582305 Symbolinen ohjelmointi
8. harjoitus, 21.11.2002

Huom! Päivän jälkimmäinen laskuharjoitustyhmä kokoontuukin jo klo 13-15, eli tuntia aiemmin kuin tavallisesti. Sali on edelleen sama eli A216. Jos pääset paikalle vasta normaaliin alkamisaikaan klo 14, niin saat silti hyvityspisteet tekemistäsi tehtävistä.

Tehtävä 8.1: Tarkastellaan jo tehtävässä 6.4 ollutta kuningatar Viktorian sukupuuta. Kirjoita seuraavat Prolog-predikaatit:

(a) `family(`M`,`D`,`C`)` joka hakee sukupuun lapsiperheet siten, että C on epätyhjä lista äidin M ja isän D yhteisistä lapsista.

(b) `familySize(`R`)` joka laskee muuttujaan R tällaisten lapsiperheiden keskimääräisen lapsiluvun.
(Laskemasi R on tosin väärin, koska aineisto on puutteellinen. Esimerkiksi Venäjän tsaari Nikolai II ja Alexandra saivat aineistossa olevan kruununprinssi Aleksein lisäksi myös neljä prinsessaa, mutta heitä ei lisätty aineistoon.)

(c) `isCarrier(`Y`)` jossa nainen Y kantaa tehtävän 6.5 mukaista verenvuototaudin aiheuttavaa geeniä. Ohjelmoi predikaattisi siten, että kukin eri Y palautetaan vain kerran, eikä useasti kuten tehtävässä 6.6.

Tehtävä 8.2: Toteuta Prolog-kielellä perustietorakenne *jono*. Tarkemmin sanoen, tee seuraavat Prolog-predikaatit:

(a) `emptyQueue(`Q`)` joka tuottaa tyhjän jonon Q.

(b) `enQueue(`Q`,`A`,`R`)` joka lisää aiemmin tehtyyn jonoon Q uuden viimeisen alkion A. Tässä R on näin syntynyt uusi yhtä alkiota pidempi jono.

(c) `deQueue(`Q`,`A`,`R`)` joka poistaa aiemmin tehdystä jonosta Q sen ensimmäisen alkion A (tai epäonnistuu jos Q on tyhjä). Tässä R on näin syntynyt uusi yhtä alkiota lyhyempi jono.

Huolehdi tehokkuudesta varmistamalla, ettei lisäys- ja poistopredikaattiesi tarvitse selata jonoa.

Voit olettaa, ettei alkuperäistä jonoa Q enää käytetä lisäyksen ja poiston jälkeen, vaan pelkästään tulosjonoa R. Tämä voi auttaa tehokkuudesta huolehtiessasi.

Tehtävä 8.3: Tämä tehtävä on niin laaja, että jokainen sen kohta on oma tehtävänsä hyvityspisteitä laskettaessa. Voit myös tehdä tämän tehtävän kohdista vain osan.

Scheme-kieli on implisiittisesti tyypitetty, joten ilmeisetkin ohjelmointivirheet, kuten yritys laskea yhteen kokonaisluku ja totuusarvo, selviävät usein vasta suoritusaikana. Jos Scheme-kieleen haluttaisiin kirjoittaa tyypintarkistusjärjestelmä, joka havaitsisi ne jo käännösaikana, niin se voitaisiin tehdä Prolog-kielellä esimerkiksi seuraavien periaatteiden mukaisesti.

Perustyypit ovat `bool` totuusarvoille ja `int` kokonaisluvuille.

Tyypin `bool` arvot ovat tutut vakiot `#t` ja `#f`.

Tyypin `int` arvot ovat vakiot $\ldots, -2, -1, 0, +1, +2, \ldots$

Rajoitutaan tarkastelemaan yhden muuttujan x nimettömiä funktioita `(lambda (`x`) `e`)`. Niiden tyyppi on $\tau_1 \to \tau_2$ missä τ_1 on argumentin x ja τ_2 tuloksen e tyyppi. Silloin esimerkiksi kirjastofunktion `positive?` tyyppi on `int` $\to$ `bool`.

Silloin voidaan sopia tyypitysperiaatteita kuten: "Ehtolausekkeessa `(if` e f g`)` on ehtona toimivan lausekkeen e oltava tyyppiä `bool`. Lisäksi sekä toden haaran f että epätoden haaran g on oltava samaa yhteistä tyyppiä τ, joka on myös koko ehtolausekkeen tyyppi."

(a) Valitse tyypitettäville Scheme-lausekkeille esitys Prolog-termeinä siten, että voit esittää ylläolevat totuusarvo- ja kokonaislukuvakiot, `if`-lausekkeet ja yhden muuttujan `lambda`-lausekkeet h sekä sellaisen kutsut `(`h a`)` argumentilla a.
Ota mukaan kokonaisluvuille myös kaksipaikkaiset perusoperaatiot `(`o p q`)` missä operaatio o on `+`, `-`,`*`, `/`, `=` tai `<`.
Valitse myös tyypeille sopiva esitys Prolog-termeinä.

(b) Edellä annettiin ehtolausekkeelle intuitiivinen mutta epätarkka tyypitysperiaate. Formaalimmin jokainen tyypitysperiaate voidaan kirjoittaa seuraavan muotoisena päättelysääntönä (samalla tapaa kuin kalvoilla II.5.5 kuvattiin Scheme-ohjelman suoritusperiaatteet):

$$\frac{x\colon \tau_1, \Gamma \vdash e\colon \tau_2}{\Gamma \vdash \texttt{(lambda (}x\texttt{) }e\texttt{)}\colon \tau_1 \to \tau_2}$$

Tämä sääntö luetaan seuraavasti: "Scheme-lauseke `(lambda (`x`) `e`)` on tyyppiä $\tau_1 \to \tau_2$ *tyyppisanakirjassa* Γ jos sen tuloslauseke e on tyyppiä τ_2 sellaisessa tyyppisanakirjassa $x\colon \tau_1, \Gamma$ joka on muuten kuin Γ paitsi että Scheme-muuttujan x tyyppi onkin τ_1."
Nämä tyyppisanakirjat ovat samanlaisia kuin kalvojen II.5.5 sanakirjatkin, paitsi että Scheme-muuttujaan x liitetäänkin nyt tyyppi τ_1 eikä vakioarvoa kuten aiemmin.
Anna tarvittavat tyypitysperiaatteet niille lausekkeille jotka esitit kohdassa (b). Vaakaviivanotaatiota ei tarvitse käyttää, mutta periaatteet on esitettävä yhtä yksityiskohtaisella tasolla.

(c) Kirjoita kohdan (b) periaatteesi Prolog-predikaattina. Toisin sanoen, toteuta tälle pienelle Scheme-osakielelle tyypintarkistusjärjestelmä Prolog-kielellä.

(d) Anna esimerkki syötteestä, jolla kohdan (c) Prolog-predikaattisi jättää tuloksenaan ilmoittamaansa tyyppiin vapaita muuttujia. Mitä ne mielestäsi tarkoittavat?

(Jos tämä alue alkoi kiinnostaa, niin tervetuloa kevään 2003 seminaariini "Tyyppiteoria ja ohjelmointikielet"...)

Tehtävä 8.4: Koska nämä ovat kurssin viimeiset laskuharjoitukset, käy kurssipalautesivulla antamassa palautetta luennoista ja harjoituksista.

(Tehtäviä yhteensä 4 kpl.)