5. TOOLBOX-ul "FUZZY LOGIC" [1]

5.1. BAZELE LOGICII FUZZY

5.1.1. FUNCTII DE APARTENENTA

function y = trimf(x, params)

```
x = (0:0.2:10)';
y1 = trimf(x, [3 4 5]);
y2 = trimf(x, [1 4 9]);
subplot(211), plot(x, [y1 y2]);
y1 = trimf(x, [2 3 5]);
y2 = trimf(x, [4 5 9]);
subplot(212), plot(x, [y1 y2]);
```

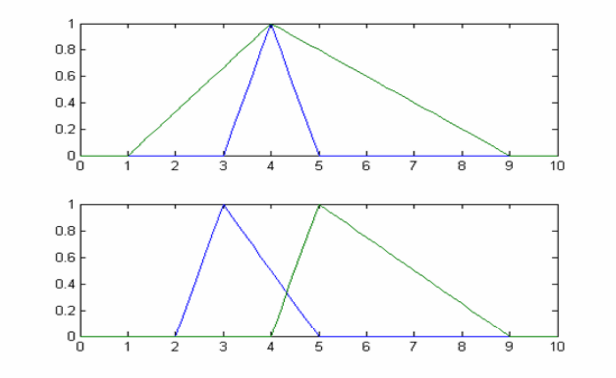


Fig. 5.1.1

function y = trapmf(x, params)

```
x = (0:0.1:10)';
y1 = trapmf(x, [2 3 7 9]);
y2 = trapmf(x, [4 5 5 7]);
y3 = trapmf(x, [5 6 4 6]);
plot(x, [y1 y2 y3]);
```

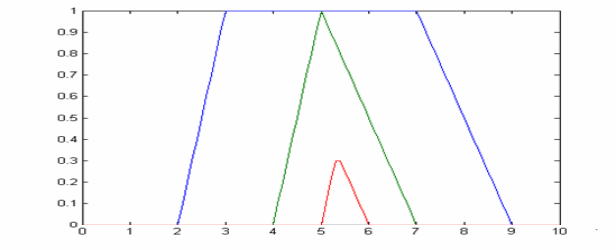


Fig. 5.1.2

Trapezul exterior este y_1, iar cel interior – y_3.

function y = gaussmf(x, params)

GAUSSMF este functia de apartenenta de forma curbei Gauss.

GAUSSMF(X, PARAMS) returneaza o matrice care este functia de apartenenta Gauss evaluata in X. PARAMS este un vector cu 2 elemente care determina forma si pozitia functiei de apartenenta.

Formula acestei functii de apartenenta este:

GAUSSMF(X, [SIGMA, C]) =
EXP(-(X - C).^2/(2*SIGMA^2));

adica $g(x) = e^{-\frac{(x-c)^2}{2\sigma^2}}$

De exemplu:

```
x = (0:0.1:10)';
y1 = gaussmf(x, [0.5 5]);
y2 = gaussmf(x, [1 5]);
y3 = gaussmf(x, [2 5]);
y4 = gaussmf(x, [3 5]);
subplot(211); plot(x, [y1 y2 y3 y4]);
y1 = gaussmf(x, [1 2]);
y2 = gaussmf(x, [1 4]);
y3 = gaussmf(x, [1 6]);
y4 = gaussmf(x, [1 8]);
subplot(212); plot(x, [y1 y2 y3 y4]);
```

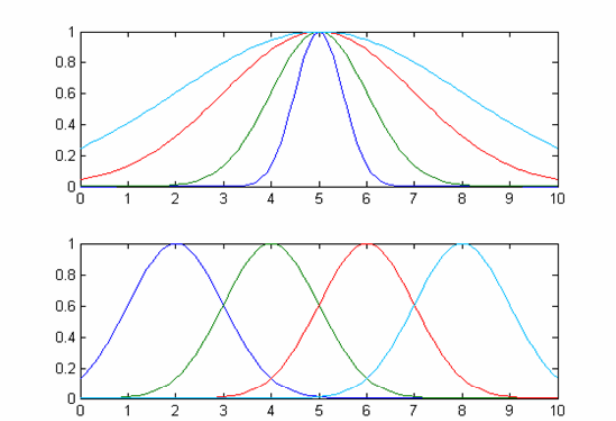


Fig. 5.1.3

In prima fig. clopotul din centru este y_1, iar cel exterior – y_4.

σ este distanta pe orizontala de la punctul de intersectie cu orizontala 0.5 pana la abscisa maximului.

function y = gauss2mf(x, params)

Functia de apartenenta de forma *doua ramuri Gauss*.

Formula: y = gauss2mf(x,params);
y = gauss2mf(x,[sig1 c1 sig2 c2])

Descriere: Functia Gauss depinde de doi parametri *sig* si *c*.

gauss2mf este o combinatie a doua functii Gauss. Prima functie, specificata de *sig1* si *c1*, determina forma ramurii stangi a curbei; a doua functie determina forma ramurii din dreapta a curbei. Daca c1<c2, functia *gauss2mf* are a valoarea maxima egala cu 1. In caz contrar, valoarea maxima este mai mica decat 1. Parametrii se listeaza in ordinea: [sig1, c1, sig2, c2]. Exemple:

```
x = (0:0.1:10)';
y1 = gauss2mf(x, [2 4 1 8]);
y2 = gauss2mf(x, [2 5 1 7]);
y3 = gauss2mf(x, [2 6 1 6]);
y4 = gauss2mf(x, [2 7 1 5]);
y5 = gauss2mf(x, [2 8 1 4]);
plot(x, [y1 y2 y3 y4 y5]);
```

Curba exterioara este y_1, iar cea interioara – y_5.

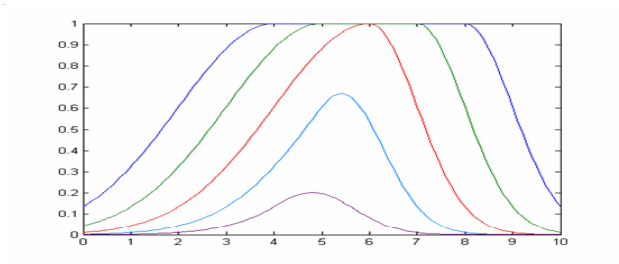


Fig. 5.1.4

function y = gbellmf(x, params)

GBELLMF Functia de apartenenta de forma curbei clopot (bell) generalizata.

GBELLMF(X, PARAMS) returneaza o matrice care este functia de apartenenta clopot generalizata evaluata la X. PARAMS este un vector cu 3 elemente care determina forma si pozitia acestei functii de apartenenta.

Formula:

GBELLMF(X, [A, B, C]) =
1./(1+ABS((X-C)/A))^(2*B);

adica $f(x) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}$

Se observa ca aceasta functie de apartenenta este o extensie a functiei de distributie de probabilitate Cauchy. De exemplu:

```
x = (0:0.1:10)';
y1 = gbellmf(x, [1 2 5]);
y2 = gbellmf(x, [2 4 5]);
y3 = gbellmf(x, [3 6 5]);
y4 = gbellmf(x, [4 8 5]);
subplot(211); plot(x, [y1 y2 y3 y4]);
y1 = gbellmf(x, [2 1 5]);
y2 = gbellmf(x, [2 2 5]);
y3 = gbellmf(x, [2 4 5]);
y4 = gbellmf(x, [2 8 5]);
subplot(212); plot(x, [y1 y2 y3 y4]);
```

In prima fig. clopotul din centru este y_1, iar cel exterior – y_4, iar in a doua fig. curba din centru sus este y_1, iar cea exterioara – y_4.

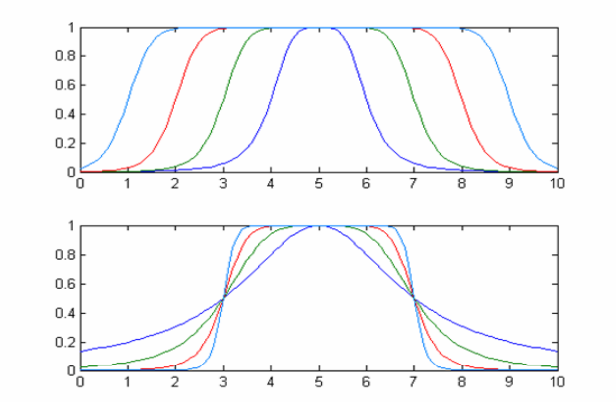


Fig. 5.1.5

A = a este distanta pe orizontala de la punctul de intersectie cu orizontala 0.5 pana la abscisa maximului.

function y = sigmf(x, params)

SIGMF Functia de apartenenta de forma curbei sigmoid.

SIGMF(X, PARAMS) returneaza o matrice care este functia de apartenenta sigmoid evaluate la X. PARAMS este un vector cu 2 elemente care determina forma si pozitia acestei functii de apartenenta.

Formula: SIGMF(X, [A, C]) =
1./(1 + EXP(-A*(X-C)))

adica $s(x) = \frac{1}{1+e^{-a(x-c)}}$

De exemplu (fig. 5.1.6):

```
x = (0:0.2:10)';
y1 = sigmf(x, [-1 5]);
y2 = sigmf(x, [-3 5]);
y3 = sigmf(x, [4 5]);
y4 = sigmf(x, [8 5]);
subplot(211); plot(x, [y1 y2 y3 y4]);
y1 = sigmf(x, [5 2]);
y2 = sigmf(x, [5 4]);
y3 = sigmf(x, [5 6]);
y4 = sigmf(x, [5 8]);
subplot(212); plot(x, [y1 y2 y3 y4]);
```

In prima fig. curba din stg. sus este y_1, apoi sunt y_2, y_4 si y_3.

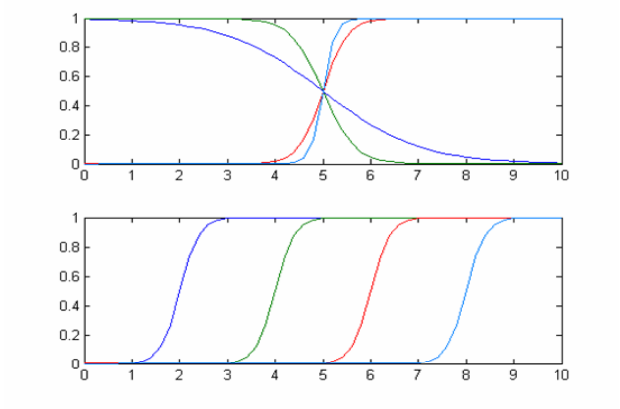


Fig. 5.1.6

function y = dsigmf(x, params)

DSIGMF Functia de apartenenta formata prin diferenta dintre doua functii de apartenenta sigmoidale.

Formula: y = dsigmf(x,[a1 c1 a2 c2])

Descriere: Functia de apartenenta sigmoidala utilizata aici depinde de doi parametrii *a* si *c*:

f(x;a,c) = 1/(1 + exp(-a(x-c))).

Functia de apartenenta *dsigmf* depinde de patru parametrii (a1, c1, a2 si c2) si este diferenta dintre doua functii sigmoidale:

f1(x; a1, c1) - f2(x; a2, c2).

Parametrii se listeaza in ordinea: [a1 c1 a2 c2]. Exemplu:

```
x=0:0.1:10;
y=dsigmf(x,[5 2 5 7]);
plot(x,y)
xlabel('dsigmf, P=[5 2 5 7]')
```

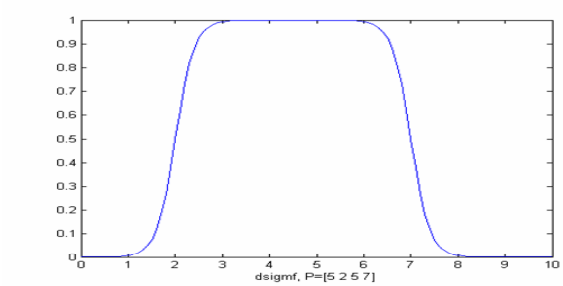


Fig. 5.1.7

function y = psigmf(x, params)

PSIGMF Produsul a doua doua functii de apartenenta sigmoidale.

PSIGMF(X, PARAMS) returneaza o matrice Y care este produsul a doua doua functii sigmoid evaluate la X. PARAMS este un vector cu 4 elemente care determina forma si pozitia acestei functii de apartenenta.

Formula: PSIGMF(X, PARAMS) =

= SIGMF(X, PARAMS(1:2)).*SIGMF(X, PARAMS(3:4));

De exemplu:

```
x = (0:0.2:10)';
params1 = [2 3];
y1 = sigmf(x, params1);
params2 = [-5 8];
y2 = sigmf(x, params2);
y3 = psigmf(x, [params1 params2]);
subplot(211);
plot(x, y1, x, y2); title('sigmf');
subplot(212);
plot(x, y3, 'g-', x, y3, 'o'); title('psigmf');
```

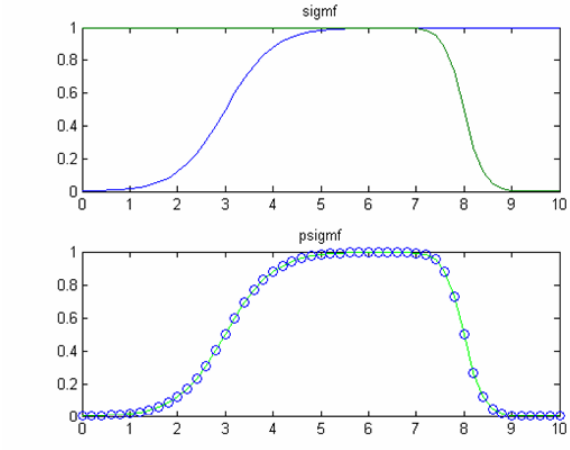


Fig. 5.1.8

function y = zmf(x, params)

ZMF Functia de apartenenta in forma de Z.

ZMF(X, PARAMS) returneaza o matrice care este functia de apartenenta Z evaluata la X. PARAMS = [X1 X0] este un vector cu 2 elemente care determina punctele de racordare ale acestei functii de apartenenta.

Daca X1 < X0, ZMF prezinta o trecere lenta de la 1 (in X1) la 0 (in X0).

Daca X1 >= X0, ZMF devine o functie treapta inversa, care sare de la 1 la 0 in (X0+X1)/2.

De exemplu:

```
x = 0:0.1:10;
subplot(311); plot(x, zmf(x, [2 8]));
subplot(312); plot(x, zmf(x, [4 6]));
subplot(313); plot(x, zmf(x, [6 4]));
```

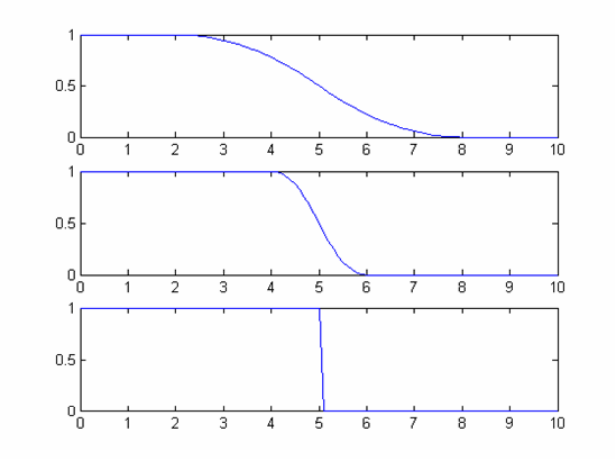


Fig. 5.1.9

function y = pimf(x, params)

PIMF Functia de apartenenta de forma lui π.

PIMF(X, PARAMS) returneaza o matrice care este functia de apartenenta π evaluata in X. PARAMS = [A B C D] este un vector cu 4 elemente care determina punctele de racordare ale acestei functii de apartenenta. Parametrii *a* and *d* indica "picioarele" curbei, iar *b* si *c* indica "umerii" ei. In fond, aceasta functie de apartenenta este produsul lui SMF (vezi mai jos) si ZMF:

PIMF(X, PARAMS) = SMF(X, PARAMS(1:2)).*ZMF(X, PARAMS(3:4))

Se observa ca aceasta functie de apartenenta π poate fi asimetrica, pentru ca are patru parametrii, spre deosebire de functia de apartenenta conventional π, care are numai doi parametrii.

De exemplu:

```
x = (0:0.1:10)';
y1 = pimf(x, [1 4 9 10]);
y2 = pimf(x, [2 5 8 9]);
y3 = pimf(x, [3 6 7 8]);
y4 = pimf(x, [4 7 6 7]);
y5 = pimf(x, [5 8 5 6]);
plot(x, [y1 y2 y3 y4 y5]);
```

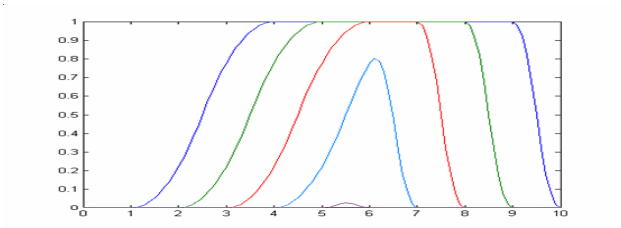


Fig. 5.1.10

Curba exterioara este y_1, iar cea interioara – y_5.

function y = smf(x, params)

SMF Functia de apartenenta in forma de S.

SMF(X, PARAMS) returneaza o matrice care este functia de apartenenta S evaluata la X. PARAMS = [X0 X1] este un vector cu 2 elemente care determina punctele de racordare ale acestei functii de apartenenta. Daca X0 < X1, SMF prezinta o trecere lenta de la 0 (in X0) la 1 (in X1).

Daca X0 >= X1, SMF devine o functie treapta, care sare de la 0 la 1 in (X0+X1)/2.

De exemplu:

```
x = 0:0.1:10;
subplot(311); plot(x, smf(x, [2 8]));
subplot(312); plot(x, smf(x, [4 6]));
subplot(313); plot(x, smf(x, [6 4]));
```

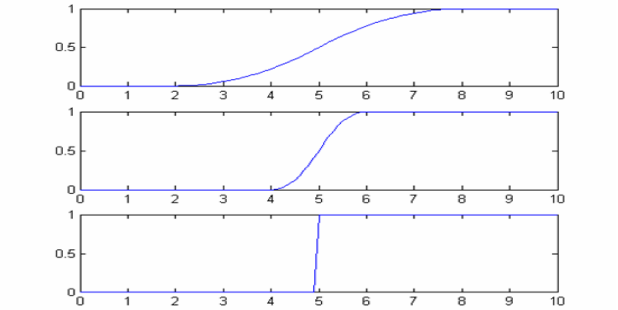


Fig. 5.1.11

5.1.2.- REGULI FUZZY *IF-THEN*
(daca-atunci)

O regula fuzzy *if-then* simpla este:

If x is A *then* y is B

unde A si B sunt valori lingvistice, definite pe multimi fuzzy considerate pe universurile de discurs X si - respectiv – Y. Partea "if" se numeste *antecedent* (premiza), iar "then" – *consecinta* (concluzie).

Ex.: "*If* temp. este moderata, *then* ventilatorul trebuie sa aiba viteza mica".

Facem observatia ca *moderat* se reprezinta ca un numar intre 0 si 1, ce este returnat de catre antecedent. *Viteza mica* este reprezentata de o multime fuzzy B, atribuita variabilei de iesire y. In regula fuzzy *if-then*, cuvantul *is* este interpretat diferit, dupa cum apare in antecedent sau in consecinta. In limbajul MATLAB aceasta distinctie se face intre testul relational "= =" (identic) si asignarea variabilei cu simbolul "=". Deci un mod mai exact de notare a regulii de mai sus este:

"*If* temp. = = moderata,
then vit. ventilatorului = mica".

La o regula fuzzy if-then, in general, intrarea este valoarea curenta a variabilei de intrare (temperatura), iar iesirea este o multime fuzzy, partea "consecinta" indicand o multime fuzzy ("mic") ce se atribuie iesirii.

In cazul logicii booleene (crisp), regula if-then se aplica usor: daca antecedentul este adevarat, atunci si consecinta este adevarata.

In cazul logicii fuzzy, daca antecedentul este adevarat **intr-un anumit grad**, atunci si consecinta este adevarata **in acelasi grad**.

Functia *implication* modifica multimea fuzzy atribuita iesirii cu gradul specificat de antecedent. Cel mai adesea aceasta modificare se face prin trunchiere, cu functia *min*, multimea fuzzy fiind "retezata", cum se va vedea mai jos.

Modul de lucru al acestei reguli se poate vedea pe graficele:

"Temp. moderata"-fig. 5.1.12

```
tempS=12:30;
modtempG=triangle(tempS,[14 21 28]);
```

si "Viteze mici"-fig. 5.1.13

```
vitezeS=0:.1:4;
vitmicG=triangle(vitezeS,[0 0 3]);
```

Pentru temperatura de 17^{o} fig. 5.1.12 ne arata

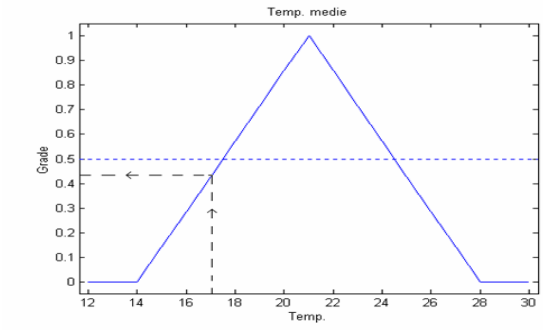


Fig. 5.1.12

ca gradul de adevar al afirmatiei "temperatura de 17^{o} este moderata" este de *43 %*. Aceasta inseamna ca si consecinta este adevarata in acelasi procentaj, multimea fuzzy "Viteze mici" trebuind sa fie ***retezata*** cu gradul antecedent *0.43* (fig. 5.1.14).

Retezarea limiteaza gradul de adevar al

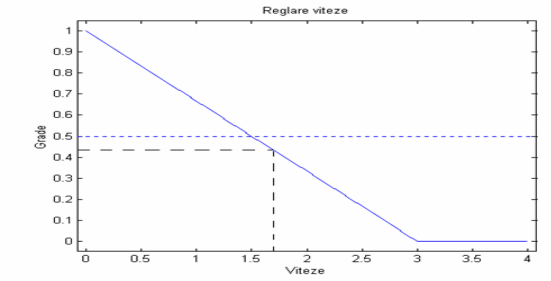


Fig. 5.1.13

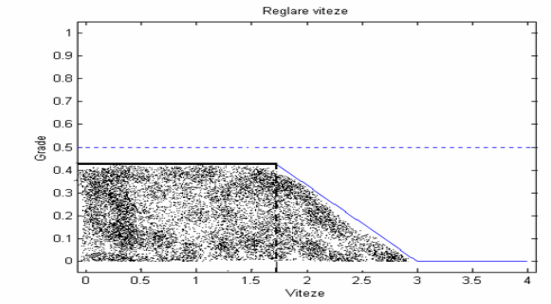


Fig. 5.1.14

consecintei la acelasi grad al antecedentului. Cu alte cuvinte, consecinta este adevarata in aceeasi masura ca si antecedentul.

Fig. 5.1.14 ne spune ca ventilatorul trebuie pus la pozitia *1.7*, sau mai mica, dar gradul de precizie al acestei indicatii este mediu (*43 %*). Pentru temperatura de 21^o procentajul de mai sus devine *100%* (fig. 5.1.13 nu se reteaza, iar ventilatorul este la pozitia 0)

Reguli *if-then* complexe

Antecedentul, sau consecinta, poate fi o expresie referitoare la mai multe variabile fuzzy. De exemplu, regula:

"*If* temperatura este moderata *or* puterea este slaba, *then* debitul combustibilului in cuptor trebuie sa fie scazut".

Dupa definirea multimilor fuzzy "Putere slaba" si "Debit scazut"(= fig. 5.1.13), pentru valori precizate privind temperatura si puterea se determina gradele fuzzy corespunzatoare si se ia maximul lor, acesta fiind gradul antecedentului, cu care se reteaza graficul din fig. 5.1.13.

Reguli *if-then* multiple

Daca ambele grade din antecedentul regulii de mai sus sunt nule, atunci si gradul antecedentului este nul si regula nu se poate aplica. Iesirea din acest impas o reprezinta formularea mai multor reguli, pentru a caracteriza iesirea din sistem pentru toate intrarile posibile.

La un sistem cu mai multe iesiri, fiecarei iesiri i se va atribui propriul set de reguli.

CONCLUZII

Interpretarea regulilor if-then este un proces compus din 3 parti:

1- *Fuzificarea intrarilor*: se rezolva toate propozitiile fuzzy din antecedent, rezultand un grad de apartenenta intre 0 si 1.

2- *Aplicarea operatorilor fuzzy pentru antecedente cu mai multe parti*: rezulta un singur numar, intre 0 si 1.

3- *Aplicarea metodei "implication"*: Multimea fuzzy de la iesire se formeaza utilizeaza gradul suportului de mai sus. Daca antecedentul este numai partial adevarat (adica are o valoare < 1), atunci multimea fuzzy de la iesire se reteaza.

In cazul mai multor reguli, multimea fuzzy corespunzatoare fiecarei iesiri sunt *agregate* si, in final, multimea fuzzy rezultanta este *defuzificata*.

5.2.- FUZZY INFERENCE SYSTEMS (**FIS**)

Inferenta fuzzy este procesul prelucrarii unei intrari date, utilizand logica fuzzy, pentru a obtine iesirea. Acest proces cuprinde toate elementele descrise mai sus: functia de apartenenta, operatori logici fuzzy, reguli if-then.

Cel mai utilizat tip de "fuzzy inference system" este Mamdani. Metoda de inferenta fuzzy a lui Mamdani a fost propusa in 1975, cu scopul de a controla un sistem tehnic prin sintetizarea unui set de reguli lingvistice de control, obtinute din experienta unor operatori umani.

5.2.1.- SISTEMUL *JOB*

Consideram un sistem (avand acest titlu), cu doua intrari si o iesire, prezentat in diagrama din fig. 5.2.1.

Natura paralela a regulilor este o caracteristica importanta a logicii fuzzy, ce permite glisarea simpla dintr-o regiune unde comportarea sistemului este dominata de una dintre reguli, catre alta regiune.

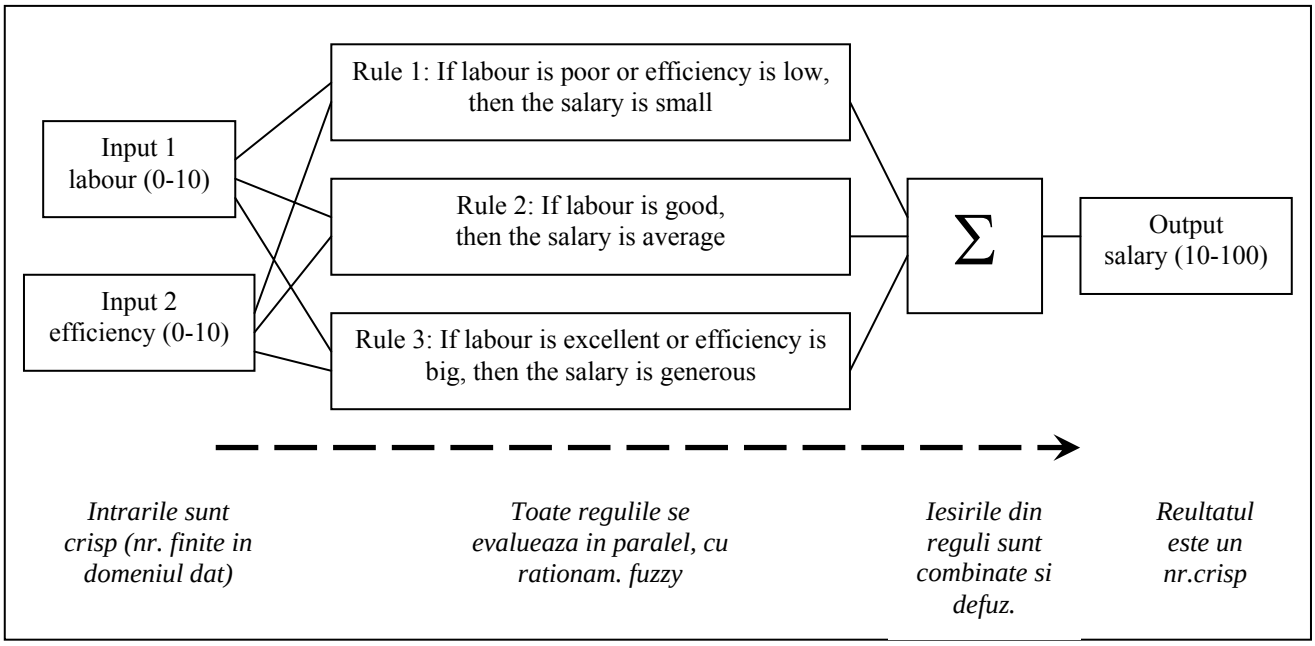
Procesul de inferenta fuzzy are 5 etape: fuzificarea variabilelor de intrare, aplicarea operatorilor fuzzy in antecedent, implicarea de la antecedent la consecinta, agregarea consecintelor si defuzificarea.

Etapa 1. Fuzificarea intrarilor

Se determina gradul in care intrarile apartin multimilor fuzzy corespunzatoare, prin

functia de apartenenta.
De exemplu, in ce masura *efficiency* este *big* (fig. 5.2.1)? In fig. 5.2.2 se prezinta cum *efficiency* (apreciata pe o scara de la 0 la 10) este caracterizata (prin functia ei de apartenenta) ca variabila lingvistica *big*. Considerand ca gradul de eficienta = 8, deducem ca ea este mare in proportie de 70%.
Etapa 2. Aplicarea operatorilor fuzzy
Acum stim gradul in care fiecare parte a antecedentului este satisfacuta, pentru fiecare

JOB-ul



Sistem cu 2 intrari, o iesire si 3 reguli
Fig. 5.2.1

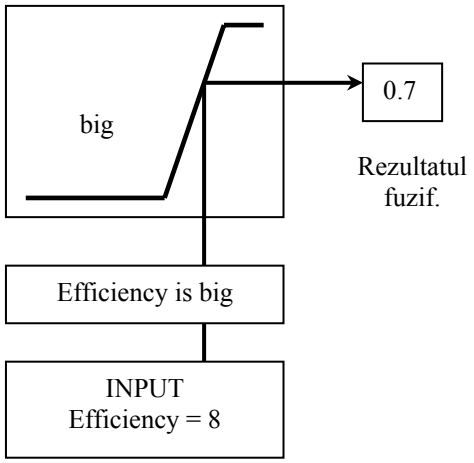


Fig. 5.2.2

regula . Daca antecedentul unei reguli are mai multe parti, se aplica *operatorii fuzzy* pentru a obtine un numar care reprezinta rezultatul antecedentului pentru aceea regula. La un operator fuzzy intrarea contine doua sau mai multe grade de apartenenta provenite din fuzificarea intrarilor, iar iesirea este un numar unic.
In fig. 5.2.3 este un exemplu de aplicare a operatorului OR (max) la calculul antecedentului regulii 3 din fig. 5.2.1.
Etapa 3. Aplicarea metodei *implication* (fig. 5.2.4)
Mai intai trebuie sa avem in vedere *ponderea regulii* (un numar subunitar) cu care se inmulteste numarul determinat de antecedent. Uzual, aceasta pondere = 1.

Multimea fuzzy din consecinta este acum prelucrata utilizand o functie asociata antecedentului.
Intrarea pentru *implication* este un numar dat de catre antecedent, iar iesirea este o multime fuzzy.
Etapa 4. Agregarea tuturor iesirilor
Agregarea este procesul prin care multimile fuzzy, reprezentand iesirile pentru fiecare regula,

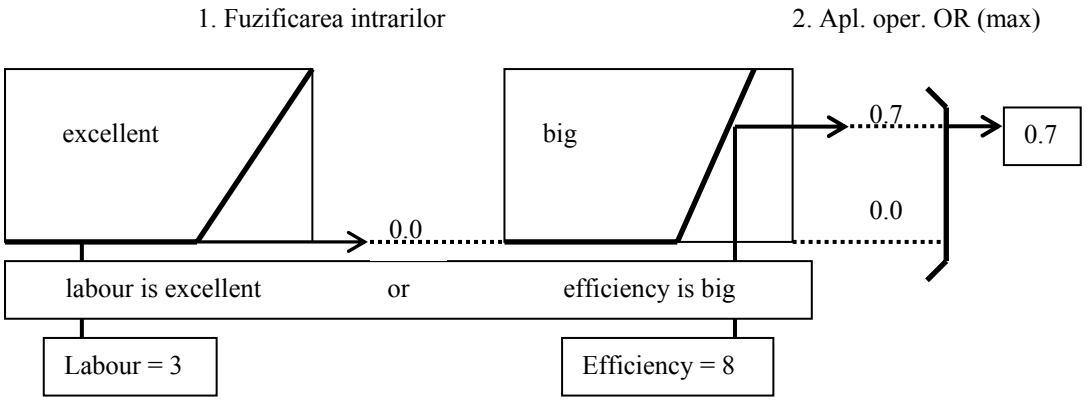


Fig. 5.2.3

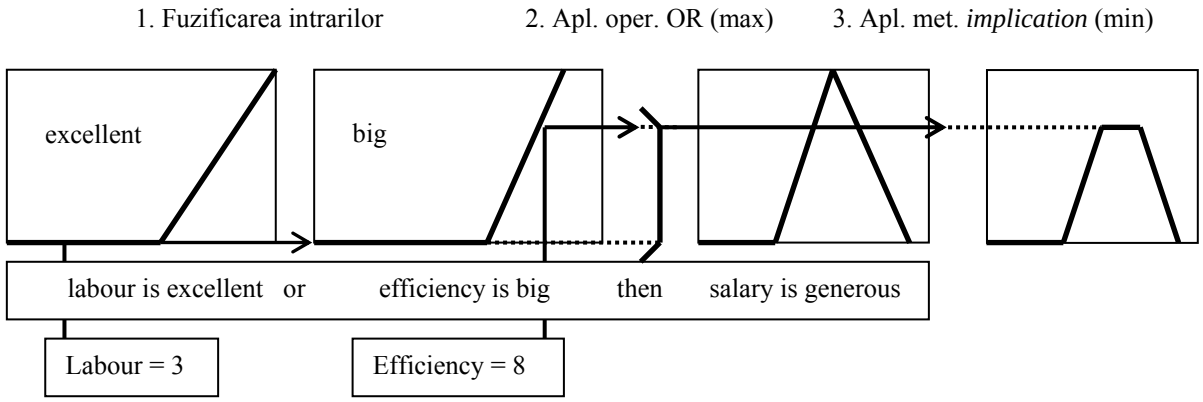
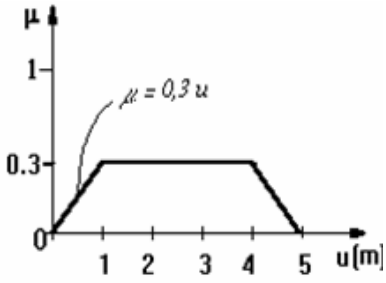
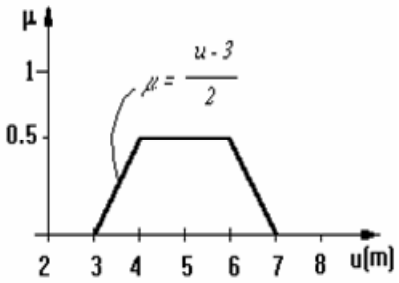


Fig. 5.2.4

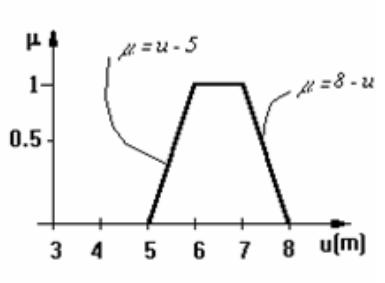
sunt combinate intr-o singura multime fuzzy.
Intrarea pentru procesul de agregare este lista iesirilor (retezate) furnizate de metoda *implication* pentru fiecare regula, iar iesirea este o multime fuzzy pentru fiecare variabila de iesire.
Cea mai utilizata este metoda *max* (maximului).
Consideram ca iesirile din cele 3 reguli din



a)



b)



c)

Fig. 5.2.5

fig. 5.2.1sunt – respectiv - multimile fuzzy retezate din fig. 5.2.5. Suprapunand aceste 3 figuri si luand la fiecare abscisa ordonata maxima, rezulta graficul din fig. 5.2.6

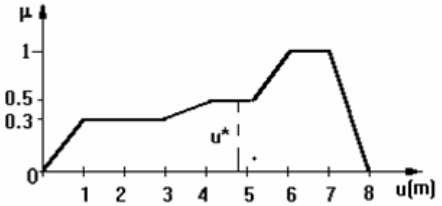


Fig. 5.2.6

Etapa 5. Defuzificarea
Intrarea in procesul de defuzificare este multimea fuzzy obtinuta prin agregarea de mai sus, iar iesirea este un numar (obtinut – de regula – cu functia *centroid*).

5.2.2.- DIAGRAMA DE INFERENTA FUZZY

Aceasta diagrama se obtine prin alaturarea figurii 2.4 (repetata – cu modificarile corespunzatoare - pentru fiecare regula din fig. 2.1) cu figurile 5.2.5,6.
Modulul "Rule Viewer" (descris mai jos) reprezinta implementarea aceastei diagrame.

5.3.- REALIZAREA FIS-ULUI

In fuzzy logic toolbox (F-L-T) sunt 5 unelte GUI (graphical user interface):

- **FIS-E** (the Fuzzy Inference System Editor);
- **MF-E** (the Membership Function Editor);
- **R-E** (the Rule Editor);
- **R-V** (the Rule Viewer);
- **S-V** (the Surface Viewer).

FIS-E nu are un numar limitat de intrari, dar acest numar trebuie limitat pentru memorie.
MF-E defineste formele functiilor de apartenenta asociate fiecarei intrari.
R-E editeaza toate regulile ce definesc comportarea sistemului.
R-V si S-V vizualizeaza FIS. R-V poate arata care reguli sunt active, sau ce forme pentru MF influenteaza rezultatul. S-V arata dependenta unei iesiri in functie de una sau doua intrari.

5.3.1.- JOB-ul

Se pune problema : "Dandu-se un numar intre 1 si 10 care reprezinta calitatea *of labour* la servici (10 fiind "excellent") si un alt numar intre 1 si 10 care reprezinta calitatea *of the efficiency* la acel job (10 fiind "big"), cat trebuie sa fie *the salary* ?"
Rezolvarea incepe cu formularea unor reguli bazate pe experienta:
1- If labour is poor or the efficiency is low, then the salary is small.
2- If labour is good, then the salary is average.

5.3.2.- FIS-E

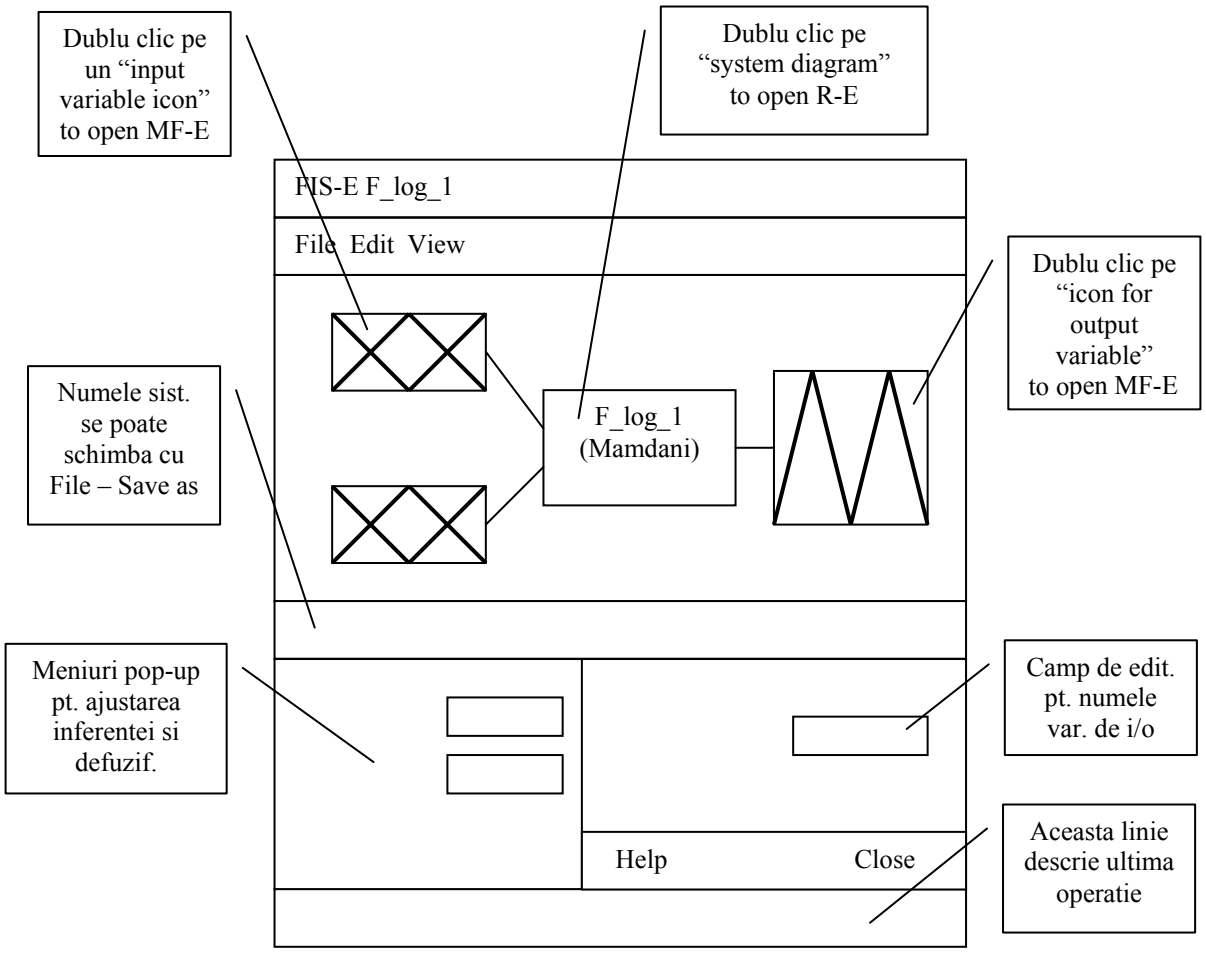


Fig. 5.3.1

3- If labour is excellent or the efficiency is big, then the salary is generous.

Admitem ca un salariu "average" este 50%, unul "generous"- 100%, iar unul "small"- 10%.

Vrem sa construim un nou FIS; se econom. timp luand unul deja construit, tastand:

>>fuzzy tipper.

Se va incarca un FIS asociat cu fisierul *tipper.fis* (fig. 5.3.1). Sus stanga sunt cele doua variabile de intrare, iar la dreapta- variabila de iesire; in centru este numele sistemului si tipul de inferenta (implicit – Mamdani; mai este tipul Sugeno, explicat in alt capitol). In stanga jos sunt meniuri derulante ce permit modificarea diferitelor componente ale procesului de inferenta, iar in dreapta jos: numele variabilei de i/o si tipul MF.

Pentru pornirea de la zero, se tasteaza >>*fuzzy* (la promptul MATLAB). Se deschide FIS-E, cu "input 1" si "output 1". Vom construi un sistem cu 2 intrari si o iesire: Edit – Add input → apare un careu galben "input 2"; schimbare numelor: clic in careul "input 1" (conturul devine rosu) – In campul de editare din dreapta jos se scrie "labour"- En; analog pentru "input 2" → "efficiency" si pentru careul din dreapta (bleu) "output 1" → "salary"- File – Save to workspace as…- In campul de editare al casetei ce se deschide se tasteaza: F_log_1 – OK.

Trecem la definirea functiei de apartenenta asociate fiecarei variabile, cu MF-E, ce se poate deschide in trei moduri:

- View – Edit MF;
- Dublu clic pe careul "salary";
- La prompt se scrie ">>mfedit". Apare figura 5.3.2.

5.3.3.- MF-E

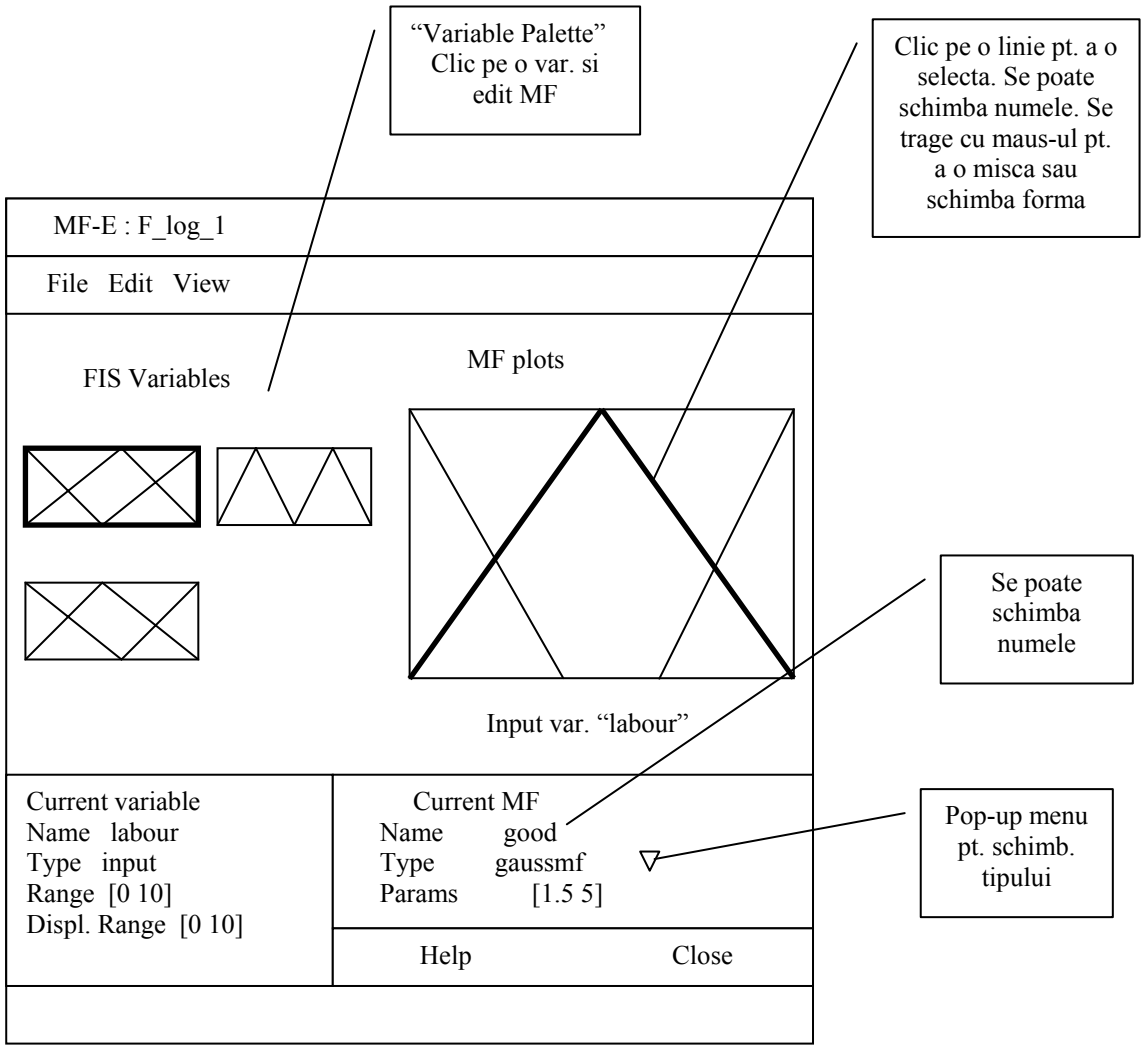


Fig. 5.3.2

MODUL DE LUCRU

1- In zona “Variable Palette” (FIS Variables) se selecteaza variabila de intrare “labour” (dublu clic pe ea) – in campurile de editare din stanga jos se scrie [0 10].

2- Edit – Add MFs – apare o fereastra in care:

3- Se alege “gaussmf” si “3” (adica 3 curbe Gauss pevtru “labour”).

4- In zona “MF plots” se face clic pe curba din stanga; in zona din dreapta jos i se schimba numele din “mf1” in “poor”- En.

Ajustarea formei MF:

- se trage cu maus-ul in stanga sau dreapta (vor fi afectati parametrii matematici);
- clic pe un mic patrat de pe curba si apoi tragere spre “outside” (pentru dilatare), sau spre “inside” (pentru contractie);
- in zona din dreapta jos se tasteaza parametrii doriti - En.

Pentru curba “poor” parametrii sunt [1.5 0].

5- Curba din centru “good” are parametrii [1.5 5], iar cea din dreapta “excel.” [1.5 10].

6- Clic pe variabila de intrare “efficiency” – in “Range” si “Displ. Range” (stanga jos) se scrie [0 10].

7- Edit – Add MFs → 2 “trapmf” (pentru “efficiency”).

8- Clic pe trapezul din stanga - i se schimba numele in “low” si param. [0 0 1 3].

9- Trapezul din dreapta→ “big” si parametrii [7 9 10 10].

10- In zona “Variable Palette” (FIS Variables) se selecteaza “salary” – in “Range” si “Displ. Range” se scrie [0 100] - Edit – Add MFs → “trimf” si “3” – MF “small” → [0 25 40]; MF “average” → [40 55 70]; MF “generous” → [70 85 100].

Apelare R-E:

- View – Edit rules. . .; - Se scrie “ruleedit” la prompt. Apare fig. 5.3.3.

5.3.4.- R-E

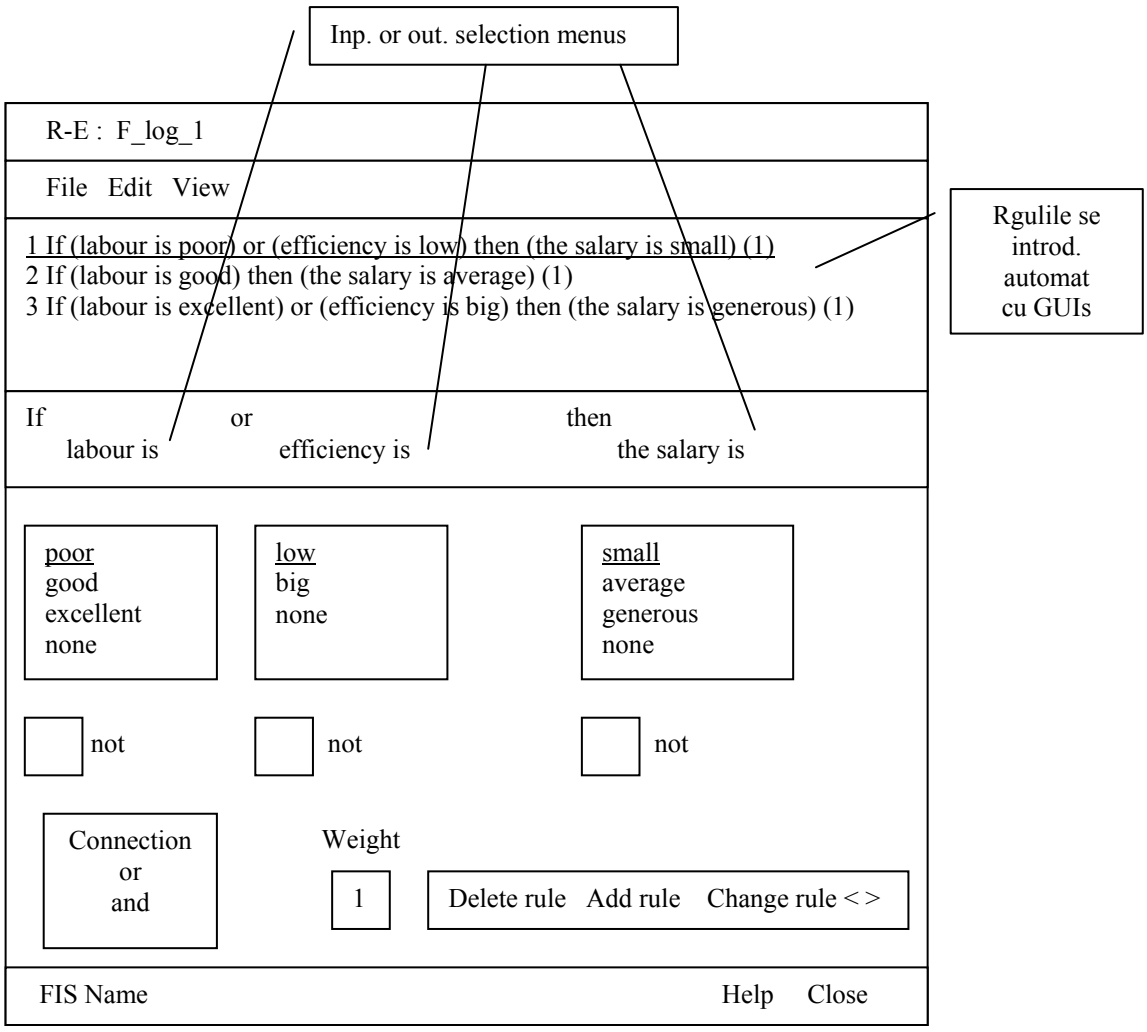


Fig. 5.3.3

Alegand “none” in meniuri, se exclude acea varianta in regula respectiva; alegand “not”, apare varianta negata.
Pentru a insera prima regula: sub “labour” se selecteaza “poor”; sub “efficiency”– low; sub “salary”- small; in zona “Connection”- or. Numarul din coada este ponderea ce se poate aplica fiecarei reguli (se poate schimba scriind sub “Weight” un numar $\in$ [0 1]) –Clic pe “Add rule”.
Analog pentru regulile 2 si 3.
Modificarea unei reguli: clic pe ea – se fac modificarile - clic pe “Change rule”.
In meniul “Options”- Format – este activat “Verbose” (adica forma lingvistica a regulilor, ca mai sus) – alegand “Symbolic”, apare:
1 (labour = = poor) or (efficiency = = low) $\rightarrow$ (salary = small) (1)
2 …
3 …
Se continua cu: View – View rules.

5.3.5.- R-V

In figura 5.3.4 fiecare regula este un rand de ploturi, iar fiecare coloana este o variabila; primele doua coloane (galbene) = antecedentul (partea “if”), iar a treia coloana (bleu) = consecinta (partea “then”). Plotul blanc de sub “efficiency” corespunde lui “none”.
Vectorul de intrare se poate introduce:
- prin scriere in zona din stanga jos;
- clic oriunde in ploturile de intrare - linia rosie se va translata pe orizontala spre punctul de clic;
- se trage cu maus-ul linia rosie.

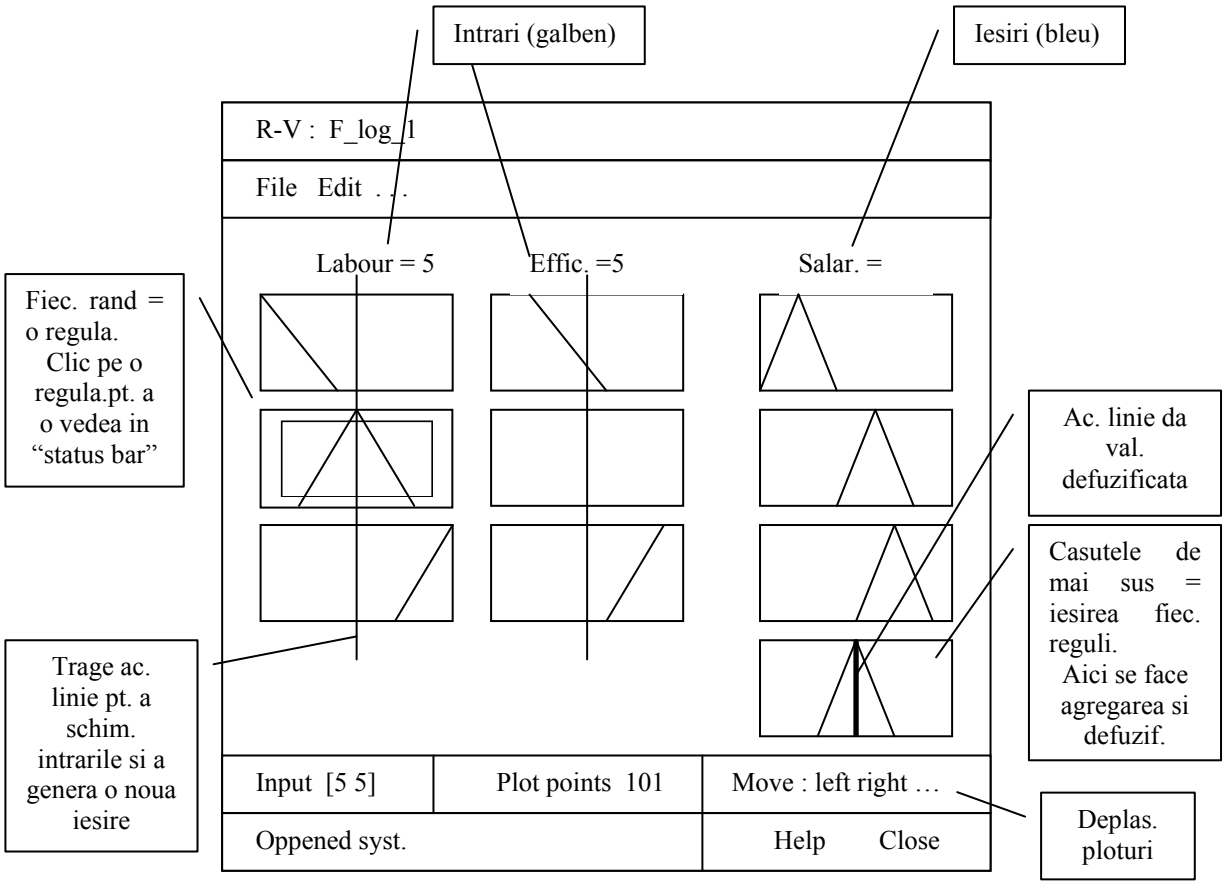


Fig. 5.3.4

Dupa asta, PC-ul realizeaza un nou calcul si se poate vedea procesul de inferenta in actiune.
La regula 1 se poate observa cum consecinta “salary is small” este retezata cu gradul rezultat din antecedent; acesta este procesul *implication* in actiune.
R-V da o interpretare intregului FIS, aratand cum forma diferitelor MF influenteaza rezultatul final. Se poate lucra cu pana la 30 reguli si 7 variabile.

R-V arata un singur calcul in timp, in detaliu. Pentru a vedea intreaga suprafata de iesire (adica intreaga anvergura a iesirii, in functie de intreaga anvergura a intrarii): View – View surface.

5.3.6.- S-V (fig. 5.3.5)

Schimbarea "X grids" (sau "Y grids"): se modif. *15* – Clic pe "X grids".

"Ref. Input": pentru sisteme care au mai multe intrari decat permite plotul; de exemplu, la un sistem cu o iesire si 4 intrari, se ploteaza o iesire in functie de 2 intrari, celelalte 2 intrari fiind constante.

5.3.7.- IMPORTUL SI EXPORTUL DIN GUI-TOOLS

Pentru redeschiderea unui FIS, el trebuie salvat pe disc (ca un text ASCII), cu sufixul "fis".

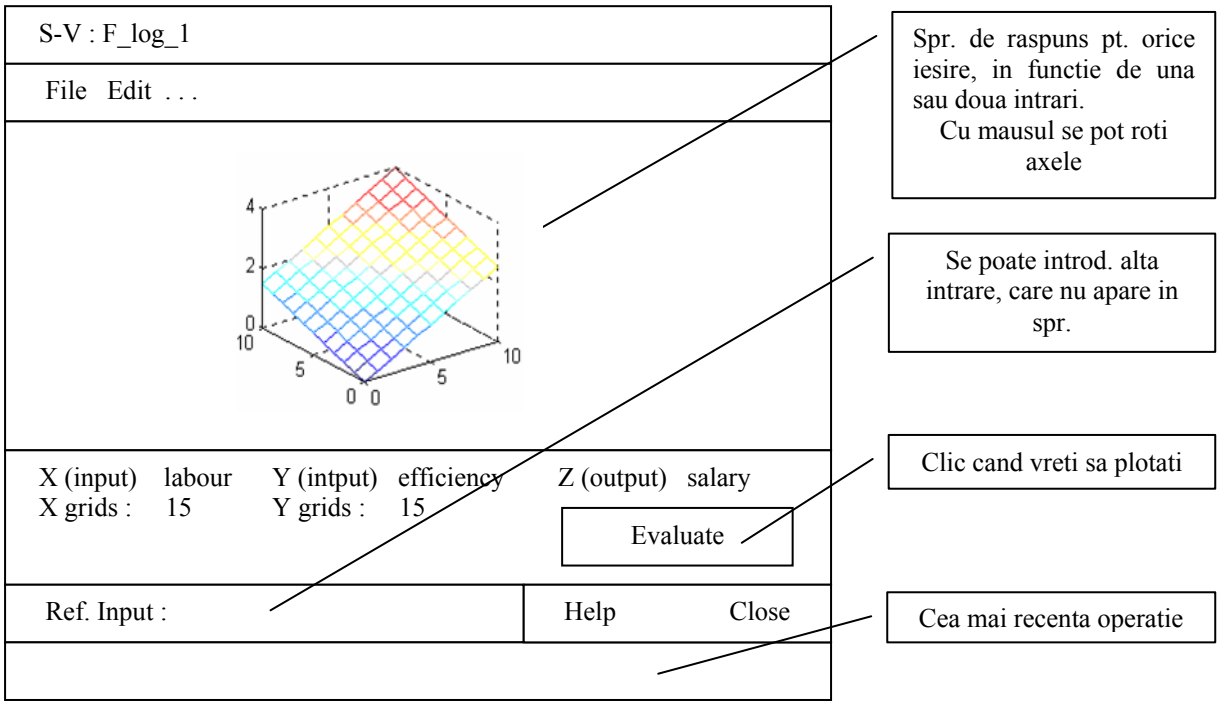


Fig. 5.3.5