KS i Programmeringsparadigm 2016, del 2: Logik-programmering
2016-10-13 08.15-10.00 med efterföljande kamraträttning

Inga hjälpmedel är tillåtna. Skriv svaren direkt på blanketten. Bonuspoäng från Prologlabbarna hösten 2016 kommer automatiskt att tillgodoräknas på denna KS. 12 poäng (utifrån max 20) krävs för godkänt. Sitt kvar ända till klockan 09.00. Då lämnar alla in samtidigt. Därefter tar kamraträttning vid.

1. (6 p) **Problemdomänbeskrivning**.

   En databas ska skapas om sjukdomar och deras symptom. En databas ska innehålla en mängd sjukdomar, och för varje sjukdom, en mängd symptom som kännetecknar sjukdomen.

   (a) Beskriv i text (utan kod!) en lämplig Prolog-representation av sådana databaser. *(1p)*

   .....

   .....

   .....

   (b) Hur skulle följande exempel skrivas som en Prolog-term i din representation? *(1p)*
   Exempel-databas: Symptom `s1`, `s2` och `s3` visas vid sjukdom `i1`, symptom `s2` och `s4` vid `i2`, och symptom `s3` och `s5` vid `i3`.

   .....

   .....

   (c) Konstruera ett Prolog-predikat `diagnose(+Symptoms, +Database, ?Diseases)`, där:
   - `Symptoms` är en lista med symptom.
   - `Database` är en sjukdoms-databas enligt din representation från (a)-uppgiften.
   - `Diseases` är en lista med sjukdomar.

   Predikatet ska vara sant om och endast om `Diseases` är exakt de sjukdomar i databasen som inkluderar alla de givna symptomen (`Symptoms`). I exempel-databasen från (b), för `Symptoms = [s3]`, ska alltså `Diseases` vara `[i1, i3]` eftersom dessa två sjukdomar har alla de givna symptomen.
   Du får definiera egna hjälp-predikat om du behöver dem. Om du använder något av de inbyggda standard-predikaten i Prolog, förklara kortfattat vad predikatet gör. *(4p)*

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

2. (8 p) **Kontrollflöde, Unifiering, Backtrackning**. Betrakta följande Prolog-program:

```
edge(a, b).
edge(b, c).

reach(X, Y) :- edge(X, Z), reach(Z, Y).
reach(X, Y) :- edge(X, Y).
```

Givet dessa deklarationer, beskriv *kontrollflöde*, *unifieringar*, och *slutsvaret* vid frågan:

```
?- reach(a, c).
```

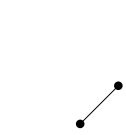
3. (6 p) **Induktiva datatyper, Rekursion**. Binära träd utan data (BTUD) kan definieras med följande BNF-grammatik, där "`nil`" representerar det tomma trädet.

```
<Btud> ::= nil | leaf | branch(<Btud>, <Btud>)
```

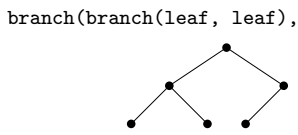
I denna uppgift betraktar vi en delmängd av alla BTUD, nämligen "välfriserade" BTUD, vilket vi definierar som att de uppfyller följande formegenskap (shape property):

> Alla nivåer i trädet, förutom möjligen den sista, är helt fyllda, och om den sista nivån av trädet inte är helt fylld, så är noderna i den nivån fyllda från vänster till höger.

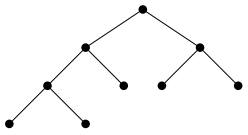
Här är några exempel på välfriserade BTUD:



`branch(leaf, nil)`



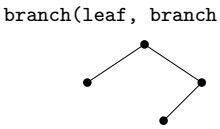
`branch(branch(leaf, leaf), branch(leaf, nil))`



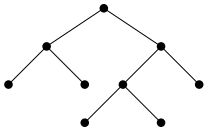
`branch(branch(branch(leaf, leaf), leaf), branch(leaf, leaf))`

Och här är några exempel på BTUD som *inte* är välfriserade:

`branch(nil, leaf)`



`branch(leaf, branch(leaf, nil))`



`branch(branch(leaf, leaf), branch(branch(leaf, leaf), leaf))`

Notera att det för varje antal noder n bara finns ett unikt välfriserat BTUD med n noder!

Vi vill definiera ett predikat `insert(+T1, ?T2)` som är sant om och endast om `T2` är det välfriserade trädet som har en nod mer än det välfriserade trädet `T1`. Du *ska* utgå från följande två hjälp-predikat, som du kan anta redan existerar:

1. `incomplete(+T)` är sant om och endast om trädet `T` *inte* är fullständigt (dvs, den sista nivån i trädet är inte helt fylld).
2. `height(+T, ?H)` är sant om och endast om `H` är höjden på trädet `T`. I den här uppgiften definieras höjden av ett träd som antalet noder på längsta stig från rot till löv. Höjden av `nil` är 0, och höjden av `leaf` är 1.

Du får inte använda eller definiera några andra hjälp-predikat. Beskriv (helst med Prolog-kod) samtliga fall som måste definieras för att konstruera ett väldefinierat predikat `insert(+T1, ?T2)` som använder de givna hjälp-predikaten.

(a) Vad ska basfallen i definitionen av `insert` vara? *(2p)*

.....

.....

(b) Vad ska de induktiva fallen i definitionen av `insert` vara? *(4p)*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....