FMI, Info, Anul II, 2018-2019
Programare logică

Seminar 4
Unificare si arborele de executie. Forma prenex. Skolemizare.

(S4.1) Folosind algoritmul de unificare, indicaţi dacă următoarea listă de termeni are unificator:

$$U = p(x, f(y, z)), p(x, a), p(x, g(h(k(x))))$$

(p, f sunt simboluri de funţii de aritate 2, g, h, k sunt simboluri de funcţii de aritate 1, a este simbol de constantă, x, y, z sunt variabile).

(S4.2) Găsiţi răspunsul dat de Prolog la următoarele întrebări:

1. 'Luke' = luke.
2. Luke = luke.
3. jedi(luke) = luke.
4. jedi(luke) = X.
5. jedi(luke) = jedi(X).
6. father(vader, A) = father(B, luke).
7. heroes(han, X, luke) = heroes(Y, ben, X).
8. heroes(han, X, luke) = heroes(Y, ben).
9. jedi(X) = X.
10. characters(hero(luke), villain(vader)) = characters(X, Y).
11. characters(hero(luke), X) = characters(X, villain(vader))

Demonstraţie: https://www.scss.tcd.ie/ dwoods/1617/CS1LL2/HT/wk4/lec4.pdf □

(S4.3) În această problemă W,R,O,N,G,I,H,T reprezintă nişte cifre distincte de la 0 la 9. Scrieţi un program în Prolog care găseşte valori pentru aceste variabile astfel încât expresia

$$\text{WRONG} + \text{WRONG} = \text{RIGHT}$$

să fie adevărată (unde WRONG şi RIGHT sunt numere cu câte 5 cifre).

De exemplu, dacă W = 1, R = 2, O = 7, N = 3, G = 4, I = 5, H = 6, T = 8 atunci avem
12734 + 12734 = 25468

(S4.4) Execution tree

```
q(a). q(b).  r(c).  r(d). s(e).
top(X,Y) :- p(X,Y).
top(X,X) :- s(X).
p(X,Y) :- q(X), r(Y).
p(X,Y) :- s(X), r(Y).

?- top(X,Y).
```

Desenaţi arborele de execuţie pentru întrebarea `top(X,Y)` indicând răspunsul pentru fiecare ramură.

Demonstraţie:

https://www.dcc.fc.up.pt/ ines/aulas/1617/PL/problems5.pdf □

(S4.5) Scrieţi în Prolog un predicat `list_poz(L,R)` care este adevărat dacă `R` este o listă de perechi care conţin elementele din `L` şi poziţia pe care se află în `L`, poziţiile fiind în ordine crescătoare:

```
list_poz([a,b,c], [(1,a),(2,b),(3,c)])
```

Predicatul trebuie să verifice, dar şi să instanţieze.

(S4.6) Implementaţi algoritmul *Quicksort* în Prolog.

Demonstraţie: https://www.cp.eng.chula.ac.th/ piak/teaching/dsys/2004/quick-prolog.htm

```
quicksort([X|Xs],Ys) :- partition(Xs,X,Left,Right), quicksort(Left,Ls), quicksort(Right,Rs), append(Ls,[X|Rs],Ys).
quicksort([],[]).
partition([X|Xs],Y,[X|Ls],Rs) :- X <= Y, partition(Xs,Y,Ls,Rs).
partition([X|Xs],Y,Ls,[X|Rs]) :- X > Y, partition(Xs,Y,Ls,Rs).
partition([],Y,[],[]).
append([],Ys,Ys).
append([X|Xs],Ys,[X|Zs]) :- append(Xs,Ys,Zs).
```

□

(S4.7) (1)Scrieţi un predicat `flatten(L,R)` unde `L` este o listă de liste, iar `R` conţine elementele lui `L`, în aceeaşi ordine, dar elimină gruparea în liste:

```
flatten([[1,2], [3,4], [5]], [1,2,3,4,5]).
```

Predicatul trebuie să facă verificare şi să găsească `R` cu `L` instanţiat.