KS i Programmeringsparadigm 2017, del 2: Logik-programmering
2017-10-13 13.15-15.00 med efterföljande kamraträttning

Inga hjälpmedel är tillåtna. Skriv svaren direkt på blanketten. Bonuspoäng från Prologlabbarna hösten 2017 kommer automatiskt att tillgodoräknas på denna KS. 12 poäng (utifrån max 20) krävs för godkänt. Sitt kvar ända till klockan 14.00. Då lämnar alla in samtidigt. Därefter tar kamraträttning vid.

1. (8 p) **Kontrollflöde, Unifiering, Backtrackning**.

   Betrakta predikatet `conc(?X, ?Y, ?Z)` som är sant om och endast om listan `Z` är konkateneringen av listan `X` med listan `Y`:

   ```
   conc([], X, X).
   conc([H | T], X, [H | TX]) :-
     conc(T, X, TX).
   ```

   Givet denna deklaration, beskriv *kontrollflödet* med (1) alla *regelinstanser*, (2) alla *unifieringar*, och (3) alla *svar* som kommer att skapas (ett i taget) under exekveringen av frågan:

   ```
   ?- conc(X, Y, [a]).
   ```

   OBS: Presentera kontrollflödet i stilen som vi har använt på föreläsningarna. **Svar baserade på lådmodellen eller trace kommer ej godkännas**.

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

2. (12 p) **Generera-och-testa, Listor, Rekursion**.

Vi har en lista över studenter, som ska tilldelas handledare ur en separat lista över handledare. Villkoret är att ingen handledare ska handleda mer än en student. Problemet ska lösas på ett enkelt (fast kanske ineffektivt) sätt, med tekniken *Generera-och-testa*. Lösningens struktur kommer alltså vara:

```
assign(Students, Supervisors, Assignment) :-
   generateAssignment(Students, Supervisors, Assignment), % Generera
   checkAssignment(Assignment).                           % Testa
```

(a) Börja med att implementera predikatet `generateAssignment(+From, +To, ?Map)`, som ska kunna generera (med backtracking) alla möjliga tilldelningar av handledare till studenter, utan inskränkning på hur många studenter blir tilldelade till varje handledare (men varje student skall tilldelas exakt en handledare). T.ex., för student-listan `[anna, bo]` och handledar-listan `[eva, lars, gudrun]` har vi tilldelningarna `[(anna, lars), (bo, gudrun)]`, `[(anna, eva), (bo, eva)]`, och sju andra tilldelningar. Du får använda det inbyggda Prologpredikatet `member`. Du får inte använda några andra inbyggda Prologpredikat. *(4p)*

(b) Implementera predikatet `checkAssignment(+Map)`, som ska vara sant om och endast om tilldelningen `Map` är *injektiv*, dvs om den tilldelar till olika studenter olika handledare. T.ex. är tilldelningen `[(anna, lars), (bo, gudrun)]` injektiv, medan `[(anna, eva), (bo, eva)]` inte är det. Du får använda `member`, men inga andra inbygga Prologpredikat. *(4p)*

(c) Detta enkla sätt att implementera tilldelningen är inte särskilt effektivt eftersom man först skapar en hel tilldelning innan man kollar om den duger (fast detta skulle kunna misslyckas med de första två studenterna om de blir tilldelade samma handledare, oavsett resten av tilldelningen). Ett mera effektivt sätt vore att direkt generera injektiva tilldelningar. Implementera predikatet `assignInjective(+From, +To, ?Map)` som åstadkommer detta genom att använda inbyggda Prologpredikatet `select(?El, ?List1, ?List2)`, som är sant om och endast om `List2` är en lista som `List1`, men med en förekomst av elementet `El` mindre. *(4p)*

.....

.....

.....

.....

.....

.....

.....

.....

.....