582305 Symbolinen ohjelmointi
2. harjoitus, 3.10.2002

Tehtävä 2.1: Kalvojen II.6.5 λ-kalkyylin keksijä Alonzo Church ehdotti, että luonnollinen luku n kirjoitettaisiin sellaisena terminä c_n, että sovellus $(c_n\ f)$ esittäisi funktiota

$$\lambda x.(\underbrace{f\ (f\ (f\ (\ldots(f}_{n \text{ kappaletta}}\ x)))))$$

eli "funktion f sovellus n kertaa". Normaalimuodoissaan (eli loppuun saakka β-redusoituina) nämä *Church-numeraalit* ovat siis

$$\begin{aligned} c_0 &= \lambda f.\lambda x.x \\ c_1 &= \lambda f.\lambda x.(f\ x) \\ c_2 &= \lambda f.\lambda x.(f\ (f\ x)) \\ &\vdots \end{aligned}$$

(a) Anna sellainen λ-termi succ, että sovelluksen $(\text{succ}\ \ c_n)$ normaalimuoto on c_{n+1}.

(b) Anna sellainen λ-termi add, että sovelluksen $(\text{add}\ \ c_m\ c_n)$ normaalimuoto on c_{m+n}.

(c) Anna sellainen λ-termi mul, että sovelluksen $(\text{mul}\ \ c_m\ c_n)$ normaalimuoto on $c_{m \cdot n}$.

(d) Anna sellainen λ-termi pow, että sovelluksen $(\text{pow}\ \ c_m\ c_n)$ normaalimuoto on c_{m^n}.

Termien käyttäytymisestä muilla kuin Church-numeraaleilla ei tarvitse huolehtia.

Tehtävä 2.2: Tehtävän 2.1 tapaan myös totuusarvot voidaan esittää λ-termein:

$$\begin{aligned} \text{totta} &= \lambda p.\lambda q.p \\ \text{valhe} &= \lambda p.\lambda q.q. \end{aligned}$$

(a) Anna sellainen λ-termi nolla, että normaalimuodossaan

$$(\text{nolla}\ \ c_n) = \begin{cases} \text{totta} & \text{kun } n = 0 \\ \text{valhe} & \text{kun } n > 0. \end{cases}$$

Termin käyttäytymisestä muilla kuin Church-numeraaleilla c_n ei tarvitse huolehtia.

(b) Anna λ-termit näiden totuusarvojen negaatiolle, konjunktiolle ja disjunktiolle. Termien käyttäytymisestä muilla kuin totuusarvoilla ei tarvitse huolehtia.

Voit tarkistaa vastauksesi tehtäviin 2.1 ja 2.2 luennon 24.9. Scheme-funktiolla `beta` joka sieventää λ-termin normaalimuotoonsa. Sen saat kurssin WWW-kotisivun kautta.

Tehtävä 2.3: Luentojen II.7.3 esimerkki yleistetyistä murtojatkeista ei itse asiassa vaadi jatkeiden käyttöä tullakseen iteratiiviseksi. Kehitä alkuperäisestä funktiosta `cf` iteratiivinen `cfi` versio käyttäen kerääjänä pelkkää lukua. Perustele miksi `cfi` laskee samat tulokset kuin `cf`.

Tehtävä 2.4: Lisää tehtävään 2.3 sellainen virheenkäsittely, joka palauttaa virhearvon

- `#f` jos nollalla jako uhkaa
- `#t` jos parametrina saatu funktio f antoikin tuloksenaan jotakin muuta kuin numeron.

Tehtävä 2.5: *Ackermannin funktio* A luonnollisille luvuille määritellään seuraavasti:

$$\begin{aligned} A(0,n) &= n+1 \\ A(m+1,0) &= A(m,1) \\ A(m+1,n+1) &= A(m,A(m+1,n)). \end{aligned}$$

Ohjelmoi funktio A Scheme-kielellä siten, että funktiossasi on vain kaksi rekursiokutsua. Selitä kummastakin kutsustasi onko se loppuyhteydessä.

Tehtävä 2.6: Tehtavän 2.5 funktio on suunniteltu siten, että sen arvo (ja suoritusaika) kasvaa hyvin nopeasti. Tee tämän vuoksi siitä sellainen versio, joka ottaa lisäparametrin l, ja keskeyttää laskennan arvoon `#f` heti jos sisäkkäisen rekursion syvyys ylittää tämän rajan l.

(Tehtäviä yhteensä 6 kpl.)