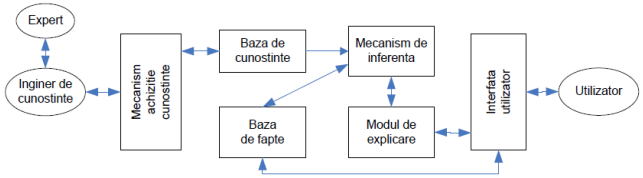
## Structura SBC




Un SBC este format din:

- sistem essential – cuprinde acele module din arhitectura unui SBC obligatorii pentru orice implementare
- module auxiliare

Componentele <u>sistemului esential</u>:

- **baza de cunostinte** – este stocata intotdeauna in memorie permanenta, e echivalenta cu memoria de lunga durata a unui individ uman; contine componente numite piese de cunoastere, care sunt independente cu toate ca intre ele avem legaturi.

Piese de cunoastere: fapte, reguli.

Ordinea pieselor de cunoastere in baza de cunostinte nu conteaza, rezultatul nu este influentat in niciun fel. Piesele de cunoastere pot sa aiba insa atasate si un coeficient de certitudine, adica nu sunt evaluate in totalitate A/F.

- **mecanism de inferenta** – este implementata ca program generic procedural, fiind capabil sa selecteze o piesa de cunoastere pe care sa o utilizeze la un moment dat

Mecanismul de inferenta poate fi de:

- control inainte -> constructie si parcurgere de arbori/grafuri
- constrol inapoi -> constructie si parcurgere de arbori/grafuri

SI/SAU

- **baza de fapte** – similara cu zona de variabile din programarea clasica, este pastrata mereu in memoria temporara

Executia unei inferente are ca efect reactualizarea bazei de fapte.

<u>Modulele auxiliare</u>:

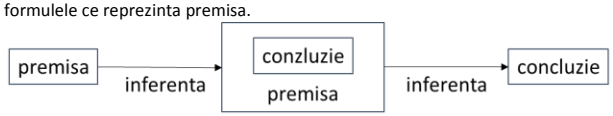
- mecanism de achizitii cunostinte (translator de limbaj) : organizarea bazei de cunostinte, controlul consistentei BC
- modul de explicatii: furnizeaza utilizatorului piesele de cunoastere utilizate, justifica rationamentul efectuat
- interfata utilizator

## Inferente, lant inferential

Regulile de inferenta sunt metode prin care se determina consecintele unor presupuneri despre formule sau enunturi.

O inferenta reprezinta o modalitate de transferare a valorii de adevar a unei formule numita si premisa la valoarea de adevar a altei formula numita si concluzie. Prin inferenta valoarea de adevar a premisei se transfera concluziei.

Un lant inferential este reprezentat de o succesiune de inferente logice in care formulele ce reprezinta concluzia unei inferente se regasesc printre formulele ce reprezinta premisa.




## Cum se produc formulele in limbaje de ordinul I

Orice limbaj natural se construieste pornind de la simboluri primitive (litere, semne de punctuatie etc.)

<u>Termen</u>

Daca a1,..an reprezinta simboluri primitive si f este un simbol functional, atunci f(a1,..an) formeaza un termen.

Daca t1,t2,..,tk sunt termeni, atunci aplicarea f(t1,..,tk) este tot un termen

<u>Formula atomica</u>

Daca t1,t2,..,tn sunt termeni si r un simbol relational => r(t1,..,tn) este formula atomica

<u>Formula</u>

Daca F1,F2 sunt formule atomice atunci F1 ∧ F2, F1 v F2, F1 -> F2 formeaza o formula.

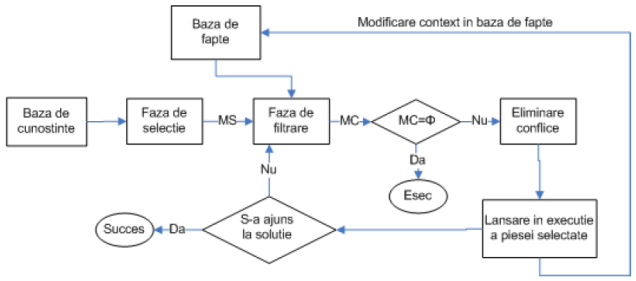
Daca F(x) este o formula atomica in care x este o variabila neinstantiata atunci ∃ xF(x), ∀ xF(x) reprezinta o formula.

## Fazele sistemelor rezolutive

Sisteme rezolutive (mecanism de inferenta)

Un sistem rezolutiv este un program procedural ce construieste si parcurge arbori sau grafuri pentru a putea relationa cu baza de cunostinte

Fazele sistemelor rezolutive:




- **selectie**: faza in care din multimea pieselor de cunoastere aflate in BC se extrage o submultime ce poarta denumirea de multime selectata (MS) ce contine piesele de cunoastere ce au legatura cu formularea problemei (valabil doar daca BC contine si metacunostinte)
- **filtrare**: din MS se extrag piesele de cunoastere ce au premisa adevarata, adica expresiile din premisa se gasesc in baza de fapte . Se obtine multimea candidata (MC) care se mai numeste si multime conflictuala intrucat contine candidatii la executie.
- **eliminare conflicte**: din multimea candidata se extrage piesa de cunoastere
- **lansare in executie**: piesa selectata este lansata in executie si concluziile sale sunt adaugate in baza de fapte, modificand contextul curent al problemei.

SR parcurge ultimele 3 faze in mod ciclic pana in momentul in care s-a demonstrat ce s-a cerut sau pana cand MC devina vida.

## Inconsistenta sistemelor de productii (Consistenta BC)

Un sistem de productie este inconsistent daca pornind dela o baza de fapte initiala se poate obtine validarea si invalidarea unei expresii.

Exemplu:

-Baza de Fapte (constructie=rezistenta, cutie=aspectoasa, conectare=puternica)

-Sistemul de productii:

P1: IF (constructie=rezistenta AND cutie=aspectoasa) THEN (confectie=aluminiu)

P2: IF (cutie=aspectoasa) THEN (surub = cap ingropat)

P3: IF (conectare=puternica)THEN (surub= NOT(cap ingropat))

Se observa ca baza de fapte initiala (constructie=rezistenta, cutie=aspectoasa, conectare=puternica) este consistenta

Aplicarea regulilor produce prin derivare urmatoarea baza de fapte (constructie=rezistenta, cutie=aspectoasa, conectare=puternica, confectie=aluminiu, surub=cap ingropat, surub=NOT(cap ingropat))

Am obtinut concluzii contadictorii, adica adevarat si fals, pentru aceeasi expresie

Obs: O astfel de situatie este rezolvata la rationarea cu incertitudini

## Logica Fuzzy

In logica fuzzy, valorile de adevar ale premiselor nu mai sunt doar A/F. ci pot lua valori pe tot intervalul [0,1].

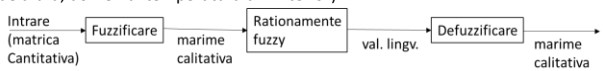
Se defineste o functie de apartenenta $\mu_C = \begin{cases} 1, x \in C \\ 0, x \notin C \end{cases}$ care ne spune ca un obiet apartine unei anumite clase C.

Functia de apartenenta poate lua orice valoare in intervalul [0,1] si este adimensionala.

Prin procesul de fuzzificare vreau sa trec de la o notiune cantitativa (de exemplu o temperatura masurata in °C) la o marime calitativa reprezentata prin intermediul unor clase (ex: clase de tipul rece, cald, foarte cald)

Tot ceea ce judecam dpdv al claselor este determinat de domeniul parametrilor.

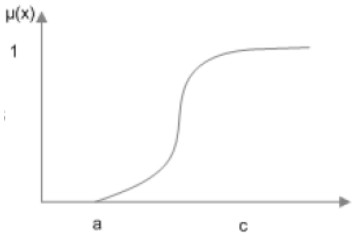
Domeniul poate sa difere in functie de circumstante (domeniul temperaturii de afara, domeniul temperatura din interior)




O valoare lingvistica nu da apartenenta la o clasa, aceasta este data de functia de apartenenta.

Impartirea in clase de face in functie de context si e subiectiva.

Ex: pentru functia de apartenenta in forma standard: S(x,a,c)




- valorile sub 0.5 apartin mai putin clasei respective
- valorile peste 0.5 apartin mai mult clasei

## Tip similitudine

Tip de similitudine = abstractizare fundamentala pe care sunt construite limbajele

σ = <I,K,H,θ,φ>in care:

•I – multimea de indexare a familiilor de relatii

•J – multimea de indexare a familiilor de functii

•K – multimea de indexare pentru elemente distincte

• θ :I→N aplicatia de aritate a familiilor de relatii, cu N multimea numerelor naturale

• φ :J→N aplicatia de aritate a familiilor de functii, cu N multimea numerelor naturale

O relatie stabileste o legatura intre variabilele functiei, pe cand o functie genereaza un nou element.

Peste (∀) tip de similitudine se poate defini un limbaj $L_\alpha$= <A, $R_i^\alpha$ , $F_j^\alpha$ , $e_k^\alpha$ > unde:

- A = multimea simbolurilor primitive
- familia {Riα|iєI} astfel incat pentru fiecare iєI, Riα este o relatie pe A cu aritatea θ(i)
- familia{Fjα|jєJ} astfel incat pentru fiecare jєJ, Fjα este o functie pe A cu aritatea φ(j)
- submultimea {ekα|kєK} a lui A, denumita si multimea elementelor distincte din A

Daca I=Ø, adica structura nu contine simboluri relationale se numeste algebra.

Daca J=Ø, adica structura nu contine simboluri functionale, structura se numeste sistem relational.

Un limbaj de ordinul intai este constituit din simboluri primitive, termeni, formule atomice, formule.

## Retele semantice

O retea semantica este un graf orientat etichetat (nodurile si arcele sunt etichetate)

**Nodurile** corespund unor concepte, instante, subclase.

**Arcele** sunt etichetate cu relatiile dintre noduri.

Legaturi intre noduri:

- relatie de apartenenta intre instanta si concept (este un / este o)
- relatie de incluziune (subset)

Nodurile de nivel superior sunt asociate conceptelor generice, iar cele de baza sunt instante.

**Modelarea evenimentelor in retele semantice**

Prin un eveniment avem:

- primitiva de tip actiune – asociata categoriei gramaticale verb
- primitiva de tip agent – entitate care genereaza un eveniment (subiect)
- primitiva de tip receptor – se refera la alta entitate care participa la eveniment, dar nu il genereaza
- primitiva de tip obiect – entitate asupra careia se desfasoara actiunea de catre agent

## Cadre (frame-uri)

Fiecare nod in modelarea prin cadre asociaza perechi de tipul proprietate-valoare.

Orice cadru are un nume.

Proprietatile aferente fiecarui cadru pot fi mosenire de la cadre de nivel superior prin specificarea tipului, iar pentru celelalte proprietati fie se mostenesc fie se asociaza demoni pentru determinarea valorii

Ex: cadru automobil -> presupunem ca in baza de gaseste si un cadru „mijloc de transport" si in acest mod se produce mostenirea proprietatilor. Daca o proprieatate nu este mostenita va fi specificata explicit in cadrul care defineste obiectul ce apartine clasei. Valoarea unei proprietati pentru o entitate este cea mai apropiata entitatii. In cadre sunt atasate fatete pentru calculul valorilor caracteristicilor. Valoarea poate fi precizata explicit sau cu ajutorul fatetelor.

| Tip | Mijloc de transport |
|---|---|
| Caroserie | Default: Sedan<br>If needed: procedura |
| Motor | Ardere interna |
| Calculator de bord | If-needed:<br>If-removed: procedura |

In cadru sunt atasate fatete (demoni) pentu calculul valorilor caracteristicilor.

Cadrele sunt specifice pentru reprezentarea cunostintelor stereotipice (cele care se refera la clase de obiecte, sunt caracterizate de aceleasi proprietati.

Cadrele imbina cunoasterea declarativa (continutul initial obtinut prin mostenire) si procedurala prin demoni.

**Fatete**

•Default – valoarea asteptata pentru o caracteristica

•If-needed - la aparitia unei probleme la folosirea fatetei default se executa o procedura care pe baza informatiilor din baza poate determina valoarea caracteristicii

•If-added – activata la introducerea valorii caracteristicii la care este atasata

•If-removed – activata la indepartarea valorii caracteristicii

Procedurile atasate fatetelor cadrelor se numesc *demoni* si intra in executie la momentul corespunzator. Este o programare orientata pe evenimente pentru ca demonii se activeaza la evenimentele care determina modificarea caracteristicilor.

Valorile caracteristicilor se obtin prin mostenire, valoarea implicita sau declansarea demonilor ce modifica valorile.

## Cautarea optimala in arbori si grafuri

Se pot stabili metrici prin care se evalueaza costul sau eficienta atingerii obiectivului.

Daca metrica se defineste ca un cost => minimizare / ca eficienta => maximizare.

Fie n un nod al unui drum f(n) = g(n) + h(n).

f(n) = costul drumul de la START la obiectiv

g(n) = costul drumului de la nodul de start la nordul n

h(n) = costul frumul de la nodul n la obieciv

Cautarea optimala se bazeaza doar pe costul drumului parcurs, adica componenta g(n).

O tranzitie in arbore de la un nod n la un nod fiu n1 incarca costul cu c(n,n1), adica executia piesei de cunoastere ce produce tranzitia. Ca urmare g(n1)=g(n)+c(n,n1)

Un nod este definit prin nod = {stare, pointer la parinte, g(n)}

**Cautarea optimala (algoritm)**

```
Initializare stare initiala, stare obiectiv
Open_list={nod_start(0)}
Closed_list=Ø
while (open_list≠Ø) {
  n=prim_nod(open_list)
  remove(n, open_list)
  add(n, closed_list)
  if(obiectiv(n)) exit(succes) //testeza potrivire
  obiectiv
  expand(n)//genereaza toate nodurile fiu ni
  calculeaza g(ni) // g(ni)=g(n)+c(n,ni)
  add(ni,open_list)
  ordonare(open_list,crescator)}
exit(esec)
```

## CF (coeficientul de certitudine)

Se definesc doua marimi:

1) masura increderii procedurii lui H cand E s-a produs

$$MB(H,E) = \begin{cases} 1, & P(H) = 1 \\ \dfrac{\max\big(P(H,E),P(H)\big) - P(H)}{1 - P(H)}, & P(H) < 1 \end{cases}$$

2) masura neincrederii producerii lui H cand E s-a produs

$$MD(H,E) = \begin{cases} 1, & P(H) = 0 \\ \dfrac{\max\big(P(H,E),P(H)\big) - P(H)}{-P(H)}, & P(H) > 0 \end{cases}$$

CF(H,E) = MB(H,E) – MD(H,E)

Cazuri posibile:

1. P(H,E) = 1, H sigur daca E s-a produs

$MB(H,E) = \frac{1-P(H)}{1-P(H)} = 1$

$MD(H,E) = \frac{P(H)-P(H)}{-P(H)} = 0$ => CF = 1

2. P(~H|E)=1, ~H este sigur daca E s-a produs. P(H|E)=0

$MB(H,E) = \frac{P(H)-P(H)}{1-P(H)} = 0$

$MD(H,E) = \frac{-P(H)}{-P(H)} = 1$ => CF = -1

3. P(H|E)=P(H), H nu este influentat de producerea lui E

$MB(H,E) = \frac{P(H)-P(H)}{1-P(H)} = 0$

$MD(H,E) = \frac{P(H)-P(H)}{-P(H)} = 0$ => CF = 0

4. Evidenta pozitiva P(H|E)>P(H), H favorizat de evenimentul E

$MB(H,E) = \frac{P(H|E)-P(H)}{1-P(H)} < 0$

$MD(H,E) = \frac{P(H)-P(H)}{-P(H)} = 0$ => CF = MB(H,E) > 0

5. Evidenta negativa P(H)>P(H|E), H este defavorizat de E

$MB(H,E) = \frac{P(H)-P(H)}{1-P(H)} = 0$

$MD(H,E) = \frac{P(H|E)-P(H)}{-P(H)} > 0$ => CF = -MD(H,E) < 0