

# Programare Logică – LISTĂ SUBIECTE DE EXAMEN

Claudia MUREȘAN, c.muresan@yahoo.com, cmuresan@fmi.unibuc.ro

UNIVERSITATEA DIN BUCUREȘTI, FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

2019–2020, Semestrul I

**Exercițiul 1.** Considerăm un limbaj de ordinul I conținând un simbol de operație binară  $f$  și două simboluri distincte de operații unare  $g$  și  $h$ . Fie  $V$ ,  $X$  și  $Y$  variabile distincte.

Să se deseneze arborii de expresii asociați următorilor doi termeni, apoi, prin aplicarea algoritmului de unificare, să se determine dacă acești termeni au unificator și, în caz afirmativ, să se determine un cel mai general unificator pentru aceștia:

$$f(f(h(X), h(h(V))), g(h(f(X, V)))) \text{ și } f(f(h(h(V)), h(h(Y))), g(h(f(X, Y)))).$$

**Exercițiul 2.** Având următoarea bază de cunoștințe în Prolog, scrisă respectând sintaxa Prolog:

```
barbat(alex). barbat(andrei).  
femeie(ana). femeie(carmen). femeie(elena). femeie(maria).  
ochi(alex, negri).  
ochi(ana, caprui).  
ochi(andrei, albastri).  
ochi(carmen, violeti).  
ochi(elena, verzi).  
ochi(maria, caprui).  
place(B, F) :- barbat(B), femeie(F), ochi(F, violeti).  
place(B, F) :- barbat(B), femeie(F), ochi(F, verzi).  
place(F, B) :- femeie(F), barbat(B), ochi(B, albastri).  
logoditi(B, F) :- place(B, F), place(F, B).  
rivali(P, Q) :- femeie(P), femeie(Q), place(B, P), place(B, Q).  
rivali(P, Q) :- barbat(P), barbat(Q), place(F, P), place(F, Q).  
să se scrie arborele de derivare prin rezoluție SLD pentru următoarea interogare:  
?- rivali(Cine, CuCine).
```

După obținerea tuturor soluțiilor interogării, dacă apar în arborele de derivare noduri pentru care expandarea (recursia) continuă la infinit, să se indice acele noduri, fără a continua cu expandarea lor.

Toate nodurile care au doar nu număr finit de descendenți vor fi expandate complet, chiar dacă produc aceleași soluții de mai multe ori.

**Exercițiul 3.** Să se scrie în Prolog un predicat binar *listintermnr(Lista, ListaElemListe)*, definit ca mai jos, precum și toate predicatele auxiliare necesare pentru definirea acestuia:

*listintermnr* să fie satisfăcut ddacă ambele sale argumente sunt liste, iar al doilea argument se obține din primul prin înlocuirea secvențelor nevide de elemente ale sale de pe poziții consecutive care nu sunt constante numerice cu listele formate din elementele din acele secvențe;

și, într-o interogare în Prolog, *listintermnr* să funcționeze sub forma: dacă primește o listă arbitrară *Listă* în primul argument, să construiască în al doilea argument lista *L* obținută din *Listă* lăsând constantele numerice din *Listă* pe loc și înlocuind șirurile (nevide) de elemente  $E_1, \dots, E_n$  din *Listă* cuprinse între constantele numerice cu câte un singur element, dat de lista formată din elementele  $E_1, \dots, E_n$ ; de exemplu:

la interogările următoare:	Prologul să răspundă:
?- <i>listintermnr</i> ([], <i>L</i> ).	$L = []$ ;
?- <i>listintermnr</i> ([1, -2.5, 0.0, 0], <i>L</i> ).	$L = [1, -2.5, 0.0, 0]$ ;
?- <i>listintermnr</i> ([0, <i>a</i> , <i>b</i> , <i>X</i> , 1, -1, <i>X</i> , <i>a</i> ], <i>L</i> ).	$L = [0, [a, b, X], 1, -1, [X, a]]$ ;

iar, la interogarea:

?- *listintermnr*([[*f*(*a*), [1, *a*, *V*, 2]], [], 0, [*a*], [1, -1], [*b*, *X*], [1, *a*, *b*, -1], 1, -1, *f*(*X*), *a*, 2], *L*).

Prologul să răspundă:

*L* = [[[*f*(*a*), [1, *a*, *V*, 2]], [], 0, [[*a*], [1, -1], [*b*, *X*], [1, *a*, *b*, -1]], 1, -1, [*f*(*X*), *a*], 2].

**Exercițiul 4.** Să se scrie în Prolog un predicat binar *argtermcomp*(*Termen*, *TermenModificat*) definit ca mai jos, precum și toate predicatele auxiliare necesare pentru definirea acestuia:

*argtermcomp* să fie satisfăcut ddacă ambele argumente ale sale sunt termeni Prolog, iar al doilea argument al său coincide cu primul, dacă primul argument este o constantă sau o variabilă, iar, dacă primul argument este un termen compus, atunci doilea argument se obține din primul prin înlocuirea, în fiecare subtermen, a secvențelor de argumente consecutive ale operatorului dominant al acelui subtermen care nu sunt termeni compuși cu un singur argument, egal cu numărul argumentelor din secvența înlocuită;

și, într-o interogare în Prolog, *argtermcomp* să funcționeze sub forma: dacă primește în primul argument un termen Prolog arbitrar *T*, să construiască în al doilea argument termenul obținut din *T* ca mai sus; de exemplu:

la interogările următoare:	Prologul să răspundă:
?- <i>argtermcomp</i> ( <i>X</i> , <i>Termen</i> ).	<i>Termen</i> = <i>X</i> ;
?- <i>argtermcomp</i> (1.5, <i>Termen</i> ).	<i>Termen</i> = 1.5;
?- <i>argtermcomp</i> ( <i>c</i> , <i>Termen</i> ).	<i>Termen</i> = <i>c</i> ;
?- <i>argtermcomp</i> ( <i>f</i> ( <i>X</i> , <i>f</i> ( <i>a</i> , <i>g</i> ( <i>b</i> )), 1), <i>Termen</i> ).	<i>Termen</i> = <i>f</i> (1, <i>f</i> (1, <i>g</i> (1)), 1);

iar, la interogarea:

?- *argtermcomp*(*f*(*X*, *X*, *Y*, 1, *f*(*a*), *f*(*a*), *u*, *V*, *f*(*a*, *b*, *g*(*f*(*a*), *c*, *V*, *h*(*a*, *X*, *b*), 2.5))), *Termen*).

Prologul să răspundă:

*Termen* = *f*(4, *f*(1), *f*(1), 2, *f*(2, *g*(*f*(1), 2, *h*(3), 1))).