Kontrollskrivning i DD1361 Programmeringsparadigmer: Prolog

2010-09-21

Vid godkänt på denna KS har du klarat Prologavsnittet på alla framtida tentor i DD1361. Det krävs 12 poäng för godkänt. Du kan tillgodoräkna dig en bonuspoäng för varje godkänd Prologlab. Inga hjälpmedel utöver skrivmedel är tillåtna.

Namn:
Poängsumma:

1. Vad är ett *mål* (eng: *goal*) i Prolog? (1p)
   (a) *Ett uttryck* som Prologsystemet söker en lösning för.
   (b) *En tilldelning* av värden till parametrar som gör att ett predikat lyckas.
   (c) *Alla tilldelningar* av värden till parametrar som gör att ett predikat lyckas.
   (d) Ett *predikat med både huvud och kropp.*
2. Vad kallas det att binda ett värde till en parameter i Prolog? (1p)
   (a) Satisfiera
   (b) *Backtrack*
   (c) Unifiera
   (d) Tilldela
3. Betrakta följande kod:

```
predikatet(X, Z) :-
   pred1(X, Y), !, pred2(Y, Z).
```

   Vilket påstående är sant? (1p)
   (a) Det räcker med att `pred1` *eller* `pred2` lyckas för att `predikatet` ska lyckas.
   (b) Det krävs att både `pred1 och pred2` lyckas för att `predikatet` ska lyckas.
   (c) Precis *ett* av `pred1 och pred2` lyckas för att `predikatet` ska lyckas.
4. Skriv ett predikat `bra` som avgör (lyckas) om ett tal är i det intervallet $[0, 1]$ (d.v.s., alla tal mellan 0 och 1) eller inte. (2p)
   Kod:

5. Antag att vi representerar mängder med listor. Predikatet `union/3`, härnedan, returnerar unionen av de två mängderna som ges av de två första parametrarna.

```
union([H|T], L2, Lut) :-
    member(H,L2), union(T, L2, Lut).
union([H|T], L2, [H|Lut]) :-
    union(T, L2, Lut).
```

Koden har två problem som du ska rätta till!

(a) Det saknas ett basfall. Hur skriver du det? (2p)

(b) Koden kan ge listor som innehåller samma element två gånger (det tillåts inte för mängder). Använd ett *snitt* (eng: *cut*) för att åtgärda detta! (Markera tydligt i koden var du lägger snittet.) (2p)

6. Antag att vi lagrar persondata som sammansatta termer `person(Firstname,Lastname,Birthyear)` i en lista, som till exempel

```
[person("Bo", "Ek", 1958), person("Alva", "Graan", 1987)].
```

Skriv ett predikat `birthyear(+PL, -Year)` som givet en lista persondata i `PL` kan hitta ettt födelseår i listan. Predikatet ska kunna exekveras som i exemplet (antag att `pl` plockar fram listan ovan!). (3p)

Exempelkörning:

```
| ?- pl(_PL), birthyear(_PL, Y).
Y = 1958 ? ;
Y = 1987 ? ;
no
| ?- pl(_PL), birthyear(_PL, 1958).
true ?
yes
```

Kod:

7. Skriv ett predikat `listlengths/2` som tar en lista av listor som första parameter och returnerar en lista av listlängder i den andra parametern. Du kan använda standardpredikatet `length/2` för att beräkna längden på en lista. (4p)

   Exempelkörning:

```
| ?- lengths([[1,1], ['kth'], [], [4,3,2,1]], L).
L = [2, 1, 0, 4] ? ;
no
```

   Kod:

8. Betrakta Prologprogrammet nedan och illustrera med lådmodellen hur `otyglad(X)` beräknas.(4p)

```
vild(ali).
vild(bo).
galen(cia).
otyglad(X) :-
        vild(X), galen(X).
```

   Lådmodell: