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 unară f, un simbol de relaţie unară p şi unul de relaţie binară q. Fie x, y şi z variabile distincte.

Să se pună următorul enunţ într–o formă Skolem, apoi să se aplice algoritmul Davis–Putnam acelei forme Skolem:

$$\exists\, x\, [p(f(x)) \vee q(x, f(x))] \to [\forall\, y\, p(f(y)) \wedge \forall\, z\, q(z, f(z))].$$

Exerciţiul 2. Având următoarea bază de cunoştinţe în Prolog, scrisă respectând sintaxa Prolog:

$are(ion, electronice, vechi)$.
$are(intel, electronice, vechi)$.
$are(intel, electronice, noi)$.
$recicleaza(mircea, electronice)$:- $are(Cineva, electronice, vechi)$.
$repara(mircea, electronice)$.
$cumpara(P, electronice)$:- $are(P, electronice, vechi)$.
$recicleaza(P, electronice)$:- $are(P, electronice, vechi)$.
$vinde(P, electronice)$:- $are(P, electronice, noi)$.
$produce(P, electronice)$:- $recicleaza(P, electronice)$.
$fabrica(P)$:- $produce(P, electronice)$, $vinde(P, electronice)$.
$electronist(P)$:- $produce(P, electronice)$, $repara(P, electronice)$.
$persoana(P)$:- $cumpara(P, electronice)$.
$persoana(P)$:- $recicleaza(P, electronice)$.

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

?- $electronist(Cine)$.

După obţinerea tuturor soluţiilor 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.

Exerciţiul 3. Să se scrie în Prolog declaraţii pentru doi operatori unari prefixaţi *lista_nr* şi *nr_elem*, ambii de precedenţă 300, şi un predicat binar

$$nrlistanrelem(ListaListe, ListaListecuNrListeisiNrdeElementealeListeiAdaugate),$$

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

nrlistanrelem să fie satisfăcut ddacă ambele sale argumente sunt liste de liste, iar al doilea argument al său se obţine din primul prin modificarea fiecărei liste L din lista de liste $ListL$ din acest prim argument astfel: în capul listei cu care va fi înlocuită L în doilea argument, să se adauge termenul *lista_nr* N, unde N este numărul care indică al câtulea element al lui $ListL$ este lista L (cu elementele lui $ListL$ numărate de la stânga la dreapta, începând de la 1 şi până la lungimea listei $ListL$), iar la sfârşitul listei cu care va fi înlocuită L în doilea argument, să se adauge termenul *nr_elem* K, unde K este numărul de elemente ale listei L (în forma ei iniţială, în care apare în $ListL$);

şi, într–o interogare în Prolog, *nrlistanrelem* să funcţioneze sub forma: dacă primeşte o listă arbitrară de liste $ListL$ în primul argument, să obţină în al doilea argument lista elementelor L ale lui $ListL$ modificate prin adăugarea la fiecare dintre aceste elemente L, în capul listei obţinute din L, a termenului *lista_nr* N, unde N este poziţia lui L in $ListL$, iar, la coada listei obţinute din L, a termenului *nr_elem* K, unde K este lungimea listei L; de exemplu:

la interogările următoare:	Prologul să răspundă:
?- $nrlistanrelem([\,], LL)$.	$LL = [\,]$;
?- $nrlistanrelem([[\,]], L)$.	$L = [[lista\_nr\ 1, nr\_elem\ 0]]$;

iar, la interogarea următoare:

?- $nrlistanrelem([[1,2,3],[a,X],[\,],[f(X)],[V,V,V,V,V],[[a,b],[c]],[[a,b],V,[V,[a,b],[c]],[[c],[c]],V],L)$.

Prologul să răspundă:

$LL = [[lista\_nr\ 1, 1, 2, 3, nr\_elem\ 3], [lista\_nr\ 2, a, X, nr\_elem\ 2], [lista\_nr\ 3, nr\_elem\ 0], [lista\_nr\ 4, f(X), nr\_elem\ 1], [lista\_nr\ 4, V, V, V, V, V, nr\_elem\ 5], [lista\_nr\ 6, [a,b], [c], nr\_elem\ 2], [lista\_nr\ 7,\ [a,b],\ V,\ [V,[a,b],[c]],\ [[c],[c]],\ V], nr\_elem\ 5]$.

Exerciţiul 4. Să se scrie în Prolog un predicat binar $elimarg(Termen, TermenModificat)$ definit ca mai jos, precum şi toate predicatele auxiliare necesare pentru definirea acestuia:

elimarg să fie satisfăcut ddacă ambele argumente ale sale sunt termeni Prolog, iar al doilea argument al său se obţine din primul modificând argumentele operatorului dominant f al fiecărui subtermen al lui T astfel: dacă f are argumente care nu sunt nici constante numerice, nici termeni compuşi, atunci acestea vor fi eliminate dintre argumentele lui f, iar numărul acestor argumente ale lui f care au fost eliminate va fi adăugat ca prim argument al lui f în acel subtermen al lui T;

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

la interogările următoare:	Prologul să răspundă:
?- $elimarg(X, Termen)$.	$Termen = X$;
?- $elimarg(c, Termen)$.	$Termen = c$;
?- $elimarg([\,], Termen)$.	$Termen = [\,]$;
?- $elimarg(f(X, c, f(V), Y), Termen)$.	$Termen = f(3, f(1))$;

iar, la interogarea următoare:

?- $elimarg(f(f(V, g(g(1.5), X, X, 10, V), c), X, 1, g(c), g(2), Y), Termen)$.

Prologul să răspundă:

$Termen = f(3, f(2, g(3, g(1.5), 10)), g(1), g(2))$.