

INTELIGENȚĂ ARTIFICIALĂ

Teme de casă - III

COSTIN BĂDICĂ

Universitatea din Craiova
Facultatea de Automatică, Calculatoare și Electronică
Catedra de Inginerie Software

25 Noiembrie 2002

1 Lista de probleme

Scopul acestei teme este efectuarea unor prelucrări asupra unor structuri simbolice reprezentate sub forma unor termeni Prolog.

Problema 1 Se consideră o bază de cunoștințe ce conține o mulțime de reguli. O regulă are o etichetă, o parte stângă și o parte dreaptă. Partea stângă este o conjuncție de condiții. O condiție exprimă o relație între o variabilă și o valoare. Concluzia exprimă adăugarea unei valori la o variabilă.

Baza de cunoștințe se reprezintă printr-un termen Prolog de forma `bc(Lista)` unde `List` este o listă de reguli sau comentarii. Un comentariu este un termen de forma `com(Text)`. Orice regulă este precedată opțional de cel mult un comentariu. O regulă are forma `regula(Eticheta,Stanga,Dreapta)`. Partea stângă este o listă de condiții. O condiție este de forma `p(Variabila,Valoare,Relatie)`. Sunt admise relațiile `maimic`, `maimare` și `egal`. O concluzie este de forma `c(Variabila,Valoare)`.

Un exemplu de bază de cunoștințe este:

```
bc([
  regula(r1,[p(a,1,egal),p(c,2,maimare)],c(b,2)),
  com('Comentariu pentru regula r2'),
  regula(r2,[p(b,3,maimic)],c(d,1))])
```

Se cere:

- i) Să se definească un predicat `baza_cun(X)` care este adevărat dacă și numai dacă `X` este un termen care reprezintă o bază de cunoștințe.
- ii) Să se definească un predicat `elimina_comentarii(Bci,Bce)` care este adevărat dacă și numai dacă `Bce` este baza de cunoștințe rezultată prin eliminarea comentariilor din baza de cunoștințe `Bci`.
- iii) Să se definească un predicat `lista_variabile(Bc,ListaVar)` care este adevărat dacă și numai dacă `ListaVar` este lista tuturor variabilelor din baza de cunoștințe `Bc`.
- iv) Se consideră o mulțime de fapte de forma unor perechi variabilă-valoare. Această mulțime de fapte se reprezintă printr-o listă de termeni de forma `f(Variabila, Valoare)`. Să se definească un predicat `determina_reguli(Fapte,Bc,Reguli)` care este adevărat dacă și numai dacă `Reguli` este lista etichetelor regulilor aplicabile. O regulă este aplicabilă dacă toate condițiile din partea sa stângă sunt adevărate pentru mulțimea dată de fapte.

Problema 2 Se consideră o bază de cunoștințe cu informații despre un proiect informatic. Un proiect informatic este caracterizat de o mulțime de resurse umane alocate și o mulțime de faze.

Un proiect informatic se reprezintă printr-un termen `proiect(Resurse,Faze)` unde `Resurse` este lista resurselor umane alocate proiectului și `Faze` este lista fazelor proiectului.

O resursă este un termen `resursa(Id,Nume,Tip)`. O resursă poate fi precedată opțional de un comentariu de descriere a resursei de forma `com(Text)`. `Id` este un cod unic de identificare a resursei, `Nume` este numele persoanei corespunzătoare resursei, `Tip` este tipul resursei (de exemplu `'Programator'` sau `'Analist'`).

O fază este un termen de forma `faza(ResurseNecesare,Nume,Inceput,Durata)`. `ResurseNecesare` este lista de resurse necesare fazei. Un element al acestei liste este de forma `necesar(Tip,Numar)` unde `Tip` este tipul resursei necesare și `Numar` este numărul de resurse necesare de tipul respectiv. `Nume` este numele fazei, `Inceput` este un text ce reprezintă data de început a fazei și `Durata` reprezintă durata fazei în număr de zile.

Un exemplu de reprezentare a unui proiect informatic este:

```
proiect(  
  [resursa(i1,'Ion','Analist'),  
   resursa(i2,'Stefan','Arhitect'),  
   resursa(i3,'Maria','Programator')],  
  [faza([necesar('Analist',1),necesar('Arhitect',1)],
```

```
'Faza 1', '10.01.2003', 30),
faza([necesar('Arhitect', 1), necesar('Programator', 1)], 'Faza
2', '10.02.2003', 30)])
```

Se cere:

- i) Să se definească un predicat `proiect(X)` care este adevărat dacă și numai dacă `X` este un termen care reprezintă un proiect informatic.
- ii) Să se definească un predicat `elimina_comentarii(Pi, Pe)` care este adevărat dacă și numai dacă `Pe` este proiectul rezultat prin eliminarea comentariilor referitoare la resurse din proiectul `Pi`.
- iii) Să se definească un predicat `lista_faze(P, ListaFaze)` care este adevărat dacă și numai dacă `ListaFaze` este lista etichetelor fazelor proiectului `P`.
- iv) Să se definească un predicat `realizabil(Proiect)` care este adevărat dacă și numai dacă `Proiect` reprezintă un proiect realizabil. Un proiect este realizabil dacă și numai dacă există resursele necesare pentru fiecare fază a sa.

Problema 3 Se consideră o bază de cunoștințe de forma unei rețele semantice. Rețeaua semantică se reprezintă sub forma unei mulțimi de arce obiect-atribut-valoare.

O rețea semantică este un termen de forma `retea(Lista)`. `Lista` este o listă de arce și comentarii. Un arc este un termen de forma `prop(IdObiect, NumeAtribut, Valoare)`. `IdObiect` este un întreg pozitiv ce reprezintă un identificator al unui obiect. `NumeAtribut` este numele unui atribut. `Valoare` reprezintă valoarea atributului `NumeAtribut` pentru obiectul identificat prin `IdObiect`. O valoare poate fi o valoare primitivă de forma `l(Valoare)` sau un alt obiect de forma `o(IdObiect)`. Un comentariu este un termen de forma `com(Text)`.

Spre exemplu baza de cunoștințe: "Arterele sunt vase de sânge. Un vas de sânge transportă sânge și are forma tubulară. Arterele transportă sânge bogat în oxigen" se reprezintă astfel:

```
retea([
  com('Clasa vaselor de sange'),
  prop(1, nume, l('VasDeSange')),
  prop(1, transporta, l(sange)),
  prop(1, forma, l(tubulara)),
  com('Clasa arterelor'),
  prop(2, nume, l('Artera')),
  prop(2, este_un, o(1)),
  prop(2, sange, l(bogat_in_oxigen))])
```

Se cere:

- i) Să se definească un predicat `retea(X)` care este adevărat dacă și numai dacă `X` este un termen ce reprezintă o rețea semantică.
- ii) Să se definească un predicat `elimina_comentarii(Ri,Re)` care este adevărat dacă și numai dacă `Re` este rețeaua rezultată prin eliminarea comentariilor din rețeaua `Ri`.
- iii) Să se definească un predicat `lista_obiecte(R,ListaObiecte)` care este adevărat dacă și numai dacă `ListaObiecte` este lista tuturor identificatorilor obiectelor din rețeaua `R`.
- iv) Să se definească un predicat `valori(IdObiect,NumeAtribut,Retea,Valori)` care este adevărat dacă și numai dacă `Valori` reprezintă lista valorilor atributului `NumeAtribut` pentru obiectul identificat prin `IdObiect` din rețeaua `Retea`. Valorile pot fi proprii obiectului respectiv sa pot fi dobândite prin moștenire de-alungul arcelor `este_un`. Se presupune ca moștenirea se face cu cumulare și că nu există cicluri cu arce de tipul `este_un`.

Problema 4 Se consideră o bază de cunoștințe de forma unei mulțimi de obiecte. Un obiect are o mulțime de atribute. Pentru fiecare atribut se specifică o listă de valori.

Baza de cunoștințe se reprezintă printr-un termen `bc(Obiecte)` unde `Obiecte` este o listă obiecte. Un obiect este un termen de forma `obiect(IdObiect,IdObiectParinte,Lista)`. `IdObiect` este un număr întreg strict pozitiv care identifică în mod unic obiectul. `IdObiectParinte` este identificatorul obiectului părinte. Se presupune că fiecare obiect are cel mult un obiect părinte. Un obiect fără părinte are în acest câmp valoarea 0. `Lista` este o listă de perechi atribut-valori sau comentarii. O pereche atribut-valori este de forma `atVal(NumeAtribut,ListaValori)`. `NumeAtribut` o listă de valori primitive (de forma `l(Valoare)`) sau de tip obiect (de forma `o(IdObiect)`). Un comentariu este de forma `com(Text)`.

Spre exemplu baza de cunoștințe "Un calculator PC rulează Windows sau Linux. Un calculator PC portabil este un calculator PC. Dell Inspiron 3100 este un calculator PC portabil. Prețul său este 1500 de lire sterline" se reprezintă astfel:

```
bc([
  obiect(1,0,[
    com('Clasa calculatoarelor PC'),
    atVal(numa,[l('Calculator PC')]),
    com('Sistemele de operare care pot rula pe PC'),
    atVal(ruleaza,[l('Windows'),l('Linux')])]),
  obiect(2,1,[
```

```

    atVal(ume, [l('Calculator PC portabil')]))],
obiect(3,2,[
    atVal(ume, [l('Dell Inspiron 3100')]),
    atVal(pret, [o(4)])),
obiect(4,0,[
    atVal(valoare, [l(1500)]),
    atVal(moneda, [l('Lira Sterlina')]))]
])

```

Se cere:

- i) Să se definească un predicat `baza_cun(X)` care este adevărat dacă și numai dacă X este un termen ce reprezintă o bază de cunoștințe de forma unei mulțimi de obiecte.
- ii) Să se definească un predicat `elimina_comentarii(Bi,Be)` care este adevărat dacă și numai dacă Be este baza de cunoștințe rezultată prin eliminarea comentariilor din baza de cunoștințe Bi .
- iii) Să se definească un predicat `lista_atribute(B,ListaNumeAtribute)` care este adevărat dacă și numai dacă `ListaNumeAtribute` este lista tuturor numelor atributelor din baza de cunoștințe B .
- iv) Să se definească un predicat `valori(IdObiect,NumeAtribut,BazaCun,Valori)` care este adevărat dacă și numai dacă `Valori` reprezintă lista valorilor atributului `NumeAtribut` pentru obiectul identificat prin `IdObiect` din baza de cunoștințe `BazaCun`. Valorile pot fi proprii obiectului respectiv sau pot fi dobândite prin moștenire. Se presupune că moștenirea se face cu cumulare și că nu există cicluri de-alungul relației părinte în cadrul bazei de cunoștințe.

Problema 5 Se consideră o bază de cunoștințe de forma unei ierarhii arborescente de obiecte.

Baza de cunoștințe se reprezintă printr-un termen de forma `bc(Ierarhie)`. `Ierarhie` este o ierarhie arborescentă de obiecte. Ea este un termen de forma `obiect(IdObiect, Lista, ListaCopii)`. `IdObiect` este un întreg pozitiv care identifică în mod unic obiectul rădăcină al ierarhiei. `Lista` este o listă de perechi atribut-valoare sau comentarii. O pereche atribut-valoare este de forma `atVal(NumeAtribut, Valoare)`. `NumeAtribut` este un nume de atribut și `Valoare` este o valoare primitivă. Un comentariu este de forma `com(Text)`. `ListaCopii` este o listă de obiecte care reprezintă rădăcinile ierarhiilor de obiecte derivate din obiectul rădăcină.

Spre exemplu baza de cunoștințe "Arterele și venele sunt vase de sânge. Un vas de sânge transportă sânge și are formă tubulară" se reprezintă astfel:

```
bc(
  obiect(0,
    [com('Clasa vaselor de sange'),
     atVal(ume,'Vas de sange'),
     atVal(transporta,sange),
     atVal(forma,tubulara)],
    [obiect(1,
      [com('Clasa arterelor'),
       atVal(ume,'Artera')],
      []),
     obiect(2,
      [atVal(ume,'Vena')],
      [])])
```

Se cere:

- i) Să se definească un predicat `baza_cun(X)` care este adevărat dacă și numai dacă X este un termen ce reprezintă o bază de cunoștințe de forma unei ierarhii arborescente de obiecte.
- ii) Să se definească un predicat `elimina_comentarii(Bi,Be)` care este adevărat dacă și numai dacă Be este baza de cunoștințe rezultată prin eliminarea comentariilor din baza de cunoștințe Bi .
- iii) Să se definească un predicat `lista_atribute(B,ListaObiecte)` care este adevărat dacă și numai dacă `ListaObiecte` este lista tuturor identificatorilor obiectelor din baza de cunoștințe B , obținută prin parcurgerea ierarhiei în preordine.
- iv) Să se definească un predicat `valori(IdObiect,NumeAtribut,BazaCun,Valori)` care este adevărat dacă și numai dacă `Valori` reprezintă lista valorilor atributului `NumeAtribut` pentru obiectul identificat prin `IdObiect` din baza de cunoștințe `BazaCun`. Valorile pot fi proprii obiectului respectiv sau pot fi dobândite prin moștenire. Se presupune că moștenirea se face cu suprascriere.

Problema 6 Se consideră o structură de reprezentare a unui circuit combinațional sub forma unei mulțimi de porți logice.

Un circuit logic combinațional se reprezintă printr-un termen de forma `circuit(Porti)` unde `Porti` este o listă de porți logice sau comentarii. Fiecare poartă este precedată opțional de un comentariu. O poartă logică este un termen de forma `poarta(Nume,Tip,Intrari,Iesire)`. `Nume` este un nume unic asociat porții respective. `Tip` este tipul porții și poate fi `or`, `and`, `xor`, `not`, `nor` sau `nand`. `Intrari` reprezintă intrările porții respective. Pentru porțile `not` există o singură intrare și ea se reprezintă printr-un simbol. Pentru

celelalte porți `Intrari` este o listă de simboluri. `Iesire` este un simbol ce desemnează ieșirea porții.

Spre exemplu un sumator complet de un bit se reprezintă astfel:

```
circuit([
  poarta(x1,xor,[i1,i2],t1),
  com('Poarta care genereaza bitul suma'),
  poarta(x2,xor,[t1,i3],o1),
  poarta(a1,and,[i1,i2],t2),
  poarta(a2,and,[i3,t1],t3),
  com('Poarta care genereaza bitul transport'),
  poarta(o1,or,[t3,t2],o2)])
```

Se cere:

- i) Să se definească un predicat `circuit(X)` care este adevărat dacă și numai dacă `X` este un termen ce reprezintă un circuit logic combinațional de forma unei liste de circuite.
- ii) Să se definească un predicat `elimina_comentarii(Ci,Ce)` care este adevărat dacă și numai dacă `Ce` este descrierea circuitului rezultată prin eliminarea comentariilor din descrierea circuitului `Ci`.
- iii) Să se definească un predicat `lista_conexiuni(C,Conexiuni)` care este adevărat dacă și numai dacă `Conexiuni` este lista tuturor numelor conexiunilor din circuitul `C`.
- iv) Să presupunem că valorile semnalelor de intrare ale circuitului sunt date sub forma unei liste `Semnale` de element de forma `semnal(Intrare,Valoare)`. Spre exemplu dacă sumatorul complet primește la intrare valorile `i1=0,i2=1,i3=1` atunci această listă va fi `[semnal(i1,0),semnal(i,1),semnal(i3,1)]`. Să se definească un predicat `semnal(Conexiune,Circuit,Intrari,Valoare)` care este adevărat dacă `Valoare` este valoarea conexiunii `Conexiune` a circuitului `Circuit` pentru intrările având valorile din lista `Intrari`.

Problema 7 Se consideră o problemă de restricții finite în care variabilele sunt numere naturale, restricțiile sunt unare sau binare și domeniile sunt de forma unor intervale. O astfel de problemă se descrie prin mulțimea restricțiilor și mulțimea domeniilor variabilelor.

O problemă de restricții se reprezintă printr-un termen `problema(Restrictii, Domenii)`.

`Restrictii` este lista restricțiilor. O restricție este de forma `r(R,Var,VarVal)` unde `R` este numele unei restricții, `Var` este un nume simbolic al unei variabile și `VarVal` este

un nume simbolic de variabilă sau o valoare întreagă. Restricțiile admise sunt `diferit`, `maimic` și `maimare`.

`Domenii` este lista domeniilor variabilelor. Un domeniu este un termen `dom(Var,Min,Max)` unde `Var` este numele unei variabile și `Min` respectiv `Max` sunt valorile minimă respectiv maximă pentru variabila `Var`.

Fiecare restricție respectiv domeniu poate fi precedat opțional în listele `Restrictii` respectiv `Domenii` de un comentariu de forma `com(Text)`.

Se consideră următorul exemplu de problemă de restricții:

```
restrictii(  
  restrictii([  
    com('a este mai mic strict decat c'),  
    r(maimic,a,c),  
    r(diferit,a,b),  
    r(maimare,b,2)]),  
  domenii([  
    dom(a,1,7),  
    dom(b,1,7),  
    com('domeniul lui c este 2..5'),  
    dom(c,2,5),  
    dom(d,1,5)]))
```

Se cere:

- i) Să se definească un predicat `problema(X)` care este adevărat dacă și numai dacă `X` este un termen ce reprezintă o problemă de restricții finite.
- ii) Să se definească un predicat `elimina_comentarii(Pi,Pe)` care este adevărat dacă și numai dacă `Pe` este problema rezultată prin eliminarea comentariilor din problema `Pi`.
- iii) Să se definească un predicat `lista_variabile_nefolosite(P,Variable)` care este adevărat dacă și numai dacă `Variable` este lista tuturor numelor variabilelor din problema `P` care nu apar în nici o restricție din `P`.
- iv) Se consideră o asignare a variabilelor unei probleme de restricții reprezentată sub forma unei liste de termeni `egal(Variabila,Valoare)`. Să se definească predicatul `verifica(A,P)` care este adevărat dacă și numai dacă asignarea `A` verifică toate restricțiile problemei `P` și dacă valorile fiecărei variabile se află în domeniul definit în problemă.

Problema 8 Să presupunem că decidem să folosim o nouă reprezentare a listelor cu ajutorul predicatelor `e` și `c`. `e(E)` este o listă formată dintr-un singur element `E` și `c(L1,L2)`

este lista formată din elementele lui L1 urmate de elementele lui L2. Reprezentarea unei liste cu **e** și **c** nu este unică. Spre exemplu lista formată din elementele **a, b, c, d** în această ordine se poate reprezenta în oricare dintre modurile:

```
c(e(a), c(c(e(b), e(c)), e(d)))
c(c(e(a), e(b)), c(e(c), e(d)))
```

Se cere:

- i) Să se definească un predicat `lista(L)` care este adevărat dacă și numai dacă L este un termen ce reprezintă o listă reprezentată cu **e** și **c**.
- ii) Să se definească un predicat `membru(E,L)` care este adevărat dacă și numai dacă E este un element al listei L reprezentată cu **e** și **c**.
- iii) Să se definească un predicat `lungime(L,N)` care este adevărat dacă și numai dacă N este numărul de elemente ale listei L reprezentată cu **e** și **c**.
- iv) Să se definească un predicat `lista_prolog(L,LP)` care este adevărat dacă și numai dacă LP este lista Prolog a tuturor elementelor din lista L reprezentată cu **e** și **c**.

Problema 9 Se consideră o bază de cunoștințe de forma unei mulțimi de clauze precise cu variabile și fără funcții.

O astfel de bază de cunoștințe se poate reprezenta sub forma unui termen `Clauze` unde `Clauze` este o listă de clauze. O clauză poate fi precedată opțional de un comentariu de forma `com(Text)`. O clauză este un termen de forma `regula(Antet, Corp)`. Corpul este o listă de formule atomice și antetul este un atom. O formula atomică este un termen complet instanțiat de forma `predicat(Arg1, ..., Argn)` unde `predicat` este un simbol de predicat iar `Argi`, $1 \leq i \leq n$ reprezintă constante sau variabile. O constantă este un termen de forma `c(NumeConstanta)` și o variabilă este un termen de forma `v(NumeVariabila)`. Spre exemplu baza de cunoștințe $\{p(X, Y) \leftarrow q(X) \wedge r(Y), q(a), r(b)\}$ se reprezintă astfel:

```
bc([
  com('0 regula'),
  regula(p(v(x), v(y)), [q(v(x)), r(v(y))]),
  regula(q(c(a)), []),
  com('Un fapt'),
  regula(r(c(b)), [])])
```

Se cere:

- i) Să se definească un predicat `baza_cun(X)` care este adevărat dacă și numai dacă X este un termen ce reprezintă o bază de cunoștințe formată de clauze precise.

- ii) Să se definească un predicat `elimina_comentarii(Bi,Be)` care este adevărat dacă și numai dacă `Be` este baza de cunoștințe rezultată prin eliminarea comentariilor din baza de cunoștințe `Bi`.
- iii) Să se definească un predicat `lista_constanta(B,C)` care este adevărat dacă `C` este lista constantelor care apar în baza de cunoștințe `B`. Fiecare constantă este considerată o singură dată.
- iv) Să se definească un predicat `predicate_recursive(B,L)` care este adevărat dacă `L` este lista tuturor predicatelor recursive din baza de cunoștințe `B`. Predicatele sunt considerate în ordinea în care apar în program. Un element al listei `L` este un termen de forma `p(SimbolPredicat,Aritate)`.

Problema 10 Un articol dintr-o revistă este compus din următoarele elemente: titlu, lista de autori, abstract, conținut și bibliografie. Conținutul este o listă de secțiuni. Fiecare secțiune are un titlu și un corp. Bibliografia este o listă de referințe bibliografice. În lista de autori, în lista de secțiuni și în lista de referințe bibliografice pot apărea opțional comentarii.

Pentru reprezentarea unui articol se folosește un termen de forma `articol(Titlu, Autori, Abstract, Continut, Bibliografie)`. Titlul este un termen de forma `titlu(Text)`. Autorii se reprezintă printr-un termen de forma `autori(ListaAutori)`. Un autor se reprezintă printr-un termen `autor(Text)`. Abstractul este un termen de forma `abstract(Text)`. Continutul este un termen de forma `continut(ListaSectiuni)`. O secțiune este de forma `sectiune(TextTitlu, TextCorp)`. Bibliografia este un termen de forma `bibl(ListaReferinte)`. O referință este de forma `ref(Text)`. Comentariile sunt termeni de forma `com(Text)`.

Un exemplu de articol este următorul:

```
articol(
  titlu('Articolul meu'),
  autori([com('adica chiar eu'), autor('Eu'), autor('Tu')]),
  abstract('abstractul meu'),
  continut([sectiune('Introducere', 'Aceasta este introducerea'),
            com('Un comentariu'), com('Altul'),
            sectiune('Concluzii', 'Acestea sunt concluziile')]),
  bibl([ref('Referinta 1'),
        ref('Referinta 2')]))
```

Se cere:

- i) Să se definească un predicat `articol(X)` care este adevărat dacă și numai dacă `X` este un termen ce reprezintă un articol.

- ii) Să se definească un predicat `elimina_comentarii(Ai,Ae)` care este adevărat dacă și numai dacă `Ae` este articolul rezultat prin eliminarea comentariilor din articolul `Ai`.
- iii) Să se definească un predicat `lista_continut(A,L)` care este adevărat dacă `L` este lista textelor secțiunilor care apar în conținutul articolului `A`.
- iv) Să se definească un predicat `contorizare(A,Na,Ns,Nr)` care este adevărat dacă `Na`, `Ns` și `Nr` sunt numărul de autori, secțiuni respectiv referințe bibliografice din articolul `A`.

Problema 11 O matrice bidimensională de numere reale este caracterizată de numărul de linii, numărul de coloane și lista liniilor sale. Fiecare linie este o listă formată dintr-un număr de numere egal cu numărul de coloane.

O matrice bidimensională de numere reale se poate reprezenta printr-un termen de forma `matrice(NrLinii,NrColoane,ListaLinii)`. `ListaLinii` este o lista formată din `NrLinii` de liste de numere reale. Fiecare astfel de listă reprezintă o linie a matricii și este formată din `NrColoane` elemente. Spre exemplu matricea:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & -2 & 6 \\ -3 & 0 & 10 \\ 3 & -4 & 5 \end{bmatrix}$$

se reprezintă prin termenul:

```
matrice(4,3,[
  [1,2,3],
  [4,-2,6],
  [-3,0,10],
  [3,-4,5]])
```

Se cere:

- i) Să se definească un predicat `matrice(X)` care este adevărat dacă și numai dacă `X` este un termen ce reprezintă o matrice bidimensională de numere reale.
- ii) Să se definească un predicat `contorizare(M,NrPoz,NrNeg,NrZero)` care este adevărat dacă și numai dacă `NrPoz`, `NrNeg` și `NrZero` reprezintă numărul de numere pozitive, negative, respectiv egale cu zero ale matricii `M`.
- iii) Să se definească un predicat `suma(M1,M2,S)` care este adevărat dacă `S` este suma matricilor `M1` și `M2`. Suma se poate calcula dacă și numai dacă cele două matrici au aceleași dimensiuni.

- iv) Să se definească un predicat `transpusa(M,Mt)` care este adevărat dacă `Mt` este transpusa matricii `M`.

Problema 12 Se consideră un document HTML simplificat. Un astfel de document conține un element rădăcină de tip `html`. Acesta conține un antet opțional (tipul `head`) și un corp obligatoriu (tipul `body`). Antetul conține în mod obligatoriu un element `title`. Corpul conține o secvență de paragrafe (elemente `p`) sau liste (elemente `ul`). O listă este o secvență de elemente `li`. Fiecare element poate conține, pe lângă elementele menționate, oricâte texte și comentarii.

Fiecare element se reprezintă printr-un termen `element(Tip,ListaContinut)`. `Tip` este un simbol ce reprezintă tipul elementului: `html`, `head`, etc. `ListaContinut` este o listă de elemente componente. Printre ele se pot afla opțional fragmente de text reprezentate prin `text(Text)` și comentarii reprezentate prin `com(Text)`.

Se consideră spre exemplu următorul document HTML:

```
<html>
  <!-- Un exemplu de document HTML simplificat -->
  <head>
    <!-- Antetul exemplului. -->
    <title>Exemplu</title>
  </head>
  <!-- Corpul incepe acum -->
  <body>
    <!-- Primul paragraf -->
    <p>Paragaraf 1</p>
    Text in afara unui paragraf
    <ul>
      <li>Primul</li>
      <li>
        <!-- al doilea din lista -->
        Al doilea
      </li>
    </ul>
    <p>Paragraf 2</p>
  </body>
</html>
```

El se reprezintă astfel:

```
element(html, [
  com('Un exemplu de document HTML simplificat'),
```

```

element(head, [
  com('Antetul exemplului.'),
  element(title, [text('Exemplu')])]),
com('Corpul incepe acum'),
element(body, [
  com('Primul paragraf'),
  element(p, [text('')]),
  text('Text in afara unui paragraf'),
  element(ul, [
    element(li, [text('Primul')]),
    element(li, [
      com('al doilea din lista'),
      text('Al doilea')])
  ]),
  element(p, [text('Paragraf 2')])
])
])

```

Se cere:

- i) Să se definească un predicat `html(X)` care este adevărat dacă și numai dacă `X` este un termen ce reprezintă un document HTML.
- ii) Să se definească un predicat `elimina_comentarii(Hi,He)` care este adevărat dacă și numai dacă `He` este documentul HTML rezultat prin eliminarea comentariilor din documentul HTML `Hi`.
- iii) Să se definească un predicat `extrage_text(H,L)` care este adevărat dacă `L` este lista textelor din documentul HTML `H`, colectate în ordinea în care apar în document sub forma de termeni `text(Text)`.
- iv) Să se definească un predicat `contorizare(H,Np)` care este adevărat dacă `Np` este numărul paragrafelor din documentul HTML `H`.

Problema 13 Lista de componente a unui anumit dispozitiv tehnic este o descriere a tuturor elementelor componente ale sale, până la cel mai înalt nivel de detaliere. O componentă poate apare de mai multe ori în cadrul unei liste de componente a unui dispozitiv principal, pe diverse nivele în structura de decompoziție a dispozitivului.

Spre exemplu lista de componente pentru un motor poate avea structura următoare:

```

1 motor
  4 bujii
  4 cilindrii

```

```

1 piston
  2 piulite
  3 garnituri
1 tija
  4 piulite
  4 suruburi
4 supape
  1 garnitura
  4 suruburi
  2 piulite
1 arbore cotit (vilbrochen)
  8 articulatii
    10 suruburi
    20 piulite

```

Lista de componente a unui dispozitiv tehnic se poate reprezenta printr-o listă Prolog cu elemente de forma `comp(P1,P2,C)`. Va exista câte un astfel de element pentru fiecare componentă P1 ce conține C părți de tipul P2. Lista poate conține și comentarii opționale de forma `com(Text)`.

Spre exemplu, motorul considerat se reprezintă astfel:

```

[com('Un motor se compune din '),
 com('4 bujii, 4 cilindrii si 4 supape'),
 comp(motor,bujie,4),
 comp(motor,cilindru,4),
 comp(motor,supapa,4),
 comp(motor,vilbrochen,1),
 com('Un cilindru se compune din'),
 com('1 piston si o tija'),
 comp(cilindru,piston,1),
 comp(cilindru,tija,1),
 comp(supapa,garnitura,1),
 comp(supapa,surub,4),
 comp(supapa,piulita,2),
 comp(vilbrochen,articulatie,8),
 comp(piston,piulita,2),
 comp(piston,garnitura,3),
 comp(tija,piulita,4),
 comp(tija,surub,4),
 comp(articulatie,surub,10),
 comp(articulatie,piulita,20)]

```

Se cere:

- i) Să se definească un predicat `disp(X)` care este adevărat dacă și numai dacă `X` este un termen ce reprezintă lista de componente a unui dispozitiv tehnic.
- ii) Să se definească un predicat `elimina_comentarii(Di,De)` care este adevărat dacă și numai dacă `De` este lista de componente rezultată prin eliminarea comentariilor din lista de componente `Di`.
- iii) Să se definească un predicat `extrage_parti(D,L)` care este adevărat dacă `L` este lista tuturor părților listei de componente `D`. Spre exemplu, pentru motorul considerat în exemplu această listă este:

```
[motor,bujie,cilindru,piston,  
piulita,garnitura,tija,surub,  
supapa,vilbrochen,articulatie]
```

Componentele se colectează prin parcurgerea în preordine a structurii arborescente a listei de decompoziție. Componentele unei anumite părți se vor vizita în ordinea în care apar în lista inițială de componente.

- iv) Să se definească un predicat `contorizare(D,L)` care este adevărat dacă `L` este lista componentelor primitive din lista de componente `D`. Pentru fiecare componentă se va indica și cantitatea corespunzătoare. Spre exemplu, pentru motorul din exemplu, `L` este:

```
[cant(bujie,4),cant(surub,168),  
cant(garnitura,16),cant(piulita,192)]
```

Elementele primitive se trec în lista `L` în ordinea în care sunt întâlnite la parcurgerea în preordine a listei de componente.

Problema 14 Se consideră un orar al activităților didactice dintr-o școală într-o anumită zi a săptămânii. Fiecare activitate are durata de o oră și se caracterizează prin următoarele informații: un identificator unic, o descriere, o sală de desfășurare, un profesor, o clasă de elevi și o oră de început.

Orarul se reprezintă printr-un termen `orar(Activitati)`. `Activitati` este lista activităților dintr-o zi a săptămânii. O activitate este un termen de forma `act(IdAct,Descr,IdSala,IdProf,IdStud,OraInceput)`.

Un exemplu de orar este următorul:

```

orar(
  [act(a1, 'Act 1', s1, p1, c1, 8),
   act(a4, 'Act 4', s1, p1, c2, 9),
   act(a6, 'Act 6', s2, p2, c2, 8),
   act(a2, 'Act 2', s3, p3, c3, 9),
   act(a5, 'Act 5', s3, p2, c1, 10),
   act(a3, 'Act 3', s2, p3, c2, 11)]

```

Se cere:

- i) Să se definească un predicat `orar(X)` care este adevărat dacă și numai dacă `X` este un termen ce reprezintă un orar.
- ii) Să se definească un predicat `determina_sali_prof_clase(O,S,P,C)` care este adevărat dacă și numai dacă `S`, `P` și `C` sunt listele de săli, profesori, clase care sunt folosite în cadrul orarului `O`. Fiecare sală, profesor, respectiv clasă se reprezintă prin identificatorul său unic.
- iii) Să se definească un predicat `conflict(A1,A2,O,C)` care este adevărat dacă `C` este lista conflictelor activităților `A1` și `A2` în orarul `O`. Două activități diferite care încep la aceeași oră pot fi în conflict de sală, profesor sau studenți. Spre exemplu activitățile `act(a20, 'Act 20', s3, p2, c1, 10)` și `act(a25, 'Act 25', s3, p3, c1, 10)` sunt în conflict de sală și clasă. Un conflict se reprezintă printr-un termen `c(Tip, Cauza)` unde `Tip` este unul din simbolurile `sala`, `prof`, `clasa` și `Cauza` este identificatorul unic al sălii, profesorului, respectiv clasei care a cauzat conflictul. Pentru exemplul considerat avem două conflicte : `c(sala, s3)` și `c(clasa, c1)`.
- iv) Să se definească un predicat `contorizare(O,N)` care este adevărat dacă `N` este numărul perechilor de activități diferite din orarul `O` care au cel puțin un conflict.

Problema 15 Se consideră problema jocului Perspico pentru o tablă de dimensiune 4.

O tablă pătrată de dimensiune 4×4 conține 15 plăcuțe glisante de dimensiuni 1×1 . Fiecare plăcuță este etichetată cu un număr natural de la 1 la 15. Un pătrățel al tablei este vid, el permițând astfel glisarea plăcuțelor cu o poziție la stânga, dreapta, în sus sau în jos.

Se pune problema determinării unei secvențe minime de mutări care să transforme o configurație inițială într-o configurație finală dată. Această problemă se poate rezolva folosind o strategie de căutare în spațiul stărilor.

O stare a problemei este dată de configurația plăcuțelor tablei. Această configurație se poate reprezenta printr-un termen `st(Lista, Pozitie)`. `Lista` este o listă de 16 numere naturale. Numerele reprezintă etichetele plăcuțelor, prin parcurgere pe linii, de la stânga la dreapta. Poziția vidă se consideră etichetată cu valoarea 0. `Pozitie` reprezintă indicele poziției vide în această listă (indicii se numără începând cu 1).

Spre exemplu, tabla următoare:

1	3	6	9
10	2	13	15
14	7		5
8	4	11	12

se reprezintă prin lista Prolog:

`st([1,3,6,9,10,2,13,15,14,7,0,5,8,4,11,12],11)`

Se cere:

- i) Să se definească un predicat `stare(X)` care este adevărat dacă și numai dacă `X` este un termen ce reprezintă o stare a problemei jocului Perspico.
- ii) Să se definească un predicat `vecini(S0,LS1)` care este adevărat dacă și numai dacă `LS1` este lista stărilor următoare stării `S0` în problema jocului Perspico. Mutările se consideră în ordinea stânga, sus, dreapta, jos.
- iii) Pentru rezolvarea problemei jocului Perspico utilizând un algoritm de căutare euristică, trebuie furnizată o funcție euristică a cărei valoare să reprezinte un estimator al numărului de mutări necesare pentru a ajunge din starea curentă în starea finală. Fie `S0` și `S1` două stări. Distanța dintre ele poate fi estimată contorizând toate perechile de plăcuțe aflate în poziții similare în cele două stări, dar care sunt etichetate cu valori diferite. Să se definească un predicat `eval_h(S0,S1,H)` care este adevărat dacă `H` este valoarea acestui estimator euristic al distanței dintre stările `S0` și `S1` ale problemei jocului Perspico.
- iv) Să se definească un predicat `vecini2(S0,LS1)` care este adevărat dacă `LS1` este lista stărilor succesoare stării `S0` obținute prin efectuarea a două mutări. Mutările se consideră în ordinea stânga, sus, dreapta, jos. Revenirea în `S0` va fi ignorată.

Problema 16 Se consideră un graf orientat etichetat reprezentat sub forma unei liste de arce. Fiecare arc are un nod sursă, un nod destinație, o etichetă și un cost pozitiv. Pentru orice arc, nodul sursă este diferit de nodul destinație.

Pentru reprezentarea unui graf orientat etichetat se folosește o listă de termeni de forma `arc(Nod0,Nod1,Cost,Eticheta)`.

Un exemplu de graf etichetat este:

```
[arc(a,b,2,et1),  
arc(b,c,3,et2),  
arc(a,d,4,et3),  
arc(b,d,1,et4),  
arc(d,a,1,et5)]
```

Se cere:

- i) Să se definească un predicat **graf**(**G**) care este adevărat dacă și numai dacă **G** este un termen ce reprezintă un graf orientat etichetat.
- ii) Să se definească un predicat **noduri**(**G**,**LN**) care este adevărat dacă **LN** este lista nodurilor grafului **G**. Nodurile se consideră în ordinea în care se întâlnesc în lista de arce a grafului.
- iii) Un graf etichetat se poate reprezenta și printr-o listă de termeni de forma **vecini**(**Nod0**,**ListaVecini**). **ListaVecini** este lista termenilor de forma **v**(**Nod1**,**Cost**,**Eticheta**) pentru care există un arc de la **Nod0** la **Nod1** de cost **Cost** și etichetat cu **Eticheta**. Să se definească un predicat **transforma**(**G0**,**G1**) care este adevărat dacă **G1** este reprezentarea cea nouă a grafului etichetat **G0**. Vecinii din **G1** se consideră în ordinea în care sunt întâlniți în lista de arce. Noua reprezentare pentru graful exemplu este:

```
[vecini(a, [v(b,2,et1),v(d,4,et3)]),  
vecini(b, [v(c,3,et2),v(d,1,et4)]),  
vecini(c, []),  
vecini(d, [v(a,1,et5)])]
```

- iv) Să se definească un predicat **drum_minim**(**N0**,**N1**,**G**,**Cale**) care este adevărat dacă **Cale** este o cale de lungime minimă de la nodul **Nod0** la nodul **Nod1** în graful **G**. O astfel de cale se va reprezenta prin lista arcelor sale. Un arc se va reprezenta printr-un termen **a**(**Ns**,**Nd**,**Et**) unde **Ns** este nodul sursă al arcului, **Nd** este nodul destinație al arcului și **Et** este eticheta arcului.

2 Modul de lucru

Pentru realizarea acestei teme va trebui să urmați cu strictețe următoarele indicații:

- i) Drept răspuns va trebui să furnizați două fișiere: **scenariu.pro** și **proiect.pro**. Împachetați aceste fișiere într-o arhivă ZIP pe care trimiteți-o prin email la adresa **c_badica@hotmail.com** până la data de 16.12.2002.

- ii) Fișierul **scenariu.pro** va conține datele de test pentru testarea programului dumneavoastră. Veți furniza date de test pentru fiecare punct al problemei urmând întocmai modelul din fișierul **scenariu.pro** furnizat ca exemplu. Pentru fiecare punct veți pregăti minim 5 seturi de date de test, din care cel puțin un test cu valori incorecte (vezi exemplul).

- iii) Fișierul **proiect.pro** va conține codul dumneavoastră. Este foarte important ca acest cod să fie cât mai bine documentat și cât mai îngrijit scris.

iv) Pentru testarea codului dumneavoastră se vor consulta întâi fișierele `proiect.pro` și `scenariu.pro`, în această ordine.

v) Orice formă de plagiat va fi depunctată drastic. În consecință, încercați să scrieți dumneavoastră codul și datele de test, nu să copiați aceste chestiuni de la colegi.