Grundläggande logik och modellteori
Höstterminen 2014

Laboration 2: Zebror

Deadline: 2014-09-26, 12:00

Inlämning: Skicka din lösning till Niklas Zechner (zechner@cs.umu.se). Ärendemeningen ska vara

[5dv102] namn-2

där 'namn' är ditt cs-användarnamn. Din *väl kommenterade* lösning ska finnas i en bifogad fil som heter

namn-2.pl

där 'namn' är ditt användarnamn. Glöm inte att klistra in dina quiz-svar i en kommentar högst upp i filen.

Prolog: Listor och Zebror

1: Zebrorna

I denna laboration tittar vi på listor i Prolog, och använder dem för att lösa ett klassiskt pussel ni säkert alla sett förr:

1. Det står fem hus i en rad.
2. Engelsmannen bor i det röda huset.
3. Spanjoren har hund.
4. I det gröna huset dricker man kaffe.
5. Ukrainaren dricker te.
6. Det gröna huset står omedelbart till höger om det vita.
7. Personen som röker Old Gold har sniglar som husdjur.
8. I det gula huset röker man Kools.
9. I huset i mitten dricker man mjölk.
10. Norrmannen bor i det första huset.
11. Personen som röker Chesterfields bor granne med huset med en räv.
12. Personen som röker Kools bor granne med huset där det hålls en häst.
13. Apelsinjos-drickaren röker Lucky Strikes.

14. Japanen röker Parliaments.

15. Norrmannen bor granne med det blå huset.

16. Någon dricker vatten och någon har en zebra, men var bor de?

- Skriv ett predikat `list5` som är sant för modeller där variabeln är en lista av längd 5. Det inbyggda predikatet `length/2` får användas.

```
?- list5([1,2,3]).
false.

?- list5([a,b,c,d,e]).
true.

?- list5(X).
X = [_G1844, _G1847, _G1850, _G1853, _G1856].
```

- Skriv ett predikat `eq_pos(L1,L2,E1,E2)` som är sant då L1 och L2 är listor sådana att L1 innehåller E1 på samma position som L2 innehåller E2.

```
?- eq_pos([1], [a,b,c], E1, E2).
E1 = 1,
E2 = a ;
false.

?- eq_pos([1,2,3], [a,a,c], 2, a).
true .

?- eq_pos([1,2,3], [a,a,c], 3, a).
false.

?- eq_pos(L, [a,a,c], 3, a).
L = [3|_G1869] ;
L = [_G1868, 3|_G1872] ;
false.
```

- Skriv ett predikat `eq_neigh_pos(L1,L2,E1,E2)` som är sant då L1 och L2 är listor sådana att L1 innehåller E1 på en position just till höger eller vänster om positionen där L2 innehåller E2. Detta predikat skrivs enklast genom att applicera `eq_pos` på argumenten med första elementet borttaget ur antingen L1 eller L2.

```
?- eq_neigh_pos([1,2,3], [a,b,c], 1, b).
true .

?- eq_neigh_pos([1,2,3], [a,b,c], 1, a).
false.

?- eq_neigh_pos([1,2,3], [a,b,c], 2, X).
X = a ;
X = c ;
false.

?- eq_neigh_pos([1,2,3], L2, 2, b).
L2 = [b|_G1869] ;
L2 = [_G1868, _G1871, b|_G1875] ;
false.
```

- Använd de ovanstående predikaten för att skriva ett predikat

  `zebra(Origin,Color,Pet,Drink,Smoke)`

  som är sant för listor Origin, Color, Pet, Drink, Smoke, som är lösningar av Zebra-pusslet.

  Predikatet `zebra` skall alltså vara sant då, t.ex., Origin är instantierad med en lista av längd fem, innehållande symbolerna `english`, `spaniard`, `ukrainian`, `norwegian` och `japanese`, sådana att ordningen i listan är lösningen på i vilket hus vilken person bor.

  Predikatet behöver bara bestå av ovanstående regler (1–15), kontrollera längden på listan och koda t.ex. regel 14 med predikatet

  `eq_pos(Origin, Smoke, japanese, parliaments)`

  om allt går rätt kommer Prolog att på frågan

  `zebra(Origin,Color,Pet,Drink,Smoke)`

  ge ett unikt svar där dess "`_G1868`"-markörer markörer avslöjar var en Zebra och vatten kan passa in.

  Tips: Vi har inget särskilt predikat för t.ex. regel 9, men det kan helt enkelt skrivas som `Drink = [_,_,milk,_,_]` i konjunktionen.

Skriv ner alla predikaten i din svarsfil. Kommentera ditt arbete noga.

2: Lite matematik

Prolog kan förstås arbeta också med siffror, för detta används `is`. Som ett exempel, predikatet nedan kontrollerar om det givna talet är en kvadrat (dvs, existerar det något heltal x sådant att talet är x^2?).

```
% square(X) checks if X is a square by recursively checking if 1 through
% X is a square
square(X) :- recurse_square(X,1).

% recurse_square(X,T) checks if X is the square of T or some number larger
% than T (but less than X)
%
% First case: If T=X it is impossible, just fail.
% If T*T is equal to X then we are done.
recurse_square(X,T) :- X is T * T.
% If not, if T is still smaller than X, add one to T and try again.
recurse_square(X,T) :-
  T < X,
  Tp is T + 1,
  recurse_square(X,Tp).
```

Er uppgift är att skriva ett predikat `prime(X)` som rapporterar om `X` är ett primtal eller inte.

Ni kommer troligen vilja använda infix-predikatet för modulo (`mod/2`), vilket skrivs t.ex. `R is 47 mod 10`, vilket binder R till resten av att dela 47 med 10, alltså i det här fallet 7. Om man hellre jämför likhet än olikhet så kan operatorn =**:**= vara av intresse. Denna kräver till skillnad från = att båda sidor evalueras till ett heltal innan jämförelse sker.

Inlämning

- Skriv lösningarna på båda deluppgifterna i din lösningsfil
- Dubbelkolla att ni lämnar in rätt kod
- Dubbelkolla att den fungerar på SWI-Prolog som installerad på Datavenskap
- Kommentera grundligt
- Kom ihåg att lämna in era quiz-lösningar som en kommentar högst upp i filen

Lycka till!