Laborator 12b

Structuri. Operatori. Liste

1. Structuri

O structura Prolog este un obiect ce desemneaza o colectie de obiecte corelate logic, care formeaza componentele structurii. Un exemplu este structura asociata obiectului carte, care este formata din componentele: titlu carte, autor si an aparitie. Un fapt ce refera relatia de posedare a unei carti de Prolog de catre Mihai poate fi exprimat astfel:

poseda(mihai, carte(prolog, clocksin, 1981)).

unde **carte(prolog, clocksin, 1981)** este o structura cu numele **carte** si cu trei componente: prolog, clocksin si 1981. Se admit si structuri imbricate, de exemplu:

poseda(mihai, carte(prolog, autori(clocksin, mellish), 1981)).

unde predicatul **poseda** are doua argumente: primul argument este constanta **mihai**, iar cel de-al doilea este structura **carte(prolog ....)**, cu doua componente, a doua componenta fiind structura **autori(clocksin, mellish)**.

In Prolog, o structura se defineste prin specificarea:

(1) numelui structurii (*functorul structurii*);

(2) elementelor structurii (*componentele structurii*).

Structurile Prolog pot fi utilizate pentru reprezentarea structurilor de date, de exemplu liste sau arbori. Un arbore binar poate fi reprezentat in Prolog tot printr-o structura, de exemplu:

arb(1, arb(2, vid, vid), vid).

reprezinta un arbore binar cu cheia din radacina 1, cu subarborele drept vid si cu subarborele stang format dintr-un singur nod cu cheia 2.

Observatie

Sintaxa structurilor este aceeasi cu cea a faptelor Prolog. Un predicat Prolog poate fi vazut ca o structura a carui functor este numele predicatului, iar argumentele acestuia reprezinta componentele structurii (structuri => date, predicate => programe).

2. Operatori

Uneori este convenabil sa se scrie anumiti functori (nume de structuri sau predicate) in forma infixata. Aceasta este o forma sintactica ce mareste claritatea programului, cum ar fi cazul operatorilor aritmetici sau al operatorilor relationali.

(1) *Operatori aritmetici*

Operatorii aritmetici binari, cum ar fi +, -, *, /, pot fi scrisi in Prolog in notatie infixata; de exemplu:

1+2*(3*4)/5.

Aceasta sintaxa este de fapt o rescriere infixata a formei prefixate a structurilor echivalente:

+(1,/(*(2,*(3,4)),5)).

Este important de retinut ca operatorii aritmetici sunt o rescriere infixata a unor structuri deoarce valoarea expresiei astfel definita nu este calculata. Evaluarea expresiei se face la cerere in cazul in care se foloseste operatorul predefinit infixat **is**, de exemplu:

X is 1 + 2.

va avea ca efect instantierea variabilei **X** la valoarea 3.

(2) *Operatori relationali*

Operatorii relationali sunt predicate predefinite infixate. Un astfel de operator este *operatorul de egalitate* =. Operatorul de egalitate functioneaza ca si cum ar fi definit prin urmatorul fapt:

X = X.

iar incercarea de a satisface un scop de tipul **X = Y** se face prin incercarea de a unifica **X** cu **Y**. Din aceasta cauza, dandu-se un scop de tipul **X = Y**, regulile de decizie care indica daca scopul se indeplineste sau nu sunt urmatoarele:

Daca **X** este variabila neinstantiata, iar **Y** este instantiata la orice obiect Prolog, atunci scopul reuseste. Ca efect lateral, **X** se va instantia la aceeasi valoare cu cea a lui **Y**. De exemplu:

?- carte(barbu, poezii) = X.

este un scop care reuseste si X se instantiaza la carte(barbu, poezii).

Daca atat **X** cat si **Y** sunt variabile neinstantiate, scopul **X = Y** reuseste, variabila **X** este legata la **Y** si reciproc. Aceasta inseamna ca ori de cate ori una dintre cele doua variabile se instantiaza la o anumita valoare, cealalta variabila se va instantia la aceeasi valoare. Exemplu:

?- X=Y,X=3.

X = 3

Y = 3

Atomii si numerele sunt intotdeauna egali cu ei insisi. De exemplu, urmatoarele scopuri au rezultatul marcat drept comentariu:

mihai = mihai % reuseste

mare = mic % esueaza

102 = 102 % reuseste

1010 = 1011 % esueaza

Doua structuri sunt egale daca au acelasi functor, acelasi numar de componente si fiecare componenta dintr-o structura este egala cu componenta corespunzatoare din cealalta structura. De exemplu, scopul:

autor(barbilian, ciclu(uvedenrode, poezie(riga_crypto))) =

autor(X, ciclu(uvedenrode, poezie(Y)))

este satisfacut, iar ca efect lateral se fac instantierile:

X = barbilian si Y = riga_crypto.

Operatorul de inegalitate \= se defineste ca un predicat opus celui de egalitate. Scopul **X \= Y** reuseste daca scopul **X = Y** nu este satisfacut si esueaza daca **X = Y** reuseste. In plus, exista predicatele relationale de inegalitate definite prin operatorii infixati **>**, **<**, **=<**, **>=**.

Un operator interesant este **=:=**. El face numai evaluare aritmetica si nici o instantiere. Exemplu:

?- 1 + 2 =:= 2 + 1.

Yes

?- 1 + 2 = 2 + 1.

No

In cele mai multe implementari Prolog exista predefinit operatorul de obtinere a modulului unui numar, **mod**; scopul

X is 5 mod 3.

reuseste si X este instantiat la 2.

Comentariile dintr-un program Prolog sunt precedate de caracterul "**%**".

Exemple:

1. O definitie posibila a predicatului **modulo(X, Y, Z)**, cu semnificatia argumentelor Z = X mod Y, presupunand X, Y > 0, este:

% modulo(X, Y, Z)

modulo(X, Y, X) **:-** X < Y.

modulo(X, Y, Z) **:-** X >= Y, X1 is X - Y, modulo(X1, Y, Z).

2. Plecand de la predicatul **modulo** definit anterior, se poate defini predicatul de calcul al celui mai mare divizor comun al doua numere, conform algoritmului lui Euclid, presupunand X > 0, Y > 0, astfel:

```
% cmmdc(X, Y, C)
cmmdc(X, 0, X).
cmmdc(X, Y, C) :- modulo(X, Y, Z), cmmdc(Y, Z, C).
```

Testare:

```
?- cmmdc(15, 25, C).
C=5
```

raspunsul sistemului este corect. In cazul in care se incearca obtinerea unor noi solutii, se observa ca sistemul intra intr-o bucla infinita datorita imposibilitatii resatisfacerii scopului **modulo(X, Y, Z)** pentru Y = 0. Daca la definitia predicatului **modulo** se adauga faptul:

```
modulo(X, 0, X).
```

atunci predicatul **modulo(X, Y, Z)** va genera la fiecare resatisfacere aceeasi solutie, respectiv solutia corecta, la infinit.

Pentru a nu permite resatisfacerea scopului cmmdc(15, 25, C) se modifica definitia lui cmmdc astfel:

```
cmmdc(X, 0, X).
cmmdc(X, Y, C) :- Y \= 0, modulo(X, Y, Z), cmmdc(Y, Z, C).
```

3. Cel mai mic multiplu comun a 2 numere se poate afla folosind predicatul cmmmc:

```
% cmmmc(X, Y, C)
cmmmc(X,Y,C):-cmmdc(X,Y,A),C is X*Y/A.
```

3. Liste

O lista este o structura de date ce reprezinta o secventa ordonata de zero sau mai multe elemente. O lista poate fi definita recursiv astfel:

(1) lista vida (lista cu 0 elemente) este o lista

(2) o lista este o structura cu doua componente: primul element din lista (capul listei) si restul listei (lista formata din urmatoarele elemente din lista).

Sfarsitul unei liste este de obicei reprezentat ca lista vida.

In Prolog structura de lista este reprezentata printr-o structura standard, predefinita, al carei functor este caracterul "**.**" si are doua componente: primul element al listei si restul listei. Lista vida este reprezentata prin atomul special **[ ]**. De exemplu, o lista cu un singur element **a** se reprezinta in Prolog, prin notatie prefixata astfel:

.(a, [ ])

iar o lista cu trei elemene, **a**, **b**, **c**, se reprezinta ca:

.(a, **.** (b, **.** (c, [ ])))

Deoarece structura de lista este foarte des utilizata in Prolog, limbajul ofera o sintaxa alternativa pentru descrierea listelor, formata din elementele listei separate de virgula si incadrate de paranteze drepte. De exemplu, cele doua liste anterioare pot fi exprimate astfel:

[a]

[a, b, c]

Aceasta sintaxa a listelor este generala si valabila in orice implementare Prolog. O operatie frecventa asupra listelor este obtinerea primului element dintr-o lista si a restului listei, deci a celor doua componente ale structurii de lista. Aceasta operatie este realizata in Prolog de operatorul de scindare a listelor "|" scris sub urmatoarea forma:

[Prim | Rest]

Variabila **Prim** sta pe postul primului element din lista, iar variabila **Rest** pe postul listei care contine toate elementele din lista cu exceptia primului. Acest operator poate fi aplicat pe orice lista care contine cel putin un element. Daca lista contine exact un element, **Rest** va reprezenta lista vida. incercarea de identificare a structurii **[Prim | Rest]** cu o lista vida duce la esec. Mergand mai departe, se pot obtine chiar primele elemente ale listei si restul listei. Iata cateva echivalente:

[a, b, c] = [a | [b, c]] = [a, b | [c]] = [a, b, c | [ ]] = [a | [b | [c]]] = [a | [b | [c | [ ]]]].

In Prolog elementele unei liste pot fi atomi, numere, liste si in general orice structuri. In consecinta se pot construi liste de liste.

Exemple:

1. Se poate defini urmatoarea structura de lista:

[carte(barbu, poezii), carte(clocksin, prolog)]

2. Considerand urmatoarele fapte existente in baza de cunostinte Prolog

pred([1, 2, 3, 4]).

pred([coco, sta, pe, [masa, alba]]).

se pot pune urmatoarele intrebari obtinand raspunsurile specificate:

?- pred([Prim | Rest]).

Prim = 1

Rest = [2, 3, 4];

Prim = coco

Rest = [sta, pe, [masa, alba]];

No

?- pred([_ , _ , _ , [_ | Rest]]).

Rest = [alba];

No

3. Urmatorul predicat testeaza apartenenta unui element la o lista:

```
% membru(Element, Lista)
membru(Element, [Element | _ ]).
membru(Element, [ _ | RestLista]) :- membru(Element, RestLista).
```

Testare:

?- membru(b, [a, b, c]).

Yes

?- membru(X, [a, b, c]).

X = a;

X = b;

X = c;

No

?- membru(b, [a, X, b]).

X = b;

X = _G160;

No

4. In continuare este prezentat predicatul de concatenare a doua liste L1 si L2. Rezultatul este L3:

```
% conc(L1, L2, L3)
conc([], L2, L2).
conc([Prim1|Rest1], Lista2, [Prim1|Rest3]) :- conc(Rest1, Lista2, Rest3).
```

Testare:

?- conc([a, b], [c, d, e], L3).

L3 = [a, b, c, d, e];

No

?- conc(L1, [c, d, e], [a, b, c, d, e]).

L1 = [a, b];

No

?- conc([a, b], L2, [a, b, c, d, e]).

L2 = [c, d, e];

No

?- conc(L1, L2, [a, b]).

L1 = [ ]

L2 = [a, b];

L1 = [a]

L2 = [b];

L1 = [a, b]

L2 = [ ];

No

Observatie: Acest laborator are ca sursa cartea "Tehnici de programare Prolog pentru inteligenta artificiala", autori: A.M. Florea, S. Radu, A.H. Mogos, Editura Printech, 2007