1. Sisteme „Expert”

Va angajati la o firma de recrutare resurse umane. Aici vi se spune sa identificati oameni ambitiosi. Pentru a va face munca mai usoara, construiti un Sistem Exper folosind urmatoarele reguli:

R0: **if** AND(?x risca mult; ?x e muncitor; ?x vrea sa fie mai bun decat ?y) **then** ?x e ambitios
R1: **if** ?x merge la extrem **then** ?x risca mult
R2: **if** ?x rezolva probleme **then** ?x e muncitor
R3: **if** ?x e inteligent **then** ?x rezolva probleme

Dupa cateva interviuri apar urmatoarele afirmatii despre candidati:

A0: Adi merge la extrem A1: Mihai merge la extrem A2: Andrei rezolva probleme
A3: Andrei vrea sa fie mai bun decat Adi A4: Andrei merge la extrem

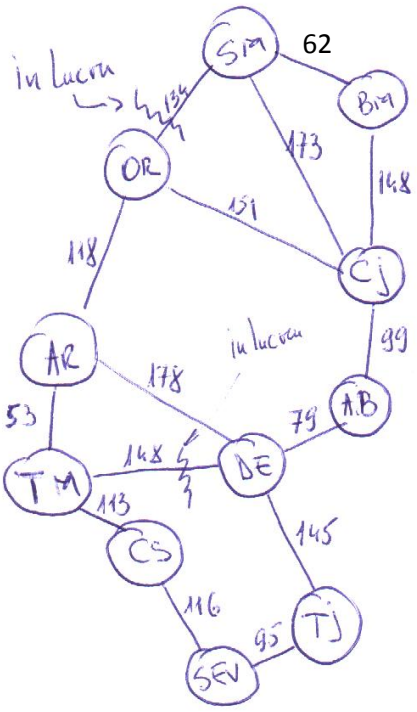
Folosind Backward chaining demonstrati ca **Andrei e ambitios**.
Folosind Forward chaining, rulati primele 3 iteratii pentru Sistemul Expert.

2. Algoritmi de cautare

Sunteti teleportat intr-un univers paralel in care nu exista Google Maps sau alte sisteme de navigatie. Va hotarati sa construiti o aplicatie de navigatie folosind cunostintele de la IA. Folositi harta zonei de vest a Romaniei (graful din figura). Punctul de plecare este Deva (DE), punctul de sosire este Timisoara (TM). Va decideti sa implementati Cautare in adancime (Depth First Search) si Cautare pe nivel (Breadth First Search). Ce rute rezulta? Va hotarati sa faceti o analiza a algoritmilor. Ce algoritm produce numarul minim de pasi tinand cont ca drumurile in lucru nu sunt practicabile.

Folosind un dispozitiv de masurare a distantei dintre orase, sunteti capabil sa adaugati la graf distantele dintre localitati. Folosind Branch&Bound, stabiliti ruta optima intre Deva (DE) si Satu Mare (SM). In acest caz drumurile nu mai sunt in lucru.
Observati ca algoritmul nu este foarte eficient si ca treceti de mai multe ori prin acelasi loc in cautare. Va decideti sa folositi B&B + Lista elementelor extinse (B&B + EL). Care este arborele de cautare? Exista imbunatatire fata de B&B ?

Minimax. Va plictisiti acasa, asa ca va decideti sa construiti un agent inteligent care sa va fie adversar in jocuri. Agentul gandeste cateva mutari in avans dupa care produce evaluari statice, astfel rezulta urmatorul arbore de joc. Aplicand algoritmul minimax, determinati valoarea din varf a maximizatorului si secventa de mutari care produce acest rezultat. Tineti cont ca factorul de ramificare este 2, si valorile statice sunt urmatoarele:
-7 76 -84 -88 -28 76 72 -65 82 -47 -20 -62 77 -11 98 24

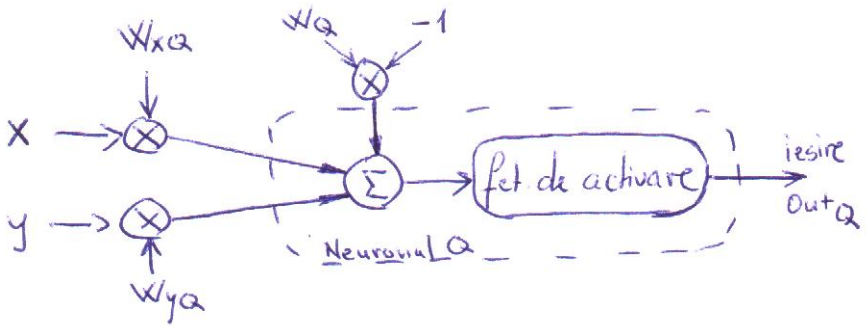


3. Algoritmi de clasificare. Retele neuronale

In cadrul experimentelor de inteligenta artificiala primiti la studiu un neuron artificial. Stabiliti comportamentul acestuia in clasificarea (iesirea din neuron) unui element, daca se foloseste o functie de activare tip treapta sau sigmoida. Neuronul va clasifica 2 clase, 0 si 1. Obiectul va face parte fie din clasa 0, fie din clasa 1.

Pentru functia treapta, reprezentati intr-un grafic xOy dreapta ce reprezinta pragul de clasificare.

Se dau: $W_{QX} = 2$; $W_{QY} = 9$; $W_Q = 0$; Elementul de clasificat 1 (X=-3; Y=2); Elementul 2 (X=1; Y=0);



$$treapta(x) = \begin{cases} 0, & \text{dacă } x < 0 \\ 1, & \text{dacă } x \geq 0 \end{cases} \qquad sigmoida(x) = \frac{1}{1+e^{-x}}$$

4. Fuzzy Logic

Vă aflați pe un vapor de croazieră când, la un moment dat, vaporul se oprește. Curios din fire, vă informați de ce s-a oprit vaporul și aflați că sistemul de menținere pe traiectorie s-a defectat, acesta avea rolul de a compensa deviațiile cauzate de vantul lateral. Vă hotarâți sa concepeti dumneavoastra un sistem nou, bazat pe Fuzzy logic, pentru a continua croaziera. In acest sens alegeti ca parametrii de control la intrare: viteza de deplasare a vaporului [noduri/h] și viteza vântului [km/h] iar ca parametru controlat poziția cârmei [%] (0%-inainte, 100%-maxim dreapta).

Construiti următoarele functii de apartenență trapezoidale

viteza vântului din dreapta [km/h]: slab(0,10,20,30), mediu(20,30,50,60), tare(50,60,100,100)

viteza vaporului [noduri/h]: mica(0,0,5,10), medie(5,10,15,20), mare(15,20,35,35)

poziția cârmei către dreapta [%]: ușor(0, 0,20,30), mediu(20,30,80,90), tare(80,90,100,100)

Stabiliți cateva reguli (2 fiind suficiente pentru a demonstra un caz particular)

R1: Dacă vânt_dreapta=slab și viteza_vapor=medie atunci cârma_dreapta=ușor

R2: Dacă vânt_dreapta=mediu și viteza_vapor=mare atunci cârma_dreapta=mediu

și vă propuneți ca acest controller să funcționeze după o normă Min/Max

Se măsoară vânt_dreapta=22 km/h și viteza_vapor=17 noduri/h, se cere ca să se determine poziția cârmei pentru aceste valori, astfel încât voi sa mențineți traiectoria.

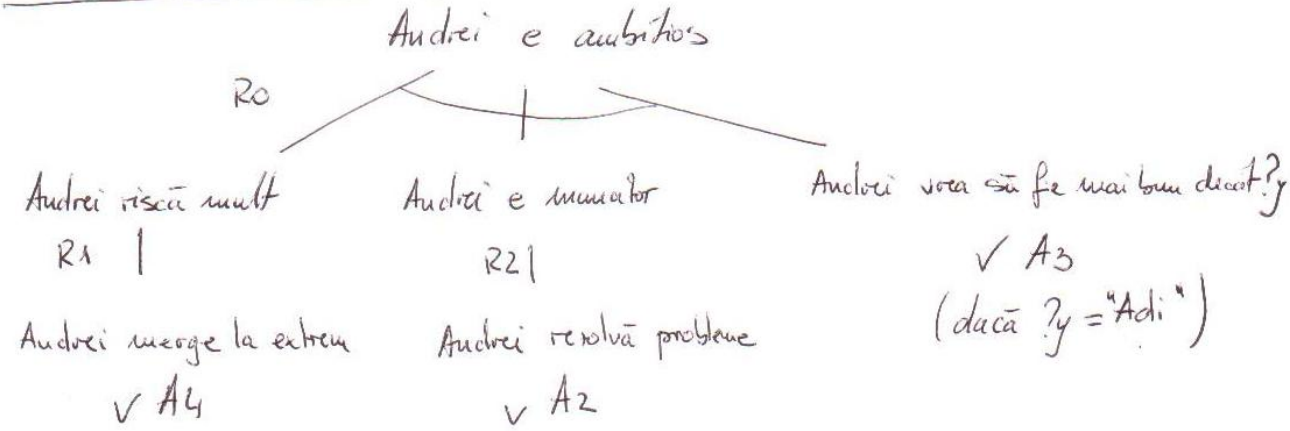
Valori ajutătoare:

Grade de apartenență $\mu_{vânt\_dreapta\_slab}(22) = 0.8$ $\mu_{vânt\_dreapta\_mediu}(22) = 0.2$

$\mu_{viteza\_vapor\_mediu}(17) = 0.6$ $\mu_{viteza\_vapor\_mare}(17) = 0.4$

1. R B E. S.

Backward Chaining



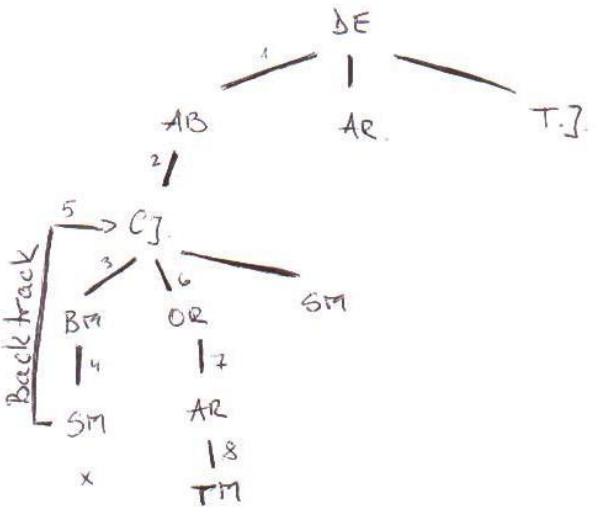
=> „Andrei e ambitios" se adevereşte

Forward Chaining

	Se poate executa	Se execută	Afirmaţie nouă	Expiră
1.	R1, R2	R1	A5. Adi riscă mult	R1 - Adi
2.	R1, R2	R1	A6. Mihai riscă mult	R1 - Mihai
3.	R1, R2	R1	A7. Andrei riscă mult	R1 - Andrei

2. Algoritmi de căutare.

DE - TM - DFS



DE - TM - BFS

Nr. de pași (căi dezvoltate în procesul de căutare):

DFS : 8 (notate pe arborele de căutare)

BFS : 6 (notate pe arborele de căutare)

calea / ruta : DFS DE - AB - CJ - OR - AR - TM

BFS. DE - AR - TM

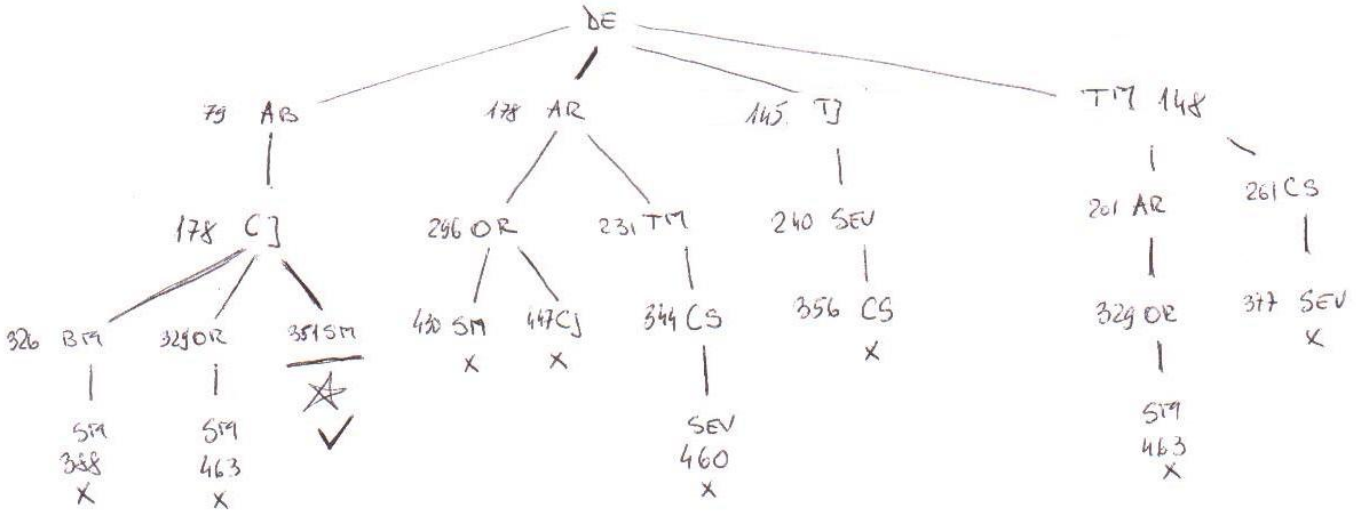
Lungimea rutei (numărată ca nr. de salturi) : DFS : 5

BFS : 2

BFS produce numărul mai mic de salturi.

2. Algoritmi de căutare. Branch & Bound

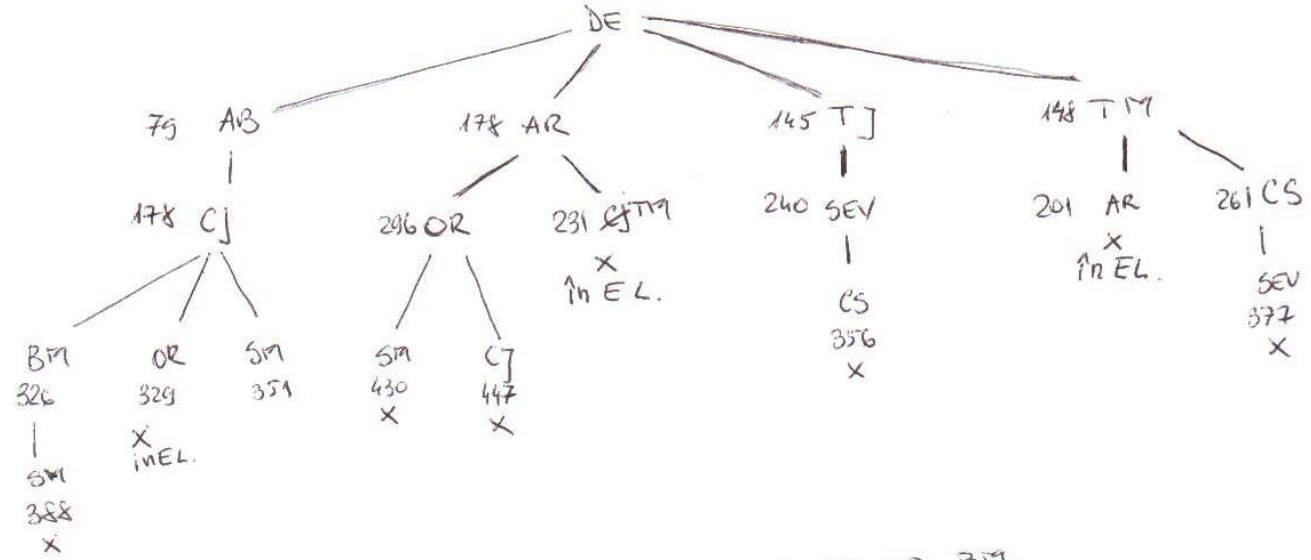
DE - SM - B&B



Ruta găsită cu B&B = DE - AB - Cj - SM
Lungime = 351

DE - SM BB + EL.

Elem. în Extended List: AB, TJ, TM, AR, Cj, SEV, CS, OR, BM



Elem. în EL: AB, TJ, TM, AR, Cj, SEV, CS, OR, BM.

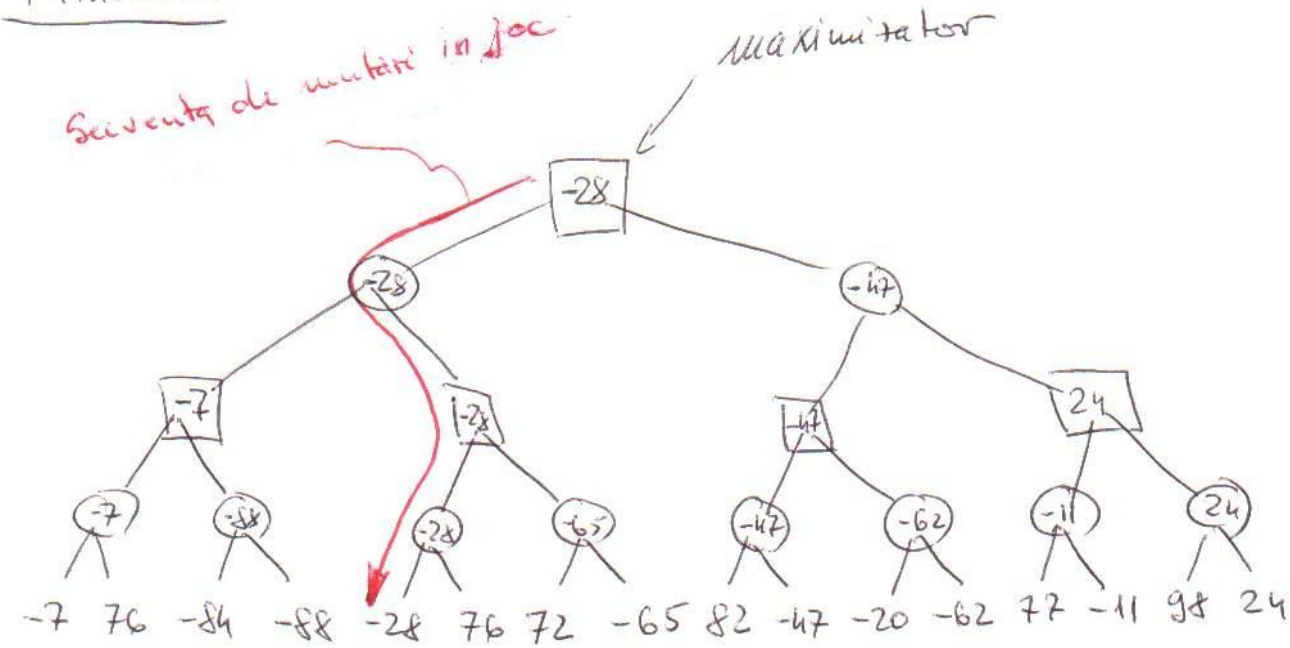
Calea descoperită cu B&B + EL: DE - AB - Cj - SM, Lungime = 351

Se observă că arborele de căutare B&B + EL e redus față de B&B

Notă: Se notează cu X ramurile care nu se mai extind (de exemplu, pt. că sunt mai lungi decât calea la final)

2. Algoritmi de căutare

Minimax



Rezultatul obținut de maximizator este **-28**

3. Rețele neuronale artificiale

Se calculează ieșirea din sumator:

(*) $x \cdot w_{xa} + y \cdot w_{ya} + (-1) \cdot w_a =$ pt. elementul 1: 12
pt elementul 2: 2

Ieșirea din sumator este intrare în funcția de activare:

Pt. elementul 1: ieșirea din sumator = 12.

La funcția treaptă: treaptă (12) = 1

sigmoidă: $\text{sigmoida}(12) = \frac{1}{1+e^{-12}} = 1$

(e^{-12} aprox 0)

=> ~~Rețeaua neuron~~ Neuronul clasifică elementul 1 (-3,2) drept făcând parte din clasa 1

Pt. elementul 2. Ieșirea din sumator = 2 $e_2(1,0)$

La funcția treaptă: treaptă (2) = 1

sigmoidă: $\text{sigmoidă}(2) = \frac{1}{1+e^{-2}} = \frac{1}{1+0,13} = 0,88 \Rightarrow$

=> neuronul mai trebuie „antrenat" pt. o clasificare corectă. (0 sau 1) („ajustate ponderile")

Reprezentarea grafică a pragului de clasificare:

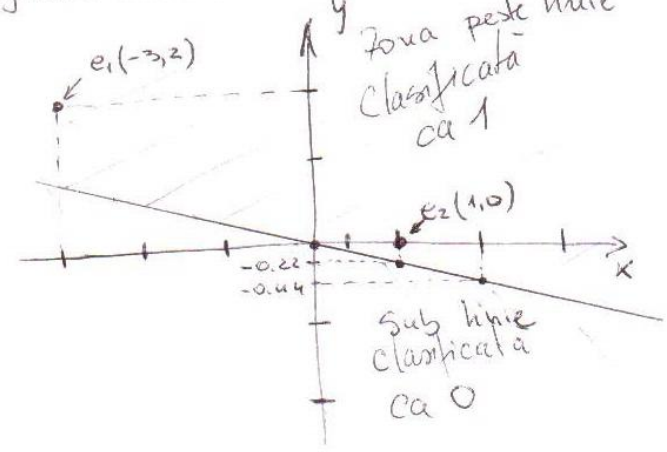
Se pleacă de la relația (*), aceasta se egalează cu zero

$x\, w_{xa} + y\, w_{ya} + (-1)\, w_a = 0 \Rightarrow$

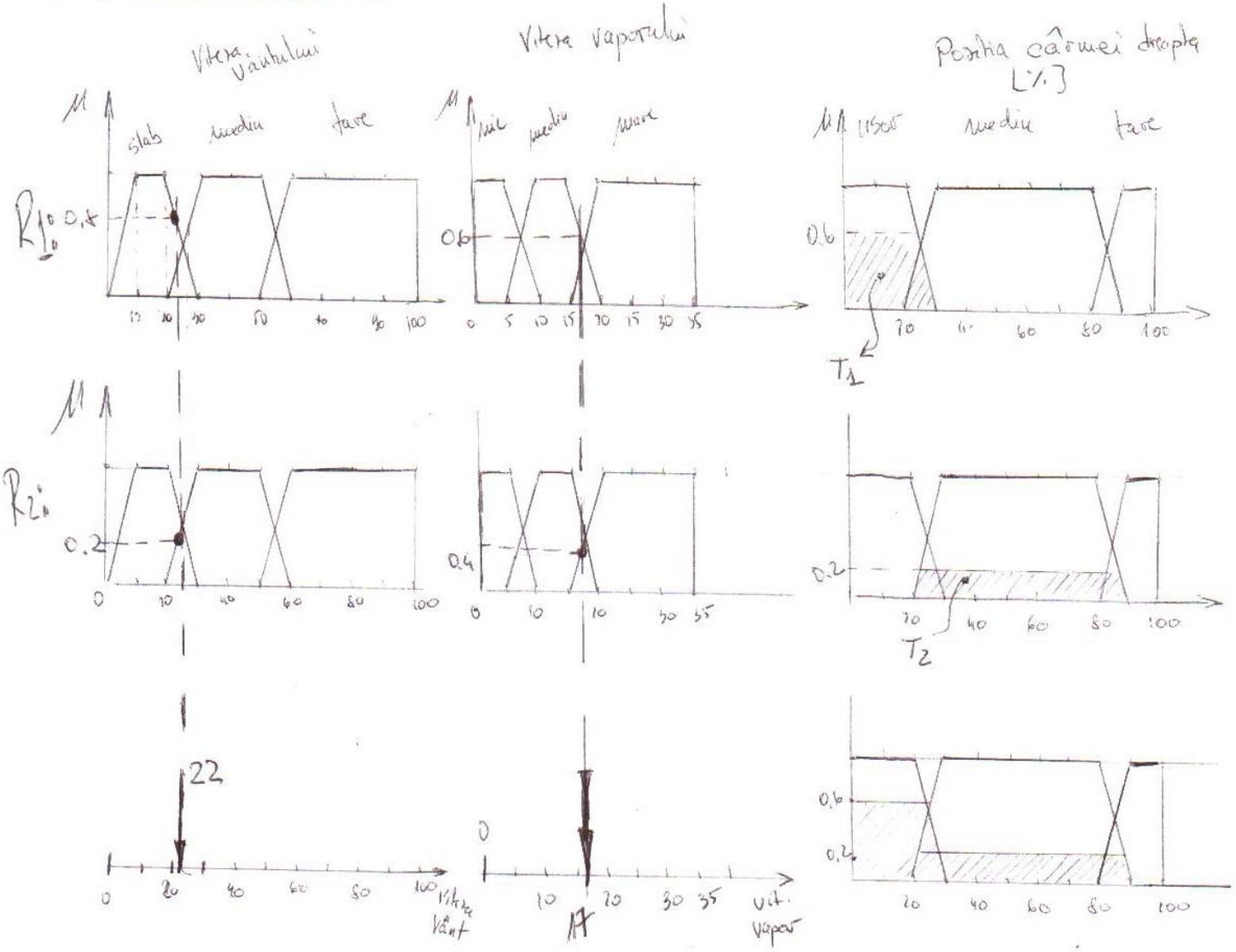
$y = \frac{-x\, w_{xa} + w_a}{w_{ya}}$ Această este o funcție de gradul I $y = f(x)$

Pt. noi => $y = \frac{-x \cdot 2 + 0}{9}$

Aceasta se reprezintă pe graficul xOy



4. Fuzzy Logic.



$$C.g.\ trapez = \frac{2ac + a^2 + cb + ab + b^2}{3(a+b)} \quad \cancel{+ offset}$$

$$C_{g T_1} = \frac{20^2 + 20 \cdot 30 + 30^2}{3(20+30)} + \underset{(0)}{offset} = \frac{400 + 600 + 900}{3 \cdot 50} = 12,5$$

$$C_{g T_2} = \frac{2 \cdot 10 \cdot 50 + 50^2 + 10 \cdot 70 + 50 \cdot 70 + 70^2}{3(50+70)} + 20 = 37,7 + 20 = 57,7$$

Se înmulțesc cu gradele de umplere

$\cdot 0,6 = 7,5$

$\cdot 0,2 = 11,54$

$$\Rightarrow \quad C_{g\ figura} = \frac{12,5 \cdot 0,6 + 57,7 \cdot 0,2}{0,8}$$

$$C_{g\ figura} = 23,8$$