

Tentamen i kurserna Beräkningsmodeller (TDA181/INN110) och Grundläggande Datalogi (TDA180)

Onsdagen den 17 januari 2007, kl 14.00 – 17.00 i V-huset.

Ansvarig lärare: Bengt Nordström, tel 0730-79 42 89.

Tillåtna hjälpmedel: Inga.

Börja varje uppgift på nytt blad. Skriv endast på en sida av papperet. Varje svar skall motiveras! Svar utan motivering ger inga poäng. Komplicerade lösningar och motiveringar kan ge poängavdrag.

Frågor om tentan kommer endast att besvaras via telefon (0730-79 42 89).

Poäng från hemuppgifter inlämnade under 2006 kan tillgodoräknas.

Kursen är värd 4 p vid Chalmers och 5 p vid universitetet. Detta förklarar följande betygsgränser: CTH: 3=80p, 4=100p, 5=120p, GU: G=100p, VG=150p. Det finns ett fåtal elever på Chalmers som läste motsvarande 3p-kurs. För dem är poänggränserna 3=60p, 4=75p, 5=90p.

Examensvisning kommer att äga rum fredagen den 19 januari kl 14.45 – i Bengt Nordströms tjänsterum. Lösningar till tentan kommer att finnas tillgängliga från kursen Beräkningsmodellens hemsida.

1. Bevisa eller motbevisa följande påståenden:

- (a) Funktionen som är odefinierad för varje argument är beräkningsbar. (5)

Svar: Funktionen är beräkningsbar eftersom den kan beräknas av ett icke terminerande program, t.ex. $(\lambda x. (x x)(x x))(\lambda x. (x x)(x x))$ i lambda-kalkyl.

- (b) Om M är normalformen av N (i lambda-kalkyl) och N är öppet, så är M öppet. (5)

Svar: Detta gäller inte, låt N vara det öppna uttrycket $(\lambda x. (\lambda y. y))z$, som har variabeln z fri. Då blir M uttrycket $\lambda y. y$ som är slutet. (Vi låter alltså N vara en funktion som inte beror på sitt argument, och applicerar det på ett öppet uttryck.)

- (c) Alla slutna uttryck i lambda-kalkyl har en normalform och denna normalform är unik. (5)

Svar: Nej, det finns ju uttryck i lambda-kalkyl som inte terminerar, t.ex. $T T$, där T är uttrycket $\lambda x. (x x)$

- (d) Mängden av totala funktioner från \mathbf{N} till \mathbf{Bool} är uppräknelig. (15)

Svar: Falskt. Mängden är inte uppräkningsbar, om de vore det skulle det finnas en uppräkningsbar f_i av dem. Men uppräkningsbarheten kan inte innehålla funktionen d som är definierad av

$$d(n) = \neg f_n(n)$$

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

- (e) Mängden av totala funktioner från \mathbf{Bool} till \mathbf{N} är uppräknelig. (15)

Svar: Ja. Intuitionen är att mängden har samma kardinalitet som $\mathbf{N} \times \mathbf{N}$, mängden av par av naturliga tal. Ett element f i funktionsmängden är ju fullständigt beskrivet av två tal, funktionens värde för de två boolska elementen. För att bevisa att mängden är uppräkningsbar räcker det att ge en total och injektiv funktion $g \in (\mathbf{Bool} \rightarrow \mathbf{N}) \rightarrow \mathbf{N}$. En sådan är:

$$g(f) = 2^{f(\mathbf{true})+1} * 3^{f(\mathbf{false})+1}$$

Den är ju total (definitionen gäller för alla f) och injektiv. Om $g(f) = g(f')$, så gäller

$$2^{f(\mathbf{true})+1} * 3^{f(\mathbf{false})+1} = 2^{f'(\mathbf{true})+1} * 3^{f'(\mathbf{false})+1}$$

Men eftersom primtalsuppdelningen är unik måste $f(\mathbf{true}) = f'(\mathbf{true})$ och $f(\mathbf{false}) = f'(\mathbf{false})$, dvs $f = f'$.

2. Enligt läroboken är en icke-tom mängd A uppräkningsbar om det finns en total surjektiv funktion $f \in \mathbf{N} \rightarrow A$.

(a) Är det väsentligt att funktionen skall vara total? (15)

(b) Är det väsentligt att funktionen skall vara surjektiv? (15)

Om svaret är ja, skall du motivera det genom att visa att de reella talen skulle vara uppräkneliga om kravet inte finns med i definitionen. Om svaret är nej, skall du visa hur man givet en funktion som inte uppfyller kravet kan konstruera en funktion med kravet uppfyllt.

Svar: Det är inte väsentligt att funktionen är total, om vi har en icke-total surjektiv funktion $f \in \mathbf{N} \rightarrow A$ så kan vi ju alltid konstruera en total funktion g genom:

$$g(x) = \begin{cases} f(x) & \text{om } f(x) \text{ är definierad,} \\ \mathbf{a} & \text{för övrigt} \end{cases} \quad (1)$$

där \mathbf{a} är ett godtyckligt element i A .

Däremot är det viktigt att funktionen är surjektiv. Annars skulle ju identitetsfunktionen räkna upp de reella talen.

3. (a) Vad är en fixpunktskombinator? (5)
- (b) Ge ett exempel på en fixpunktskombinator och visa att det är en sådan! (5)
- (c) Varför är fixpunktskombinatorer viktiga? (5)

Svar: Se läroboken!

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

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

och semantiken beskrivs informellt som:

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

Ge en informell beskrivning av den konstruktion som saknas!

Visa också hur man kan uttrycka fakultetsfunktionen som definieras av att $f(0) = 1$ och $f(n) = 1 * 2 * \dots * n$ 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.

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

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

vars semantik beskrivs informellt som:

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

Vi ska nu skriva ett program f för fakultetsfunktionen. Om vi försöker uttrycka f på en primitivt rekursiv form ser vi att

$$\begin{aligned} f(0) &= 1 \\ f(n + 1) &= (n + 1) * f(n) \end{aligned}$$

Om vi ansätter

$$f =_{\text{def}} \mathbf{rec}(g, h)$$

där $g \in \mathbf{PRF}_0$ och $h \in \mathbf{PRF}_2$, så vet vi att $g[0] = 1$ måste gälla. Det är uppfyllt om

$$g =_{\text{def}} \mathbf{comp}(s, [z]).$$

Vi vet att följande skall gälla för funktionen h :

$$\begin{aligned} f[n + 1] &= \mathbf{rec}(g, h)[n + 1] \\ &= h[n, f(n)] \\ &= (n + 1) * f(n) \end{aligned}$$

Vi vill alltså konstruera ett program h i \mathbf{PRF}_2 så att $h[n, f(n)] = (n + 1) * f(n)$. Detta är uppfyllt om funktionen h uppfyller $h[n, m] = \mathbf{mul}[n + 1, m]$, där \mathbf{mul} är det program som utför multiplikation.

Om vi nu försöker ansätta att h måste ha formen

$$h = \mathbf{comp}(\mathbf{mul}, [e_1, e_2])$$

för några program e_1 och e_2 , så vet vi att följande måste gälla:

$$\begin{aligned} \mathbf{comp}(\mathbf{mul}, [e_1, e_2])[n, m] &= \mathbf{mul}(e_1[n, m], e_2[n, m]) \\ &= \mathbf{mul}(n + 1, m). \end{aligned}$$

Detta är uppfyllt om

$$\begin{aligned} e_1[n, m] &= n + 1 \\ e_2[n, m] &= m. \end{aligned}$$

Detta är uppfyllt om e_2 är en projektion, nämligen

$$e_2 =_{\text{def}} \mathbf{proj}(, 1)^1$$

och om e_1 är en komposition:

$$e_1 =_{\text{def}} \mathbf{comp}(s, \mathbf{proj}(, 1)^0)$$

ty $\mathbf{comp}(s, \mathbf{proj}(, 1)^0)[n, m] = s(\mathbf{proj}(, 1)^0[n, m]) = n + 1$

För att sammanfatta så kan vi alltså definiera fakultetsfunktionen som

$$\begin{aligned} f =_{\text{def}} \mathbf{rec}(\mathbf{comp}(s, z), \\ \mathbf{comp}(\mathbf{mul}, [\mathbf{comp}(s, \mathbf{proj}(, 1)^0), \\ \mathbf{proj}(, 1)^1])) \end{aligned}$$

5. Lös en av följande uppgifter (beroende på om du studerat χ eller PCF):

(a) Följande uppgift är för de som har studerat språket χ :

i. Skriv ett program \mathbf{prod} i χ (utan syntaktiskt socker) som är definierat så att

(8)

$$\mathbf{prod} \ n = (0 + 1) * (1 + 2) * \dots * (n - 1 + n)$$

Du kan anta att vi har definierat funktionerna \mathbf{add} och \mathbf{mult} som utför addition respektive multiplikation. Förklara hur du representerar de naturliga talen (om du inte vill behöver du inte använda standard-representationen).

ii. Bevisa (med induktion) att ovanstående gäller! (12)

Svar: Vi använder standard-representationen av tal, dvs talet 0 representeras av $\text{zero}\langle \rangle$ och talet $n + 1$ av $\text{succ}\langle n' \rangle$, där n' är representationen av n . Vi ser att programmet skall uppfylla följande ekvationer:

$$\begin{aligned}\text{prod}(\text{zero}\langle \rangle) &= \text{one} \\ \text{prod}(\text{succ}\langle n \rangle) &= (\text{mult}(\text{prod } n))(\text{add } n (\text{succ } n))\end{aligned}$$

Låt oss införa förkortingen

$$f \ m \ n = (\text{mult } m)(\text{add } n (\text{succ } n))$$

Vi skall alltså lösa ekvationerna

$$\begin{aligned}\text{prod}(\text{zero}\langle \rangle) &= \text{one} \\ \text{prod}(\text{succ}\langle n \rangle) &= f(\text{prod } n) \ n\end{aligned}$$

Vi kan förenkla problemet till att lösa ekvationen

$$\begin{aligned}\text{prod } z &= \text{case } z \ \text{of}\{ \\ &\quad \text{zero}\langle \rangle : \text{one}, \\ &\quad \text{succ}\langle n \rangle : f(\text{prod } n) \ n\}\end{aligned}$$

vilken löses av

$$\begin{aligned}\text{prod} &=_{\text{def}} \text{rec } p = \lambda z \rightarrow \text{case } z \ \text{of}\{ \\ &\quad \text{zero}\langle \rangle : \text{one}, \\ &\quad \text{succ}\langle n \rangle : f(p \ n) \ n\}\end{aligned}$$

vilket utan syntaktiskt socker skrives:

$$\begin{aligned}\text{prod} &=_{\text{def}} \text{rec } p = \lambda z \rightarrow \text{case } z \ \text{of}\{ \\ &\quad \text{zero} : \lambda u \rightarrow \text{one}, \\ &\quad \text{succ} : \lambda n \rightarrow f(p \ n.\text{arg1}) \ n.\text{arg1}\}\end{aligned}$$

Denna definition av konstanten prod är korrekt ty, i basfallet får vi:

$$\begin{aligned}\text{prod}(\text{zero} \ \langle \rangle) \\ &= \{\text{enl definitionen av prod}\} \\ &(\text{rec } p = \lambda z \rightarrow \text{case } z \ \text{of} \{ \\ &\quad \text{zero} : \lambda u \rightarrow \text{one}, \\ &\quad \text{succ} : \lambda n \rightarrow f(p \ n.\text{arg1}) \ n.\text{arg1}\})\end{aligned}$$

```

        zero: \u -> one;
        succ: \n -> f (p n.arg1) n.arg1})(zero <>)
= {enl beräkningsregel för rec}
\z -> case z of {
        zero: \u -> one;
        succ: ...} (zero <>)
= {enl beräkningsregel för applikation}
case zero <> of {
        zero: \u -> one;
        succ: ... }
= {enl beräkningsregel för case}
(\u -> one) <>
=
one

```

I efterföljarfallet får vi följande beräkningar:

```

prod (succ <n>)
= {enl definitionen av prod}
(rec p = \z -> case z of {
        zero: \u -> one;
        succ: \n -> f (p n.arg1) n.arg1})(succ <n>)
= {enl beräkningsregel för rec}
\z -> case z of {
        zero: \u -> one;
        succ: \n -> f (prod n.arg1) n.arg1})(succ <n>)
= {enl beräkningsregel för applikation}
case succ <n> of {
        zero: \u -> one;
        succ: \n -> f (prod n.arg1) n.arg1})
= {enl beräkningsregel för case}
(\n -> f (prod n.arg1) n.arg1) <n>
=
f (prod n) n)

```

(b) Följande uppgift är för de som studerat PCF:

- i. Define a PCF-program that behaves as `prod`, for any $n > 0$ (8)

$$\text{prod } n = (0 + 1) * (1 + 2) * \dots * (n - 1 + n)$$

- ii. What is the output of your PCF-program when applied to the value `Zero`? Justify by showing the main steps in the reduction of `prod Zero`. Explain. Here you should assume that (7)

both `add` and `mult` have the expected semantics. You can use either big or small semantics.

- iii. What is the purpose of the `fix` operator in PCF? (5)
Give a PCF-program that contains no `fix` operator and that does not terminate. Justify!

Lycka till!