Tenta i Programmeringsparadigm - DD1361 2019-08-16 08.00-13.00

Inga hjälpmedel är tillåtna. Skriv inga svar på blanketten. För godkänt på en del krävs utförliga svar med kompletta meningar på samtliga (del)uppgifter för den delen. Ett grovt fel ger underkänt. En del blir godkänd även med ett halvgrovt fel. Varje del bedöms oberoende av resultat på de andra delarna. Logikdelen bedöms annorlunda. Här behövs 12 poäng godkänt av maximal 20 poäng. Eventuella bonuspoäng gäller inte längrepå någon del. Lycka till önskar Dilian, Marcus och Per!

Funktionell programmering

1. Funktionen sorted i Python tar in en lista och returnerar en sorterad kopia. Om man vill att en lista med strängar ska sorteras på längd så kan man skicka med en funktion som används som jämförelse. Exempel:

```
>>> v = ["Sophia", "Marcus", "Anna", "Gunnar"]
>>> sorted(v, key=len)
['Anna', 'Sophia', 'Marcus', 'Gunnar']
>>> sorted(v)
['Anna', 'Gunnar', 'Marcus', 'Sophia']
>>> sorted(v, key=lambda x:x[-1]=='a')
['Marcus', 'Gunnar', 'Sophia', 'Anna']
>>>
```

   (a) Vad kallas en funktion som tar en funktion som parameter?

   (b) Vilken notation används för anonyma funktioner i Haskell?

2. Studera följande Haskellkod:

```
f::Integer->Integer
f n
    |n==0 = 1
    |otherwise = n * f (n-1)

g::Integer->Integer
g n = h 1 n

h::Integer->Integer->Integer
h a 0 = a
h a n = h (a*n) (n-1)
```

   Vilken eller vilka av f, g och h i programmet ovan är svansrekursiv? Motivera ditt svar noggrant med kursens terminologi.

3. En av Haskells inbyggda funktioner heter `filter`. Du behöver inte skriva dess parametrar i rätt ordning i någon av uppgifterna, men du behöver veta vilka de är.

   (a) Vad är typsignaturen för filter?

   (b) Beskriv i detalj vad filter gör.

En uppgift till på nästa sida.

4. En föreläsning handlade om olika sorters polymorfism. För var och en av dessa olika sorters polymorfism, namnge vilken sort det är på formen "X polymorfism". Förklara i detalj varför exemplet beskriver just denna sorts polymorfism.

   (a) Ett verksamhetssystem är programmerat i ett objektorienterat språk. En superklass heter `Anställd` som innehåller metoden `agera`. Subklasserna till `Anställd` heter `Programmerare`, `Chef` och `Projektledare`. Varje subklass till `Anställd` har en egen metod `agera` som fungerar annorlunda än den i superklassen `Anställd`. Programmet kan innehålla en lista över flera objekt som är instanser av subklasser till `Anställd` och anropa metoden `agera` på dessa.

   (b) Haskells inbyggda funktion map kan ta in såväl strängar som listor av heltal som en av sina parametrar.

   (c) Additionsoperatorn kan överlagras i vissa språk så att `1+1 == 2` och

   ```
   "1"+"1" == "11"
   ```

   fungerar.

   (d) I vissa språk kan det förekomma två funktioner med samma namn men olika parameteruppsättning. Vid funktionsanrop väljer kompilatorn vilken av dem som ska anropas.

Formella språk och syntaxanalys

5. Vad är definitionen av ett *formellt språk* (inom datavetenskap/matematik)?

6. Följande meningar använder terminologin från kursavsnittet, men på ett konstigt och/eller felaktigt sätt som indikerar att personen som yttrat meningarna inte riktigt vet vad hen pratar om. För var och en av meningarna, förklara vad som är konstigt.

   För de av er som är mer bekväma med kursens terminologi på engelska ges även engelska översättningar, du kan själv välja om du vill utgå från de engelska eller svenska versionerna i dina svar.

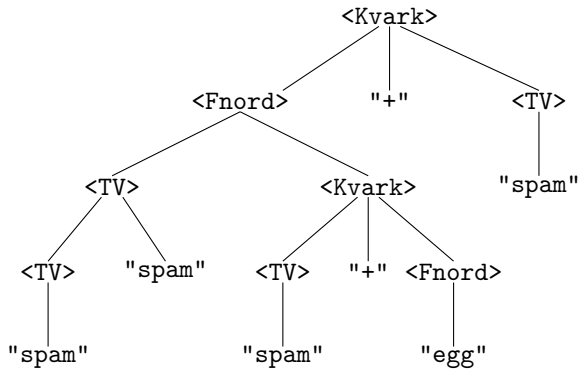
   (a) "Den lexikala analysen visade att strängen 'aabaab' var syntakiskt korrekt."
   (Engelska: "The lexical analysis showed that the string 'aabaab' was syntactically correct.")

   (b) "Eftersom språket är reguljärt kan vi köra det genom den ändliga automaten."
   (Engelska: "Since the language is regular we can run it through the finite automaton.")

   (c) "Eftersom automaten har en loop så följer det från pumping-lemmat att språket är oändligt stort."
   (Engelska: "Since the automaton has a loop it follows from the pumping lemma that the language is infinite.")

En uppgift till på nästa sida.

7. Betrakta följande syntaxträd som erhållits genom att parsa någon sträng x enligt någon grammatik.



   (a) Vad är strängen x som parsats?

   (b) Hur ser grammatiken ut som använts vid parsningen av x? (Du behöver bara beskriva de delar av grammatiken som går att utläsa från syntaxträdet.)

   (c) Är språket som grammatiken beskriver oändligt? Det finns tre möjliga svar på denna fråga: "ja", "nej", och "det finns inte tillräckligt mycket information för att avgöra detta".

Logikprogrammering

8. **Kontrollflöde** *(6p)*

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

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

   Givet denna deklaration, beskriv *kontrollflödet* vid exekveringen av följande fråga:

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

   Du behöver bara beskriva kontrollflödet fram tills och med dess att vi får första svaret. Din beskrivning *ska* innehålla: (1) alla *regelinstanser*, (2) alla *unifieringar*, och (3) själva svaret.

   Presentera kontrollflödet i stilen som vi har använt på föreläsningarna. Unifieringar utan regelinstanser är tvetydiga och ger poängavdrag.

9. **Logisk versus procedurell läsning** *(4p)*

   (a) Vad menas med en logisk läsning av ett Prolog-program och vad menas med en procedurell läsning? *(2p)*

   (b) Ge ett exempel på två program som har samma (dvs ekvivalent) logiska läsning, men olik procedurell läsning, så att de två programmen ger olika resultat när de exekveras. Förklara ditt tänkesätt. *(2p)*

En uppgift till på nästa sida.

10. **Induktiva datatyper, Rekursion** *(10p)*

När ett program i ett språk som C, Java, eller Python exekverar, kallas den tillfälliga tilldelningen för ett *tillstånd* (dvs vilka variabler som har vilka värden i ett visst steg under exekveringen). Om ett program använder sig av till exempel heltalsvariablerna `x` och `y` (i denna uppgift begränsar vi oss till att bara betrakta heltalsvariabler), så kan vi i Prolog representera tillståndet där `x` har värdet 5 och `y` värdet -7 med listan:

```
[val(x, 5), val(y, -7)].
```

Tillstånd kan *uppdateras* genom att ange ett nytt värde till en given variabel. Det resulterande tillståndet behåller den ursprungliga tilldelningen för alla andra variabler.

(a) Definiera ett predikat `update(+OldState, +Variable, +Integer, -NewState)` som beräknar det uppdaterade tillståndet `NewState` när den givna variabeln sätts till det givna värdet i det givna tillståndet `OldState`. Till exempel ska:

```
?- update([val(x, 5), val(y, -7)], x, -3, X).
```

resultera i:

```
X = [val(x, -3), val(y, -7)].
```

(4p)

Predikatet behöver bara hantera fallet när variabeln som ska uppdateras redan har ett existerande värde i `OldState`.

(b) Aritmetiska *heltalsuttryck* med addition, subtraktion och multiplikation som operationer kan representeras som Prolog-termer enligt följande BNF:

```
A ::= num(I) | var(V) | add(A, A) | sub(A, A) | mul(A, A)
```

där `I` sträcker sig över heltal (t.ex. 3 eller -7), och `V` över heltalsvariabler (t.ex. `x` eller `y`). Till exempel kan aritmetiska uttrycket $(4 + y) \cdot (-3)$ representeras med Prolog-termen `mul(add(num(4), var(y)), num(-3))`.

För att kunna *utvärdera* sådana uttryck, som kan involvera variabler, behöver vi veta i vilket tillstånd utvärderingen sker; då är det tillståndet som bestämmer värdet på variablerna i uttrycket.

Definiera ett predikat `eval(+Exp, +State, -Integer)` som beräknar aritmetiska uttryck gentemot ett givet tillstånd, med hjälp av de inbyggda aritmetiska operatörerna i Prolog. Till exempel ska:

```
eval(mul(add(num(4), var(y)), num(-3)), [val(x, 5), val(y, -7)], X).
```

resultera i:

```
X = 9.
```

Korrekta basfall ger 2 poäng. Korrekta induktiva fall ger 4 poäng. *(6p)*