

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 ternară f , unul de operație binară g , unul de operație unară h și un simbol de constantă c . Fie X , Y și Z 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:

$f(g(h(X), g(X, Y)), f(X, h(X), g(X, X)), h(h(Y)))$ și $f(g(h(c), g(h(c), h(c))), f(c, h(Z), g(h(c), h(c))), h(h(Z)))$.

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

joc(sah, 2). joc(reversi, 2). joc(solitaire, 1).

joc(cartiJoc, OricatiJucatori).

prefera(ana, Joc) :- joc(Joc, 1).

prefera(george, Joc) :- joc(Joc, 5).

prefera(victor, Joc) :- joc(Joc, 2).

prefera(victor, Joc) :- prefera(george, Joc).

să se scrie arborele de derivare prin rezoluție SLD pentru următoarea interogare:

?- *prefera(victor, Ce).*

și să se determine soluțiile date de Prolog acestei interogări folosind acest arbore de derivare.

Exercițiul 3. Să se scrie în Prolog un predicat binar

listmaxfelemlist(ListaListe, CMLungaListaFaraElementeListe),

definit ca mai jos, precum și toate predicatele auxiliare necesare pentru definirea acestuia:

listmaxfelemlist să fie satisfăcut primul său argument este o listă LL de liste având elemente care nu au elemente de tip listă, iar al doilea argument al său este una dintre listele din LL care nu are elemente de tip listă la rândul ei și este de lungime maximă raportat la această proprietate;

și, într-o interogare în Prolog, *listmaxfelemlist* să funcționeze sub forma: dacă primește, în primul argument, o listă arbitrară de liste LL , atunci să întoarcă false dacă LL nu conține liste care să nu conțină liste, iar, în caz contrar, să obțină în al doilea argument o listă L din LL de lungime maximă astfel încât L să nu conțină liste; de exemplu:

la interogările următoare:	Prologul să răspundă:
?- <i>listmaxfelemlist([], Lista).</i>	<i>false;</i>
?- <i>listmaxfelemlist([[], [1, 2, [V]]], Lista).</i>	<i>false;</i>
?- <i>listmaxfelemlist([[], [1, 2, [V]]], Lista).</i>	<i>Lista = [];</i>
?- <i>listmaxfelemlist([[], [1, 2], [], [], X], [a, V], [x, [1, 2], c], Lista).</i>	<i>Lista = [1, 2] sau Lista = [a, V];</i>
?- <i>listmaxfelemlist([[], [1, 2], [], [], X], [a, b, c, A], [x, [1, 2], c], Lista).</i>	<i>Lista = [a, b, c, A].</i>

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

adaugnrvar să fie satisfăcut ddacă ambele argumente ale sale sunt termeni Prolog, iar al doilea argument al său se obține din primul în modul următor: pentru fiecare subtermen care nu e nici variabilă, nici

constantă, se va adăuga, ca prim argument al operatorului dominant al acelui subtermen, numărul de variabile care apar în acel subtermen, indiferent de numele acestora (așadar, o variabilă cu același nume este numărată de ori de câte ori apare);

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

la interogările următoare:	Prologul să răspundă:
?- <i>adaugnrvar</i> (ct , $Termen$).	$Termen = ct$;
?- <i>adaugnrvar</i> ($f(ct)$, $Termen$).	$Termen = f(0, ct)$;
?- <i>adaugnrvar</i> ($f(V)$, $Termen$).	$Termen = f(1, ct)$;
?- <i>adaugnrvar</i> ($f(V, g(V, X), h(a), h(g(c, V)))$, $Termen$);	$Termen = f(4, V, g(2, V, X), h(0, a), h(1, g(1, c, V)))$.