11. CADRE MINSKI

În acest capitol prezentăm o metodă de reprezentare a cunoştinţelor bazată pe ideea de ***cadru*** ("***frame***") introdusă de Minski, precum şi o modalitate de implementare a acestui formalism în PROLOG.

11.1. Aspecte formale

Un cadru se caracterizează prin:

- ***nume*** - prin intermediul căruia poate fi identificat cadrul respectiv
- mulţime de ***părinţi*** (***antecedenţi***) - cadre ale căror proprietăţi sunt moştenite (asumate implicit pentru cadrul cu părinţii respectivi)
- mulţime de proprietăţi (***atribute***)

Fiecare atribut este caracterizat printr-un ***nume*** şi o ***valoare*** (perechea nume-valoare este desemnată prin termenul ***slot***). Structura unui cadru poate fi reprezentată prin:

Nume: nume_cadru
Părinte: nume_cadru_parinte_1
⋮
Părinte: nume_cadru_parinte_k
Atribut: nume_atribut_1 valoare: val_1
⋮
Atribut: nume_atribut_n valoare: val_n

Un sistem de cadre poate fi perceput ca o bază de date ale cărei înregistrări sunt cadrele din sistem. Într-un sistem de cadre înregistrările sunt interconectate prin intermediul ***relaţiilor de moştenire***. Fiecare cadru moşteneşte proprietăţile părinţilor lui care la rândul lor moştenesc proprietăţile părinţilor lor ş.a.m.d. Dacă orice cadru are cel mult un cadru părinte, atunci se spune

că avem un sistem de cadre cu ***moştenire simplă***. Dacă există cel puţin un cadru care are mai mult de un părinte, sistemul se zice cu ***moştenire multiplă***.

Un sistem de cadre poate fi reprezentat grafic printr-un graf orientat aciclic. Nodurile acestui graf sunt cadrele, iar arcele sunt legăturile dintre cadre. Mai precis, (c1, c2) este un arc dacă c1 este părinte pentru c2 (i.e. apare în mulțimea de părinți asociată lui c2). Dacă în graful orientat asociat există un drum de la cadrul c1 la cadrul c2, atunci c1 se numeşte ***predecesor*** al lui c2, iar c2 se numeşte ***succesor*** al lui c1. Se numeşte distanță dintre două cadre minimul lungimilor drumurilor care leagă două cadre (dacă nu există nici un drum între cele două cadre de consideră distanța egală cu ∞). Cel mai apropiat predecesor al unui cadru relativ la o anumită proprietate este un predecesor al cadrului respectiv care îndeplineşte proprietatea respectivă şi care se găseşte la distanță minimă (în raport cu toate cadrele care îndeplinesc proprietatea respectivă). Dacă nu se impune nici o restricție sistemului de cadre, în raport cu proprietate oarecare, un cadru poate să nu aibă un cel mai apropiat predecesor, poate să aibă unul unic sau poate să aibă mai mulți. În continuare vom presupune că sistemul de cadre îndeplineşte următoarele condiții:

- fiecare cadru este unic determinat de numele lui
- pentru fiecare cadru trebuie specificate mulțimea de părinți şi mulțimea de atribute; aceste mulțimi pot fi vide
- valoarea unui atribut poate fi
  - valoare directă
  - demon (va fi definită în cadrele care moştenesc cadrul respectiv)
  - nume de procedură (procedura va indica modalitatea de calcul a valorii respective)
- dacă un atribut asociat unui cadru c1 are valoarea demon, atunci nici un predecesor al cadrului c1 nu conține atributul respectiv.
- dacă un atribut cu numele a1 asociat unui cadru c1 are valoarea demon, şi a1 apare în mulțimea de atribute asociate unui succesor al lui c1, atunci valoarea lui a1 în acea mulțime trebuie să fie valoare directă sau nume de procedură
- mulțimea celor mai apropiați predecesori ai unui cadru, care conțin un anumit atribut trebuie să aibă cel mult un element.
- dacă valoarea unui atribut este numele unei proceduri, atunci procedura respectivă nu trebuie să facă apel la numele acestui atribut

Să considerăm, spre exemplu, un cadru referitor la împrumuturi acordate persoanelor fizice de către o bancă, presupunând că rambursarea se face în tranşe lunare egale. Un astfel de cadru poate avea structura

Nume: imprumut
Părinte: -
Atribut: suma_imprumutată valoare: demon
Atribut: rata_anuala valoare: demon
Atribut: rata_curenta valoare: demon
Atribut: client valoare: demon
Atribut: dt valoare: demon

Atributul dt semnifică dobânda totală. Vom defini două cadre subordonate acestui cadru, unul pentru clienții cu credite preferențiale, iar altul pentru clienții obişnuiți. Structurile celor două cadre sunt

Nume: imprumut_pref
Părinte: imprumut
Atribut: rata_anuala valoare: 30%

Nume: imprumut_o
Părinte: imprumut
Atribut: rata_anuala valoare: 50%

Să presupunem că Popescu este un client preferențial, iar Ionescu un client obişnuit. Cadrele asociate sunt următoarele

Nume: credit_c103
Părinte: imprumut_pref
Atribut: suma_imprumutată valoare: 10 000 000
Atribut: rata_curenta valoare: 1 000 000
Atribut: client valoare: Popescu

Atribut: dt valoare: proc(dobanda_totala)

Nume: credit_c200
Părinte: imprumut_o
Atribut: suma_imprumutată valoare: 50 000 000
Atribut: rata_curenta valoare: 5 000 000
Atribut: client valoare: Ionescu
Atribut: dt valoare: proc(dobanda_totala)

Procedura dobanda_totala specifică modalitatea de calcul a dobânzii totale.

11.2. Implementarea cadrelor Minski în PROLOG

Pentru reprezentarea cadrelor în PROLOG avem în vedere următoarele două probleme:

- reprezentarea cadrelor
- definirea unei mulțimi de predicate pentru manipularea cadrelor

Fiecare cadru se reprezintă cu ajutorul unui fapt de forma:

cadru(nume, lista_parinti, lista_atribute).

Sistemul de cadre se reprezintă printr-o bază dinamică de fapte de forma de mai sus.

Se definesc următoarele operații asupra unui sistem de cadre:

- adăugarea unui cadru
- adăugarea unui slot (atribut)
- modificarea valorii unui slot (atribut)
- ştergerea unui slot (atribut)
- ştergerea unui cadru
- calcularea valorii unui atribut asociat unui cadru

Operația de ştergere a unui cadru afectează succesorii cadrului respectiv. Astfel înainte de a elimina cadru c din baza dinamică se efectuează următoarele operații:

- se identifică descendenții (succesorii) direcții ai cadrului c şi în lista de părinți ai fiecărui descendent direct c se înlocuieşte cu părinții lui c

- la lista de atribute a fiecărui descendent direct al lui c se adaugă atributele lui c (care nu sunt în lista respectivă)

Operația de calcul al valorii val a unui atribut cu nume na asociat unui cadru c presupune următorii paşi

1. dacă c conține un slot de forma

   Atribut: na valoare: v (valoare directă)

   atunci val = v
2. dacă c conține un slot de forma

   Atribut: na valoare: demon

   atunci val = demon
3. dacă c conține un slot de forma

   Atribut: na valoare: proc(nume_procedura)

   atunci val este valoarea obținută cu ajutorul procedurii indicate; procedura respectivă poate face apel la calculul valorilor altor atribute asociate aceluiaşi cadru sau unor cadre diferite
4. dacă c nu conține nici un slot de forma

   Atribut: na valoare: _

   atunci se identifică (dacă există) cel mai apropiat predecesor al lui c care conține un atribut cu numele na, şi se calculează valoarea atributului na asociat acestui cadru utilizând pasul 1, 2 sau 3
5. dacă nu există nici un predecesor al lui c care să conțină un atribut cu nume na, sau dacă nu există nici un cadru cu numele c, atunci val = necunoscută

Prezentăm implementarea în PROLOG a operațiilor de mai sus.

```
domains
  nume=symbol
  lista_nume=nume*
  valoare=proc(nume);re(real);si(string);demon;necunoscuta
  slot=reference atribut(nume,valoare)
  lista_atribute=slot*
database
```

```
   cadru(nume,lista_nume,lista_atribute)
predicates
   start
   fer
   fer1
   fer2
   sterge_ferestre
   baza(string)
   salvare(string)
   meniu(string)
   selectie(string,char)
   citeste_valoare_atribut(valoare)
   val_atrib(char,valoare)
   citeste_parinti(lista_nume)
   citeste_atribute(lista_atribute)
   modifica_atribut(lista_atribute,nume,valoare,lista_atribute)
   elimin_slot(slot,lista_atribute,lista_atribute)
   apartine1(nume,lista_nume)
   apartine2(slot,lista_atribute)
   concatenare1(lista_nume,lista_nume,lista_nume)
   adaug_atrib(lista_atribute,lista_atribute,lista_atribute)
   inlocuiesc_nume_cu_parinti(nume,lista_nume,lista_nume,lista_nume)
   prelucrare(nume,lista_nume,lista_atribute)
   gaseste_descendenti(nume,lista_nume,lista_nume)
   descendenti(nume,lista_nume)
   modifica(nume,lista_nume,lista_nume,lista_atribute)
   scrie_valoare(valoare)
   calculeaza_valoare_directa(nume,nume,valoare)
   calculeaza_valoare(nume,nume,valoare)
   calculeaza_valoare_mostenita(nume,lista_nume,valoare)
   calculeaza_valoare_parinti(nume,lista_nume,valoare)
   gaseste_stramosi(lista_nume,lista_nume,lista_nume)
```

```
   calcul(nume,nume,valoare)
   dobanda(real,real,real,real)
 clauses
    start:-fer,fer1,fer2,shiftwindow(1),baza(X), meniu(X),sterge_ferestre.
    fer:-makewindow(1,113,36,"***Meniu***",0,0,25,80).
    fer1:- makewindow(2,30,97,"...",8,10,14,60).
    fer2:- makewindow(3,78,97,"Valoare atribut",15,11,7,58).
    baza(X):-shiftwindow(2),write("Baza dinamica = "),readln(X),
           consult(X),shiftwindow(1).
   meniu(X):-shiftwindow(1),clearwindow,cursor(8,10),write("a --> adauga slot"),
            cursor(9,10),write("b --> adauga cadru"),
            cursor(10,10),write("c --> modifica valoare atribut"),
            cursor(11,10),write("d --> sterge slot"),
            cursor(12,10),write("e --> sterge cadru"),
            cursor(13,10),write("f --> calculeaza"),
            cursor(14,10),write("q --> iesire "),
            readchar(R),R<>'q',selectie(X,R),meniu(X).
    meniu(_).
  sterge_ferestre:-shiftwindow(3),removewindow,shiftwindow(2),removewindow,
                   shiftwindow(1),removewindow.
  salvare(X):-concat("del ",X,S),system(S),save(X).
  /* ============se adauga un
slot======================================== */
  selectie(X,'a'):-shiftwindow(2),clearwindow,write("Numele cadrului="),readln(Nc),
              write("Nume atribut="),readln(Na),
              citeste_valoare_atribut(Va),A=atribut(Na,Va),
              cadru(Nc,P,La),retract(cadru(Nc,P,La)),L1=[A|La],asserta(cadru(Nc,P,L1)),
              write("Continuati acesta operatie? [d|n]: "),readchar(R),
              R='d',selectie(X,'a').
  selectie(X,'a'):-salvare(X).
  /* ============se adauga un
cadru======================================== */
```

```
  selectie(X,'b'):-shiftwindow(2),clearwindow,write("Numele cadrului="),readln(Nc),
                clearwindow,write("Introduceti parintii:"),nl,citeste_parinti(P),
                clearwindow,write("Introduceti atributele:"),nl,citeste_atribute(A),
                asserta(cadru(Nc,P,A)),
                write("Continuati acesta operatie? [d|n]: "),readchar(R),
                R='d',selectie(X,'b').
  selectie(X,'b'):-salvare(X).
  /* ============se modifica valoarea unui
atribut============================== */
  selectie(X,'c'):-shiftwindow(2),clearwindow,write("Numele cadrului="),readln(Nc),
                write("Nume atribut="),readln(Na),
                citeste_valoare_atribut(Va),cadru(Nc,_,La),
                modifica_atribut(La,Na,Va,La_modif),
                retract(cadru(Nc,P,La)),asserta(cadru(Nc,P,La_modif)),
                write("Continuati acesta operatie? [d|n]: "),readchar(R),
                R='d',selectie(X,'c').
  selectie(X,'c'):-salvare(X).
  /* ============se sterge un
slot=========================================== */
  selectie(X,'d'):-shiftwindow(2),clearwindow,write("Numele cadrului="),readln(Nc),
                write("Nume atribut="),readln(Na),A=atribut(Na,_),
                cadru(Nc,_,La),elimin_slot(A,La,La1),retract(cadru(Nc,P,La)),
                asserta(cadru(Nc,P,La1)),
                write("Continuati acesta operatie? [d|n]: "),readchar(R),
                R='d',selectie(X,'d').
 selectie(X,'d'):-salvare(X).
  /* ============se sterge un
cadru========================================== */
  selectie(X,'e'):-shiftwindow(2),clearwindow,write("Numele cadrului="),readln(Nc),
                cadru(Nc,P,A),prelucrare(Nc,P,A),
                retract(cadru(Nc,P,A)),
                write("Continuati acesta operatie? [d|n]: "),readchar(R),
```

```
            R='d',selectie(X,'e').
  selectie(X,'e'):-salvare(X).
  /* =========se calculeaza valoarea asociata unui atribut================== */
  selectie(X,'f'):-shiftwindow(2),clearwindow,write("Numele cadrului="),readln(Nc),
            write("Nume atribut="),readln(Na),calculeaza_valoare(Nc,Na,Val),
            scrie_valoare(Val),nl,
            write("Continuati acesta operatie? [d|n]: "),readchar(R),
            R='d',selectie(X,'f').
  selectie(_,'f').

/*================================a================================
=*/
  citeste_valoare_atribut(Va):-shiftwindow(3),clearwindow,
                        cursor(1,3),write("r/s --> valoare imediata reala/sir caractere"),
                        cursor(2,3),write("d   --> demon"),
                        cursor(3,3),write("p   --> procedura"),
                        readchar(R),shiftwindow(2),val_atrib(R,Va).
  val_atrib('r',Va):-write("Valoarea atributului="),readreal(V),Va=re(V).
  val_atrib('s',Va):-write("Valoarea atributului="),readln(V),Va=si(V).
  val_atrib('d',demon).
  val_atrib('p',Va):-write("Numele procedurii="),readln(Np),Va=proc(Np).

/*============================b================================
==*/
  citeste_parinti([X|P]):-write("Nume parinte="),readln(X),X<>"",citeste_parinti(P),!.
  citeste_parinti([]):-!.
  citeste_atribute([X|A]):-write("Nume atribut="),readln(Na),
                        Na<>"",citeste_valoare_atribut(Va),!,
                        X=atribut(Na,Va),citeste_atribute(A).
  citeste_atribute([]):-!.
```

```
/*==============================c================================
===*/
  modifica_atribut([atribut(Na,_)|L],Na,Va,[atribut(Na,Va)|L]):-!.
  modifica_atribut([atribut(N,V)|La],Na,Va,[atribut(N,V)|L]):-
Na<>N,modifica_atribut(La,Na,Va,L).
  /*=========================d==============================
========*/
  elimin_slot(X,[X|Y],Y):-!.
  elimin_slot(X,[H|Y],[H|Z]):-elimin_slot(X,Y,Z).

/*==============================e================================
===*/
  apartine1(X,[X|_]):-!.
  apartine1(X,[_|Y]):- apartine1(X,Y).
  apartine2(X,[X|_]):-!.
  apartine2(X,[_|Y]):- apartine2(X,Y).
  concatenare1([],Y,Y).
  concatenare1([H|X],Y,[H|Z]):-concatenare1(X,Y,Z).
  adaug_atrib([],Y,Y).
  adaug_atrib([H|X],Y,[H|Z]):-not(apartine2(H,Y)),adaug_atrib(X,Y,Z).
  adaug_atrib([H|X],Y,Z):-apartine2(H,Y),adaug_atrib(X,Y,Z).
  inlocuiesc_nume_cu_parinti(X,[X|Y],P,Z):-concatenare1(P,Y,Z),!.
  inlocuiesc_nume_cu_parinti(X,[H|Y],P,[H|Z]):-inlocuiesc_nume_cu_parinti(X,Y,P,Z).
  prelucrare(Nc,P,A):-descendenti(Nc,L),modifica(Nc,L,P,A),!.
  descendenti(Nc,L):-gaseste_descendenti(Nc,[],L).
  gaseste_descendenti(Nc,L1,L2):-cadru(X,Lx,_),apartine1(Nc,Lx),
                        not(apartine1(X,L1)),gaseste_descendenti(Nc,[X|L1],L2).
 gaseste_descendenti(_,L,L).
 modifica(Nc,[X|L],P,A):-cadru(X,Lp,La),inlocuiesc_nume_cu_parinti(Nc,Lp,P,Lp1),
                adaug_atrib(A,La,La1),
                retract(cadru(X,Lp,La)),asserta(cadru(X,Lp1,La1)),
```

```
                modifica(Nc,L,P,A).
  modifica(_,[],_,_).

/*==============================f================================
======*/
  scrie_valoare(demon):-write("Este un demon"),!.
  scrie_valoare(X):-write("Valoare: ",X).
  calculeaza_valoare_directa(Nc,Na,Val):-cadru(Nc,_,La),apartine2(atribut(Na,V),La),
                          V=proc(P),calcul(Nc,P,Val),!.
  calculeaza_valoare_directa(Nc,Na,Val):-cadru(Nc,_,La),apartine2(atribut(Na,Val),La).
  calculeaza_valoare(Nc,Na,Val):-calculeaza_valoare_directa(Nc,Na,Val),!.
  calculeaza_valoare(Nc,Na,Val):-
cadru(Nc,P,_),calculeaza_valoare_mostenita(Na,P,Val),!.
  calculeaza_valoare(_,_,necunoscuta).
  calculeaza_valoare_mostenita(Na,P,Val):-calculeaza_valoare_parinti(Na,P,Val),!.
  calculeaza_valoare_mostenita(Na,P,Val):-gaseste_stramosi(P,[],S),S=[_|_],
                          calculeaza_valoare_mostenita(Na,S,Val).
  calculeaza_valoare_mostenita(_,_,necunoscuta):-write("Nume de atribut gresit!"),nl,!.
  calculeaza_valoare_parinti(Na,[X|_],Val):-calculeaza_valoare_directa(X,Na,Val),!.
  calculeaza_valoare_parinti(Na,[_|P],Val):-calculeaza_valoare_parinti(Na,P,Val).
  gaseste_stramosi([X|P],Y,S):-
cadru(X,T,_),concatenare1(T,Y,S1),gaseste_stramosi(P,S1,S).
  gaseste_stramosi([],S,S).
  calcul(X,dobanda_totala,Val):- calculeaza_valoare(X,suma_imprumutata,S1),
                S1=necunoscuta,Val=necunoscuta,!.
  calcul(X,dobanda_totala,Val):-
  calculeaza_valoare(X,rata_anuala,Ra1),Ra1=necunoscuta,Val=necunoscuta,!.
  calcul(X,dobanda_totala,Val):-calculeaza_valoare(X,rata_curenta,Rc1),
                Rc1=necunoscuta,Val=necunoscuta,!.
  calcul(X,dobanda_totala,Val):-calculeaza_valoare(X,suma_imprumutata,S1),
                S1=demon,Val=demon,!.
```

```
  calcul(X,dobanda_totala,Val):-
calculeaza_valoare(X,rata_anuala,Ra1),Ra1=demon,Val=demon,!.
  calcul(X,dobanda_totala,Val):-
calculeaza_valoare(X,rata_curenta,Rc1),Rc1=demon,Val=demon,!.
  calcul(X,dobanda_totala,Val):-calculeaza_valoare(X,suma_imprumutata,S1),
                          S1=re(S),write("S=",S,"\n"),
                          calculeaza_valoare(X,rata_anuala,Ra1),
                          Ra1=re(Ra),write("Ra=",Ra,"\n"),
                          calculeaza_valoare(X,rata_curenta,Rc1),
                          Rc1=re(Rc),write("Rc=",Rc,"\n"),
                       dobanda(S,Rc,Ra,Dob_tot),Val=re(Dob_tot).
  dobanda(Imprumut,Rata_curenta,Rata_anuala_dobanda,Dobanda_totala):-
                                          Dobanda_totala=Imprumut*Rata_anuala_dobanda*
                                             (Imprumut+Rata_curenta)/(2400*Rata_curenta).
```

Singura procedură descrisă în program este cea a calculului dobânzii totale corespunzătoare exemplului din subcapitolul precedent. Baza dinamică asociată aceluiaşi exemplu poate avea următorul conţinut:

```
cadru("credit_c103", ["imprumut_pref"], [atribut("suma_imprumutata",re(10000000)),
          atribut("rata_curenta",re(1000000)), atribut("client",si("Popescu")),
          atribut("dt",proc("dobanda_totala"))])
cadru("credit_c200", ["imprumut_o"], [atribut("suma_imprumutata",re(50000000)),
          atribut("rata_curenta",re(5000000)), , atribut("client",si("Ionescu"))]
          atribut("dt",proc("dobanda_totala"))])
cadru("credit_c302",["imprumut_o"],[ atribut("suma_imprumutata",re(5000000)),
          atribut("rata_curenta",re(1000000)), atribut("client",si("Georgescu")),
          atribut("dt",proc("dobanda_totala"))])
cadru("credit_c301", ["imprumut_o"], [atribut("suma_imprumutata",re(10000000)),
         atribut("rata_curenta",re(1000000)), atribut("client",si("Vasilescu"),
         atribut("dt",proc("dobanda_totala"))])
cadru("credit_c300", ["imprumut_o"], [atribut("suma_imprumutata",re(700000000)),
```

```
        atribut("rata_curenta",re(10000000)), atribut("client",si("Avram"),
        atribut("dt",proc("dobanda_totala"))])
cadru("imprumut_o",["imprumut"],[atribut("rata_anuala",re(50))])
cadru("imprumut_pref",["imprumut"],[atribut("rata_anuala",re(30))])
cadru("imprumut", [], [atribut("suma_imprumutata",demon),
atribut("rata_anuala",demon),
         atribut("rata_curenta",demon),atribut("client",demon),atribut("dt",demon)])
```