

Luento 11: Ohjelmien oikeellisuustarkastelut

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

1. OHJELMOINTIKIELI

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

Määritelmä 15.1 (Kokonaislukulausekkeet)

1. Mikä tahansa *kokonaisluku* $\dots, -1, 0, 1, \dots$ on kokonaislukulauseke.
2. *Kokonaislukumuuttujat* $x, y, \dots$ 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.

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.
- Formaalisissa spesifioinnissa etuna on määritelmien yksikäsitteisyys.
- *Turvallisuuskriittiset järjestelmät* (esim. lentokoneen ohjausjärjestelmä) vaativat perinpohjaista määrittelyä ja verifiointia.
- Hyvin määritellyn ohjelman uudelleenkäyttö on helpompaa.

Loogisten ehtojen ilmaiseminen

Määritelmä 15.3 (Boolean lausekkeet)

1. Boolean vakiot false ja true ovat Boolean lausekkeita.
2. Jos E_1 ja E_2 ovat kokonaislukulausekkeita, niin $E_1 > E_2$ on Boolean lauseke.
3. Jos B_1 ja B_2 ovat Boolean lausekkeita, niin myös *negatio* $!B_1$, *konjunktio* $B_1 \&\& B_2$ ja *disjunktio* $B_1 \mid \mid B_2$ ovat Boolean lausekkeita.

Joitain lyhennysmerkintöjä: $E_1 == E_2$ $!(E_1 > E_2) \&\& !(E_2 > E_1)$

$E_1 != E_2$ $!(E_1 == E_2)$

$B_1 \rightarrow B_2$ $!B_1 \mid \mid B_2$

$B_1 \leftrightarrow B_2$ $(B_1 \rightarrow B_2) \&\& (B_2 \rightarrow B_1)$

Ohjelmat

Määritelmä 15.4 (Komennot)

1. Jos x on kokonaislukumuuttuja ja E on kokonaislukulauseke, niin *sijoituslause* $x=E$ on komento.
2. Jos B on Boolean lauseke sekä C_1 ja C_2 ovat komentoja, niin myös
 - *ketjulause* $C_1 ; C_2$,
 - *ehtolause* $\text{if}(B) \text{ then } \{C_1\} \text{ else } \{C_2\}$ ja
 - *toistolause* $\text{while}(B) \{C_1\}$
 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 $x > 0$. ■

Komentojen vaikutus tilaan

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

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

Ohjelman suorituksen tilan esittäminen

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

1. Struktuurin 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.

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

- Kokonaislukulausekkeen E arvo tilassa S on kokonaisluku E^S .
- Boolean lauseke B on tosi tilassa $S \iff S \models B$.

2. BOOLEAN 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.
- Boolean lausekkeiden ekvivalenssin osoittamiseen voidaan käyttää sekä lauselogiikan että predikaattilogiikan menetelmiä.
- Jos Boolean 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.

Kytkenät predikaattilogiikkaan

- Boolean lauseke B on $\mathbb{Z}$ -*pätevä* (merkitään $\models_{\mathbb{Z}} B$), jos ja vain jos $S \models B$ kaikissa $\mathbb{Z}$ -struktuureissa S .
- Boolean 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 S , $S \models B_1 \iff S \models B_2$.
- Huomaa, että

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

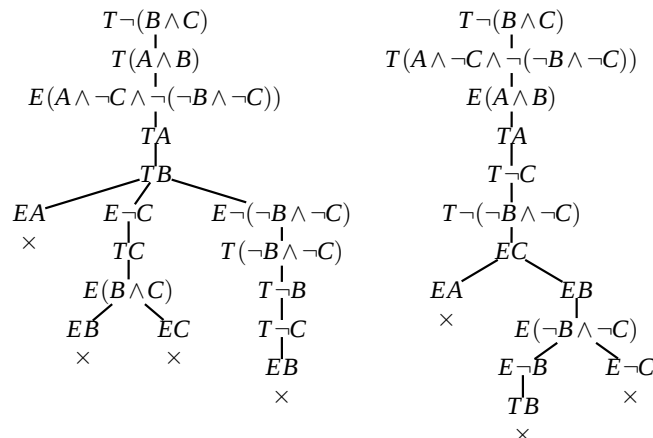
mutta käänteiset implikaatiot eivät välttämättä ole voimassa.

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

Huomio Gödelin epätäydellisyyslusten perusteella virheetön ja täydellinen todistusjärjestelmä $\mathbb{Z}$ -pätevyydelle on mahdoton.

Esimerkki (jatkuu)

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



3. OHJELMIEN ESI- JA JÄLKIEHDOT

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

$$\mathcal{M}_{\mathbb{Z}}(B) = \{S \mid S \text{ on } \mathbb{Z}\text{-struktuuri ja } S \models B\}$$
 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 $\text{if}(x==0) \text{ then } \{y=1\} \text{ else } \{y=x+1\}$ voidaan antaa spesifikaatio $[\text{true}] \text{ Succ } [y==x+1]$.

Ohjelman osittainen oikeellisuus

► Olkoon P ohjelma sekä B_1 ja B_2 kaksi Boolean lauseketta.

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

- ohjelman P suoritus aloitetaan tilasta S , missä $S \models B_1$, ja
- ohjelman P *suoritus päättyy äärellisellä askelmäärällä* tilaan S' .

Esimerkki 15.15 Ohjelman osittainen oikeellisuus ei edellytä ohjelman suorituksen päättymistä:

$$\models_P [\text{true}] \text{ while}(x \neq y) \{z = x; x = y; y = z\} [x == y].$$

Esimerkki 15.15

Todistetaan

$$\models_P [(x == n) \&\& (y == m)] z = x; x = y; y = z [(x == m) \&\& (y == n)].$$

Kirjoitetaan todistus klassisen sekvenssin muotoon:

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

Päätelysäännöt osittaiselle oikeellisuudelle

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

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

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

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

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

Implikaatio: $\frac{\models_{\mathbb{Z}} B_1 \rightarrow B_2 \quad [B_2] C [B_3] \quad \models_{\mathbb{Z}} B_3 \rightarrow B_4}{[B_1] C [B_4]}$

Heikoimmat esiehdot

- Olkoon P ohjelma $C_1; \dots; C_n$, missä $C_1, \dots, C_n$ ovat järjestyksessä peräkkäin suoritettavat komennot.
- Spesifikaation $\models_P [B_0] P [B_n]$ todistaminen voidaan pilkkoa osaongelmiin: tulisi löytää $B_1, \dots, B_{n-1}$ siten, että $\models_P [B_{i-1}] C_i [B_i]$ kaikille $i \in \{1, \dots, n\}$.
- 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 $[B_0] C_1; [B_1] C_2; [B_2] \dots C_{n-1}; [B_{n-1}] C_n [B_n]$.

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 $B\{x/E\}$ eli $\models_p [B\{x/E\}] x=E [B]$.
 2. Jos $\models_p [B_1] C [B_2]$ on jo osoitettu ja B_0 on esiehdon B_1 vahvennus ($\models_{\mathbb{Z}} B_0 \rightarrow B_1$), 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 $\text{if}(B) \text{ then } \{C_1\} \text{ else } \{C_2\}$ heikoimmaksi esiehdoksi kirjataan $(B \& \& B_1) \mid \mid (!B \& \& B_2)$.

4. TOISTOLAUSEIDEN INVARIANTIT

- Ohjelmointikielten keskeisiä primitiivejä ovat myös toistolauseet, joiden avulla komentoja voidaan toistaa haluttu määrä.
- $$z=0; v=0; \text{while}(! (z==x)) \{z=z+1; v=v+y\}$$
- 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 $\text{while}(B) \{C\}$ invariantti I on mikä tahansa Boolean lauseke siten, että $\models_p [I \& \& B] C [I]$.

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]
```

Toistolauseen yleinen spesifikaatio

- Spesifikaation $\models_p [B_1] \text{while}(B) \{C\} [B_2]$ todistaminen voi perustua sopivan invariantin I käyttöön:
 1. $\models_{\mathbb{Z}} B_1 \rightarrow I$,
 2. $\models_{\mathbb{Z}} (I \& \& !B) \rightarrow B_2$, ja
 3. $\models_p [I] \text{while}(B) \{C\} [I \& \& !B]$.
- Kuten aiemminkin, oikeellisuustodistusta voi hakea *takaperin*:
 - A1. Valitaan sopiva invariantti I siten, että $\models_{\mathbb{Z}} (I \& \& !B) \rightarrow B_2$.
 - A2. Haetaan heikoin esiehto I' siten, että $\models_p [I'] C [I]$.
 - A3. Osoitetaan $\models_{\mathbb{Z}} I \& \& B \rightarrow I'$, minkä nojalla $\models_p [I \& \& B] C [I]$ ja edelleen $\models_p [I] \text{while}(B) \{C\} [I \& \& !B]$.
 - A4. Osoitetaan $\models_{\mathbb{Z}} B_1 \rightarrow I$.

Esimerkki 15.21

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

$$[\text{true}] [0 == 0 * y] \ z = 0; [0 == z * y] \ v = 0; [v == z * y]$$

```
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 $(x \geq 0)$.

Esimerkki 15.24

Todistetaan $\models_t [0 \leq x] \text{Multi } [v == x * y]$:

```
[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]
```

5. TÄYSI OIKEELLISUUS

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

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

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 tämän lisäksi ohjelman P *suoritus päättyy äärellisellä askelmäärällä*, jos $S \models B_1$ alkutilassa S .

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 $\text{while}(B) \{C\}$ esimerkiksi osittaisen oikeellisuuden $\models_p [I] \text{while}(B) \{C\} [I \& \& !B]$ tapauksessa.

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