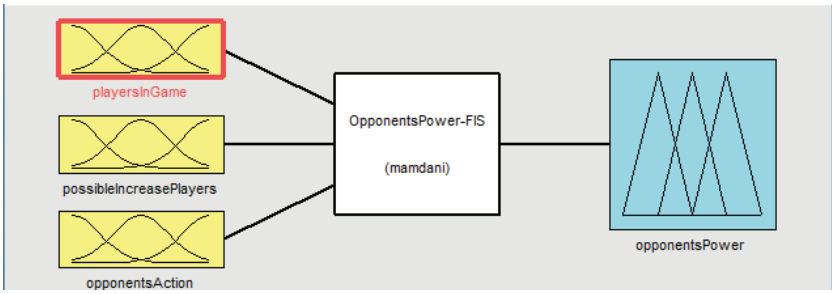
Rule Editor (editor pravidel).
Fuzzy Logic Toolbox umožňuje dvě možnosti zobrazení: [5, 9]
Rule Viewer (grafické zobrazení procesu inference),
Surface Viewer (zobrazení plochy ohraničující prostor generovaných akčních zásahů).

Jednotlivé funkce fuzzy systému se dají spravovat i pomocí příkazové řádky. Potřebujeme-li otevřít v MATLABU počáteční okno FIS editoru, aktivujeme ho v příkazovém řádku příkazem „*fuzzy*“. Již existující inferenční systém aktivujeme příkazem „*fuzzy název-souboru.fis*“. [5, 9]

3.4 FIS jako počítačový hráč v Texas Hold'em pokeru

Nyní v této fázi se dostávám k představení vlastního fuzzy regulátoru. Podrobně zde popíšu fuzzy inferenční systém (FIS), tj. počítačového hráče v Texas Hold'em pokeru, který jsem vytvořil pomocí nástroje Fuzzy Logic Toolbox. V nastavení hry, kterou jsem naprogramoval, je možnost volby mezi dvěma druhy počítačového hráče. Fuzzy systém je tedy rozdělen na dvě nezávislé části: *AggressivePlayer, ConservativePlayer*. Oba z těchto fuzzy systémů obsahují vložený fuzzy systém, který má za úkol generovat jednu ze vstupních proměnných: *OpponentsPower*.

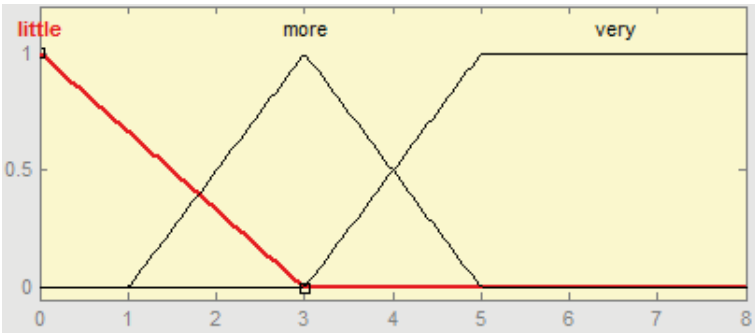
3.4.1 Fuzzy systém OpponentsPower



Obrázek 3.3: Fuzzy systém OpponentsPower.

Výstup (akční zásah) z tohoto fuzzy systému představuje ostrou hodnotu jedné ze vstupních proměnných nadřazeného fuzzy systému (AggressivePlayer resp. ConservativePlayer). OpponentsPower má 3 vstupní proměnné a 1 výstupní proměnnou (viz obr. 3.3).

Vstupní proměnná „*playersInGame*“



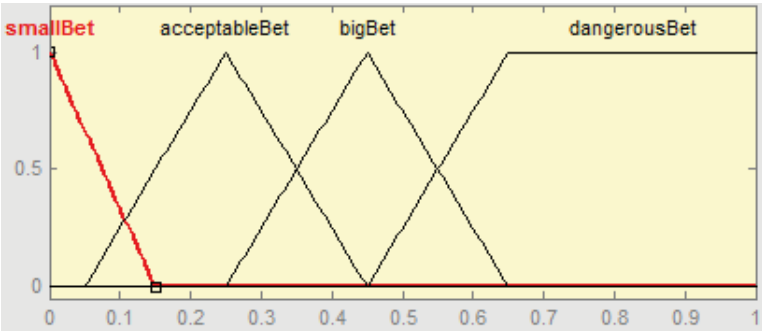
Obrázek 3.4: Funkce příslušnosti vstupní proměnné „*playersInGame*“.

Ostrá hodnota této proměnné představuje počet hráčů, kteří aktuálně už vstoupili do hry. V nastavení hry lze zvolit počet hráčů 2 - 9. Takže hráčů (nepočítá se hráč na tahu), kteří už jsou ve hře, může být 0 - 8. Tvary funkcí příslušnosti jsou u všech použitých proměnných buď trojúhelníkové nebo trapezoidní. V tomto případě funkce příslušnosti i s použitými termy ukazuje obr. 3.4.

Vstupní proměnná „*possibleIncreasePlayers*“

Tato proměnná společně s „*playersInGame*“ vypovídá o tom, jak je silná pozice hráče na tahu (kde u stolu sedí, tj. kolik hráčů už hraje a kolik ještě může vstoupit do hry). Ostrá hodnota této proměnné je tedy hodnota možného přírůstku hráčů, kteří se ještě mohou hry zúčastnit (opět jich může být 0 - 8). Funkce příslušnosti i použité termy jsou stejné jako u proměnné „*playersInGame*“ (obr. 3.4).

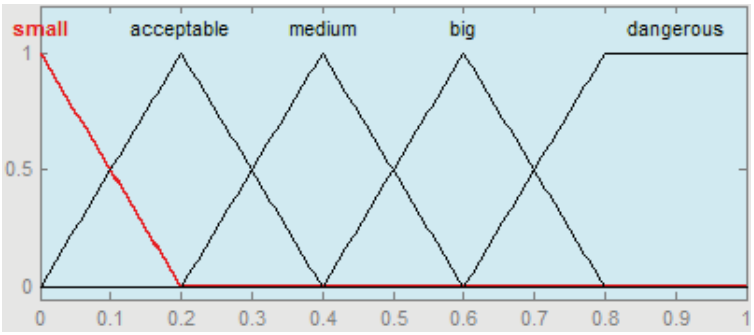
Vstupní proměnná „*opponentsAction*“



Obrázek 3.5: Funkce příslušnosti vstupní proměnné „*opponentsAction*“.

Ostrou hodnotou je zde vypočítaná hodnota, tj. poměr sázky toho protihráče, který vsadil zatím nejvíce (celkem od 1. kola sázení) a zbývajícího počtu žetonů hráče na tahu. K této sumě je přičteno číslo zohledňující aktuální výši povinné sázky (pokud by povinná sázka byla příliš vysoká, sázka protihráče může být tímto vynucená). Výsledná hodnota bude číslo v intervalu $(0, 1)$. Pokud je tedy sázka protihráče zanedbatelná vzhledem ke zbývajícím žetonům hráče na tahu, ostrá hodnota této proměnné se bude blížit k nule. V opačném případě se tato hodnota bude blížit k číslu jedna. Na obr. 3.5 jsou vidět použité termy a funkce příslušnosti.

Výstupní proměnná „*opponentsPower*“



Obrázek 3.6: Funkce příslušnosti výstupní proměnné „*opponentsPower*“.

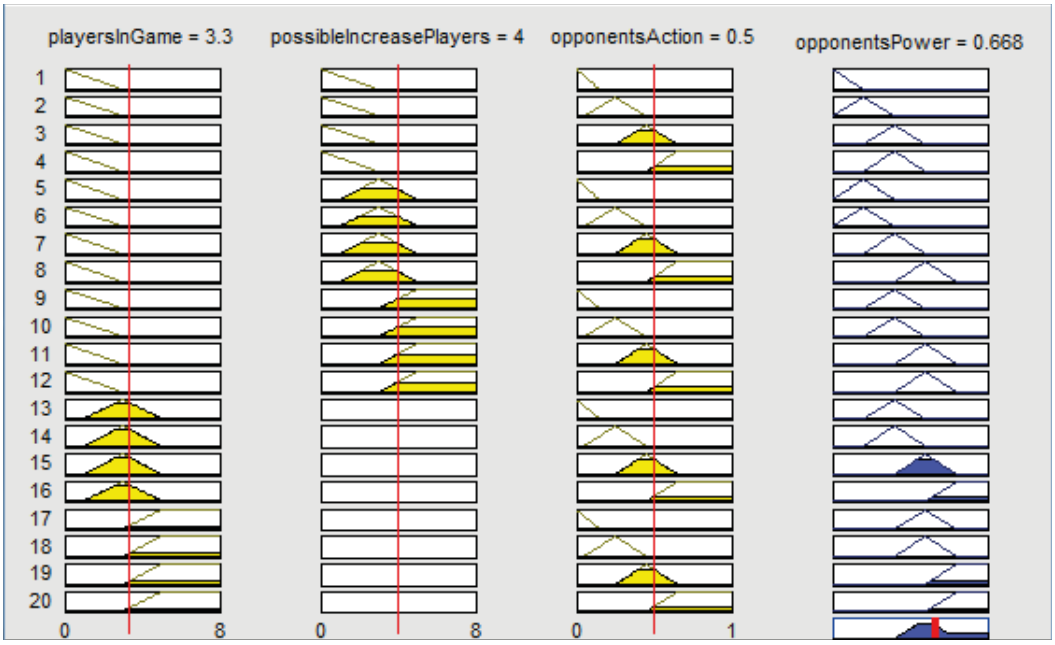
Výstup z tohoto fuzzy systému by se dal nazvat „síla protihráčů“. Sděluje informaci, jakým způsobem a kolik dalších protihráčů už vstoupilo, popř. nevstoupilo do hry. Funkce příslušnosti s použitými termy ukazuje obr. 3.6.

Báze pravidel fuzzy systému OpponentsPower

Báze pravidel obsahuje 20 pravidel. Všechny jsou uvedeny v tabulce 1. Od 13. pravidla je ignorována vstupní proměnná „*possibleIncreasePlayers*“. Její hodnota v následujících pravidlech nemá vliv na akční zásah, není v nich tedy zahrnuta. Z prvního řádku můžeme vyčíst např. pravidlo:

IF „*playersInGame*“ IS „*little*“ AND „*possibleIncreasePlayers*“ IS „*little*“ AND „*opponentsAction*“ IS „*smallBet*“ THEN „*opponentsPower*“ IS „*small*“.

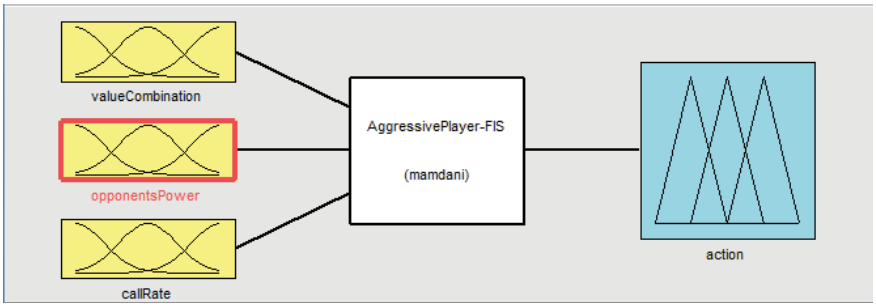
Na obrázku 3.7 je graficky znázorněn na konkrétním příkladu zadaných ostrých hodnot výstup inferenčního procesu.



Obrázek 3.7: Grafické zobrazení procesu inference s konkrétními hodnotami v „*rule viewru*“.

3.4.2 Fuzzy systémy AggressivePlayer a ConservativePlayer

Fuzzy systémy AggressivePlayer a ConservativePlayer jsou na sobě nezávislé. Jejich použití závisí na tom, jaký uživatel zvolí druh počítačového hráče v nastavení hry. Vstupní proměnné, výstupní proměnná a funkce příslušnosti obou fuzzy systémů jsou stejné. Liší se pouze bází pravidel. Jsou zde opět 3 vstupní proměnné („*opponetsPower*“, v tomto případě vstupní proměnná, je popsaná v kapitole 3.4.1) a 1 výstupní proměnná (obr. 3.8).



Obrázek 3.8: Fuzzy systém AggressivePlayer.

Vstupní proměnná „*valueCombination*“

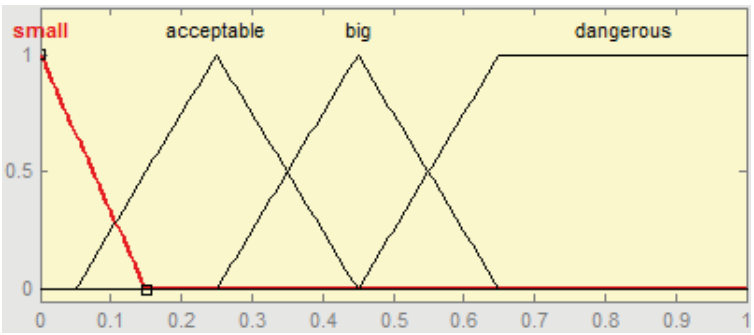
Obrázek 3.9: Funkce příslušnosti vstupní proměnné „*valueCombination*“.

Účelem této proměnné je zjistit sílu kombinace dvou karet, které drží hráč na ruce (*hole cards*). Ostrá hodnota je získáná pomocí vzorce, který vymyslel pokerový hráč Bill Chen. Jedná se o bodovací systém zohledňující hodnotu vyšší karty, jestli jsou obě karty stejné barvy a jak daleko jsou hodnoty obou karet od sebe.

Z vyšší karty se skóre vypočítá takto: [10]
A = 10 bodů, K = 8 bodů, Q = 7 bodů, J = 6 bodů, 10 až 2 = polovina z této hodnoty. Dále pokud jsou obě karty v páru, tak se body zdvojnásobí (např. KK = 16). Minimum pro pár je 5 bodů, tj. i páry 2 - 4 budou mít 5 bodů.
Karty stejné barvy s hodnotou u sebe: + 2 body
Karty s hodnotou u sebe: + 1 bod
Jedna mezera mezi hodnotami karet: - 1 bod
Dvě mezery: - 2 body
Tři mezery: - 4 body
Čtyři nebo více mezer: - 5 bodů

Takto získaná hodnota je pak převedena na číslo v intervalu $(0, 1)$ (tj. nejlepší kombinace AA = 20 bodů bude rovna 1) a vypovídá o hratelnosti kombinace (tvary funkcí příslušnosti ukazuje obr. 3.9).

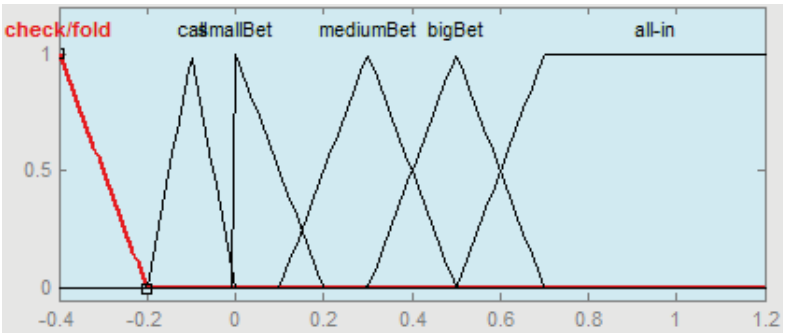
Vstupní proměnná „*callRate*“



Obrázek 3.10: Funkce příslušnosti vstupní proměnné „*callRate*“.

Ostrá hodnota je tvořena poměrem sázky, kterou je potřeba v aktuálním kole dorovnat a zbývajícího počtu žetonů hráče na tahu. Je zde opět zohledněna aktuální výše povinné sázky. Výsledek je číslo v intervalu $(0, 1)$. Pokud je tedy sázka potřebná pro dorovnání zanedbatelná vzhledem ke zbývajícím žetonům hráče na tahu, ostrá hodnota této proměnné se bude blížit k nule. V opačném případě se tato hodnota bude blížit k číslu jedna (použité funkce příslušnosti jsou vidět na obr. 3.10).

Výstupní proměnná „*action*“



Obrázek 3.11: Funkce příslušnosti výstupní proměnné „*action*“.

Výstup představuje akční zásah počítačového hráče. V programu se pak převede tato hodnota do přijatelné podoby, tj. do příslušné pokerové akce. Hráč buď složí karty (fold), dorovná sázku (call) nebo navýší sázku maximálně do výše svých žetonů (all-in). Obr. 3.11 znázorňuje škálu pokerových akcí.

Báze pravidel fuzzy systémů AggressivePlayer a ConservativePlayer

Báze pravidel obou fuzzy systémů se skládá z 53 pravidel. V tabulce 2 jsou uvedena pravidla fuzzy systémů ConservativePlayer a v tabulce 3 jsou popsána pravidla fuzzy systému AggressivePlayer.

4 Programátorská a uživatelská část

4.1 Programátorská část

K vývoji své aplikace jsem zvolil jazyk *Java* s použitím platformy *JavaFX* pro tvorbu grafického uživatelského rozhraní *(GUI)*. Java je objektově orientovaný jazyk a obsahuje značné množství knihoven usnadňující programování. K vývoji aplikací v Javě je nutné mít nainstalovanou sadu *Java Development Kit (JDK)* od společnosti *Oracle*. JDK obsahuje celou řadu nástrojů. Ke spouštění aplikací i vývojových nástrojů slouží běhové prostředí *Java Runtime Environment (JRE)*. Součástí balíčku je tzv. *virtuální stroj* a překladač zdrojového kódu Javy do mezikódu (tzv. *bytecode*). Sada JDK obsahuje také nástroje pro ladění programů, vytváření programové dokumentace a spoustu dalších.

V mé aplikaci je použito několik knihoven, které jsou součástí JDK 1.8 (např. knihovny pro vytváření animací, dialogových oken aj.). Při programování počítačového hráče jsem využil externí knihovnu *Fuzzy Logic Engine - JFuzzinator*. [11]

K vývoji mi posloužilo vývojové prostředí *NetBeans IDE 8.0.2*. Vytváření GUI mi usnadnil nástroj *JavaFX Scene Builder 2.0*. Výstup z tohoto programu je ukládán do souboru typu *FXML*, což je nástavba *XML* a pomocí *CSS* je pak možné ladit celkový vzhled.

4.1.1 Java - vlákna (*threads*) a balíčky (*package*)

Java umožňuje vícevláknové programování (*multitasking*), tj. provádění více úloh současně. Každá aplikace vytvořená v Jave disponuje hlavním vláknem (v případě JavaFX je to *JavaFX Application Thread*), které se stará o obsluhu GUI a všech událostí v něm. Při programování rozsáhlejších aplikací je užitečné oddělit logiku programu od grafického uživatelského rozhraní, tedy rozdělit program do více vláken. Pokud je na daném počítači procesor s více jádry, souběh vláken zajistí rychlejší vykonávání programu.

Java disponuje mimo jiné mechanismem pro seskupení souvisejících částí programu do tzv. balíčků (*package*). Je to užitečné k pojmenování kolekce tříd a hlavně k řízení přístupu. Data a metody třídy uvnitř jednoho balíčku mohou být soukromé a tedy z třídy v jiném balíčku nepřístupné. Tohoto principu jsem bohatě využil i ve své aplikaci.

4.1.2 Rozdělení aplikace *Texas Hold'em poker* do „*package*“

V následující části textu popíšu rozdělení programu v rámci balíčků a nejdůležitější třídy a jejich funkce.

Package „*gui*“

Balíček obsahuje třídy, které přímo pracují s GUI.

- Třída ***TexasHoldemController*** - definuje všechny obsluhy událostí v hlavním okně a všechny komponenty a kontejnery (grafické prvky) v hlavním okně (pomocí nástroje JavaFX Scene Builder 2.0 se ukládá výstup z výstavby GUI do souboru *TexasHoldem.fxml* a lazení vzhledu probíhá v souboru *TexasHoldemStyle.css*). O veškerou činnost týkající se úpravy GUI se stará hlavní vlákno programu *JavaFX Application Thread*.

- Třída ***TexasHoldem*** - zde běží hlavní metoda *main* (vstupní bod programu).

- Třída ***ShowScene*** - komunikuje s třídou *TexasHoldemController* pomocí tzv. *data bindingu*, což je obecná technika spojující logiku a uživatelské rozhraní. Zde jsou obsažena data, která jsou svázaná se všemi grafickými prvky v hlavním okně, tzn. jakákoliv změna se projeví i v rámci GUI. Třída definuje metody s návratovou hodnotou *Runnable*, pomocí nichž komunikuje s vedlejším vláknem programu (kde běží logická část), které vyžaduje okamžité překreslení GUI.

- Třídy ***SaveGameWindow*** a ***OpenGameWindow*** - v balíčku se nacházejí i třídy pro správu dialogových oken k uložení hry resp. načtení hry do souboru resp. ze souboru XML.

- Třída ***OptionsWindowController*** - tato třída spravuje okno nastavení (*Options*) podobným způsobem jako *TexasHoldemController* hlavní okno aplikace.

- Třída ***GameVariables*** - zpracovává hodnoty atributů hry z *Options* potřebné pro spuštění nové hry.

Package „*game*“

Zde jsou 2 třídy řídící logiku hry, kde běží vedlejší vlákno programu.

- Třída ***PokerManager*** - v této třídě jsou definované veškeré metody pro řízení hry. Obsahuje dva konstruktory přijímající parametry pro vytvoření nové hry a pro načtení hry ze souboru. Pomocí *runLater* je navazováno spojení s *JavaFX Application Thread* (metody v této třídě vyžadující překreslení GUI volají metody z třídy *ShowScene*).

- Třída ***GameRun*** - dědí všechna data a metody z třídy *PokerManager* a implementuje rozhraní *Runnable*. Když uživatel spustí novou nebo načte uloženou hru, v této třídě se vytvoří nové vlákno, v kterém běží logická část hry a spustí se hlavní smyčka hry v metodě *run*.

Package „*fuzzy*“

Obsahuje třídy definující fuzzy regulátor a pomocné třídy k získání akčního zásahu počítačového hráče.

- Třída ***AdaptedVariables*** - definuje metody, které transformují data z průběhu aktuální hry (z třídy *PokerManager*) do požadované podoby. Návratové hodnoty jsou pak ostré hodnoty vstupních proměnných jednotlivých fuzzy systémů.

- Třída ***Entry*** - vstupní brána do fuzzy systému. Je zde metoda, která předá transformovaná data z třídy *PokerManager* třídám *FIS_PcPlayer* a *FIS_OpponentsPower* a po průchodu inferenčním mechanismem vrátí akci počítačového hráče.

- Třída ***FIS_OpponentsPower*** - v této třídě je vytvořen pomocný fuzzy systém *OpponentsPower*, jehož výstup slouží jako vstupní proměnná hlavního fuzzy systému. S pomocí externí knihovny Javy *JFuzzinator* je vytvořena celá struktura FIS (funkce příslušnosti jednotlivých proměnných, inferenční mechanismus, báze pravidel).

- Třída ***FIS_PcPlayer*** - zde je vytvořen hlavní fuzzy systém *ConservativePlayer* resp. *AggressivePlayer* (liší se pouze bází pravidel) podobně jako *OpponentsPower*.

Package „*poker*“

V tomto balíčku jsou třídy charakterizující poker. Reprezentují hráče, kartu a herní stůl.

- Třída ***Board*** - reprezentuje hrací stůl. Data představují balíček karet, společné karty na stole a pot (v případě potřeby rozdělení banku obsahuje dílčí výhry). Je zde uložena i informace o tom, jaký hráč na jaký pot dosahuje (pokud se zúčastní vyhodnocení více hráčů, kteří vsadili všechny svoje žetony a každý má přirozeně jiný počet). Třída definuje metody pro rozdání karet hráčům, rozdání karet na stůl, míchání karet, přidávání sázek do potu, rozdělení potu apod.

- Třída ***Card*** - objekt této třídy reprezentuje jednu kartu z balíčku. Třída tedy obsahuje výčet padesáti dvou karet a data uchovávající informaci o hodnotě karty a barvě.

- Třída ***Player*** - data třídy obsahují informace o hráči: jaké drží karty na ruce, jméno, počet žetonů, kolik má vsazeno (celkově i v rámci jednoho kola sázení), jestli je člověk nebo počítač, druh hráče (konzervativní, agresivní), jestli složil karty.

Package „*evaluation*“

Součástí balíčku jsou 2 třídy, jejichž hlavním účelem je zjistit výherní kombinaci v aktuální hře, identifikovat vítězné karty, zjistit vítěze.

- Třída ***ValueCombination*** - ve třídě jsou metody pro rozpoznání jednotlivých pokerových výherních kombinací. Hodnota kombinace získaná z karet hráče i ze společných karet na stole je testována postupně jednotlivými metodami až do doby, než se najde ta správná. Návratovou hodnotou je pole čísel charakterizující výherní kombinaci.

- Třída ***WinningAttributes*** - třída má za úkol zjistit vítěze a vítězné karty. Data uchovávají hráče, kteří se zúčastní vyhodnocení. Obsahují informaci o tom, které z karet hráčů i ze společných karet jsou součástí výherní kombinace. Připravená data se pak předávají třídě *PokerManager*, která komunikuje s grafickým uživatelským rozhraním (ve hře musí být nějak graficky znázorněno, kdo vyhrál a jaké karty jsou vítězné). Obsahuje i metodu, která převádí vítěznou kombinaci do slovní podoby (historie výher je videtelná během aktuální hry stejně tak jako historie sázení).

Package „*animation*“

Jedná se o kolekci tříd, kde je využíváno animací z JavaFX. Jednotlivé třídy mají na starost práci s určitým typem animace, aby bylo docíleno efektivního vzhledu. Např. při vyhodnocení kombinací na konci hry, při zvýraznění vítězného hráče, při sázení nebo připsání výhry apod. (*FadeTransition, PathTransition, Timeline, RotateTransition*).

Package „*help*“

Zde jsou třídy spravující okno nápovědy (*HelpController*) a okno o aplikaci (*AboutWindowController*) podobně jako u hlavního okna nebo okna nastavení.

4.2 Uživatelská část

4.2.1 Spuštění a ovládání aplikace

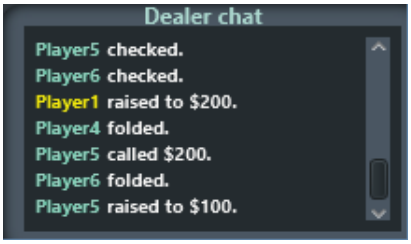
Samotná hra nevyžaduje instalaci, pouze se zkopírují potřebné soubory. Ke spuštění aplikace je nutné mít nainstalované běhové prostředí Javy *Java Runtime Environment (JRE)*, které většina operačních systémů stejně vyžaduje. Pokud je na daném počítači JRE nainstalováno, stačí spustit konkrétní soubor s příponou *jar*. V mém případě je to v kořenové složce soubor „*TexasHoldem.jar*“.

Ovládání hry je intuitivní. Položky menu lze vybrat pomocí myši i podle standardních klíčových kláves. Ve hře jsou pak tlačítka pro příslušnou akci *(CHECK/CALL, FOLD, RAISE)* a posuvník pro výběr výše sázky. Výběr lze provést myší nebo pomocí šipek, tabulátoru a mezerníku na klávesnici.

V pravém spodním rohu hlavního okna aplikace se během hry zobrazuje historie sázení aktuální hry (*Dealer chat*) viz obr. 4.1.

4.2.2 Položky menu *Game* a *Help*

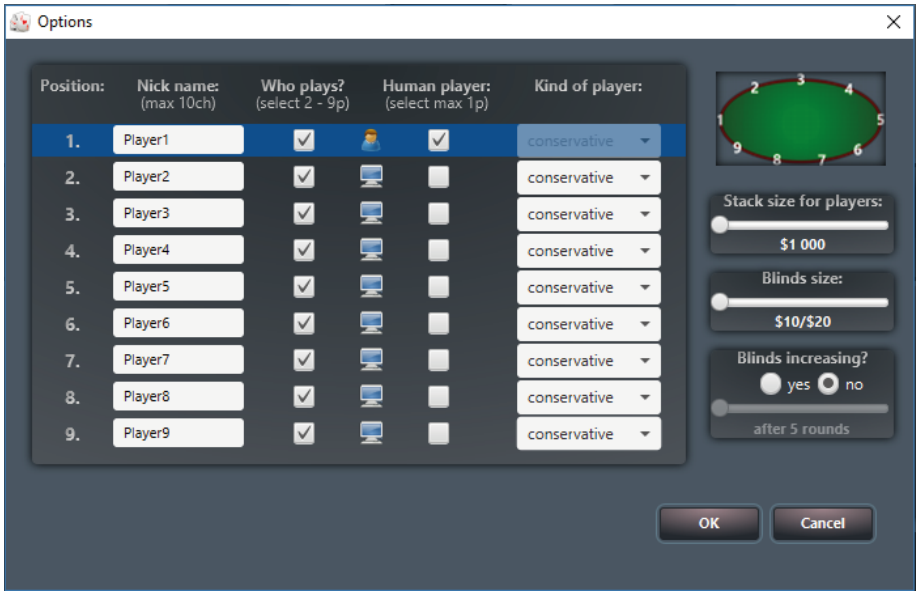
- ***New game*** - spustí novou hru s nastavenými atributy v *options*. Pokud položku vybereme v době, kdy je rozehraná jiná hra, zobrazí se dialogové



Obrázek 4.1: Historie sázení - *Dealer chat.*

okno s varováním přerušení hry, popř. s nabídkou akce uložení rozehrané hry.

- ***Open game*** - otevře dialogové okno, kde je možnost vybrat soubor s uloženou hrou ve formátu *XML*. V případě rozehrané hry se opět otevře dialog s nabídkou příslušné akce.
- ***Save game*** - otevře dialogové okno, kde lze zadat název souboru a po potvrzení uložit rozehranou hru pouze ve formátu *XML*. Hra se uloží do stavu před aktuálním rozdání karet. Při načtení uložené hry začne nové míchání a rozdají se znovu karty. Uložit rozehranou hru lze pouze v případě, když je na řadě lidský hráč. Pokud provádí nějakou akci počítačový hráč, položka *Save game* je zašedlá a není umožněn její výběr.
- ***Options*** - otevře dialogové okno, kde je možnost nastavení atributů hry (viz obr. 4.2). V levé části okna lze vybrat hráče (2-9), jejich jména, zvolit typ hráče (člověk, počítač) a u každého je možné zvolit druh hráče *(conservative, aggressive).*



Obrázek 4.2: Dialogové okno *Options.*

Pokud zvolíme menší počet hráčů než 9, je možný výběr místa u stolu (označení pozic je vidět v pravém horním rohu okna). Lidský hráč může být maximálně jeden nebo žádný, tj. probíhá souboj pouze počítačových hráčů. V pravé části okna lze vybrat výši *stacku* (počet žetonů), který budou mít na začátku hry všichni stejný. Dále lze zvolit výši *blindů* (povinných sázek) a po kolika kolech se mají blindy zvyšovat, popř. nezvyšovat vůbec.

- ***Exit*** - uzavře hlavní okno hry. Pokud je rozehraná hra, vyskočí varovné okno, popř. otázka s nabídkou příslušné akce.
- ***Texas Hold'em Poker Help*** - zobrazí okno nápovědy, kde je možné dozvědět se něco o Texas Hold'em pokeru (pravidla, jak jdou po sobě výherní kombinace). Dále nápověda obsahuje popis jednotlivých položek menu „*Game*“ a popis ovládání aplikace.
- ***About Texas Hold'em Poker*** - zobrazí okno s informacemi o aplikaci (autora, datum vytvoření apod.).

Závěr

První fáze mé bakalářské práce spočívala ve studiu fuzzy logiky a fuzzy regulace. Při vývoji vlastního fuzzy regulátoru, tj. počítačového hráče v pokeru, jsem čerpal pouze ze svých poznatků získaných právě tímto studiem.

Během studování problematiky fuzzy regulace jsem si uvědomil skutečnost, že při programování počítačového hráče nebudu mít možnost uvažovat zkušenosti získané pozorováním minulých her (jestli hráč často blafuje apod.). Vhodnějším způsobem pro vytvoření počítačového hráče v pokeru by asi bylo využití např. genetických algoritmů. Vývoj počítačového hráče jako fuzzy regulátoru spočíval v neustálém variování ostrých hodnot vstupních proměnných a sledování změny výstupu (akčního zásahu). Na základě tohoto pozorování jsem pak stanovil finální tvary a rozsahy funkcí příslušnosti jednotlivých proměnných a finální podobu báze pravidel. Můj počítačový hráč sice nerozezná blafujícího protihráče, ale dokáže reagovat na standardní pokerové situace. Uvažuje hodnotu vstupní karetní kombinace, zohledňuje pozici hráče u stolu, rozhoduje se podle vsazených žetonů (svých i protihráčů) apod.

Ve hře je možnost u počítačového hráče zvolit, jestli bude hráč konzervativní nebo agresivní (tzn. jak sází v určitých herních situacích). Zahrát si může člověk proti 1 - 8 počítačovým protivníkům nebo může jen sledovat, jak hra probíhá bez jeho účasti. Aplikace by se jistě dala rozšířit např. o nějaké jiné varianty pokeru nebo o počítání statistik výher apod.

V samotném závěru mohu směle říci, že zadání bakalářské práce jsem splnil. Řekl jsem něco obecně o fuzzy regulátorech a navrhl jsem svůj vlastní fuzzy regulátor implementovaný do vlastní aplikace „Texas Hold'em poker“. Hra je vhodná pro začátečníka, ale i zkušenější hráč si touto aplikací může zpříjemnit ne jedno odpoledne.

Conclusions

The first phase of my bachelor thesis consisted in the study of fuzzy logic and fuzzy regulation. When developing my own fuzzy controller, a computer poker player, I have drew only from of my knowledge gained from this study.

While studying fuzzy regulation I realized the fact that when programming a computer player, I will not be able to consider the experience gained from watching past games (whether a player often bluffs, etc.). A more appropriate way to create a computer poker player would be to use, for example, genetic algorithms. The development of a computer player as a fuzzy controller consisted of constantly varying the crisp values of variables and monitoring the change of output. Based on this observation, I then determined the final shapes and ranges of the membership functions variables and the final form of the rules base. My computer player does not recognize a bluffing opponent but can respond to standard poker situations. It considers the value of the incoming card combination, takes into account the position of the player at the table, decides on the stacked chips, etc.

In the game, it is possible for a computer player to choose whether the player is conservative or aggressive (i.e. betting in certain game situations). You can play against 1 - 8 computer opponents or just watch how the game takes place without his participation. The application would certainly be expanded, for example, by some other variations of poker or counting winning statistics, etc.

In the end I can say boldly that I have fulfilled assignment of bachelor thesis. I have said something about fuzzy controllers in general and have designed my own fuzzy controller implemented into my own application "Texas Hold'em poker". The game is good for a beginner, but even a more experienced player can spend more than one enjoyable afternoon playing it.

Literatura

[1] NOVÁK, Vilém. *Základy fuzzy modelování.* 1. vyd. Praha: BEN, 2000. 176 s. ISBN 80-7300-009-1.

[2] VYSOKÝ, Petr. *Fuzzy řízení.* 1. vyd. Praha: ČVUT Praha, 1997. 131 s. ISBN 80-01-01429-8.

[3] ZADEH, Lotfi A. *Fuzzy sets. Information and Control.* 1965, vol. 8, issue 3. Dostupné z: http://www.cs.berkeley.edu/∼zadeh/papers/Fuzzy%20Sets-Information%20and%20Control-1965.pdf

[4] JURA, Pavel. *Základy fuzzy logiky pro řízení a modelování.* 1. vyd. Brno: VUTIUM, 2003. 132 s. ISBN 80-214-2261-0.

[5] MODRLÁK, Osvald. *Fuzzy řízení a regulace – Teorie automatického řízení II.* Liberec: Technická univerzita v Liberci, 2002, 25 s. Skripta.

[6] TALAŠOVÁ, Jana. *Fuzzy metody vícekriteriálního hodnocení a rozhodování.* 1. vyd. Olomouc: Vydavatelství UP, 2003. 179 s. ISBN 80-244-0614-4.

[7] J & L SOLUTIONS LIMITED. *Pravidla pokeru Texas Hold'em.* [online]. © 2009 - 2017 [cit. 2017-01-28]. Dostupné z: http://www.pokerarena.cz/rubriky/poker/pravidla-pokeru-texas-holdem_5827.html

[8] MATHWORKS - Makers of MATLAB and Simulink. *MATLAB Speaks Math.* [online]. © 1994-2017. Dostupné z: https://www.mathworks.com/products/matlab/features.html#matlab-speaks-math

[9] MATHWORKS - Makers of MATLAB and Simulink. *Build Mamdani Systems Using Fuzzy Logic Designer.* [online]. © 1994-2017 [cit. 2014-04-29]. Dostupné z: http://www.mathworks.com/help/fuzzy/building-systems-with-fuzzy-logic-toolbox-software.html

[10] SIMPLYHOLDEM.com. *The Bill Chen Formula.* [online]. © 2004 - 2010. Dostupné z: http://www.simplyholdem.com/chen.html

[11] SLASHDOT MEDIA. *Fuzzy Logic Engine - JFuzzinator.* [online]. © 2017 [registered 2010-06-07]. Dostupné z: https://sourceforge.net/projects/jfuzzinator/

A Báze pravidel jednotlivých fuzzy systémů

playersInGame	possibleIncreasePlayers	opponentsAction	opponentsPower
little	little	smallBet	small
little	little	acceptableBet	acceptable
little	little	bigBet	medium
little	little	dangerousBet	medium
little	more	smallBet	acceptable
little	more	acceptableBet	acceptable
little	more	bigBet	medium
little	more	dangerousBet	big
little	very	smallBet	medium
little	very	acceptableBet	medium
little	very	bigBet	big
little	very	dangerousBet	big
more		smallBet	medium
more		acceptableBet	medium
more		bigBet	big
more		dangerousBet	dangerous
very		smallBet	big
very		acceptableBet	big
very		bigBet	dangerous
very		dangerousBet	dangerous

Tabulka 1: Báze pravidel fuzzy systému OpponentsPower.

valueCombination	opponentsPower	callRate	action
unplayable			check/fold
bad			check/fold
good	small	small	smallBet
good	small	acceptable	smallBet
good	small	big	call
good	small	dangerous	call
good	acceptable	small	smallBet
good	acceptable	acceptable	call
good	acceptable	big	call
good	acceptable	dangerous	call
good	medium	small	call
good	medium	acceptable	call
good	medium	big	check/fold
good	medium	dangerous	check/fold
good	big		check/fold
good	dangerous		check/fold
better	small	small	smallBet
better	small	acceptable	smallBet
better	small	big	call
better	small	dangerous	call
better	acceptable	small	smallBet
better	acceptable	acceptable	smallBet
better	acceptable	big	smallBet
better	acceptable	dangerous	call
better	medium	small	smallBet
better	medium	acceptable	smallBet
better	medium	big	call
better	medium	dangerous	call
better	big	small	call
better	big	acceptable	check/fold
better	big	big	check/fold
better	big	dangerous	check/fold
better	dangerous		check/fold
best	small	small	smallBet
best	small	acceptable	smallBet
best	small	big	smallBet
best	small	dangerous	call
best	acceptable	small	smallBet
best	acceptable	acceptable	smallBet
best	acceptable	big	smallBet
best	acceptable	dangerous	call
best	medium	small	mediumBet
best	medium	acceptable	smallBet
best	medium	big	call
best	medium	dangerous	call
best	big	small	smallBet
best	big	acceptable	call
best	big	big	check/fold
best	big	dangerous	check/fold
best	dangerous	small	call
best	dangerous	acceptable	check/fold
best	dangerous	big	check/fold
best	dangerous	dangerous	check/fold

Tabulka 2: Báze pravidel fuzzy systému ConservativePlayer.

valueCombination	opponentsPower	callRate	action
unplayable			check/fold
bad			check/fold
good	small	small	smallBet
good	small	acceptable	mediumBet
good	small	big	all-in
good	small	dangerous	all-in
good	acceptable	small	smallBet
good	acceptable	acceptable	mediumBet
good	acceptable	big	check/fold
good	acceptable	dangerous	check/fold
good	medium	small	smallBet
good	medium	acceptable	check/fold
good	medium	big	check/fold
good	medium	dangerous	check/fold
good	big		check/fold
good	dangerous		check/fold
better	small	small	smallBet
better	small	acceptable	mediumBet
better	small	big	all-in
better	small	dangerous	all-in
better	acceptable	small	smallBet
better	acceptable	acceptable	mediumBet
better	acceptable	big	all-in
better	acceptable	dangerous	check/fold
better	medium	small	smallBet
better	medium	acceptable	mediumBet
better	medium	big	check/fold
better	medium	dangerous	check/fold
better	big	small	smallBet
better	big	acceptable	check/fold
better	big	big	check/fold
better	big	dangerous	check/fold
better	dangerous		check/fold
best	small	small	smallBet
best	small	acceptable	mediumBet
best	small	big	all-in
best	small	dangerous	all-in
best	acceptable	small	smallBet
best	acceptable	acceptable	mediumBet
best	acceptable	big	all-in
best	acceptable	dangerous	all-in
best	medium	small	smallBet
best	medium	acceptable	call
best	medium	big	all-in
best	medium	dangerous	all-in
best	big	small	smallBet
best	big	acceptable	check/fold
best	big	big	check/fold
best	big	dangerous	check/fold
best	dangerous		check/fold

Tabulka 3: Báze pravidel fuzzy systému AggressivePlayer.

B Obsah přiloženého DVD

bin/

Aplikace Texas Hold'em poker, spustitelná přímo z DVD souborem TexasHoldem.jar v tomto adresáři. Adresář obsahuje i všechny knihovny a další soubory potřebné pro bezproblémový běh aplikace z DVD.

doc/

Dokumentace práce ve formátu PDF, vytvořená s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF souboru dokumentace (v ZIP archivu), tj. zdrojový text dokumentace, vložené obrázky, apod.

src/

Kompletní zdrojové texty aplikace Texas Hold'em poker se všemi potřebnými zdrojovými texty, knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření spustitelné verze aplikace.

readme.txt

Instrukce pro spuštění aplikace Texas Hold'em poker, včetně všech požadavků pro její bezproblémový provoz.