Göteborgs Universitet och Chalmers Tekniska Högskola 2000–01–01
Datavetenskap INN11

Tentamen i Beräkningsmodeller

Måndagen den 16 december 2000, kl 8.45 – 13.45 i V-huset
Ansvarig lärare: Bengt Nordström, tel 1033 eller 0707 - 87 77 29
Tillåtna hjälpmedel: Inga.
Börja varje uppgift på nytt blad. Skriv endast på en sida av papperet. Varje svar skall motiveras! Den här skriftliga tentamen utgör en del (75 %) av den totala examinationen, den andra delen (dvs. 25 %) består av de inlämingsuppgifter som har delats ut under kursens gång. För årets och förra årets elever gäller alltså att summan av poängen från inlämningsuppgifterna och den skriftliga tentan skall vara minst 100 för att få godkänt på kursen. Examensvisning kommer att äga rum måndagen den 22 januari kl 15.30 i MD8. Datum för omtentamen kommer att annonseras på kursens hemsida. Lösningar till den här tentan kommer också att finnas tillgänglig från kursens hemsida.

1. Ge ett exempel på ett λ-uttryck som har variabeln x fri, men vars värde (normalform) ej beror på x. (15)

   Svar: I uttrycket $(\lambda z.\lambda y.y)x$ finns variabeln x fri. Uttryckets normalform är $\lambda y.y$, som ej beror på x.

2. Skriv ett program i χ som ej innehåller **rec**-konstruktionen och som ej terminerar. (15)

   Svar: Eftersom χ är ett otypat språk kan vi använda följande term (som ej terminerar) från λ-kalkyl:

   $$(\lambda x.xx)(\lambda x.xx)$$

3. Kan man räkna upp $\wp(\mathbf{N})$, mängden av alla delmängder av $\mathbf{N}$? Motivera! (20)

   Svar: Det kan man inte. Vi kör vårt gamla diagonaliseringsargument igen: Om man kunde det fanns det en total surjektiv funktion $\mathsf{F} \in \mathbf{N} \to \wp(\mathbf{N})$ som räknar upp alla delmängder. Vi kan nu definiera en mängd D som inte finns med i uppräkningen genom att låta talet k finnas i mängden D om och endast om $\mathsf{k} \notin \mathsf{F}(\mathsf{k})$. Om D skulle finnas med i uppräkningen skulle $\mathsf{D} = \mathsf{F}(\mathsf{i})$, för något i. Vi får nu en motsägelse eftersom vi har att $\mathsf{i} \in \mathsf{D}$ om och endast om $\mathsf{i} \notin \mathsf{F}(\mathsf{i})$.

4. Bevisa att man i Haskell (eller något annat funktionellt språk) inte kan skriva en funktion `halt :: (Nat -> Nat) -> Nat -> Bool` som är sådan att `(halt f i)` evaluerar till `true` om `(f i)` terminerar och annars evaluerar till `false` . (30)

Svar: Detta är bara en enkel variant av det extensionella stoppproblemet. Om vi har en funktion `halt` enl ovan skulle vi kunna definiera en funktion `termcnv` genom

```
termcnv f i = if (halt f i) then loop else 1
```

som har egenskapen att `termcnv f i` terminerar om och endast om `f i` ej terminerar. Detta gäller för alla funktioner `f`. Speciellt för den funktion `strange` som är definierad av

```
strange i = termcnv strange i
```

Definitionen av `strange` visar att `termcnv strange i` terminerar samtidigt som `strange i`, vilket motsäger egenskapen ovan.

5. I den här uppgiften skall ni formalisera ett litet språk för aritmetik. Språkets konkreta syntax kan beskrivas på följande sätt: (40)

$$e ::= x \mid n \mid e + e \mid e * e \mid e + e \mid \sum_{x=e}^{e} e$$

vilket betyder att ett uttryck antingen är en variabel (t.ex. i), ett tal (t.ex.314), en addition (t.ex. $45 + i$), en multiplikation (t.ex. $5 * 3$) eller en summation (t.ex.$\sum_{x=1}^{1000} 1 + x * i$). Vi antar att vi i en summation $\sum_{x=d}^{e} f$ inte har några förekomster av variabeln x i uttrycken d och e.

(a) Ge den abstrakta syntaxen för språket! Ni kan utgå från att vi redan har definierat Z, mängden av heltal.

(b) Vad betyder det att variabeln x är fri i uttrycket e?

(c) Definiera substitutionsoperationen $e_1[x \leftarrow e_2]$, det uttryck man får genom att substituera uttrycket e_2 för alla fria förekomster av variabeln x i uttrycket e_1. Vi kan anta att uttrycket e_2 är slutet.

(d) Varför blir det mer komplicerat att definiera substitutionsoperationen om uttrycket e_2 inte är slutet?

Svar:

(a) Vi antar att vi har en given oändlig mängd **Id**. Den abstrakta syntaxen definieras genom följande induktiva definition av mängden Exp:

$$\begin{aligned}
\mathbf{var}(i) &\in \mathsf{Exp} \quad \text{om} \quad i \in \mathbf{Id} \\
\mathbf{int}(j) &\in \mathsf{Exp} \quad \text{om} \quad j \in Z \\
\mathbf{plus}(d, e) &\in \mathsf{Exp} \quad \text{om} \quad d, e \in \mathsf{Exp} \\
\mathbf{mult}(d, e) &\in \mathsf{Exp} \quad \text{om} \quad d, e \in \mathsf{Exp} \\
\mathbf{sum}(i, d, e, f) &\in \mathsf{Exp} \quad \text{om} \quad i \in \mathbf{Id}, d, e, f \in \mathsf{Exp}
\end{aligned}$$

(b) Summationsuttrycket $\sum_{x=e_1}^{e_2} e_3$ binder variabeln x i uttrycket e_3. Vi kan ju byta ut variabeln x mot någon annan variabel om vi bara gör samma byte i uttrycket e_3. Därför får vi följande definition av att en variabel är fri i ett uttryck:

Variabeln x är fri i uttrycket x.

Variabeln x är fri i uttrycket $d + e$ om x är fri i d eller i e.

Variabeln x är fri i uttrycket $d * e$ om x är fri i d eller i e.

Variabeln x är fri i uttrycket $\sum_{y=d}^{e} f$ om x inte är lika med y och x är fri i d eller i e.

(c) Jag ger en definition av substitutionsoperationen med hjälp av följande induktion över den abstrakta syntaxen:

$$
\begin{aligned}
\mathbf{var}(j)[i \leftarrow e] &= \mathbf{var}(j) \quad \text{om } i \neq j \\
\mathbf{var}(i)[i \leftarrow e] &= e \\
\mathbf{int}(j)[i \leftarrow e] &= \mathbf{int}(j) \\
\mathbf{plus}(d, f)[i \leftarrow e] &= \mathbf{plus}(d[i \leftarrow e], f[i \leftarrow e]) \\
\mathbf{mult}(d, e)[i \leftarrow e] &= \mathbf{mult}(d[i \leftarrow e], f[i \leftarrow e]) \\
\mathbf{sum}(j, d, f, g)[i \leftarrow e] &= \mathbf{sum}(j, d[i \leftarrow e], f[i \leftarrow e], g[i \leftarrow e]) \\
&\qquad \text{om } i \neq j \\
\mathbf{sum}(i, d, f, g)[i \leftarrow e] &= \mathbf{sum}(i, d, f, g)
\end{aligned}
$$

(d) Antagandet att det uttryck som man substituerar skall vara slutet leder till en enklare definition av substitutionsoperationen av samma skäl som i lambda-kalkyl. Dvs man kan lätt hamna i en situation där de fria variablerna skulle bindas vid substitutionen. Man blir i så fall tvungen att först byta namn på variablerna.

6. Beskriv språket PRF, mängden av de primitivt rekursiva funktionerna. Ge den abstrakta syntaxen, ge en informell beskrivning av semantiken för de olika konstruktionerna i språket. Ge ett exempel (bevis ej nödvändigt) på ett program som ej kan uttryckas i språket. (30)

   Svar: Se föreläsningsanteckningarna eller läroboken (sid 23). Ett exempel på ett program som inte kan uttryckas i språket är ett program som inte terminerar.

Lycka till!