10. REȚELE SEMANTICE

O rețea semantică utilizează grafuri pentru reprezentarea cunoştinţelor. Conceptul de rețea semantică a fost introdus aproximativ în perioada 1965-1970. Scopul primelor rețele semantice a fost reprezentarea cuvintelor din limbajul natural.

O rețea semantică are forma unui ansamblu de noduri şi arce, orientate şi etichetate. Nodurile reprezintă obiectele. Un obiect poate fi un concept abstract sau particular, un atribut, ş.a.m.d. Arcele sunt utilizate pentru a reprezenta legăturile care există între aceste obiecte.

Primele tipuri de rețele semantice au utilizat un raționament bazat pe proprietatea de moştenire. Mai precis, acest raționament poate fi descris în felul următor:

- orice x este y; orice y este z; atunci orice x este z.
- x este y; orice y este z; atunci x este z.

10.1. Rețele semantice simple

O ***relație binară*** pe o mulțime X este o submulțime $\rho \subset X \times X$. Dacă $\rho = \varnothing$, atunci ρ se numeşte ***relație vidă***.

Compunerea a două relații ρ_1 şi ρ_2 pe X este o relație pe X notată cu $\boldsymbol{\rho_1 \circ \rho_2}$ şi definită prin:

$$(x,y) \in \rho_1 \circ \rho_2 \overset{def}{<=>} (\exists)\ z \in X \text{ a.î. } (x,z) \in \rho_1 \text{ şi } (z,y) \in \rho_2$$

Compunerea relațiilor este o operație asociativă (i.e. $(\rho_1 \circ \rho_2) \circ \rho_3 = \rho_1 \circ (\rho_2 \circ \rho_3)$ pentru orice relații ρ_1, ρ_2 şi ρ_3). Compunerea a două relații poate fi relația vidă fără ca vreuna din cele două relații să fie vidă, după cum se observă din exemplul următor:

$$\rho_1 = \{(x,y), (x, z)\}\ ,\ \rho_2 = \{(x,v)\},\ \rho_1 \circ \rho_2 = \varnothing$$

Se numeşte ***graf etichetat*** un cvadruplu $G = (X, L_0, T_0, f_0)$, unde

- X, L_0 sunt mulțimi finite cu proprietatea că $X \cap L_0 = \varnothing$
- T_0 este o mulțime de relații binare pe X cu proprietatea că $\varnothing \notin T_0$.
- $f_0 : L_0 \rightarrow T_0$ este o funcție surjectivă

Elementele lui X se numesc ***etichete de noduri***, iar elementele lui L_0 se numesc ***etichete de arce***. Un graf etichetat se reprezintă grafic după cum urmează:

- fiecare nod este reprezentat printr-un dreptunghi ce conține eticheta nodului
- un nod $x \in X$ este legat de un nod $y \in X$ printr-un arc etichetat cu a dacă şi numai dacă $(x,y) \in f_0(a)$ (figura 1)

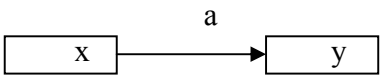

figura 1

Se consideră că fiecare element din X etichetează un nod şi numai unul. Cu alte cuvinte corespondența dintre mulțimea de noduri şi X este bijectivă. Spunem că $a \in L_0$ este o etichetă a relației binare $f_0(a) \in T_0$. Surjectivitatea funcției f_0 implică faptul că fiecare relație din T_0 are cel puțin o etichetă.

Dacă $T_0 \subset 2^{X \times X}$ este o mulțime de relații binare pe mulțimea finită X, atunci se notează cu ***$C(T_0)$*** cea mai mică (în sensul relației de incluziune) mulțime $A \subset 2^{X \times X}$ cu următoarele proprietăți:

- $T_0 \subset A$
- dacă $\rho_1, \rho_2 \in A$ şi $\rho_1 \circ \rho_2 \neq \emptyset$, atunci $\rho_1 \circ \rho_2 \in A$

O ***rețea semantică simplă*** este un tuplu (G, T, L, f, φ, g, S), unde

- $G = (X, L_0, T_0, f_0)$ este un graf etichetat
- $L_0 \subset L$ şi $\varphi : L \times L \to L$ este o aplicație parțial definită
- $T_0 \subset T \subset C(T_0)$ şi T are proprietatea că $\rho \in T$ dacă şi numai dacă există un număr natural n şi un tuplu de relații, $(\rho_1, \rho_2, \ldots, \rho_n)$ a.î. $\rho = \rho_n$ şi pentru fiecare $i = \overline{1,n}$ fie relația $\rho_i \in T_0$ fie există $u < n$ şi $v < n$ a.î. $\rho_i = \rho_u \circ \rho_v$.
- $f : L \to T$ este o extensie surjectivă a funcției f_0 a.î. dacă (a,b) aparține domeniului lui φ atunci $f(\varphi(a,b)) = f(a) \circ f(b)$
- S este o mulțime care defineşte ***spațiul semantic***
- $g : X \times L \times X \to S$ este o aplicație numită ***aplicația semantică***

În general, deoarece răspunsul la o interogare se dă în limbajul natural, spațiul semantic S este o mulțime de propoziții în acest limbaj. Intuitiv, elementul g(x,e,y) specifică semantica faptului că e este eticheta pentru o anumită relație binară care conține perechea (x,y).

Pentru o rețea semantică simplă, (G, T, L, f, φ, g, S), răspunsul la o interogare este dat de aplicația

$$Ans : X \times X \to 2^S \cup \{unknown\} ,$$

definită prin:

$$Ans(x, y) = \begin{cases} unknown, \text{ dacă } deduc(x, y) = \varnothing \\ deduc(x,y), \text{ altfel} \end{cases}$$

unde

$$deduc(x, y) = \{g(x, \varphi(\ldots\varphi(e_1,e_2)\ldots,e_n) ,y\} | \; d = [(x_1, e_1, x_2), (x_2, e_2, x_3), \ldots, (x_{n-1}, e_{n-1}, x_n)] \text{ este un drum în G cu } x_1 = x \text{ şi } x_n = y\}$$

(semnificația unui triplet (x_i, e_i, x_{i+1}) este următoarea: (x_i, x_{i+1}) este arc în graful G, iar e_i este eticheta arcului respectiv).

Vom considera un exemplu de rețea semantică pentru a clarifica noțiunile de mai sus. Vom construi o rețea semantică asociată operației "CICLOP S.A. cumpără acțiuni ASTRA" (cf. exemplului din [14]/p. 46). Graful etichetat asociat rețelei este următorul:

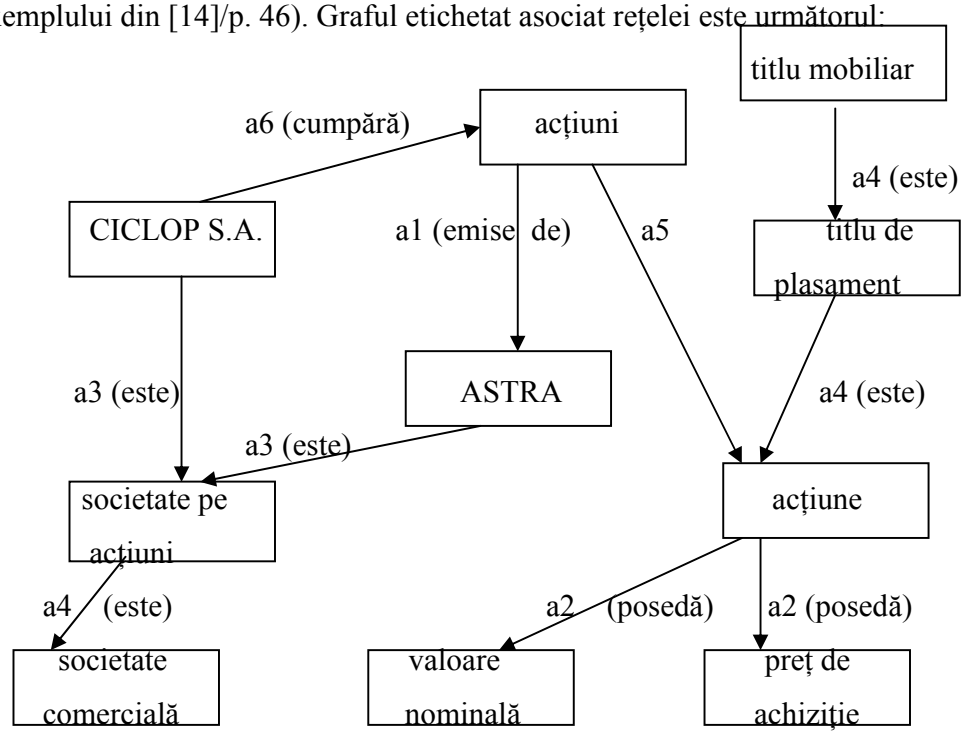

figura 2

Alături de etichetele arcurilor s-au trecut (în paranteză) şi semnificațiile legăturilor dintre noduri. Notăm $G = (X, L_0, T_0, f_0)$ graful etichetat de mai sus. Mulțimea obiectelor este

X = {“CICLOP S.A.”, ASTRA”, “acțiuni”, ”acțiune”, ”valoare nominală”,
”preț de achiziție”, “societate pe acțiuni”, “societate comercială”,
”titlu mobiliar”, ”titlu de plasament”}

Mulțimea relațiilor este

$$T_0 = \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6\}$$

unde

ρ_1 = {(“acțiuni”,”ASTRA”) }

ρ_2 = {(“acțiune”,”valoare nominala”), (“acțiune”, ”preț de achiziție”) }

ρ_3 = {(“CICLOP S.A.”, “societate pe acțiuni”), (“ASTRA”, “societate pe acțiuni”)}

ρ_4 = {(“societate pe acțiuni”,”societate comercială”),
(“titlu mobiliar”, “titlu de plasament”), (titlu de plasament”, “acțiune”)}

ρ_5 = {(“acțiuni”,”acțiune”) }

ρ_6 = {(“CICLOP”, “acțiuni”) }

Se observă apartenența unei perechi (x,y) la o relație binară din cele de mai sus are o anumită semnificație. De exemplu, faptului că (x, y) aparține ρ_3 i se asociază semnificația “x este y”, faptului că (x, y) $\in \rho_4$ i se asociază semnificația “orice x este y”, iar pentru (x,y) $\in \rho_6$ semnificația este “x este pluralul lui y”. Mulțimea etichetelor este

$$L_0 = \{a_1, a_2, a_3, a_4, a_5, a_6\}$$

Funcția $f_0 : L_0 \to T_0$ este definită prin $f_0(a_i) = \rho_i$ pentru orice $i = \overline{1,6}$. Intuitiv fiecare etichetă a indică o anumită semnificație pentru perechile conținute în relația $f_0(a)$. În consecință, trebuie combinate aceste semnificații pentru a obține proprietăți noi ale perechilor de obiecte din X, proprietăți care nu sunt specificate direct. De exemplu trebuie să obținem proprietăți de forma “ASTRA este societate comercială” sau “titlu mobiliar posedă preț de achiziție”.

Continuăm cu definirea celorlalte elemente ale rețelei semantice (G, T, L, φ, g, S). Considerăm $L = L_0 \cup \{a_7\}$. Eticheta a_7 este ataşată semnificației “ x este client pentru y”. Cu ajutorul funcției φ se precizează ce se obține prin compunerea relațiilor asociate etichetelor. Astfel φ(a3,a2) = a2 (i.e. dacă x este y şi y posedă z atunci x posedă z), φ(a3, a4, a3) (i.e dacă x este y şi orice y este z atunci x este z), ş.a.m.d. Funcția $g : X \times L \times X \to S$, este definită după cum urmează:

g(x,a1,y) =”x emise de y"

g(x,a2,y) =" x poseda y"

g(x,a3,y) =" x este y"

g(x,a4,y) =" orice x este y"

g(x,a5,y) =" x este pluralul lui y"

g(x,a6,y) =" x cumpără y"

g(x,a7,y) =" x clientul lui y"

10.2. Implementarea în PROLOG a rețelelor semantice simple

Fie (G, T, L, φ, g, S) o rețea semantică simplă. Prezentăm implementarea în PROLOG a acestei rețele. Funcției g i se asociază predicatul semantica(X, Y, A) care determină afişarea lui g(X,A,Y). Regulile asociate predicatului semantica vor fi memorate într-un fişier "sem.pro" ce trebuie inclus în programul care implementează rețelele semantice. Conținutul acestui program pentru exemplul anterior este:

```
predicates
   semantica(nod,nod,nod)
clauses
     semantica(X,Y,"a1"):-write(X," emise de ",Y,"\n").
     semantica(X,Y,"a2"):-write( X," poseda ",Y,"\n").
     semantica(X,Y,"a3"):-write(X," este ",Y,"\n").
     semantica(X,Y,"a4"):-write("orice ", X," este ",Y,"\n").
     semantica(X,Y,"a5"):-write(X, " este pluralul de la ",Y,"\n").
     semantica(X,Y,"a6"):-write(X," cumpara ", Y,"\n").
     semantica(X,Y,"a7"):-write(X," este clientul lui ", Y,"\n").
```

Graful etichetat poate fi reprezentat prin intermediul unei baze de date dinamice. Faptele din această baza dinamică sunt de forma d(x,y,a) (cu semnificația (x, y) este arc în graf etichetat cu eticheta a) şi fi(a, b, c) (cu semnificația φ(a, b) = c). Conținutul de fapte al bazei dinamice asociate rețelei anterioare este următorul

```
d("actiuni","ASTRA","a1")
d("actiuni","actiune","a5")
d("actiune","pret de achizitie","a2")
d("actiune","valoare nominala","a2")
d("ASTRA","societate pe actiuni","a3")
d("societate pe actiuni","societate comerciala","a4")
d("titlu mobiliar","titlu de plasament","a4")
d("titlu de plasament","actiune","a4")
d("CICLOP S.A.","actiuni","a6")
d("CICLOP S.A.","societate pe actiuni","a3")
fi("a3","a2","a2")
fi("a4","a2","a2")
fi("a4","a4","a4")
fi("a3","a4","a3")
fi("a5","a2","a2")
fi("a5","a3","a3")
fi("a5","a4","a4")
fi("a6","a1","a7")
fi("a6","a5","a6")
fi("a1","a3","a1")
fi("a1","a4","a1")
```

Următorul program realizează un motor de inferență utilizând o rețea semantică simplă.

```
domains
  nod=string
  lista_noduri=nod*
include "sem.dat"
database
  fi(nod,nod,nod)
  d(nod,nod,nod)
predicates
```

```
  apartine(nod,lista_noduri)
  ultim_elem(lista_noduri,nod)
  nr_elem(lista_noduri,integer)
  legatura(nod,nod)
  drum(nod,nod,lista_noduri,lista_noduri)
deduc(lista_noduri,nod)
compun(integer,lista_noduri,nod)
explic(lista_noduri,lista_noduri,char)
  start
  fer1
  fer2
  fer3
  ferestre
  sterge_ferestre
  nume_baza_dinamica
  interogare
  continua(char)
 clauses
start:-ferestre,nume_baza_dinamica,interogare,sterge_ferestre.
ferestre:-fer1,fer2,fer3.
sterge_ferestre:-shiftwindow(1),removewindow,
shiftwindow(2),removewindow,shiftwindow(3),removewindow.
fer1:-makewindow(1,30,64,"BAZA DE DATE DINAMICA...",10,10,3,60).
fer2:-makewindow(2,30,64,"OBIECTELE...",9,10,5,60).
fer3:-makewindow(3,113,37,"********",0,0,25,80).
nume_baza_dinamica:-shiftwindow(1),write("Baza de date dinamica: "),
                    readln(M),consult(M).
interogare:-shiftwindow(2),write(" Obiect 1: "),readln(X),
          write(" Obiect 2: "),readln(Y),
          shiftwindow(3),legatura(X,Y),readchar(_),clearwindow,
          shiftwindow(2),clearwindow,continua(R),
          R<>'n',clearwindow,interogare.
```

```
interogare.
  continua(X):-write("Continuati? [d|n]: "),readchar(X).
legatura(X,Y):-drum(X,Y,[],[X]),write("\n\nAlte legaturi...\n"),fail.
legatura(_,_):-write("Nu (mai) exista legaturi.
                                        Apasati orice tasta pentru a  continua...").
drum(X,X,E,[H|L]):-deduc(E,K),ultim_elem(L,U),semantica(U,H,K),
          write("\nExplicatii?[d/n]: \n"),
          readchar(R),explic(E,[H|L],R).
drum(X,Y,E,L):-d(X,T,Z),not(apartine(T,L)),
        drum(T,Y,[Z|E],[T|L]).
deduc(L,K):-nr_elem(L,N),compun(N,L,K).
compun(1,[X],X).
compun(N,[U,V|L],K):-N>=2,fi(V,U,T),N1=N-1,
          compun(N1,[T|L],K).
explic([E1|E],[U,V|L],'d'):-explic(E,[V|L],'d'),
              semantica(V,U,E1).
  explic([],_,'d').
  explic(_,_,_).
  apartine(X,[X|_]).
  apartine(X,[_|Y]):-apartine(X,Y).
  ultim_elem([X],X):-!.
  ultim_elem([_|Y],X):-ultim_elem(Y,X).
  nr_elem([],0).
  nr_elem([_|X],N):-nr_elem(X,N1),N=N1+1.
```

Programul se interoghează astfel

```
Goal: start
```

În încheiere considerăm încă un exemplu. Se presupune că trei agenţii de turism, A, B şi C, organizează excursii pentru vizitarea obiectivelor notate simbolic cu x1, x2,…, x9. În contractul dintre cele trei agenţii se prevede agenţia C nu preia turiştii agenţiilor A sau B. Un

turist doreşte să ştie dacă este posibil să plece din x şi să ajungă în y, şi care sunt traseele posibile. Considerăm următoarele relaţii binare, ρ_1, ρ_2 şi ρ_3, cu semnificaţia $(x, y) \in \rho_1$(resp. ρ_2, resp. ρ_3) dacă şi numai dacă deplasare de la x la y se poate realiza prin agenţia A (resp. B, resp. C). Se presupun cunoscute aceste trei relaţii binare:

ρ_1 ={(x1,x2), (x2,x3), (x2,x6)}

ρ_2 ={(x3,x4), (x4,x5), (x6,x7) , (x7,x8) , (x8,x9) , (x9,x5)}

ρ_3 ={(x1,x3), (x3,x5), (x7,x4)}

Putem rezolva problema cu ajutorul unei reţele semantice. Eticheta a asociată unui arc (x, y) înseamnă că deplasarea de la x la y se face prin agenţia A. O interpretare similară au etichetele b şi c. Graful etichetat asociat este:

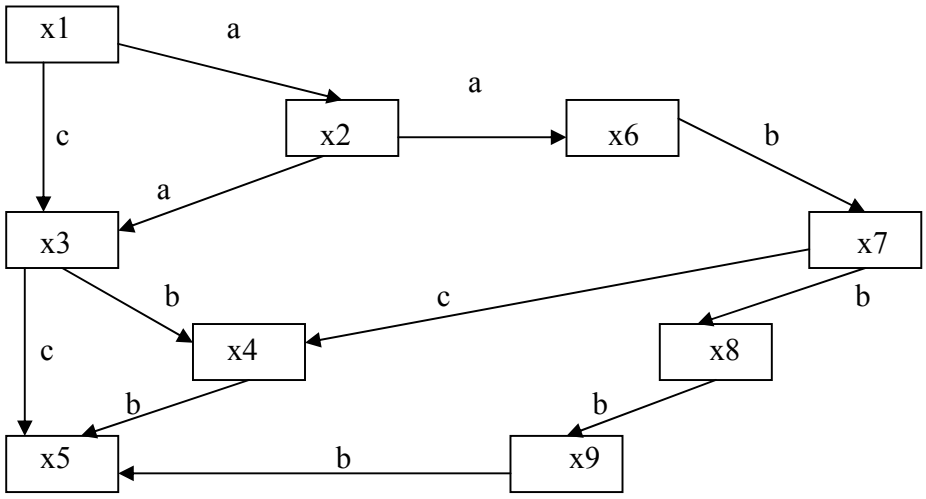

figura 3

Baza dinamică asociată acestui graf trebuie să aibă următorul conţinut:

```
d("x1","x2","a")
d("x2","x3","a")
d("x2","x6","a")
d("x3","x4","b")
d("x4","x5","b")
```

```
d("x6","x7","b")
d("x7","x8","b")
d("x8","x9","b")
d("x9","x5","b")
d("x1","x3","c")
d("x3","x5","c")
d("x7","x4","c")
fi("a","a","a")
fi("b","b","b")
fi("c","c","c")
fi("a","b","ab")
fi("a","ab","ab")
fi("b","a","ab")
fi("b","ab","ab")
fi("c","b","cb")
fi("cb","b","cb")
fi("c","cb","cb")
fi("cb","a","cba")
fi("c","cba","cba")
fi("cba","a","cba")
fi("cba","b","cba")
fi("c","a","ca")
fi("c","ca","ca")
fi("ca","a","ca")
fi("ca","b","cba")
```

Fişierul “sem.pro” care conține descrierea funcției semantice este următorul

```
predicates
        semantica(nod,nod,nod)
clauses
```

```
semantica(X,Y,"a"):-write(X," ---> ",Y," prin agentia A\n").
semantica(X,Y,"b"):-write(X," ---> ",Y," prin agentia B\n").
semantica(X,Y,"c"):-write(X," ---> ",Y," prin agentia C\n").
semantica(X,Y,"ab"):-write(X," ---> ",Y," prin agentiile A si B\n").
semantica(X,Y,"cba"):-write(X," ---> ",Y," prin agentiile C, A si B\n").
semantica(X,Y,"ca"):-write(X," ---> ",Y," prin agentiile C si A\n").
semantica(X,Y,"cb"):-write(X," ---> ",Y," prin agentiile C si B\n").
```

Să presupunem că un turist doreşte să cunoască dacă este posibil să plece din x şi să ajungă în y astfel încât să treacă prin cel mult două obiective intermediare. Pentru a rezolva această problemă baza de fapte dinamice trebuie să aibă conținutul:

```
d("x1","x2","a")
d("x2","x3","a")
d("x2","x6","a")
d("x3","x4","b")
d("x4","x5","b")
d("x6","x7","b")
d("x7","x8","b")
d("x8","x9","b")
d("x9","x5","b")
d("x1","x3","c")
d("x3","x5","c")
d("x7","x4","c")
fi("a","a","aa")
fi("a","aa","aaa")
fi("aa","a","aaa")
fi("b","b","bb")
fi("b","bb","bbb")
fi("bb","b","bbb")
fi("c","c","cc")
fi("c","cc","ccc")
```

```
fi("cc","c","ccc")
fi("a","b","ab")
fi("a","bb","abb")
fi("bb","a","abb")
fi("a","ab","aab")
fi("ab","a","aab")
fi("b","a","ab")
fi("b","ab","abb")
fi("ab","b","abb")
fi("c","b","cb")
fi("cb","b","cbb")
fi("c","cb","ccb")
fi("cb","a","cba")
fi("c","a","ca")
fi("c","ca","cca")
fi("ca","a","caa")
fi("ca","b","cba")
```

Conținutul noului fişier “sem.pro” este

```
predicates
    semantica(nod,nod,nod)
clauses
    semantica(X,Y,"a"):-write(X," ---> ",Y," prin agentia A\n").
    semantica(X,Y,"aa"):-write(X," ---> ",Y," prin agentia A
                                                (1 obiectiv intermediar)\n").
    semantica(X,Y,"aaa"):-write(X," ---> ",Y," prin agentia A
                                              (2 obiective intermediare)\n").
  semantica(X,Y,"b"):-write(X," ---> ",Y," prin agentia B\n").
  semantica(X,Y,"bb"):-write(X," ---> ",Y," prin agentia B
                                                (1 obiectiv intermediar)\n").
     semantica(X,Y,"bbb"):-write(X," ---> ",Y," prin agentia B
```

```
                                                    (2 obiective intermediare)\n").
    semantica(X,Y,"c"):-write(X," ---> ",Y," prin agentia C\n").
 semantica(X,Y,"cc"):-write(X," ---> ",Y," prin agentia C
                                                    (1 obiectiv intermediar)\n").
 semantica(X,Y,"ccc"):-write(X," ---> ",Y," prin agentia C
                                                    (2 obiective intermediare)\n").
semantica(X,Y,"ab"):-write(X," ---> ",Y," prin agentiile A si B\n").
semantica(X,Y,"cb"):-write(X," ---> ",Y," prin agentiile C si B\n").
semantica(X,Y,"cba"):-write(X," ---> ",Y," prin agentiile C, A si B\n").
semantica(X,Y,"ca"):-write(X," ---> ",Y," prin agentiile C si A\n").
semantica(X,Y,"cb"):-write(X," ---> ",Y," prin agentiile C si B\n").
semantica(X,Y,"abb"):-write(X," ---> ",Y," prin agentiile A si B (2 etape)\n").
semantica(X,Y,"aab"):-write(X," ---> ",Y," prin agentiile A (2 etape) si B\n").
semantica(X,Y,"caa"):-write(X," ---> ",Y," prin agentiile C si A (2 etape)\n").
semantica(X,Y,"cbb"):-write(X," ---> ",Y," prin agentiile C si B (2 etape)\n").
semantica(X,Y,"cca"):-write(X," ---> ",Y," prin agentiile C (2 etape) si A\n").
semantica(X,Y,"ccb"):-write(X," ---> ",Y," prin agentiile C (2 etape) si B\n").
```