T-79.144 **Syksy 2004**
Logiikka tietotekniikassa: perusteet
Laskuharjoitus 12 (opetusmoniste, kappaleet 9.1 – 9.5)
30.11. – 3.12.2004

1. Osoita lauselogiikan avulla oheisten ehtolausekkeiden ekvivalenssi.

(a) `!(a==b||a<b)`

(b) `a!=b&&!(b>a)`

Opetusmoniste määritteli Boolen lauseet tiettyjen perustapausten avulla, muiden ollessa lyhennysmerkintöjä. Näin ollen:

$$\texttt{a==b} \equiv_{def} \texttt{!(a>b) \&\& !(b>a)}$$

ja

$$\begin{aligned} \texttt{a<b} &\equiv_{def} \texttt{b>a} \\ \texttt{a!=b} &\equiv_{def} \texttt{!(a==b)} \end{aligned}$$

Määriteltäköön atomilauseet $A =$"`a>b`" ja $B =$"`b>a`". Näin a-kohdan lause saa muodon

$$\neg((\neg A \wedge \neg B) \vee B)$$

ja b-kohdan vastaavasti:

$$\neg(\neg A \wedge \neg B) \wedge \neg B$$

Havaitaan, että kun ensimäiseen sovelletaan kerran De Morganin teoreemaa saadaan jälkimmäinen. Näin ollen ne luonnollisesti ovat ekvivalentit.

2. Osoita osittainen oikeellisuus seuraavissa tapauksissa:

Esitetyt ohjelmat koostuvat peräkkäisistä sijoituslausekkeista. Näin ollen ne voidaan ratkaista käyttämällä hyväksi opetusmonisteessa annettuja päättelysääntöjä:

Sijoituslauseke: $\dfrac{}{[B\{\texttt{x}/E\}]\ \texttt{x=}E\ [B]}$

Kompositio: $\dfrac{[B_0]\ C_1\ [B_1] \qquad [B_1]\ C_2\ [B_2]}{[B_0]\ C_1\ \texttt{;}\ C_2\ [B_2]}$

(a) $\models_p$ [x>0] y=x+1 [y>1]

Ratk. Lähtemällä jälkiehdosta ja soveltamalla sijoituslausekkeen päättelysääntöä taaksepäin saadaan muoto [x+1>1] y=x+1 [y>1]

x>0 on ekvivalentti lauseke x+1>1 kanssa, joten erityisesti siis sen vahvennus, jolloin väittämä pätee.

(b) $\models_p$ [true] y=x; y=x+x+y [y==3*x]

Ratk. Soveltamalla kahteen kertaan sijoituslausekkeen päättelysääntöä saadaan muoto:

[x+x+y==3*x] y=x+x+y [y==3*x]
[x+x+x==3*x] y=x [x+x+y==3*x]

ja edelleen komposition sääntöä soveltamalla:

[x+x+x==3*x] y=x; y=x+x+y [y==3*x]

Lauseke x+x+x==3*x on luonnollisesti aina totta kokonaislukujen joukossa, jolloin väittämä pätee.

(c) $\models_p$ [x>1] a=1; y=x; y=y-a [y>0&&x>y]

Samoin kuin b-kohdassa sijoituslausekkeen päättelysääntöä tulee soveltaa peräkkäin. Saadaan:

[y-a>0&&x>y-a] y=y-a [y>0&&x>y]
[x-a>0&&x>x-a] y=x [y-a>0&&x>y-a]
[x-1>0&&x>x-1] a=1 [x-a>0&&x>x-a]

Kompositiosäännön suoraviivainen sovellus sivuutetaan. Nyt ensimmäisen ehdon x-1>0&&x>x-1 jälkimmäinen konjunkti on tutkittavissa kokonaislukustruktuureissa aina tosi. Edelleen ehto x-1>0 on yhtäpitävä ehdon x>1 kanssa, joka esiintyy väitteessä.

3. Osoita, että $\models_p$ [true] P [z== min (x,y)], missä P on seuraava ohjelma:

```
if(x > y) then {
    z = y
} else {
    z = x
}
```

Ratk. Tehtävän ratkaisuun tarvitaan edelläesitettyjen päättelysääntöjen lisäksi seuraava sääntö:

Ehtolauseke: $$\frac{[B_1 \,\texttt{\&\&}\, B]\ C_1\ [B_2] \qquad [B_1 \,\texttt{\&\&}\, !B]\ C_2\ [B_2]}{[B_1]\ \texttt{if}\,(B)\ \texttt{then}\,\{C_1\}\ \texttt{else}\,\{C_2\}\ [B_2]}$$

Pyritään osoittamaan $\texttt{z} == \min(\texttt{x},\texttt{y})$. Tämä on siis ylläolevan säännön ehto B_2. Tutkitaan komentoja C_1 ja C_2 erikseen. C_2:sta saadaan.

$$[\texttt{x} == \min(\texttt{x},\texttt{y})]\ \texttt{z} = \texttt{x}\ [\texttt{z} == \min(\texttt{x},\texttt{y})]$$

Ehto $\texttt{x} == \min(\texttt{x},\texttt{y})$ on yhtäpitävää lausekkeen `x <= y` tai lausekkeen `! (x > y)` kanssa. Luonnollisesti lauseke voidaan kirjoittaa myös muotoon `true && ! (x > y)`. Komennosta C_1 saadaan vastaavasti:

$$[\texttt{y} == \min(\texttt{x},\texttt{y})]\ \texttt{z} = \texttt{y}\ [\texttt{z} == \min(\texttt{x},\texttt{y})]$$

Ja edelleen samalla päättelyllä esiehdoksi `true && (x > y)`. Nyt voidaan soveltaa ehtolausekkeen päättelysääntöä (viivan yläpuolella olevat lausekkeet on johdettu) ja saadaan:

```
[true]
if(x > y) then {
    z = y
} else {
    z = x
}  [z == min(x, y)]
```

4. Osoita ohjelmasta `Sum` seuraavat ominaisuudet:

(a) $\models_p [\texttt{true}]\ \texttt{Sum}\ [\texttt{z} == \texttt{x} + \texttt{y}]$

(b) $\models_t [0 <= \texttt{y}]\ \texttt{Sum}\ [\texttt{z} == \texttt{x} + \texttt{y}]$

`Sum` on seuraava ohjelma:

```
z=x;
v=y;
while(!(v==0)){
    z=z+1;
    v=v-1
}
```

Tehtävän a-kohdassa pitää todistaa muotoa $[B_1]$ `while(`B`)` $\{C\}$ $[B_2]$ oleva osittainen oikeellisuus. Tähän on annettu seuraava menettely:

A1. Valitaan I siten, että $\models_{\mathbb{Z}}$ `(`I`&&!`B`)->`B_2.

A2. Haetaan heikoin esiehto I' siten, että $\models_p [I']\ C\ [I]$.

A3. Osoitetaan $\models_{\mathbb{Z}}$ I`&&`B`->`I', minkä nojalla $\models_p [I\texttt{\&\&}B]\ C\ [I]$ ja edelleen $\models_p [I]$ `while(`B`)` $\{C\}$ $[I\texttt{\&\&!}B]$.

A4. Osoitetaan $\models_{\mathbb{Z}} B_1$ `->`I.

Menettelyssä I on invariantti, jonka pitäisi olla aina voimassa, silmukan lauseiden suorituksen jälkeen. Nyt silmukkaa tutkimalla havaitaan, että muuttujan z arvo kasvaa ja v pienenee, erityisesti siis niiden summa pysyy vakiona. Kun niille lisäksi annetaan alkuarvot x ja y on summa näiden vakioiden summan suuruinen. Olkoon siis I lauseke `z+v==x+y`. Nyt havaitaan, että menetelmän kohta A1 pätee, koska `!`B on lauseke `v==0`, josta luonnollisesti seuraa, että `z==x +y`. Lähdetään siis hakemaan silmukan komentojen heikoimpia esiehtoja. Saadaan:

$$[\texttt{z+v-1==x+y}]\ \texttt{v=v-1}\ [\texttt{z+v==x+y}]$$
$$[\texttt{z+1+v-1==x+y}]\ \texttt{z=z+1}\ [\texttt{z+v-1==x+y}]$$

Saadussa esiehdossa `+1` ja `-1` kumoutuvat, jolloin esiehto on sama kuin invariantti I. Menetelmän kohdassa A3 tulee osoittaa, että I`&&`B`->`I', so. lause:

$$\texttt{z+v==x+y\&\&!(v==0)} \rightarrow \texttt{z+v==x+y}$$

Tämä luonnollisesti pätee. Näin ollen pätee myös:

```
[z +v ==x +y]
while(!(v ==0)){
    z=z+1;
    v=v-1
} [z +v ==x +y &&v ==0]
```

Menetelmän kohta A4 redusoituu lauseen $I \rightarrow I$ todistamiseen. Todistuksen loppuunsaattamiseksi pitää vielä tutkia alun sijoituslauseet. Saadaan:

$$[\texttt{z+y==x+y}]\ \texttt{v=y}\ [\texttt{z+v==x+y}]$$
$$[\texttt{x+y==x+y}]\ \texttt{z=x}\ [\texttt{z+y==x+y}]$$

Ensimmäinen ehto `x+y==x+y` on luonnollisesti aina totta $\square$.

(b) $\models_t [\texttt{0<=y}]\ \texttt{Sum}\ [\texttt{z==x+y}]$

Tehtävän b-kohdassa pyydetään todistamaan, että ohjelma myös päättyy ja että tällöin tarvitaan vahvempi esiehto. Täydellisen oikeellisuuden todistamisessa tarvitaan seuraavaa vahvempaa päättelysääntöä:

$$\frac{[B_1\ \texttt{\&\&}\ B_2\ \texttt{\&\&}\ (E\texttt{==n})]\ C\ [B_1\ \texttt{\&\&}\ (E\texttt{<n})]}{[B_1]\ \texttt{while}(B_2)\ \{C\}\ [B_1\ \texttt{\&\&}\ \texttt{!}B_2]}$$

Säännössä B_1 sisältää aiemman invariantin lisäksi vaatimuksen siitä, että kokonaislukulauseke E`>=0`. Sääntö sanoo, että aina kun toistolausekkeen komentoja C suoritetaan invariantin ja toistolausekkeen ehdon ollessa voimassa niin valitaanpa kokonaislukumuuttujan `n` arvo miten tahansa, niin aina, kun esiehto on voimassa niin jälkiehto on myös. Jälkiehto puolestaan sanoo, että invariantti säilyy ja kokonaislukulauseke pienenee aidosti (säilyen kuitenkin ei-negatiivisena). Näin ollen komentojen C toistaminen johtaa väistämättä siihen, että toistolausekkeen ehto menee lopulta epätodeksi ja ohjelman suoritus täten päättyy.

Esimerkkiohjelmassa lausekkeeksi E voidaan valita `v`. Tällöin päättely komennoille C saa muodon:

$$[\texttt{z+v-1==x+y\&\&0<=v-1\&\&v-1<n}]\ \texttt{v=v-1}\ [\texttt{z+v==x+y\&\&0<=v\&\&v<n}]$$
$$[\texttt{z+v==x+y\&\&0<=v-1\&\&v-1<n}]\ \texttt{z=z+1}\ [\texttt{z+v-1==x+y\&\&0<=v-1\&\&v-1<n}]$$

Jotta päättelysääntöä voisi käyttää täytyy todistaa, että saatu esiehto on päättelysäännössä tarvittavan esiehdon B_1 `&&` B_2 `&&` $(E$`==n)` looginen seuraus. Kyseinen esiehto saa muodon:

$$\texttt{z+v==x+y\&\&0<=v\&\&!(v==0)\&\&v==n}$$

Looginen seuraavuus pätee, koska muuttujalle `n` voidaan valita vain positiivisia arvoja, jotta vasen puoli saadaan todeksi. Näin ollen voidaan päätellä:

```
[z + v == x + y && 0 <= v]
while(!(v == 0)) {
     z = z + 1;
     v = v - 1
} [z + v == x + y && 0 <= v && v == 0]
```

Nyt `0<=v` on jälkiehdon totuusarvon kannalta luonnollisesti redundantti. Päättelyn loppuunsaattamiseksi pitää vielä tutkia alun sijoituslauseet:

$$[z+y==x+y\,\&\&\,0<=y]\ v=y\ [z+v==x+y\,\&\&\,0<=v]$$
$$[x+y==x+y\,\&\&\,0<=y]\ z=x\ [z+y==x+y\,\&\&\,0<=y]$$

Saatu esiehto redusoituu siis muotoon `0<=y` eli ohjelma päättyy vain jos muuttujan `y` arvo on ei-negatiivinen, kun ohjelman suoritus aloitetaan. □