Luento 12: Ohjelmien oikeellisuustarkastelut

1. Ohjelmointikieli
2. Boolen lausekkeiden ekvivalenssi
3. Ohjelmien esi- ja jälkiehdot
4. Toistolauseiden invariantit
5. Täysi oikeellisuus

Motivaatio

Miksi tietokoneohjelmille tulisi kirjoittaa formaaleja spesifikaatioita?

- Spesifikaatiota laadittaessa joudutaan suunnittelemaan ennalta varsin tarkkaan mitä ohjelmiston on tarkoitus tehdä.
- Järjestelmän toteutus voidaan *verifioida* eli todeta määrittelynsä mukaiseksi vasta, kun spesifikaatio on tehty.
- Formaalissa spesifioinnissa etuna on määritelmien yksikäsitteisyys.
- *Turvallisuuskriittiset järjestelmät* (esim. lentokoneen ohjausjärjestelmä) vaativat perinpohjaista määrittelyä ja verifiointia.
- Hyvin määritellyn ohjelman uudelleenkäyttö on helpompaa.

1. OHJELMOINTIKIELI

- Määritellään kokonaislukujen käsittelyyn riittävä osajoukko tyypillisistä lausekielistä kuten Pascal, C, C++ ja Java.

Määritelmä 15.1 (Kokonaislukulausekkeet)

1. Mikä tahansa *kokonaisluku* ..., `-1`, `0`, `1`, ... on kokonaislukulauseke.
2. *Kokonaislukumuuttujat* `x`, `y`, ... ovat kokonaislukulausekkeita.
3. Jos E_1 ja E_2 ovat kokonaislukulausekkeita, niin myös *summa* (E_1`+`E_2), *erotus* (E_1`-`E_2) ja *tulo* (E_1`*`E_2) ovat kokonaislukulausekkeita.

Esimerkki 15.2 Merkkijono `((x-y)*x)` on kokonaislukulauseke.

Loogisten ehtojen ilmaiseminen

Määritelmä 15.3 (Boolen lausekkeet)

1. Boolen vakiot `false` ja `true` ovat Boolen lausekkeita.
2. Jos E_1 ja E_2 ovat kokonaislukulausekkeita, niin E_1`>`E_2 on Boolen lauseke.
3. Jos B_1 ja B_2 ovat Boolen lausekkeita, niin myös *negaatio* `!`B_1, *konjunktio* B_1`&&`B_2 ja *disjunktio* B_1`||`B_2 ovat Boolen lausekkeita.

Joitain lyhennysmerkintöjä:

E_1`==`E_2	`!(`E_1`>`E_2`)&&!(`E_2`>`E_1`)`
E_1`!=`E_2	`!(`E_1`==`E_2`)`
B_1`->`B_2	`!`B_1`||`B_2
B_1`<->`B_2	`(`B_1`->`B_2`)&&(`B_2`->`B_1`)`

Ohjelmat

Määritelmä 15.4 (Komennot)

1. Jos x on kokonaislukumuuttuja ja E on kokonaislukulauseke, niin *sijoituslause* x=E on komento.
2. Jos B on Boolen lauseke sekä C_1 ja C_2 ovat komentoja, niin myös
   — *ketjulause* C_1 ; C_2,
   — *ehtolause* if(B) then {C_1} else {C_2} ja
   — *toistolause* while(B) {C_1}
   ovat komentoja.

Esimerkki 15.5 Ohjelma

```
y=1;z=1;while(z!=x){z=z+1;y=y*z}
```

laskee muuttujan y arvoksi muuttujan x arvon kertoman, kun x>0. ■

Ohjelman suorituksen tilan esittäminen

Ohjelman suorituksen tila voidaan rinnastaa $\mathbb{Z}$-*struktuuriin*:

- Kokonaislukulausekkeen E arvo tilassa $\mathcal{S}$ on kokonaisluku $E^{\mathcal{S}}$.
- Boolen lauseke B on tosi tilassa $\mathcal{S}$ $\iff$ $\mathcal{S} \models B$.

Määritelmä 15.6 Struktuuri $\mathcal{S}$ on $\mathbb{Z}$-struktuuri, jos ja vain jos:

1. Struktuurin $\mathcal{S}$ universumina U on kokonaislukujen joukko $\mathbb{Z}$.
2. Jokaisen kokonaisluvun (vakiosymboli tarkasteltavassa kielessä) tulkintana on kyseinen kokonaisluku itse.
3. Funktiosymboleilla +, - ja * on standarditulkinnat.
4. Predikaattisymbolin > tulkintana on suurempi kuin -relaatio kokonaislukujen joukossa.

Komentojen vaikutus tilaan

Määritelmä 15.8 Jos komennon C *suoritus päättyy äärellisellä askelmäärällä* tilasta $\mathcal{S}$, niin saavutettava tila $\mathcal{S}'$ määräytyy seuraavasti:

1. Jos C on sijoituslause x=E, niin $\mathcal{S}'$ on tila $\mathcal{S}[\mathrm{x} \mapsto E^{\mathcal{S}}]$.
2. Jos C on ketjulause C_1 ; C_2, niin $\mathcal{S}'$ on tila, joka saavutetaan suorittamalla ensin C_1 ja sitten C_2.
3. Jos C on ehtolause if(B) then {C_1} else {C_2}, niin $\mathcal{S}'$ on tila, joka saavutetaan suorittamalla C_1, jos $\mathcal{S} \models B$, ja C_2, jos $\mathcal{S} \not\models B$.
4. Jos C on toistolause while(B) {C_1} ja $\mathcal{S} \not\models B$, niin tila $\mathcal{S}' = \mathcal{S}$.
5. Jos C on toistolause while(B) {C_1} ja $\mathcal{S} \models B$, niin $\mathcal{S}'$ on tila, joka saavutetaan suorittamalla C_1 ; while(B) {C_1}.

2. BOOLEN LAUSEKKEIDEN EKVIVALENSSI

- Ohjelmointikielissä käytetään paljon ehtolauseita kontrolloimaan, mitä komentoja suoritetaan ja millä ehdoilla.
- Jos ehtolauseita muutetaan esim. optimointitarkoituksessa, halutaan varmistua että komennot suoritetaan samoilla ehdoilla.
- Boolen lausekkeiden ekvivalenssin osoittamiseen voidaan käyttää sekä lauselogiikan että predikaattilogiikan menetelmiä.
- Jos Boolen lausekkeiden evaluoinnilla on *sivuvaikutuksena* muutoksia ohjelman tilaan, pelkkä loogisen ekvivalenssin tarkastaminen ei välttämättä riitä.

Esimerkki (15.11)

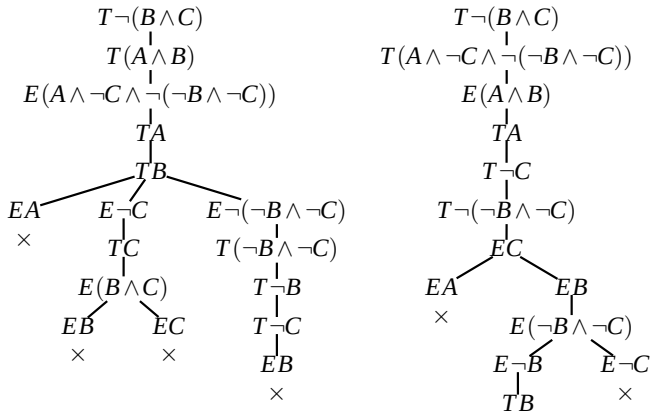
```
if((x>0)&&!(y>x)) then {
    if(x!=y) then {z=x} else {z=y}
} else {z=0}

if(x>0) then {
    if(x>y) then {z=x} else {z=y}
} else {z=0}
```

- Valitaan atomiset lauseet $A =$ "x>0", $B =$ "x>y" ja $C =$ "y>x".
- Nyt esimerkiksi sijoituslause z=x suoritetaan näissä ohjelmissa seuraavilla ehdoilla: $A \wedge \neg C \wedge \neg(\neg B \wedge \neg C)$ ja $A \wedge B$.
- Lauselogiikan nojalla näiden välinen ekvivalenssi on looginen seuraus lauseesta $\neg(B \wedge C)$, joka kuvaa lauseiden B ja C suhteen.

Esimerkki (jatkuu)

Todistetaan $\{\neg(B \wedge C)\} \models (A \wedge B) \leftrightarrow (A \wedge \neg C \wedge \neg(\neg B \wedge \neg C))$:



Kytkennät predikaattilogiikkaan

- Boolen lauseke B on $\mathbb{Z}$-*pätevä* (merkitään $\models_{\mathbb{Z}} B$), jos ja vain jos $\mathcal{S} \models B$ kaikissa $\mathbb{Z}$-struktuureissa $\mathcal{S}$.
- Boolen lausekkeet B_1 ja B_2 ovat $\mathbb{Z}$-*ekvivalentit* (merk. $B_1 \equiv_{\mathbb{Z}} B_2$), jos ja vain jos kaikille $\mathbb{Z}$-struktuureille $\mathcal{S}$, $\mathcal{S} \models B_1 \iff \mathcal{S} \models B_2$.
- Huomaa, että

  $$\models B \implies \models_{\mathbb{Z}} B \text{ ja } B_1 \equiv B_2 \implies B_1 \equiv_{\mathbb{Z}} B_2,$$

  mutta käänteiset implikaatiot eivät välttämättä ole voimassa.
- Gödelin epätäydellisyystulosten perusteella virheetön ja täydellinen todistusjärjestelmä $\mathbb{Z}$-pätevyydelle on mahdoton.

Esimerkki 15.12 $\models_{\mathbb{Z}}$!((x>y)&&(y>x)), mutta $\not\models$!((x>y)&&(y>x)). Vastamallina on $\mathcal{S}$, missä $U = \{0\}$, $\mathrm{x}^{\mathcal{S}} = \mathrm{y}^{\mathcal{S}} = 0$ ja $>^{\mathcal{S}} = \{\langle 0,0\rangle\}$.

3. OHJELMIEN ESI- JA JÄLKIEHDOT

- Tarkasteltavan ohjelmointikielen ohjelmilla on ääretön tila-avaruus, joiden läpikäyminen tila tilalta on käytännössä mahdotonta.
- Yksi mahdollisuus on tarkastella tilajoukkoja

  $$\mathrm{Mod}_{\mathbb{Z}}(B) = \{\mathcal{S} \mid \mathcal{S} \text{ on } \mathbb{Z}\text{-struktuuri ja } \mathcal{S} \models B\}$$

  ja analysoida ohjelman näihin aiheuttamia muutoksia.
- Mille tahansa ohjelmalle P voidaan asettaa *esi- ja jälkiehdot* B_1 ja B_2 kirjoittamalla ns. *Hoaren kolmikko* $[B_1]$ P $[B_2]$.
- Karkeasti ottaen ajatuksena on, että esiehto B_1 takaa jälkiehdon B_2 voimaantulon ohjelman P suorituksen päättyessä.

Esimerkki 15.13 Ohjelmalle if(x==0) then {y=1} else {y=x+1} voidaan antaa spesifikaatio [true] Succ [y==x+1].

Ohjelman osittainen oikeellisuus

➤ Olkoon P ohjelma sekä B_1 ja B_2 kaksi Boolen lauseketta.

Määritelmä 15.14 Ohjelma P on *osittain oikeellinen* annettujen esi- ja jälkiehtojen B_1 ja B_2 suhteen (merkitään $\models_p [B_1]\ \mathtt{P}\ [B_2]$), jos ja vain jos $\mathcal{S}' \models B_2$ pätee saavutettavalle tilalle $\mathcal{S}'$ aina silloin, kun

— ohjelman P suoritus aloitetaan tilasta $\mathcal{S}$, missä $\mathcal{S} \models B_1$, ja

— ohjelman P *suoritus päättyy äärellisellä askelmäärällä* tilaan $\mathcal{S}'$.

Esimerkki 15.15 Ohjelman osittainen oikeellisuus ei edellytä ohjelman suorituksen päättymistä:

$$\models_p [\mathtt{true}]\ \mathtt{while(x!=y)\{z=x;x=y;y=z\}}\ [\mathtt{x==y}].$$

Päättelysäännöt osittaiselle oikeellisuudelle

Boolen lausekkeille B, B_0, B_1 ja B_2 sekä komennoille C, C_1 ja C_2:

Sijoituslause: $$\frac{}{[B\{\mathtt{x}/E\}]\ \mathtt{x=}E\ [B]}$$

Kompositio: $$\frac{[B_0]\ C_1\ [B_1] \qquad [B_1]\ C_2\ [B_2]}{[B_0]\ C_1\ \mathtt{;}\ C_2\ [B_2]}$$

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

Toistolause: $$\frac{[B_1 \mathtt{\&\&} B_2]\ C\ [B_1]}{[B_1]\ \mathtt{while(}B_2\mathtt{)\{}C\mathtt{\}}\ [B_1 \mathtt{\&\&\ !}B_2]}$$

Implikaatio: $$\frac{\models_{\mathbb{Z}} B_1 \mathtt{->} B_2 \qquad [B_2]\ C\ [B_3] \qquad \models_{\mathbb{Z}} B_3 \mathtt{->} B_4}{[B_1]\ C\ [B_4]}$$

Esimerkki (15.15)

Todistetaan

$$\models_p [\mathtt{(x==n)\&\&(y==m)}]\ \mathtt{z=x;x=y;y=z}\ [\mathtt{(x==m)\&\&(y==n)}].$$

Kirjoitetaan todistus klassisen sekvenssin muotoon:

1. $[\mathtt{(x==m)\&\&(z==n)}]\ \mathtt{y=z}\ [\mathtt{(x==m)\&\&(y==n)}]$
2. $[\mathtt{(y==m)\&\&(z==n)}]\ \mathtt{x=y}\ [\mathtt{(x==m)\&\&(z==n)}]$
3. $[\mathtt{(y==m)\&\&(z==n)}]\ \mathtt{x=y;y=z}\ [\mathtt{(x==m)\&\&(y==n)}]$
4. $[\mathtt{(y==m)\&\&(x==n)}]\ \mathtt{z=x}\ [\mathtt{(y==m)\&\&(z==n)}]$
5. $[\mathtt{(y==m)\&\&(x==n)}]\ \mathtt{z=x;x=y;y=z}\ [\mathtt{(x==m)\&\&(y==n)}]$
6. $[\mathtt{(x==n)\&\&(y==m)}]\ \mathtt{z=x;x=y;y=z}\ [\mathtt{(x==m)\&\&(y==n)}]$

Heikoimmat esiehdot

➤ Olkoon P ohjelma $C_1\ \mathtt{;}\ \ldots\ \mathtt{;}\ C_n$, missä $C_1, \ldots, C_n$ ovat järjestyksessä peräkkäin suoritettavat komennot.

➤ Spesifikaation $\models_p [B_0]\ \mathtt{P}\ [B_n]$ todistaminen voidaan pilkkoa osaongelmiin: tulisi löytää $B_1, \ldots, B_{n-1}$ siten, että $\models_p [B_{i-1}]\ C_i\ [B_i]$ kaikille $i \in \{1, \ldots, n\}$.

➤ Usein tällaiset ehdot voidaan löytää analysoimalla komentosekvenssiä *takaperin*: haetaan komennolle C_i *heikoin esiehto* B_{i-1} siten, että $\models_p [B_{i-1}]\ C_i\ [B_i]$.

➤ Todistussekvenssit voidaan kirjoittaa (rekursiivisesti) muotoon

$$[B_0]\ C_1\ \mathtt{;}\ [B_1]\ C_2\ \mathtt{;}\ [B_2] \ldots\ C_{n-1}\ \mathtt{;}\ [B_{n-1}]\ C_n\ [B_n].$$

Heikoimpien esiehtojen hakeminen

- Sijoitus- ja ehtolauseiden heikoimmat esiehdot voidaan muodostaa varsin systemaattisesti.
- Toistolauseiden käsittely vaatii *invarianttien* käyttöä.

1. Jos B on jälkiehto sijoituslauseelle `x=`E, heikoimmaksi esiehdoksi voidaan kirjata $B\{\texttt{x}/E\}$ eli $\models_p [B\{\texttt{x}/E\}]$ `x=`E $[B]$.
2. Jos $\models_p [B_1]\ C\ [B_2]$ on jo osoitettu ja B_0 on esiehdon B_1 vahvennus ($\models_{\mathbb{Z}} B_0$ `->`B_1), kirjataan $[B_0]\ [B_1]\ C\ [B_2]$, koska $\models_p [B_0]\ C\ [B_2]$.
3. Jos $\models_p [B_1]\ C_1\ [B_3]$ ja $\models_p [B_2]\ C_2\ [B_3]$ ovat jo osoitetut jälkiehdolle B_3, niin ehtolauseen `if(`B`) then {`C_1`} else {`C_2`}` heikoimmaksi esiehdoksi kirjataan $(B\texttt{\&\&}B_1)\texttt{||}(\texttt{!}B\texttt{\&\&}B_2)$.

Esimerkki (15.19)

Tarkastellaan ohjelman Succ muunnelmaa:

```
[true]
[(x==0)||!(x==0)]
[(((x+1)-1==0)&&(x==0))||(!((x+1)-1==0)&&(x+1==x+1))]
z=x+1;
[((z-1==0)&&(x==0))||(!(z-1==0)&&(z==x+1))]
if(z-1==0) then {
    [x==0] [1==x+1]  y=1 [y==x+1]
} else {
    [z==x+1]  y=z [y==x+1]
}
[y==x+1]
```


4. TOISTOLAUSEIDEN INVARIANTIT

- Ohjelmointikielten keskeisiä primitiivejä ovat myös toistolauseet, joiden avulla komentoja voidaan toistaa haluttu määrä.

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

- Ongelmana on, millaisilla periaatteilla voidaan osoittaa toistorakenteita sisältävien algoritmien oikeellisuus.
- Toistorakenteelle halutaan tyypillisesti todistaa *invariantti* eli ominaisuus, joka säilyy voimassa toistorakenteen suorituksen ajan.

Määritelmä 15.20 Toistolauseen `while(`B`) {`C`}` invariantti I on mikä tahansa Boolen lauseke siten, että $\models_p [I\texttt{\&\&}B]\ C\ [I]$.

Toistolauseen yleinen spesifikaatio

- Spesifikaation $\models_p [B_1]$ `while(`B`) {`C`}` $[B_2]$ todistaminen voi perustua sopivan invariantin I käyttöön:
  1. $\models_{\mathbb{Z}} B_1$ `->`I,
  2. $\models_{\mathbb{Z}} (I\texttt{\&\&}\texttt{!}B)$ `->`B_2, ja
  3. $\models_p [I]$ `while(`B`) {`C`}` $[I\texttt{\&\&}\texttt{!}B]$.
- Kuten aiemminkin, oikeellisuustodistusta voi hakea *takaperin*:

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

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

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

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

Esimerkki (15.21)

- Todistetaan kertolaskuohjelman `Multi` osittainen oikeellisuus spesifikaation $\models_p$ `[true] Multi [v==x*y]` muodossa.
- Lähtökohtana on invariantti `v==z*y`:

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

Huomio: Ohjelman suoritus päättyy, jos ja vain jos `(x>=0)`.

5. TÄYSI OIKEELLISUUS

- Tieto osittaisesta oikeellisuudesta ($\models_p [B_1]\ C\ [B_2]$) on hyödyllinen ainoastaan, mikäli komennon C suoritus todella päättyy.
- *Täyttä oikeellisuutta* ($\models_t$) varten joudutaan todistamaan erikseen, että toistolauseiden suoritukset päättyvät loppujen lopuksi.
- Tätä varten tarvitaan vahvennettua päättelysääntöä

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

Määritelmä 15.22 Ohjelma P on *täysin oikeellinen* suhteessa annettuihin esi- ja jälkiehtoihin B_1 ja B_2 (merk. $\models_t [B_1]$ P $[B_2]$), jos ja vain jos $\models_p [B_1]$ P $[B_2]$ ja ohjelman P *suoritus päättyy äärellisellä askelmäärällä*, jos $\mathcal{S} \models B_1$ alkutilassa $\mathcal{S}$.

Esimerkki (15.24)

Todistetaan $\models_t$ `[0<=x] Multi [v==x*y]`:

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


TAVOITTEET

- Osaat muodostaa ohjelmille spesifikaatioita esi- ja jälkiehtojen avulla (Hoaren kolmikot).
- Osaat hakea heikoimpia esiehtoja sijoitus- ja ehtolauseista koostuville ohjelmille ao. päättelysääntöjen avulla.
- Ymmärrät osittaisen ja täyden oikeellisuuden välisen eron.
- Tiedät mitä invariantilla tarkoitetaan ja miksi niiden käyttö ohjelmankehityksessä on hyödyllistä.
- Osaat hakea yksinkertaisille toistolauseille invariantteja.

PÄIVÄN PÄHKINÄ

Mieti tarkemmin invariantin muodostamista toistolauseelle `while`(B) $\{C\}$ esimerkiksi osittaisen oikeellisuuden $\models_p [I]$ `while`(B) $\{C\}$ $[I$ `&&` $!B]$ tapauksessa.

- Voiko invariantti I olla toistolauseen `while`(B) $\{C\}$ heikoin esiehto?
- Analysoi tässä mielessä kertolaskuohjelman `Multi` oikeellisuustodistusta (esimerkki 15.15).