Göteborgs Universitet och Chalmers Tekniska Högskola — 20 december 2002
Datavetenskap — INN111/TIN120

Tentamen i Beräkningsmodeller

Fredagen den 20 december 2002, kl 8.45 – 12.45.
Ansvarig lärare: Bengt Nordström, tel 1033 eller 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 10 januari kl 11.00 i MD 9. Lösningar till tentan kommer att finnas tillgängligt från kursens hemsida.

1. Finns det en injektiv och surjektiv funktion i $N \tilde{\to} N$ som inte är total? (5)

   Svar: Funktionen som definieras av $f(n+1) = n$ är injektiv ty om $n-1 = m-1$ så måste $n = m$. Den är även surjektiv eftersom varje tal n är bilden av talet $n+1$ och den är inte total eftersom den inte är definierad för 0. Andra exempel är $f(2*n) = n$, $f(2^n) = n$ etc. etc.

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

   (a) Alla partiella funktioner från N till $\{1\}$ är beräkningsbara. (10)

   Svar: Falskt. Mängden av alla program är uppräkningsbar (det är en delmängd av mängden av den uppräkningsbara mängden av alla text-strängar). Men funktionerna är inte uppräkningsbara, om de vore det skulle det finnas en uppräkning f_i av dem. Men denna uppräkning kan inte innehålla funktionen d som är definierad av

$$d(n) = \begin{cases} 1 & \text{om } f_n(n) \text{ är odefinierad,} \\ \text{odefinierad} & \text{om } f_n(n) = 1 \end{cases}$$

eftersom den skiljer sig från alla funktioner i uppräkningen. Om den skulle vara lika med funktionen f_j skulle $d(i) = f_j(i)$ för alla i. Men detta gäller inte för $i = j$.

(b) Alla totala funktioner från N till $\{1\}$ är beräkningsbara. (10)

Svar: Sant. Det finns bara en total funktion från N till $\{1\}$ nämligen funktionen f definierad av $f(i) = 1$ för alla i. Den funktionen är trivialt beräkningsbar.

(c) Om variabeln x är fri i lambda-uttrycket e och om e's normalform är d så är x fri i d. (10)

Svar: Falskt. Låt $e = (\lambda y.\lambda z\ y)\ u\ x$ som har variabeln x fri och normalform u.

3. Visa ett sätt att representera ett naturligt tal som en positionerad remsa! (5)

Svar: Om vi har alfabetet $\{1\}$ så kan vi representera talet 0 som $\ldots(\square)\ldots\square$ och 4 som $\ldots(\square)1111\square\ldots$ där parenteserna visar aktuell symbol.

4. Konstruera en Turing-maskin som beräknar funktionen $a \in N \to N$ som definieras av $a(n) = n + 3$. (15)

Svar: De två instruktionerna $(S_0, \square, 1, S_0), (S_0, 1, \leftarrow, S_1)$ adderar 1 till en positionerad remsa. Vi kan nu upprepa detta tre gånger:

$$(S_0, \square, 1, S_0), (S_0, 1, \leftarrow, S_1)$$
$$(S_1, \square, 1, S_1), (S_1, 1, \leftarrow, S_2)$$
$$(S_2, \square, 1, S_2), (S_2, 1, \leftarrow, S_3)$$

5. Ge standard-representationen för programmen $\lambda x \to x$ och $\mathsf{True}\langle\rangle$ i χ. (15)

Svar: Programmet $\lambda x \to x$ står för det matematiska objektet lambda(“x”, var(“x”)) där vi har använt text-strängar för att representera mängden av identifierare. Detta matematiska objekt har en standard-representation i χ, nämligen $\mathsf{Lambda}\langle\text{“}x\text{”}, \mathsf{Var}\langle\text{“}x\text{”}\rangle\rangle$. På analogt sätt: $\mathsf{True}\langle\rangle$ i χ står för det matematiska uttrycket constr(“True”, struct(ϕ)), där ϕ är den tomma tabellen. Detta objekt har en standard-representation i χ, nämligen $\mathsf{Constr}\langle\text{“True”}, \mathsf{Struct}\langle\mathsf{Empty}\langle\rangle\rangle\rangle$,

6. Det finns fem programkonstruktioner i språket PRF, de första fyra har en syntax som kan beskrivas informellt på följande sätt: (30)

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

och semantiken beskrivs informellt som:

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

Ge en informell beskrivning av den konstruktion som saknas! Visa också hur man kan uttrycka funktionen som räknar ut predecessorn till ett naturligt tal i PRF ! (Predecessorn är den funktion som för argumentet 0 ger svaret 0 och för argumentet $n + 1$ returnerar n). För den sista uppgiften är det viktigt att du motiverar svaret, dvs antingen visa att programmet verkligen uppfyller de ekvationer som skall gälla för predecessor-funktionen 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.

Svar: Den konstruktion som saknas är operatorn för primitiv rekursion med syntax:

$$\mathrm{rec}(g, h) \in \mathrm{PRF}_{n+1} \text{ if } g \in \mathrm{PRF}_n, h \in \mathrm{PRF}_{n+2}$$

vars semantik beskrivs informellt som:

$$
\begin{aligned}
\mathrm{rec}(g, h)(0, j_1, \ldots, j_n) &= g(j_1, \ldots, j_n) \\
\mathrm{rec}(g, h)(y + 1, j_1, \ldots, j_n) &= h(y, \mathrm{rec}(g, h)(y, j_1, \ldots, j_n), j_1, \ldots, j_n)
\end{aligned}
$$

Vi vill alltså konstruera ett ett program p för vilket $\mathrm{p}(0) = 0$ och $\mathrm{p}(n + 1) = n$ gäller. Det är naturligt att ansätta

$$\mathrm{p} =_{\mathsf{def}} \mathrm{rec}(g, h)$$

där g och h är okända program. Vi vet att p skall ha aritet 1, vilket medför att g måste ha aritet 0 och h aritet 2. Vi vet också att g måste vara z, detta följer av rec's första semantiska klausul. Av dess andra klausul följer att $p(n + 1) = h(n, p(n))$, vilket blir uppfyllt om vi väljer $h =_{\mathsf{def}} \mathrm{proj}_0^1$. Vi kan alltså definiera

$$p =_{\mathsf{def}} \mathrm{rec}(z, \mathrm{proj}_0^1)$$

7. Skriv ett program isnat$_1$ i språket χ med följande egenskap: (42)

$$\mathrm{isnat}_1\ n = \mathsf{True}\langle\rangle \quad \text{om } n \text{ är ett naturligt tal}$$

Man skulle ju vilja kunna skriva ett program isnat$_2$ med egenskapen:

$$\mathrm{isnat}_2\, n = \begin{cases} \mathsf{True}\langle\rangle & \text{om } n \text{ är ett naturligt tal,} \\ \mathsf{False}\langle\rangle & \text{om } n \text{ evalueras till en konstruerar-applikation som inte är ett tal} \end{cases}$$

men det verkar omöjligt. Föreslå en utvidgning av språket så att man lätt kan uttrycka sådana program. Föreslå ny konkret syntax, ny abstrakt syntax och ge (informell och formell) semantik för utvidgningen!

Språkets operationella semantik ges av följande definition:

$$\frac{e_1 \longrightarrow \mathrm{lambda}(i, e_3) \qquad e_3[i \leftarrow e_2] \longrightarrow d}{\mathrm{apply}(e_1, e_2) \longrightarrow d}$$

$$\frac{e \longrightarrow \mathrm{constr}(i, e') \qquad \mathrm{lookup}(t, i, e'') \qquad \mathrm{apply}(e'', e') \longrightarrow d}{\mathrm{case}(e, t) \longrightarrow d}$$

$$\frac{e \longrightarrow \mathrm{struct}(t) \qquad \mathrm{lookup}(t, i, e') \qquad e' \longrightarrow d}{\mathrm{proj}(e, i) \longrightarrow d}$$

$$\frac{e[i \leftarrow \mathrm{rec}(i, e)] \longrightarrow d}{\mathrm{rec}(i, e) \longrightarrow d}$$

$$\mathrm{string}(i) \longrightarrow \mathrm{string}(i)$$

$$\mathrm{lambda}(i, e) \longrightarrow \mathrm{lambda}(i, e)$$

$$\mathrm{constr}(i, e) \longrightarrow \mathrm{constr}(i, e)$$

$$\mathrm{struct}(t) \longrightarrow \mathrm{struct}(t)$$

Svar: Denna uppgift är en av övningarna på hemsidan. Där finns också ett förslag till lösning.

Lycka till!