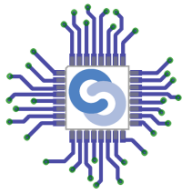


Sisteme Încorporate

Cursul 7

Sisteme de control

Facultatea de Automatică și Calculatoare
Universitatea Politehnica București

http://dilbert.com/strips/comic/1997-05-13/

Recapitulare

- Control Proporțional (P)
- Control Proporțional Diferențial (PD)
- Control Proporțional Integral (PI)

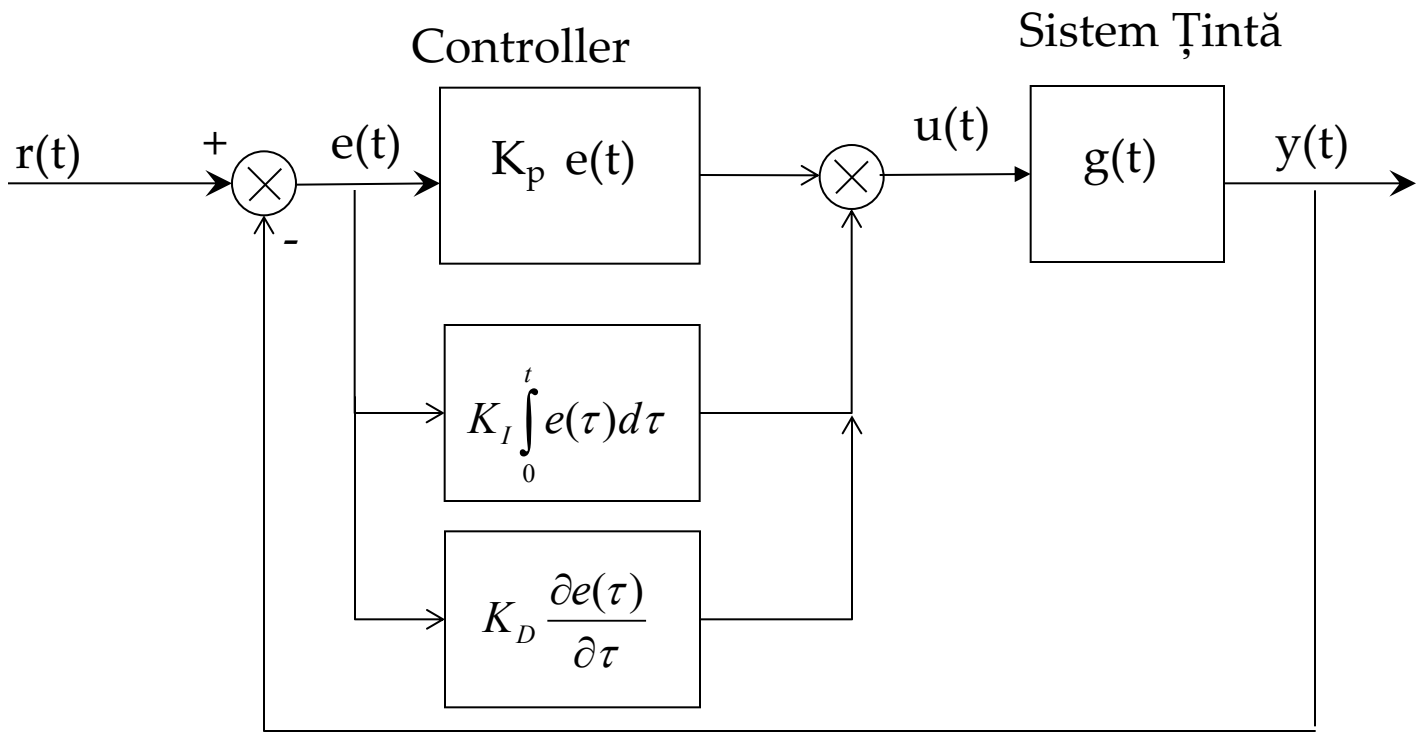
- Analiza din punct de vedere al stabilității și acurateței răspunsului sistemului la o excitație de tip treaptă

Controlul PID

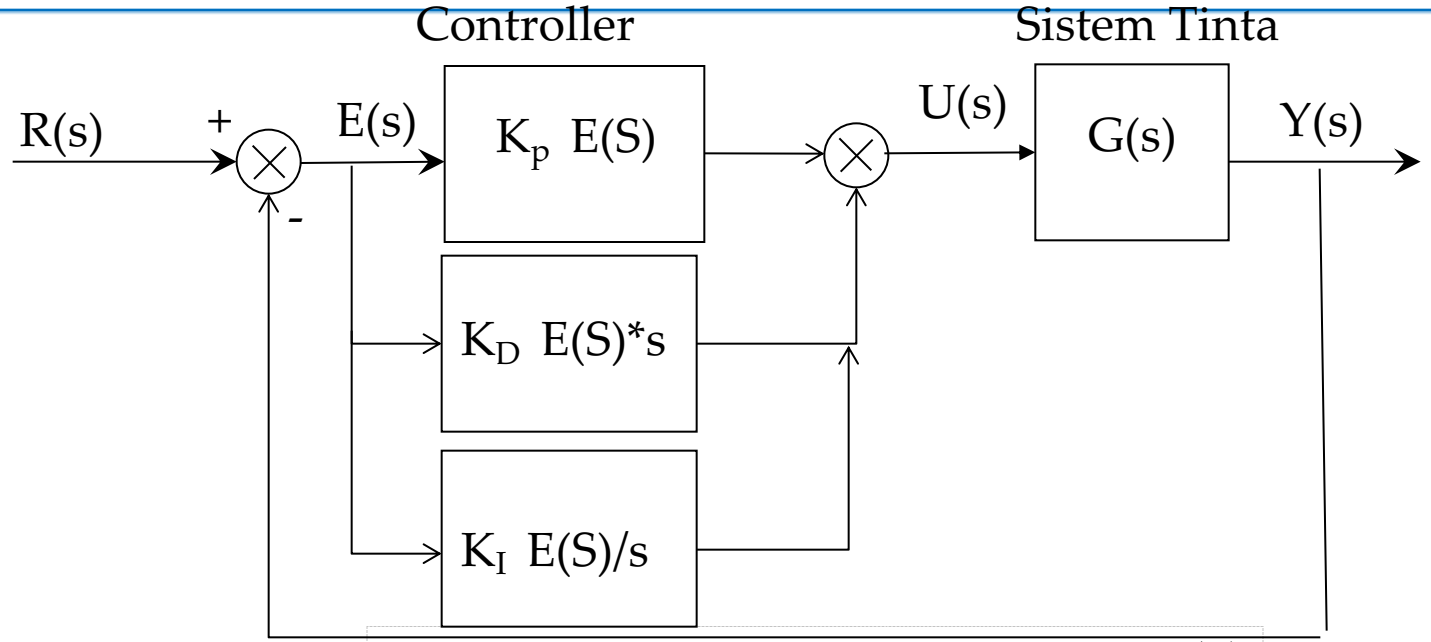
- Folosește toate cele trei tipuri de control studiate anterior
- Este cel mai utilizat tip de control în industrie, datorită stabilității lui.
- Performanțe crescute din punctul de vedere al timpului de răspuns, stabilității și erorii de aproximare.

Controlul PID

$$u(t) = K_P e(t) + K_D \frac{\partial e(\tau)}{\partial \tau} + K_I \int_0^t e(\tau) d\tau$$



Funcția de transfer PID



$$E(s) = R(s) - Y(s)$$

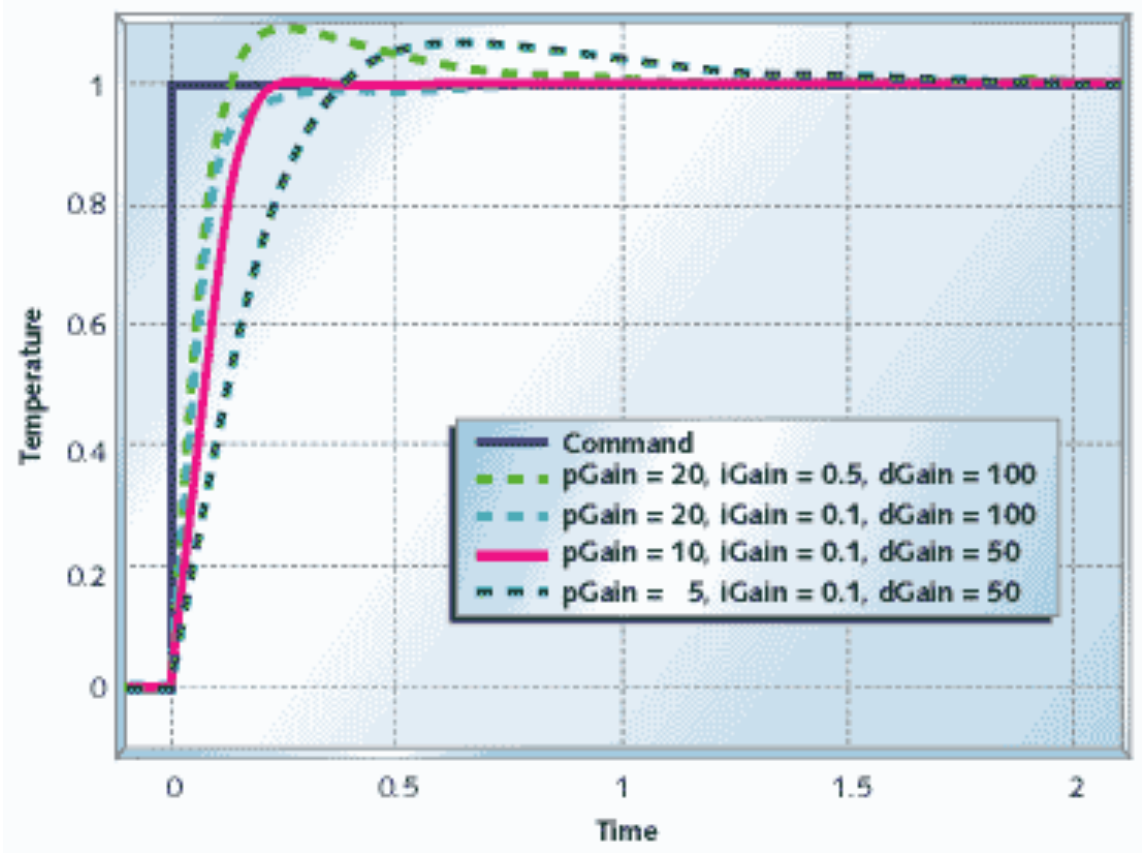
$$U(s) = K_P \cdot E(s) + K_D \cdot sE(s) + K_I \frac{E(s)}{s}$$

$$Y(s) = G(s)U(s)$$

$$H(s) = \frac{(K_P + sK_D + \frac{K_I}{s}) \cdot G(s)}{1 + \left(K_P + sK_D + \frac{K_I}{s} \right) \cdot G(s)}$$

Exemplu

- Pentru controllerul de temperatură:



www.embedded.com

Acuratețe

Ecuația caracteristică: $1+\left(K_P+sK_D+\frac{K_I}{s}\right)\cdot G(s)=0$

$$e_{ss}=\lim_{k->\infty} e(k)$$

$$e_{ss}=\lim_{s->0}\frac{1}{1+\left(K_P+sK_D+\frac{K_I}{s}\right)G(s)}$$

Estimarea coeficienților (Loop Tuning)

- Dacă parametrii controlului PID sunt aleși incorect, sistemul poate să devină instabil și să devieze de la răspunsul optim cu sau fără oscilație.
- Mai multe metode de estimare:
  - Estimare Manuală
  - Ziegler-Nichols
  - Software Tuning
  - Cohen Coon

Manual Tuning

- Obținerea matematică a K_P, K_I si K_D nu poate fi posibilă
  - Sistemul este prea complex pentru a fi modelat matematic
  - Costurile de modelare sunt prea mari
- Metoda ad hoc pentru obținerea unor valori rezonabile ale K_P, K_I, și K_D
  - Inițial K_P foarte mic, $K_I = K_D = 0$
  - Mărește K_P până când apar oscilații
    - Setează K_P la jumatate din valoarea obținută
  - Maresțe K_I până când răspunsul sistemului este egal cu valoarea dorită
    - K_I mare poate induce oscilații
  - Măreste K_D până când sistemul are un timp de răspuns suficient de mic pentru orice perturbații

Metoda Ziegler-Nichols

- Produce rezultate acceptabile, dar nu optime
- Algoritm:
  - Inițial K_P foarte mic, $K_I = K_D = 0$
  - Mărește K_P până când apar oscilații (K_C – câștigul critic)
    - Măsoară perioada oscilațiilor sistemului (T_C)
  - Determină coeficienții folosind următoarele relații:

	K_P	K_I	K_D
P	$0.5 \cdot K_c$	-	-
PI	$0.45 \cdot K_c$	$1.2K_p / T_c$	-
PID	$0.6 \cdot K_c$	$2K_p / T_c$	$K_pT_c / 8$

Implementare Software

- Buclă principală, ciclează la infinit
  - Măsoară variabila de ieșire a sistemului
    - De cele mai multe ori convertor AD
  - Măsoară valoarea de referință curentă
  - Apelează PidUpdate pentru a calcula comanda efectorului
  - Setează valoarea curentă pentru efector
    - Convertor DA

```
void main()
{
    double sensor_value, actuator_value, error_current;
    PID_DATA pid_data;
    PidInitialize(&pid_data);
    while (1) {
        sensor_value = SensorGetValue();
        reference_value = ReferenceGetValue();
        actuator_value =
         PidUpdate(&pid_data,sensor_value,reference_value);
        ActuatorSetValue(actuator_value);
    }
```

Implementare Software

- Pgain, Dgain, Igain sunt constante stabilite cu un algoritm de tuning
- sensor_value_previous
  - Pentru controlul Diferențial
- error_sum
  - Pentru controlul Integral

```
typedef struct PID_DATA {
    double Pgain, Dgain, Igain;
    double sensor_value_previous; // find the derivative
    double error_sum; // cumulative error
}
```

Calcularea coeficienților

- $u_t=P*e_t+I*(e_0+e_1+...+e_t)+D*(e_t-e_{t-1})$

```
double PidUpdate(PID_DATA *pid_data, double sensor_value,
                 double reference_value)
{
    double Pterm, Iterm, Dterm;
    double error, difference;

    error = reference_value - sensor_value;
    Pterm = pid_data->Pgain * error; /* proportional term*/
    pid_data->error_sum += error; /* current + cumulative*/
    // the integral term
    Iterm = pid_data->Igain * pid_data->error_sum;
    difference = pid_data->sensor_value_previous -
                 sensor_value;
    // update for next iteration
    pid_data->sensor_value_previous = sensor_value;
    // the derivative term
    Dterm = pid_data->Dgain * difference;
    return (Pterm + Iterm + Dterm);
}
```

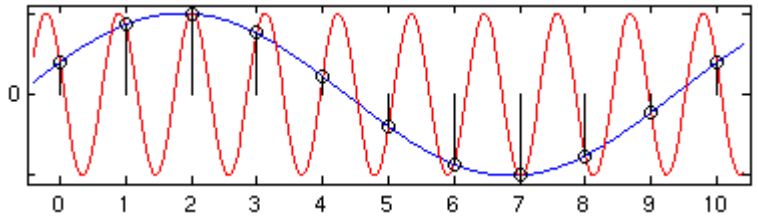
Deficiențe în controlul numeric

- Erori de reprezentare
  - Nu toate valorile numerice pot fi reprezentate intern datorită limitărilor de memorie
- Overflow
  - Depășirea superioară sau inferioară a domeniului de reprezentare
- Aliasing
- Viteza de procesare

Aliasing

- Două sau mai multe semnale periodice devin indistinctibile atunci când sunt eșantionate la o anumită frecvență.

  - Exemplu:
  - Eșantionate la 2.5 Hz, următoarele semnale nu pot fi diferențiate
    - $y(t)=1.0*\sin(6\pi t)$, f = 3 Hz
    - $y(t)=1.0*\sin(\pi t)$, f = 0.5 Hz
  - Conform teoremei Nyquist, frecvența minimă de eșantionare pentru primul semnal: 3*2 = 6 Hz



Viteza de procesare

- Produce întârzieri inevitabile
  - Acțiunea calculată se petrece mai târziu
- Întârzierile trebuiesc prevăzute și estimate în implementarea oricărui sistem de control.
- Întârzierile hardware sunt de obicei ușor de calculat
  - Structura sincronă a procesoarelor ajută
- Întârzierile software sunt mai greu de prevăzut
  - Codul trebuie organizat a.i. întârzierile să fie minime
  - Software cu întârzieri predictibile
    - Rutine declanșate la intervale regulate de timp
    - Limbaj de programare sincron (Esterel, ChucK)

Beneficiile controlului numeric

- Cost redus
  - Un control analogic este foarte scump mai ales dacă este proiectat să fie imun la perturbații
    - Uzură, temperatură, erori de fabricație
  - Controlul numeric înlocuiește circuitele analogice complexe cu programe complexe
- Programabilitate
  - Controlul poate fi “upgradat”
    - Schimbări la tipul controlului, câștig, sunt ușor de făcut
  - Se adaptează la schimbarile sistemului
    - Datorate îmbătrânirii, temperaturii etc.
  - “future-proof”
    - Ușor de adaptat la un alt standard

Sisteme de Control Fuzzy

Ce este Logica Fuzzy?

- Logica clasică:

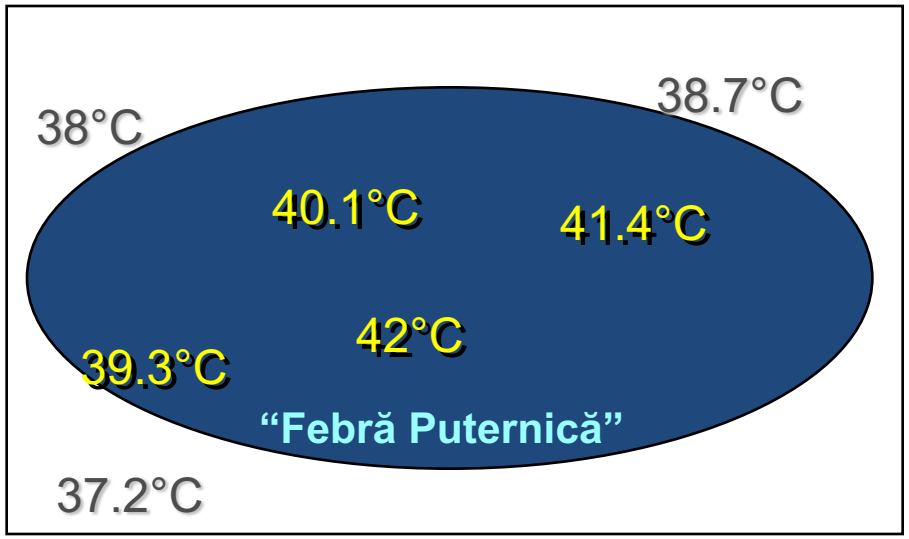
Un trandafir este Roșu… sau nu este Roșu

Ce culoare are trandafirul acesta?

Ce este Logica Fuzzy?

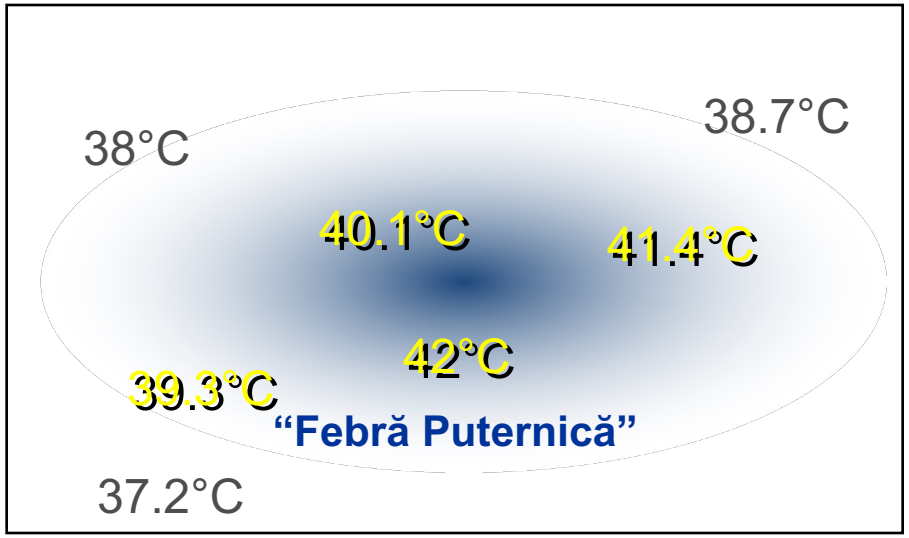
Embedded Systems Laboratory

Logica convențională (booleană):



“Mai mult sau mai puțin” în locul “ori așa – ori altfel” !

Teoria Fuzzy:



Ce este Logica Fuzzy?

- O extindere a logicii booleene care recunoaște mai multe valori de adevăr decât *adevărat* și *fals*.
- În logica fuzzy propozițiile pot avea mai multe grade de adevăr sau falsitate
  - Exemplu: “Afară este soare.”
    - 100% adevărat dacă afară este senin
    - 80% adevărat dacă este ușor înorat
    - 50% adevărat dacă este parțial înorat
    - 0% adevărat dacă plouă toată ziua
- Permite dispozitivelor de calcul să “gândească” la fel ca oamenii.

Logica Fuzzy

- Este o teorie matematică ce permite lucrul cu informații imprecise sau/și subiective
- Pornește de la o extensie a mulțimilor clasice, prin aceea că în LF un element aparține *într-un anumit grad* unei mulțimi

- Prezintă importanță practică deosebită inferența fuzzy:
  - Se bazează pe variabile lingvistice (vârstă, distanșă, viteză, etc), care au termeni (tânăr, batrân sau viteza mică, medie, mare, etc)
  - Și pe un mecanism de inferență fuzzy
    - Reguli de tip IF THEN, activate de fapte în diverse grade, combinate conform unor relatii matematice

Istoric Fuzzy

- **1965** Lucrarea “Fuzzy Logic” scrisa de Prof. Lotfi Zadeh, Faculty in Electrical Engineering, U.C. Berkeley, pune fundațiile teoriei fuzzy
- **1970** Prima aplicare a Controlului Fuzzy Logic (Europa)
- **1975** Introducerea Fuzzy Logic în Japonia
- **1980** Verificarea empirică a Fuzzy Logic în Europa
- **1985** Aplicarea la scară largă a Fuzzy Logic în Japonia
- **1990** Aplicarea la scară largă a Fuzzy Logic în Europa
- **1995** Aplicarea la scară largă a Fuzzy Logic în U.S.A.
- **2000** Fuzzy Logic devine o tehnologie standard și este aplicată în analiza de date și semnale. Aplicații ale Fuzzy Logic în finanțe și business.

Controllere Fuzzy

- Controllerele fuzzy sunt cele mai importante aplicații ale logicii fuzzy și a teoriei din spatele ei
- Ele funcționează foarte diferit de controllerele tradiționale
  - În loc de ecuații diferențiale, sistemul este modelat cu ajutorul cunoștințelor dobândite în timp de către experți
  - Aceste cunoștințe sunt exprimate într-un mod foarte natural folosind variabile lingvistice care sunt descrise de mulțimi fuzzy

Controllere Fuzzy – Aplicații

- Deși aplicarea fuzzy logic în rezolvarea și controlul sistemelor industriale a produs de foarte multe ori rezultate superioare controlului clasic, procedurile de design sunt limitate de regulile euristice ale sistemului.

- Această constrângere implicită limitează numărul de aplicații ale unui controller fuzzy
  - De cele mai multe ori, majoritatea controllerelor fuzzy au fost folosite în procese statice și bazate pe reguli derivate din cunostințele empirice ale unor operatori experimentați.

Mulțimi Fuzzy

- Teoria mulțimilor fuzzy se ocupă de caracterizarea submulțimilor unui domeniu de reprezentare U.
- O mulțime fuzzy $F \in U$ este o generalizare a conceptului de mulțime obisnuită și este identificată printr-o funcție de apartenență

  $\mu_F : U \to [0,1]$ în loc de valorile 0 și 1 ale unei mulțimi booleene obișnuite.

F este reprezentată de obicei ca o mulțime de perechi ordonate alcătuite din elementele u și gradul lor de apartenență:

$$F = \{(u, \mu_F(u)) | u \in U\}$$

Dacă U este un domeniu continuu, F poate fi exprimată dupa formula

$$F = \int_U \mu_F(u)/u$$

Dacă U este discret, atunci:

$$F = \sum \mu_F(u_i)/u_i$$

Apartenența la un set Fuzzy

Valori Discrete:

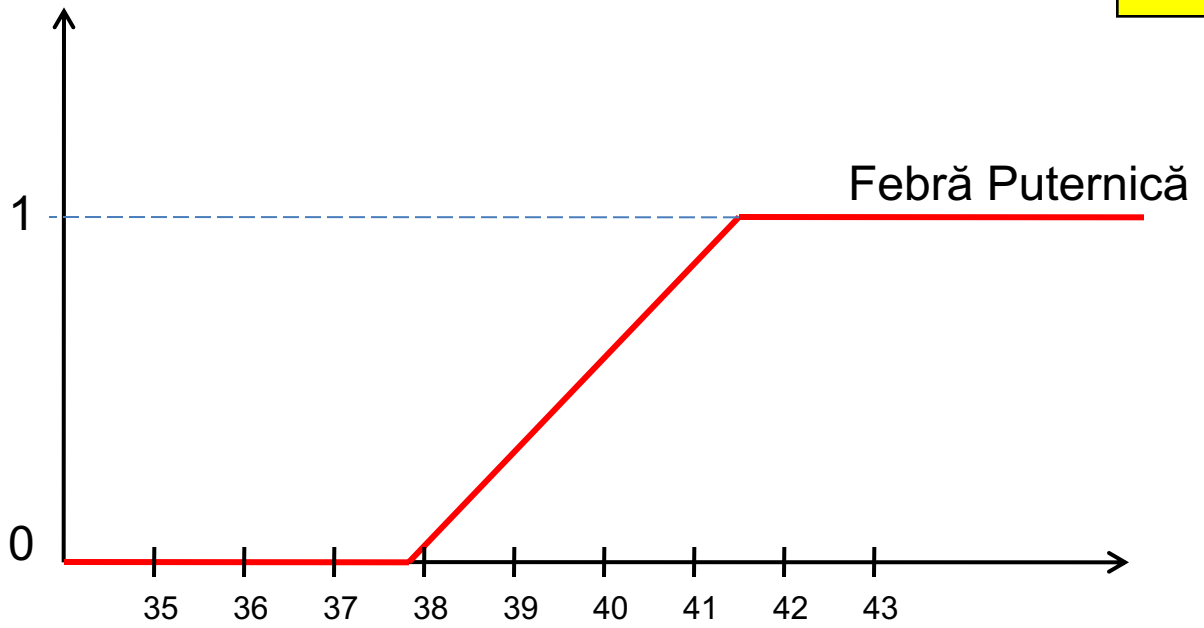
$\mu_{FP}(35°C) = 0$ $\mu_{FP}(38°C) = 0.1$ $\mu_{FP}(41°C) = 0.9$

$\mu_{FP}(36°C) = 0$ $\mu_{FP}(39°C) = 0.35$ $\mu_{FP}(42°C) = 1$

$\mu_{FP}(37°C) = 0$ $\mu_{FP}(40°C) = 0.65$ $\mu_{FP}(43°C) = 1$

Definiție Continuă:

Fără praguri "abrupte"



Operații pe seturi fuzzy

- Fie $A, B \in U$ două mulțimi fuzzy cu funcțiile de apartenență asociate u_A și u_B
- **Se pot defini următorii operatori:** $\mu_A(u) = \mu_B(u) \quad \forall u \in U$
- Două mulțimi A și B sunt egale dacă și numai dacă:
- Reuniunea celor două mulțimi are funcția de apartenență

$$\mu_{A \cup B}(u) = \max(\mu_A(u), \mu_B(u)) = \mu_A(u) \vee \mu_B(u) \quad \forall u \in U.$$

- Intersecția celor două mulțimi are funcția de apartenență:

$$\mu_{A \cap B}(u) = \min(\mu_A(u), \mu_B(u)) = \mu_A(u) \wedge \mu_B(u) \quad \forall u \in U.$$

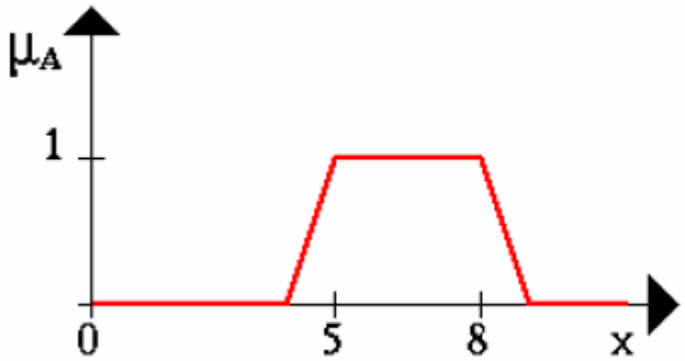
Operații pe seturi fuzzy

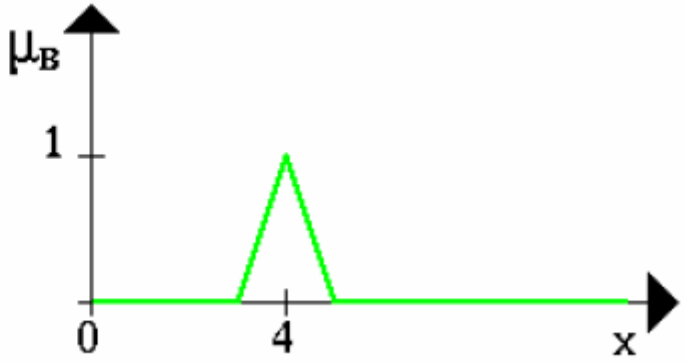
- Complementul lui A, A’ are funcția de apartenență:

$$\mu_{A'}(u) = 1 - \mu_A(u) \quad \forall u \in U.$$

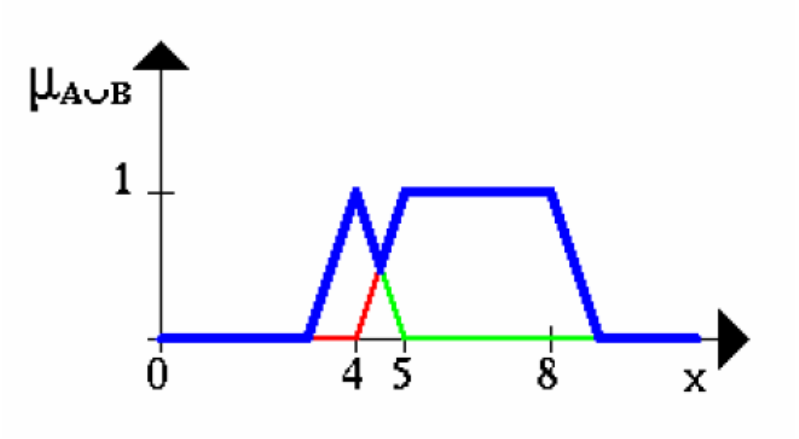
- **Exemplu:**

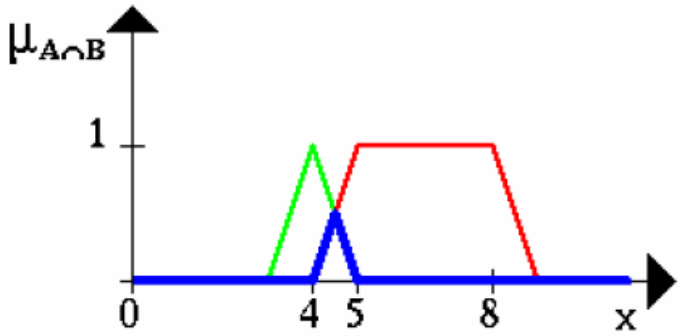
Fie A mulțimea “între 5 si 8” și B “aproape 4”

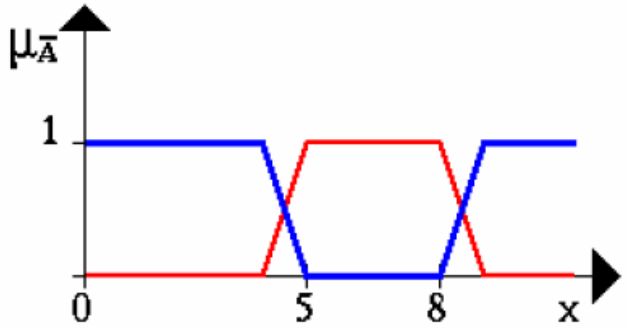




Operații pe seturi fuzzy

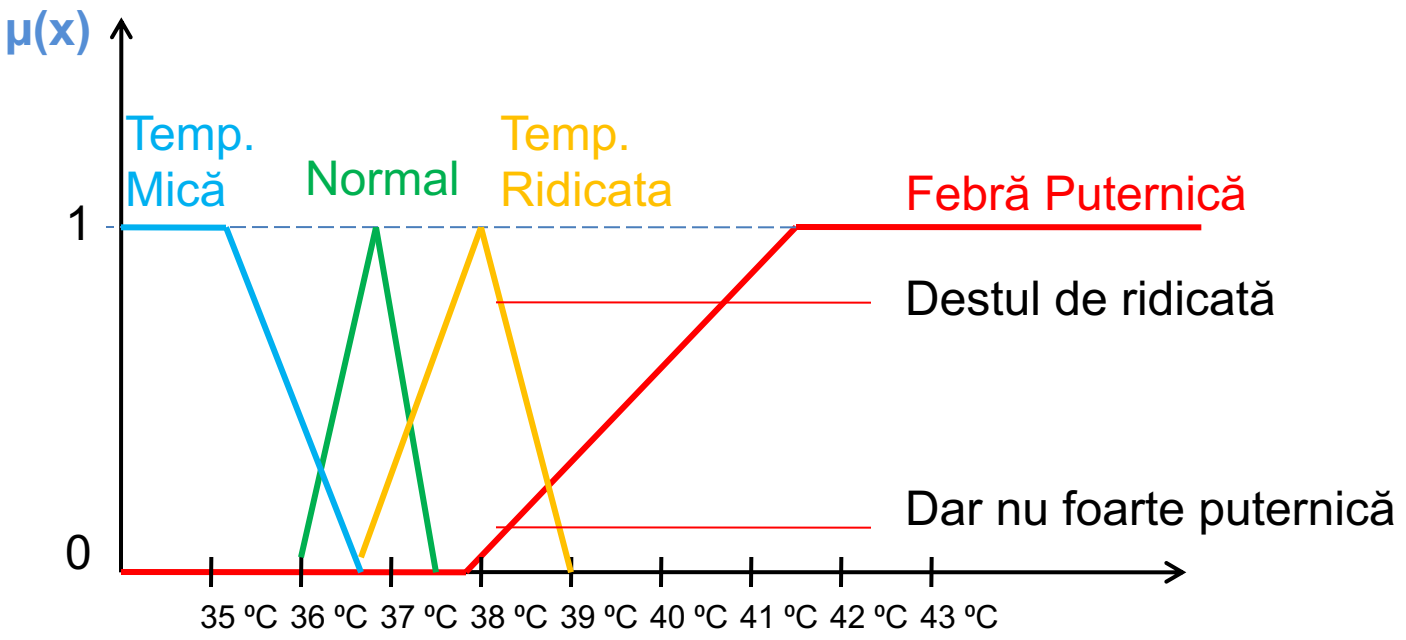






Variabile Lingvistice

- O variabilă lingvistică definește un concept din limbajul natural.

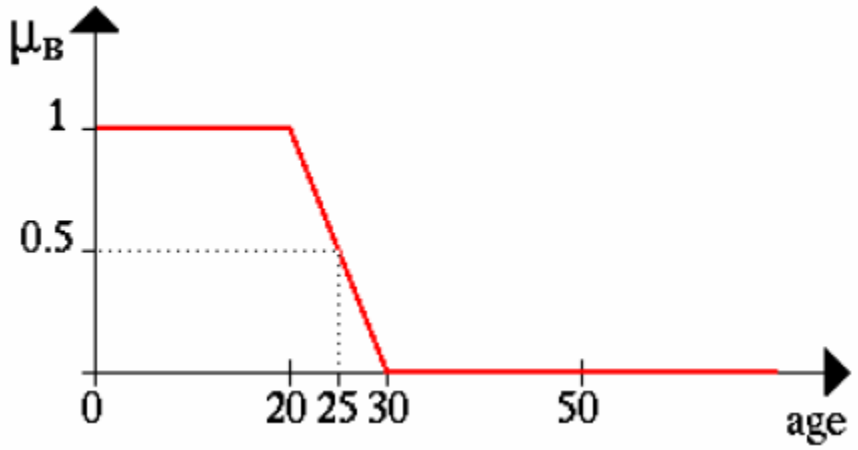


Variabile Lingvistice

- **O variabilă lingvistică asociază cuvinte sau propoziții cu o funcție de apartenență**
- Mulțimea de valori pe care fiecare variabilă lingvistică le poate avea se cheamă setul de termeni.
- Fiecare valoare din mulțime este o variabilă fuzzy definită peste o variabilă de bază
- Variabilele de bază definesc domeniul de reprezentare pentru toate variabilele fuzzy din mulțimea de termeni
- **O variabilă lingvistică este defapt un cvintuplu [*X, T(X), U, G, M] unde***
  - *X este numele variabilei*
  - *T(X) este mulțimea de termeni (mulțimea numelor pentru valorile lingvistice ale lui X)*
  - *U domeniul de reprezentare,*
  - *G este gramatica cu ajutorul căreia se generează numele*
  - *M este un set de reguli semantice care asociază fiecare X cu înțelesul său*
- Ierarhia: variabilă lingvistică -> variabilă fuzzy -> variabilă de bază

Variabile Lingvistice

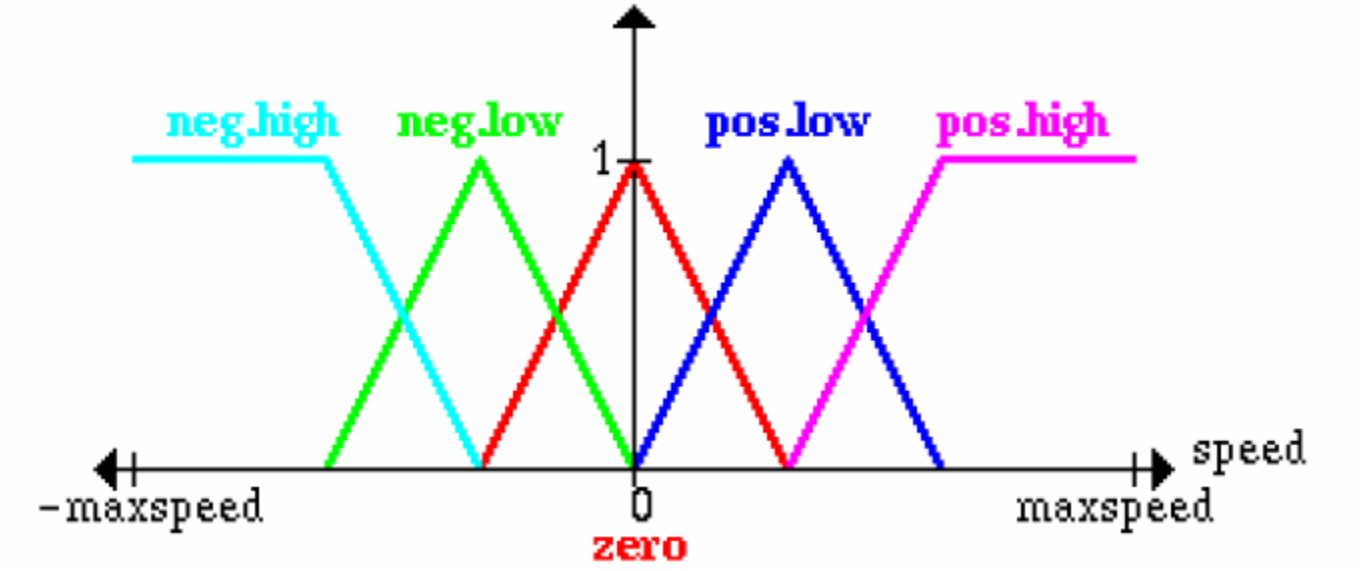
- **Fie x o variabilă lingvistică denumită “Vârstă”.**
- Termenii lui x, care sunt mulțimi fuzzy, pot fi: “bătrân”, “tânăr” și “foarte tânăr”din mulțimea de termeni **T= {Bătrân*, FoarteBătrân, NuPreaBătrân, MaiMultSauMaiPuținTânăr, Tânăr, FoarteTânăr}***
- Fiecare termen este o variabilă fuzzy definită peste variabila de bază, care poate fi o scală de la 0 la 100



Funcția de apartenență pentru x= “FoarteTânar”

Exemplul 2

- Fie x o variabilă lingvistică numită “viteza”.
- Termenii lui x, care sunt mulțimi fuzzy, pot să fie*”positive* low”, ”negative high” din mulțimea de termeni T= {*PositiveHigh, PositiveLow, NegativeLow, NegativeHigh, Zero}*
- Fiecare termen este o variabilă fuzzy definită peste o variabilă de bază care poate fi o scală cu toate vitezele relevante aplicației:



Controlul Fuzzy

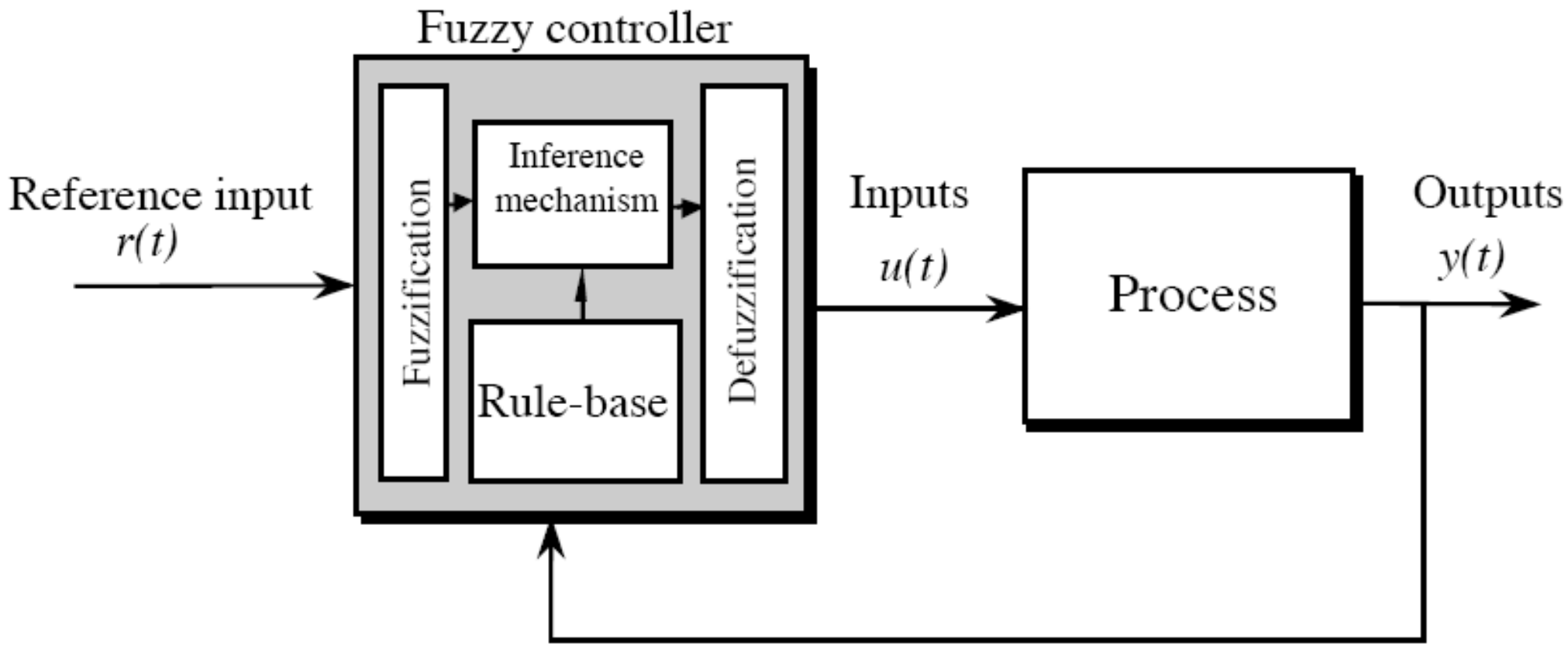
- Abordarea bazată pe fuzzy logic pentru rezolvarea problemelor de control este foarte indicată în sistemele foarte complexe, neliniare și care prezintă incertitudini pentru parametrii interni sau de intrare
- Un controller fuzzy poate fi văzut ca un sistem expert de timp real care folosește logica fuzzy pentru a analiza raportul intrare/ieșire al sistemului.
- Controllerele fuzzy pun la dispoziție o metodă pentru convertirea unei strategii de control empirice și specificate doar la nivel lingvistic (ex. “dacă sună sirena atunci apasă butonul roșu”) într-o strategie automată de control care poate să furnizeze evoluția în timp a sistemului controlat și să facă o estimare a performanțelor acestuia

Controlul Fuzzy

- Elementele unui controller fuzzy:

1. Un *set de reguli* (reguli If-Then) ce reprezintă cuantificarea descrierii lingvistice a expertului despre cum se poate obține un control bun asupra sistemului.
2. Un *mecanism de inferență* (motor de inferență, modul de inferență fuzzy) care emulează procesul prin care expertul ia decizii prin interpretarea și aplicarea cunoștintelor despre cum trebuie să fie controlat sistemul.
3. O interfață de *fuzzificare* – convertește datele de la intrările controllerului într-o formă în care mecanismul de inferență poate să activeze și să aplice anumite reguli.
4. O interfață de *defuzzificare* – convertește concluziile mecanismului de inferență în comenzi și date de intrare pentru sistemul controlat.

Controller-ul Fuzzy

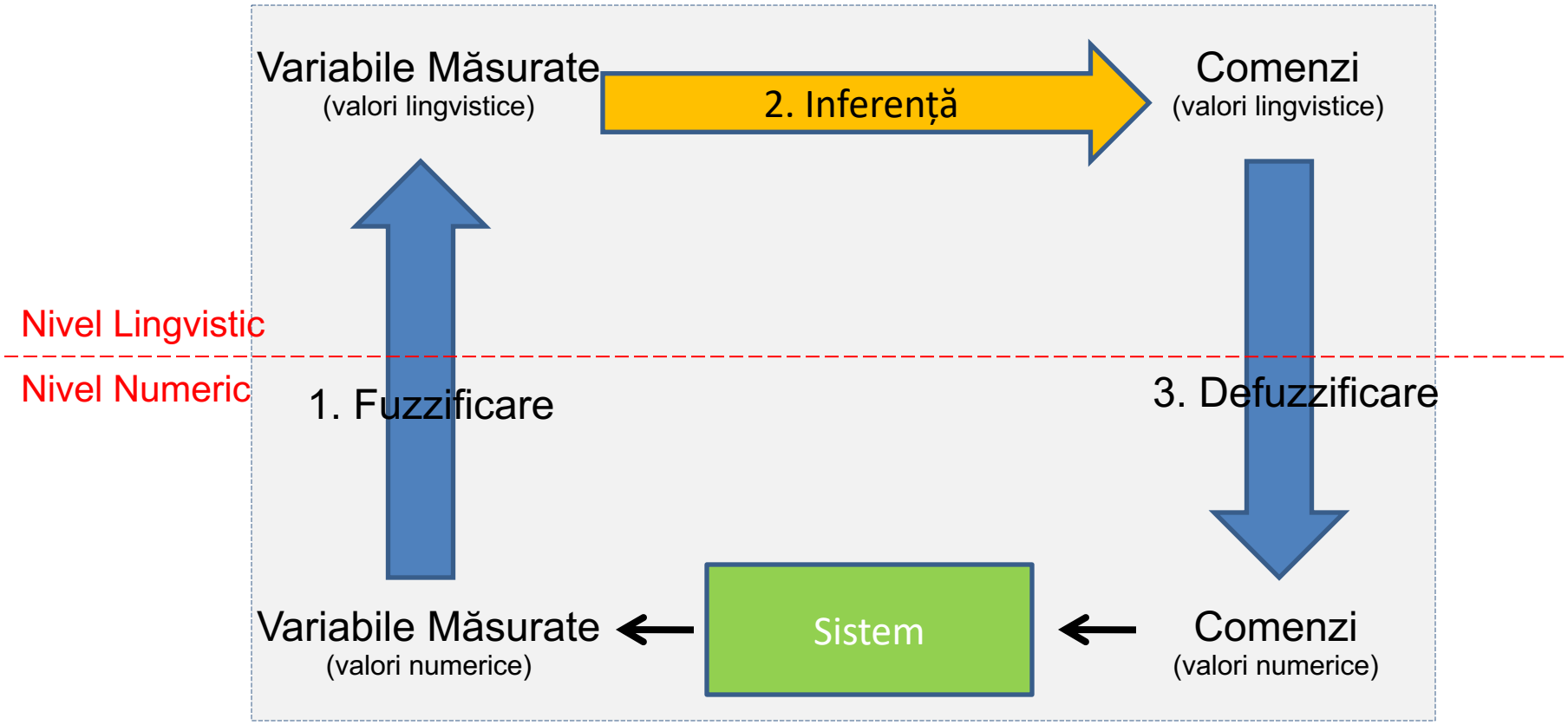


Elementele de Bază ale unui Sistem de Control Fuzzy

Embedded Systems Laboratory

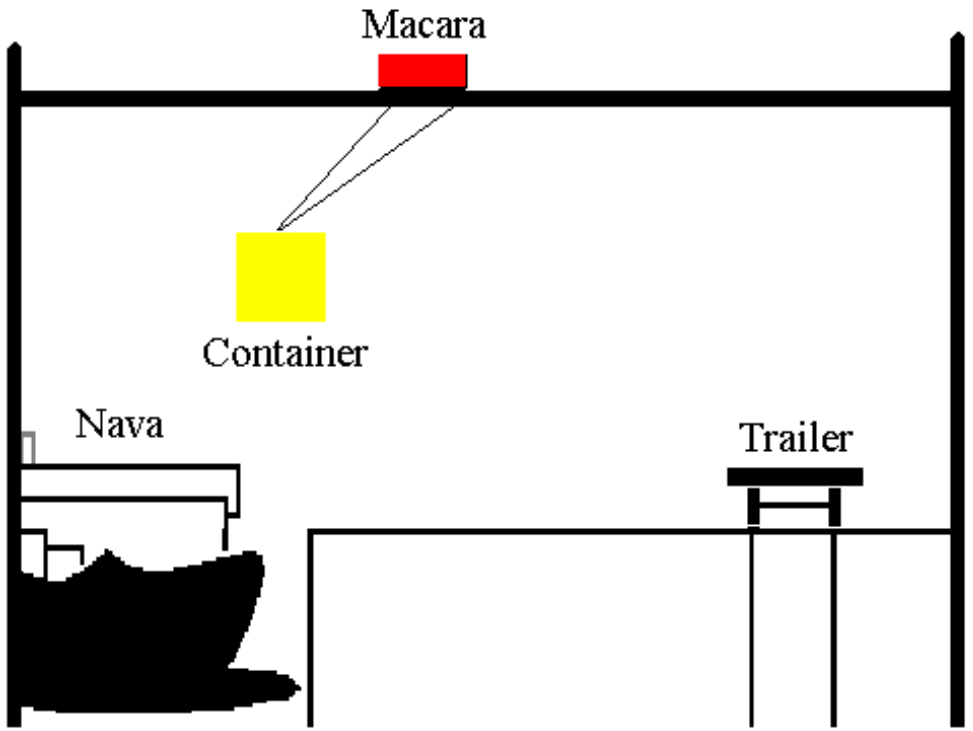
Fuzzificare, inferență, defuzzificare:

Logica Fuzzy definește strategia de control la nivel lingvistic!



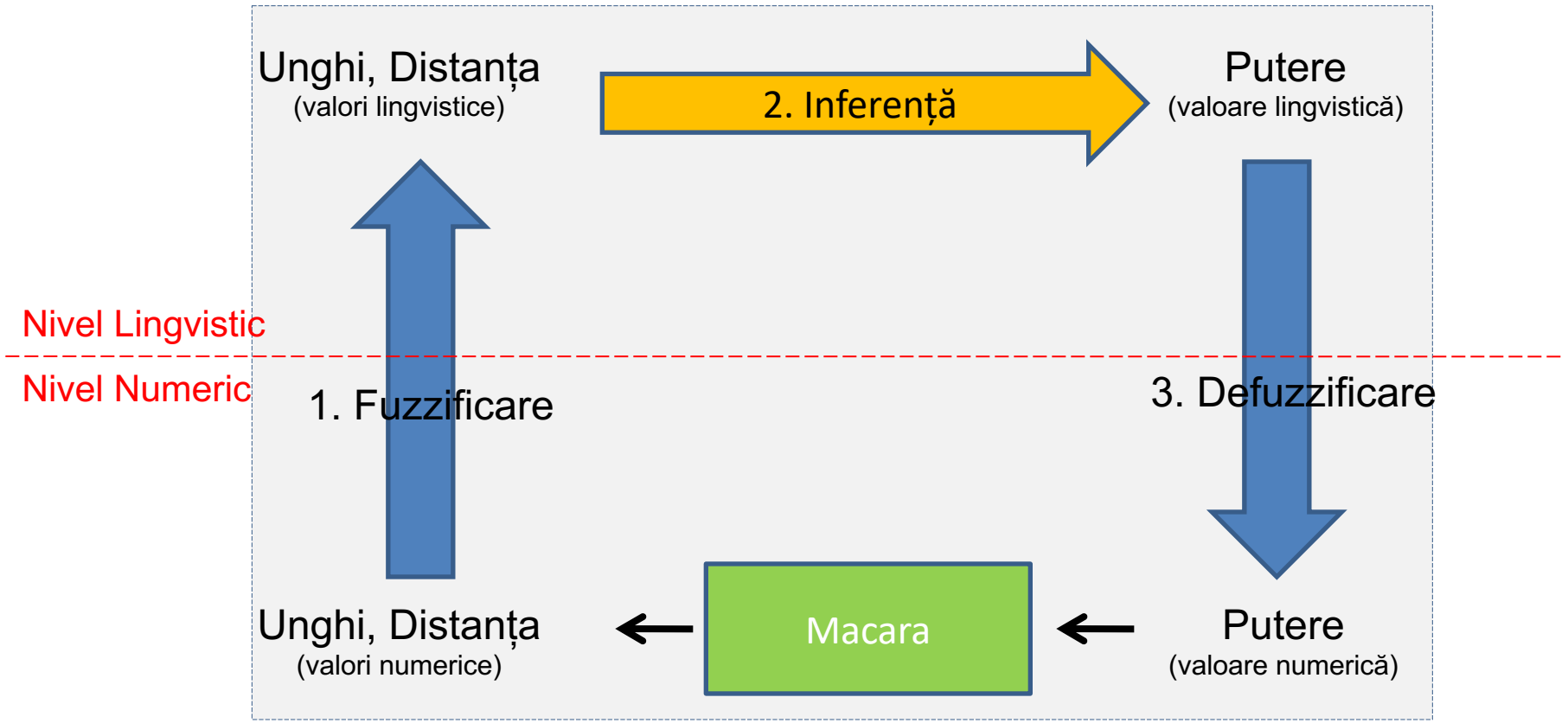
Exemplu: Controlul unei macarale

Două variabile de măsură și o variabilă de comandă



Fuzzificarea Sistemului

Fuzzificare, inferență, defuzzificare:

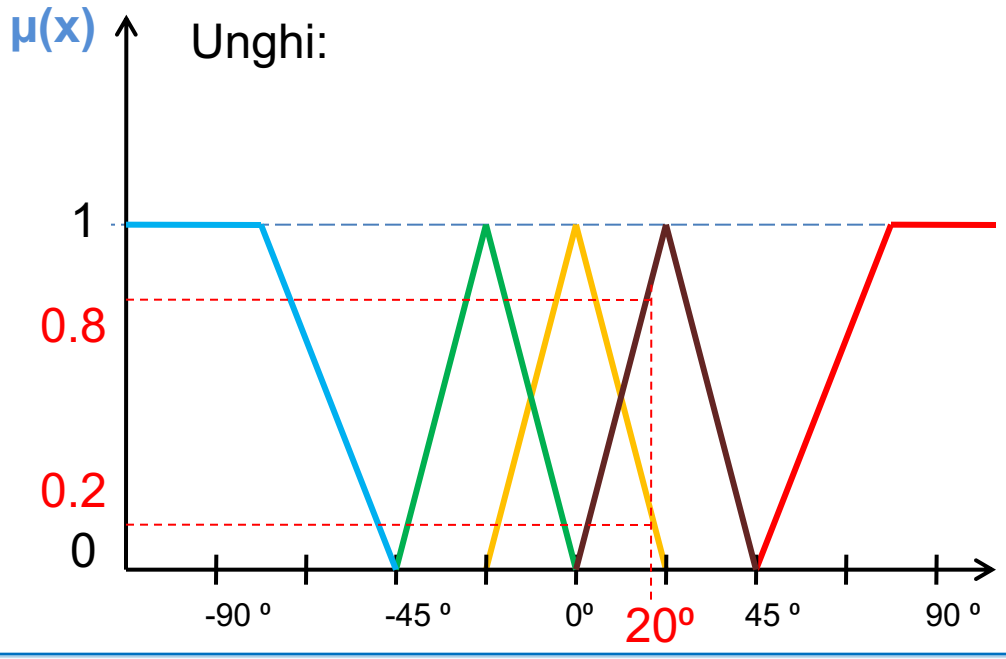


Variabile Lingvistice

Definiția Termenilor:

Variabilele lingvistice sunt "vocabularul" sistemului fuzzy.

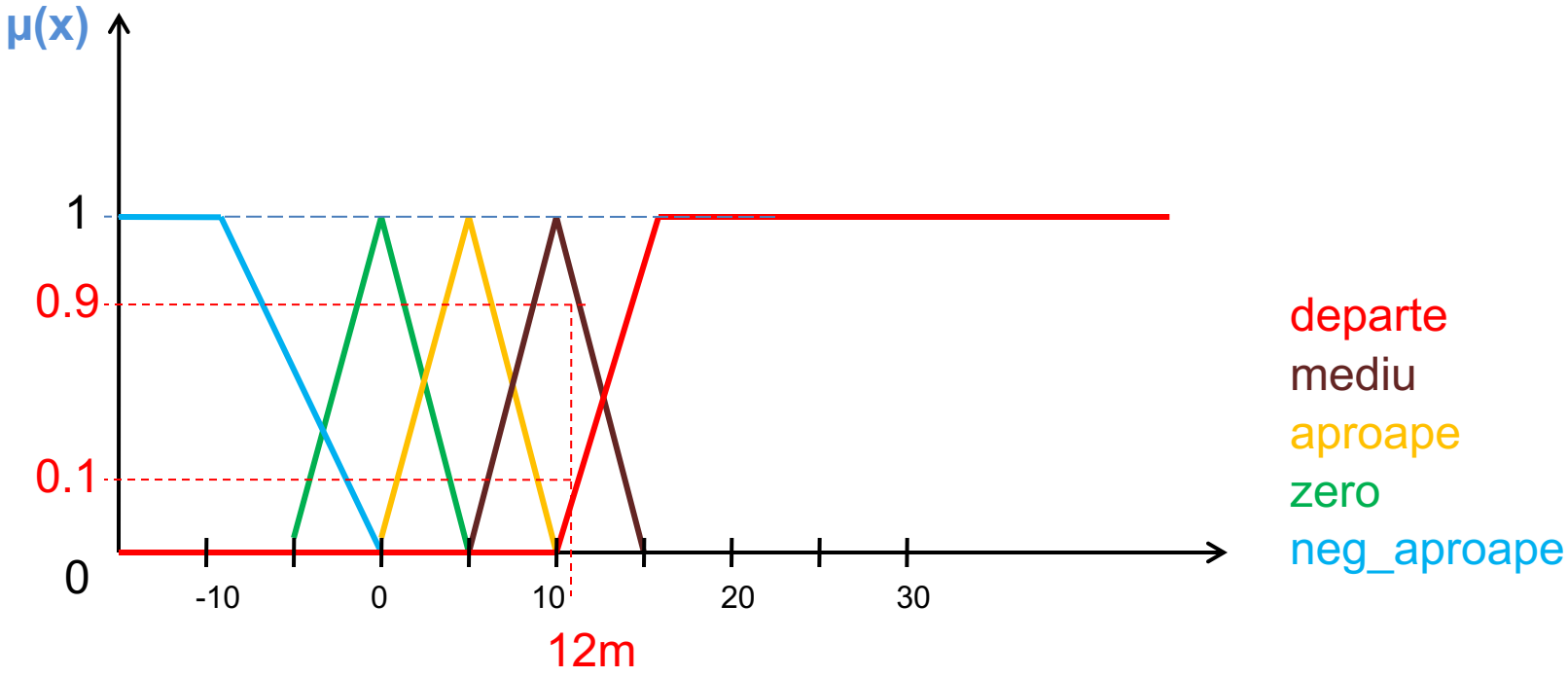
Distanță := {departe, mediu, aproape, zero, neg_aproape}
Unghi := {poz_mare, poz_mic, zero, neg_mic, neg_mare}
Putere := {poz_mare, poz_mediu, zero, neg_mediu, neg_mare}



poz_mare neg_mare
poz_mic neg_mic
zero

Variabile Lingvistice

Distanța:



Inferența Fuzzy: Reguli IF-THEN

Stabilirea regulilor “IF-THEN”:

#1: IF Distanța = medie AND Unghi = poz_mic THEN Putere = poz_mediu

#2: IF Distanța = medie AND Unghi = zero THEN Putere = zero

#3: IF Distanța = departe AND Unghi = zero THEN Putere = poz_mediu

- **Agregare: Evaluarea părții “IF”**
- **Compoziție: Evaluarea părții “THEN”**

Regulile sistemului fuzzy sunt “legile” pe care acesta le execută.

Inferența Fuzzy: Agregare

Embedded Systems Laboratory

Logica Booleană definește operatori doar pentru 0 si 1:

A	B	AvB
0	0	0
0	1	0
1	0	0
1	1	1

Logica Fuzzy oferă o extensie continuă:

- AND: $\mu_{A\&B} = \min\{ \mu_A; \mu_B \}$
- OR: $\mu_{A+B} = \max\{ \mu_A; \mu_B \}$
- NOT: $\mu_{-A} = 1 - \mu_A$

Agregarea părții “IF”

#1: min{ 0.9, 0.8 } = 0.8

#2: min{ 0.9, 0.2 } = 0.2

#3: min{ 0.1, 0.2 } = 0.1

Agregarea calculează cât de potrivită este fiecare regulă pentru situația curentă.

Inferența Fuzzy: Compoziția

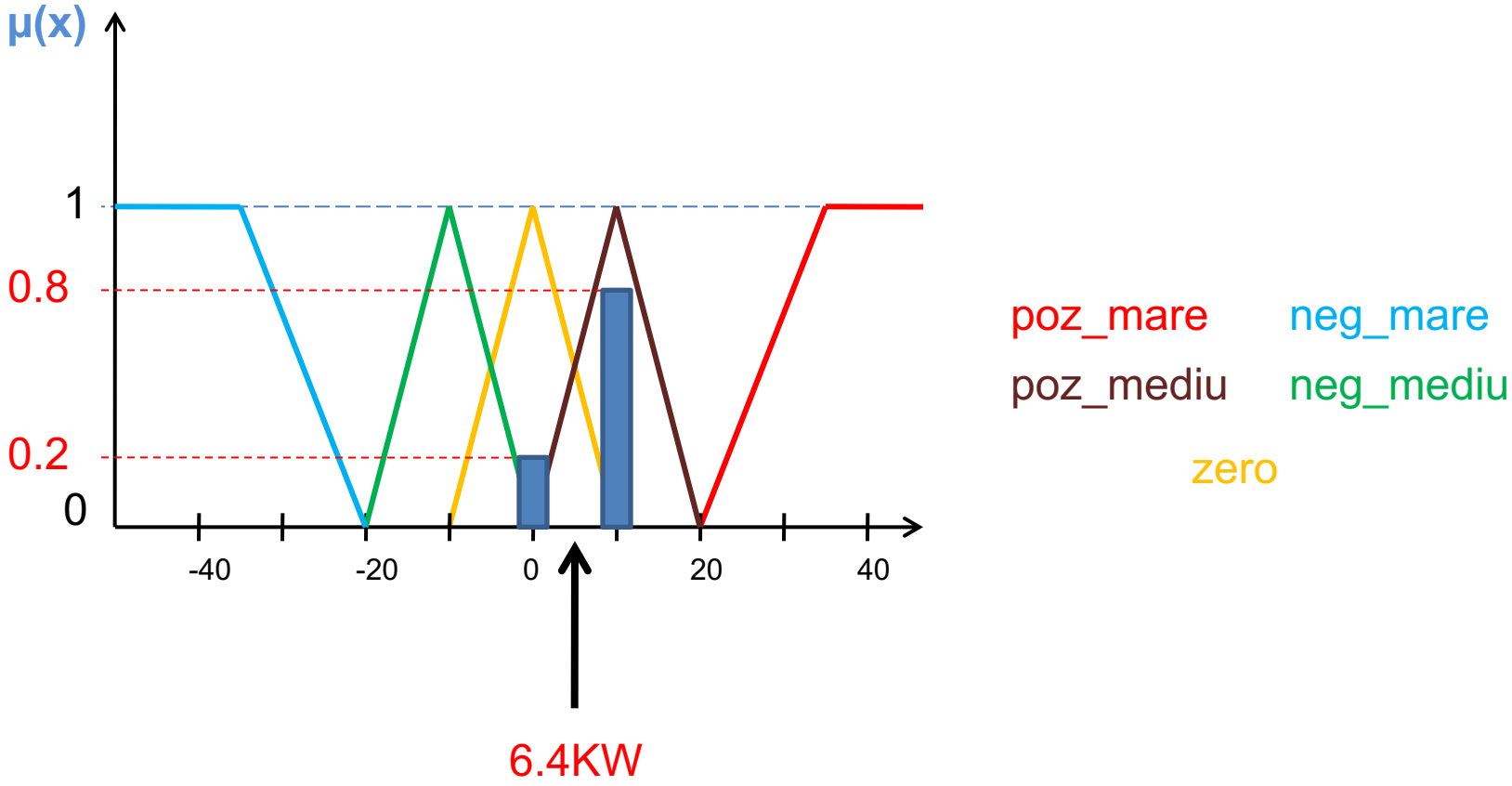
Rezultate pentru variabila lingvistică “Putere”:

poz_mare	**cu gradul 0.0**	
poz_mediu	**cu gradul 0.8**	**(= max{ 0.8, 0.1 })**
zero	**cu gradul 0.2**	
neg_mediu	**cu gradul 0.0**	
neg_mare	**cu gradul 0.0**	

Compozitia estimează cum fiecare regulă modifică rezultatul de la ieșire

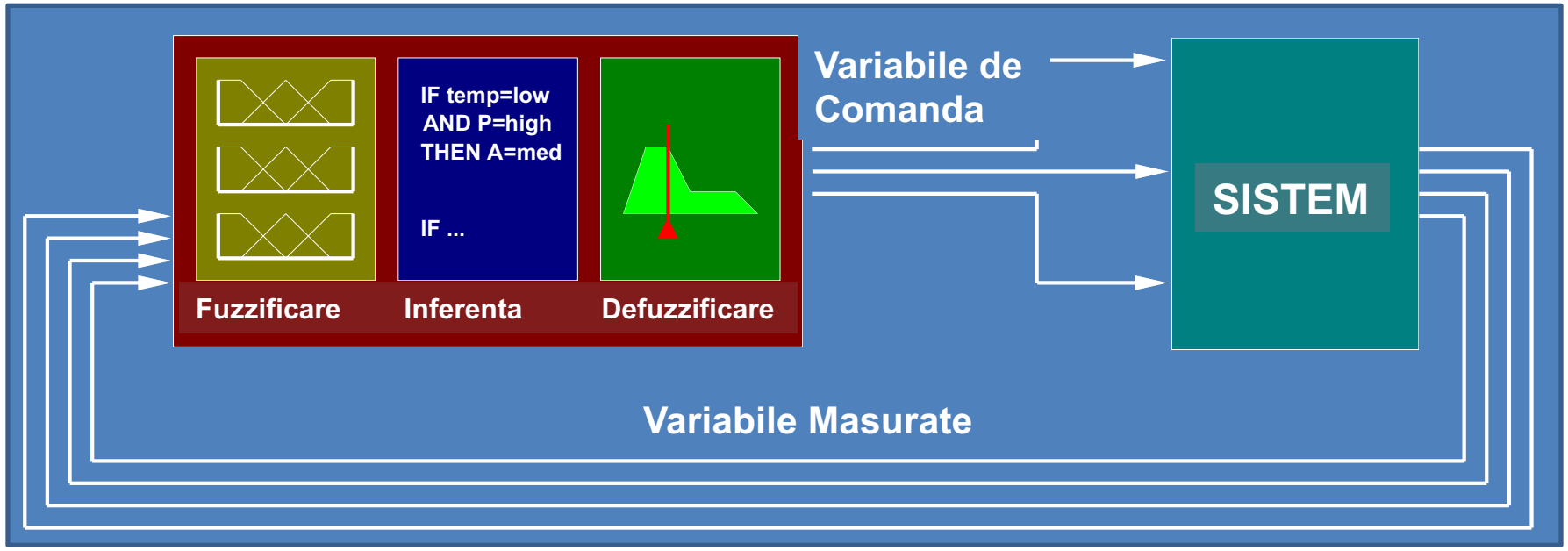
Defuzzificare

Putere: Găsește un compromis folosind o valoare medie ponderată



Tipuri de Control Fuzzy: Controller Direct

Ieșirile de comandă ale sistemului fuzzy logic controlează direct sistemul

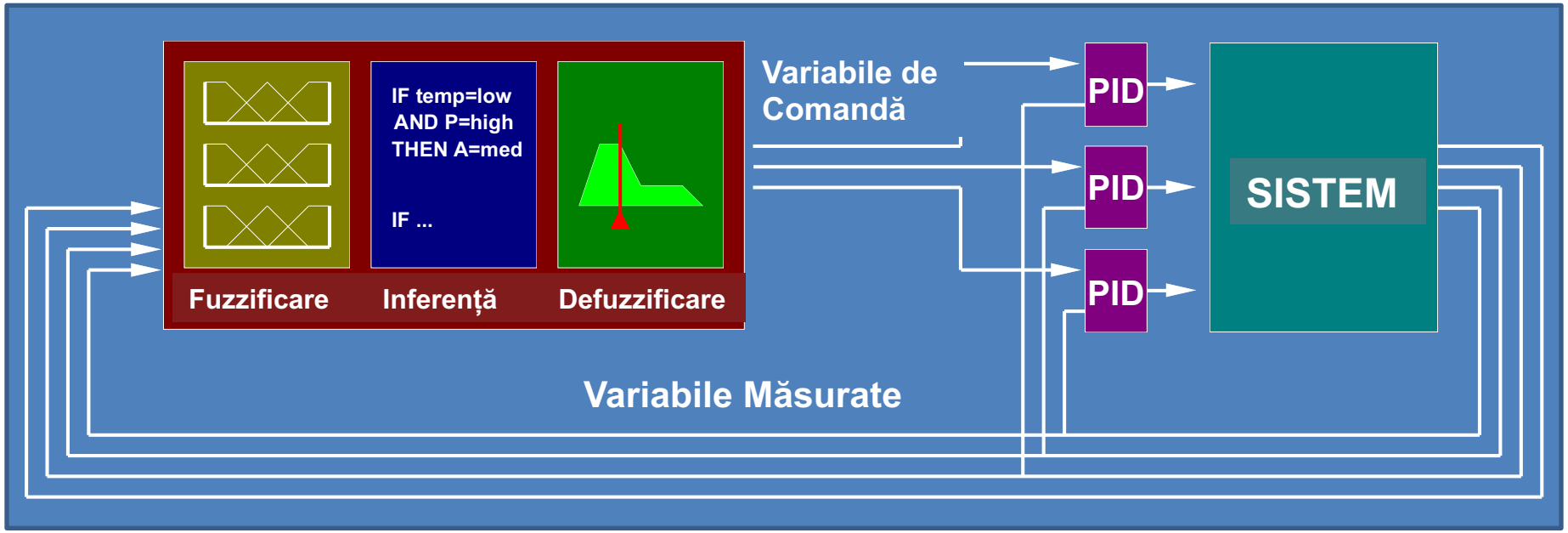


Regulile Fuzzy produc valori exacte!

Tipuri de Control Fuzzy: Control cu supervizare

Embedded Systems Laboratory

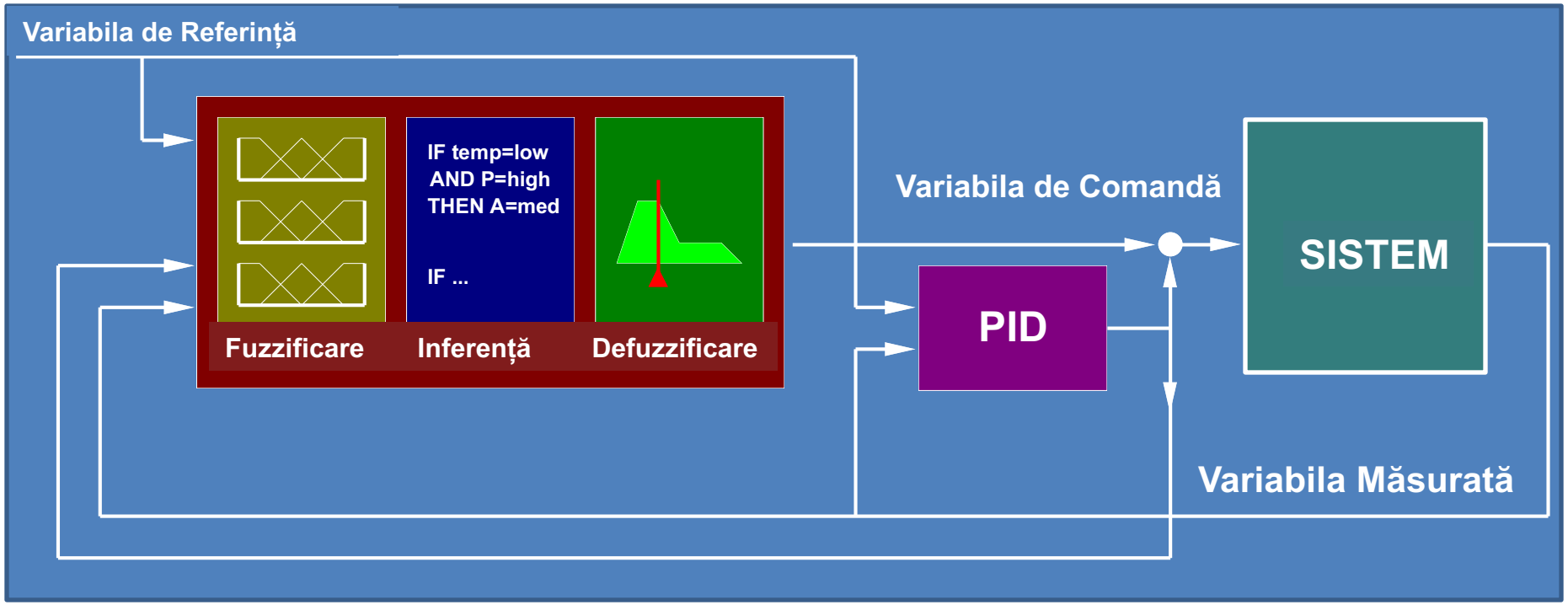
Controllerul Fuzzy Logic produce valori de setare pentru Controllerele PID:



Control de tipul "Operator Uman"

Tipuri de Control Fuzzy: Intervenția Fuzzy

Controllerul Fuzzy Logic și Controllerul PID lucrează în paralel:



Intervenția controlului fuzzy la perturbații majore.

Aplicații ale Controlului Fuzzy

- Stabilizarea imaginii:

  “Daca toți vectorii de mișcare ai unei imagini sunt aproape paraleli și variația lor în raport cu timpul este relativ mică, atunci este detectat tremurul mâinii și direcția de mișcare a mâinii este inversă direcției de mișcare a vectorilor.

Fuzzy Logic: Aplicații

- Business
  - Luarea de decizii
  - Sisteme de data mining
- Chimie
  - Dozarea substanțelor în reacții
  - Reglarea condițiilor de reacție
- Comunicații
  - QoS
  - Filtre adaptative
- Finanțe
  - Managementul fondurilor
  - Previziuni la bursă
- Robotică
  - Controlul efectoarelor
  - Determinarea poziției
- Transporturi
  - Sisteme de transport fără pilot
  - Controlul sistemelor de trafic
- Medical
  - Controlul presiunii arteriale în timpul operației
  - Diagnosticarea cancerului, bolii Alzheimer, diabetului
- Electronică
  - Sisteme de climatizare
  - Sisteme de temporizare: cuptoare, mașini de spălat
- Industrie
  - Controlul temperaturii în furnale
  - Controlul tratamentului apelor curate/uzate
  - Controlul calității