Göteborgs Universitet och Chalmers Tekniska Högskola 19 januari 2005
Datavetenskap INN111/TIN120

Tentamen i Beräkningsmodeller

Lördagen den 1 februari 2003, kl 8.45 – 12.45.
Ansvarig lärare: Bengt Nordström, tel 0708 - 96 69 14.
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 inlämningsuppgifter 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 fredagen den 7 februari kl 11.00 i Bengt Nordströms tjänsterum. Lösningar till tentan kommer att finnas tillgängliga från kursens hemsida.

1. Vad säger Church-Rossers sats? (10)

2. I λ-kalkyl byggs uttryck upp med hjälp av variabler, abstraktion och applikation. Trots att detta verkar primitivt i överkant kan ju enligt Churchs tes alla beräkningsbara funktioner uttryckas i λ-kalkyl.

   (a) Hur uttrycks talet 3 i λ-kalkyl? (10)

   (b) Ge ett exempel på ett program i λ-kalkyl som inte terminerar. (10)

   (c) Fungerar det exemplet i Haskell (som också har variabler, abstraktioner och applikationer)? (2)

3. Ange om följande påståenden är sanna eller falska samt ge ett bevis för detta!

   (a) Mängden av alla uttryck i λ-kalkyl som har en normalform är uppräknelig. (15)

   (b) Alla positiva rationella tal $\mathbf{Q}_+$ (mängden av tal som kan skrivas som $\frac{\mathfrak{i}}{\mathfrak{j}}$ där $\mathfrak{i}$ och $\mathfrak{j}$ är naturliga tal, $\mathfrak{j} \neq 0$) är uppräkningsbar. (15)

4. En partiell funktion i $\mathbf{N} \tilde{\to} \mathbf{N}$ är Turing-beräkningsbar om det finns en Turing-maskin som beräknar den. Ge en förklaring av detta! Dvs förklara vad det betyder att en Turing maskin beräknar en funktion. (10)

5. Ge ett exempel på ett program i språket χ som terminerar till svag huvud normal form, men vars fullständiga evaluering ej terminerar. Motivera! (10)

6. Det finns fem programkonstruktioner i språket **PRF**, de första fyra har en syntax som kan beskrivas informellt på följande sätt: (40)

$$\begin{aligned}
\mathbf{z} &\in \mathbf{PRF}_0 \\
\mathbf{s} &\in \mathbf{PRF}_1 \\
\mathbf{proj}(n, i) &\in \mathbf{PRF}_{n+1} \text{ if } i \leq n \\
\mathbf{comp}(g, f_1, \ldots, f_m) &\in \mathbf{PRF}_n \text{ if } g \in \mathbf{PRF}_m, f_i \in \mathbf{PRF}_n, 1 \leq i \leq m
\end{aligned}$$

och semantiken beskrivs informellt som:

$$\begin{aligned}
\mathbf{z}() &= 0 \\
\mathbf{s}(j) &= j + 1 \\
\mathbf{proj}(n, i)(j_0, \ldots, j_n) &= j_i \\
\mathbf{comp}(g, f_1, \ldots, f_m)(j_1, \ldots, j_n) &= g(f_1(j_1, \ldots, j_n), \ldots, f_m(j_1, \ldots, j_n))
\end{aligned}$$

Ge en informell beskrivning av den konstruktion som saknas!

Visa också hur man kan uttrycka fakultetsfunktionen som definieras av att $f(0) = 1$ och $f(n) = 1 * 2 * \cdots n$ för alla naturliga tal $n > 0$. För den sista uppgiften är det viktigt att du motiverar svaret, dvs antingen visa att programmet verkligen uppfyller de ekvationer som skall gälla för fakultetsfunktionen eller också visa att ditt sätt att komma fram till programmet är sådant att programmet är korrekt. Det räcker alltså inte att bara ge programmet. Du kan anta att multiplikationsfunktionen redan är definierad.

7. 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` . (20)

Lycka till!