Programare Logică – listă subiecte de examen

Claudia MUREŞAN, c.muresan@yahoo.com, cmuresan@fmi.unibuc.ro
Universitatea din Bucureşti, Facultatea de Matematică şi Informatică

2019–2020, Semestrul I

Exerciţiul 1. Considerăm un limbaj de ordinul I conţinând un simbol de operaţie ternară f, unul de operaţie binară g, unul de operaţie unară h şi un simbol de constantă c. Fie X, Y şi Z variabile distincte.

Să se deseneze arborii de expresii asociaţi următorilor doi termeni, apoi, prin aplicarea algoritmului de unificare, să se determine dacă aceşti termeni au unificator şi, în caz afirmativ, să se determine un cel mai general unificator pentru aceştia:

$$f(g(h(X), g(X, Y)), f(X, h(X), g(X, X)), h(h(Y))) \text{ şi } f(g(h(c), g(Z, Z)), f(c, h(Z), g(Z, Z)), h(h(Z))).$$

Exerciţiul 2. Având următoarea bază de cunoştinţe în Prolog, scrisă respectând sintaxa Prolog:
$joc(sah, 2).$ $joc(solitaire, 1).$
$joc(cartiJoc, OricatiJucatori).$
$prefera(ana, Joc)$:- $joc(Joc, 1).$
$prefera(victor, Joc)$:- $joc(Joc, 2).$
$prefera(george, Joc)$:- $prefera(ana, Joc).$
$prefera(maria, Joc)$:- $prefera(victor, Joc).$
să se scrie arborele de derivare prin rezoluţie SLD pentru următoarea interogare:
?- $prefera(Cine, sah).$
şi să se determine soluţiile date de Prolog acestei interogări folosind acest arbore de derivare.

Exerciţiul 3. Să se scrie în Prolog un predicat binar

$$listmaxfelemlist(ListaListe, CMLungaListaFaraElementeListe),$$

definit ca mai jos, precum şi 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 şi este de lungime maximă raportat la această proprietate;

şi, într–o interogare în Prolog, $listmaxfelemlist$ să funcţioneze sub forma: dacă primeşte, în primul argument, o listă arbitrară de liste LL, atunci să întoarcă false dacă LL nu conţine liste care să nu conţină liste, iar, în caz contrar, să obţină în al doilea argument o listă L din LL de lungime maximă astfel încât L să nu conţină liste; de exemplu:

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

Exerciţiul 4. Să se scrie în Prolog un predicat binar $adaugnrvar(Termen, TermenModificat)$ definit ca mai jos, precum şi 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 obţine 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 (aşadar, o variabilă cu acelaşi nume este numărată de ori de câte ori apare);

şi, într–o interogare în Prolog, *adaugnrvar* să funcţioneze sub forma: dacă primeşte un termen arbitrar T în primul argument, să obţină, în al doilea argument, termenul construit din T ca mai sus; de exemplu:

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