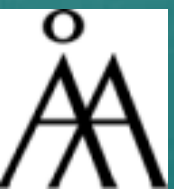
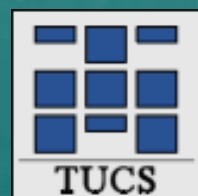
Matematiikan ohjelmointi

Joakim von Wright

Formaali menetelmä käytännössä

- miten todistetaan ohjelman oikeellisuus?
- miltä todistus näyttää?
- isot ohjelmat?
  - "miljoona riviä koodia"
- nykyajan ohjelmat?
  - rinnakkaisuus, vuorovaikutteisuus, ...





Todistamisen asema

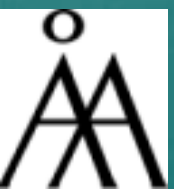
- todistaminen on formaalin menetelmän ydin
- mitä tarkoittaa "todistus"?
  - matematiikan epäformaali todistus
  - geometrian aksiomaattinen todistus
  - logiikan formaali todistus

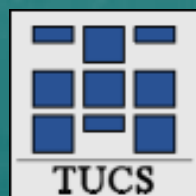
Epäformaali ja formaali todistus

- matematiikan kuva todustuksesta
  - vakuuttava
  - periaatteessa formalisoitavissa
  - havainnollinen (surveyable)
- täysin formaali todistus matematiikassa?
  - ei käytännöllistä

Miiksi formaali todistus?

- ohjelman oikeellisuustodistus on erilainen
  - todistuksen luotettavuus tärkeämpi kuin kauneus
  - todistus pitää saada heti oikeaksi
  - todistukset suuria...
  - ...mutta sisältävät paljon pieniä rutiinitodistuksia





Ohjelma matemaattisena oliona

- ohjelmointikielen näkökulma

```
i := 1
s := 0
while i <= n:
    s := s+i
    i := i+1
```

- algebrallinen näkökulma
  - i=1 ; s=0 ; while (i≤n) (s=s+i ; i=i+1)

Ohjelmointilogiikka

- ohjelmia varten tarvitaan ohjelmointilogiikka
  - tavoitteena oikeellisuuden todistaminen
  - pohjalla klassinen logiikka
  - lisäksi erikoispiirteet
    - muuttujan käsite erilainen kuin matematiikassa

Ohjelmointilogiikan perusideat

- oikeellisuusväittämä (ns Hoare-logiikka)

  {pre} program {post}
- "jos pre pätee alussa niin ohjelman suoritus päättyy tilassa jossa post pätee"
- eri ohjelmointirakenteille omat säännöt
- todistus jakautuu pienempiin osatodistuksiin
- pohjatason osatodistukset usein triviaaleja

Ohjelmointilogiikan säännöt

- sijoitussääntö

$$\frac{pre \rightarrow post[x:=E]}{\{pre\}\ x=E\ \{post\}}$$

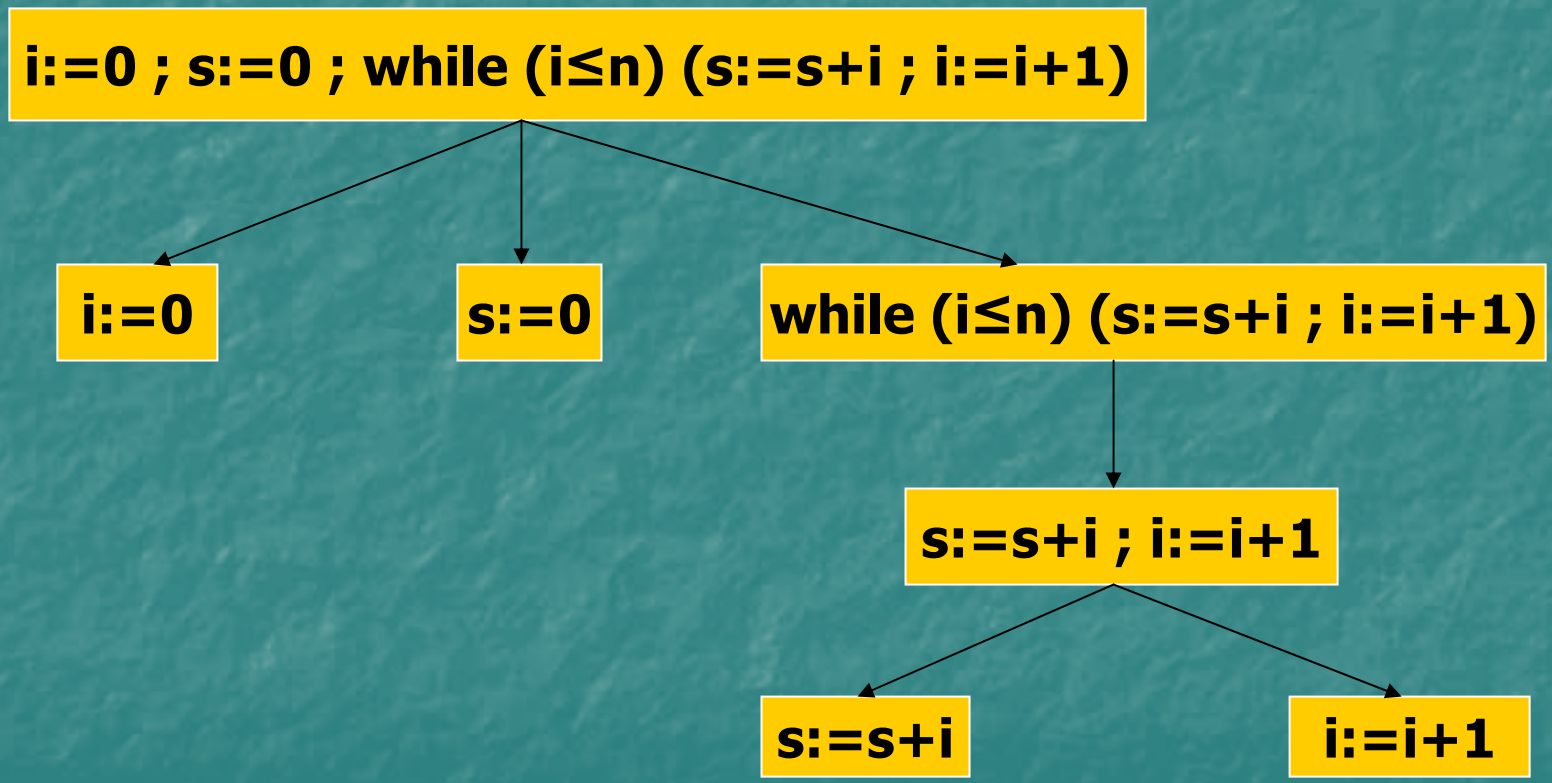
- sekvenssisääntö

$$\frac{\{pre\}\ S1\ \{mid\} \qquad \{mid\}\ S2\ \{post\}}{\{pre\}\ S1;S2\ \{post\}}$$

- toistosääntö

$$\frac{p \rightarrow inv \qquad \{C\ \&\ inv\}\ S\ \{inv\} \qquad inv\ \&\ \neg C \rightarrow q \qquad \{C\ \&\ inv\}\ S\ \{inv\}}{\{p\}\ while\ C\ S\ \{q\}}$$

Osatodistukset

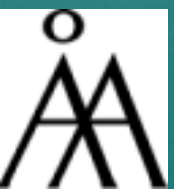


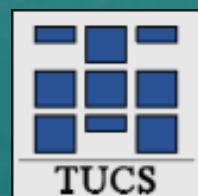
Oikeellisuustodistus

- todistettava:
  - {n≥0} s:=0;i:=0;while(i≤n)(s:=s+i;i:=i+1) {s=1+2+...+n}
- sekvenssisääntö: kolme osatodistusta
  - {n≥0} s:=0 {n≥0 & s=0}
  - {n≥0 & s=0} i:=0 {n≥0 & s=0 & i=0}
  - {n≥0 & s=0 & i=0} while(i≤n)(s:=s+i;i:=i+1) {s=1+2+...+n}
- väliin tulevat väiittämät täytyy keksiä itse

Osatodistus 1

- $\{n \geq 0\}\ s := 0\ \{n \geq 0 \ \&\ s = 0\}$
- sijoitussäännön avulla saadaan
  - $n \geq 0 \rightarrow n \geq 0 \ \&\ 0 = 0$
- peruslogiikka todistaa tämän
- ensimmäinen osatodistus hoidettu, rutiininomaisesti





Todistuksen esittäminen

- voiko todistus samalla olla
  - formaali ja
  - ihmiselle ymmmärrettävä?
- ryhmässämme kehitetty esitysmuoto: rakenteinen johto (structured derivation)

Rakenteisten johtojen selailu

- html-teknologia
  - todistusta voi katsoa webiltä
- todistusta voi selailla
  - osatodistus avataan: yksityiskohdat nakyvät
- todistusta voi kehittää

Rakenteisten johtojen käyttö

- sopiva matemattisten ratkaisujen esitysmuoto
- ryhmässämme kehitetään ratkaisumetodiikkaa joka käyttää logiikkaa ja rakenteisia johtoja
  - yhtälöiden ratkaiseminen
  - lausekkeiden sievennys
  - analyysi, ym
- testaus: yo-matematiikan tehtäviä

Koulumatematiikka-projekti

- yhteisprojekti ÅA - TY - Kupittaan lukio
- pitkän matematiikan ryhmä käyttää metodiikkaa läpi koko lukion (op. Mia Peltomäki)
- vertausryhmällä perinteinen opetus
- kokeilu aloitetty syksyllä 2001, alustavat tulokset lupaavia

Ohjelmointimetodiikka

- suuren ohjelman oikeaksi todistaminen jälkeenpäin (verifiointi) ei onnistu
- todistaminen on osa ohjelman rakentamista
- ryhmässämme kehitetään tarkennuskalkyyli
  - Dijkstran weakest precondition-käsitteeseen perustuva ohjelmointilogiikka ja -metodiikka
  - voimaakkaampi kuin Hoare-logiikka

Tarkennuskalkyyli

- ohjelmointiilogiikassa määritellään tarkennusrelaatio (refinement, $\leq$)
- $S1 \leq S2$ tarkoittaa että S2:lla on kaikki S1:n oikeellisuusominaisuudet
  - S2 on S1:n uusi/parannettu/laajennettu versio
- kalkyylin säännöt kertovat miten saadaan S2 kun tiedetään S1 ja toivottu lisäys

Tarkennusmetodiikka

Käsin todistamisen rajat

- käsin voidaan todistaa
  - yleisiä sääntöjä
  - pieniä esimerkkejä
  - erityisen vaativia yksityiskohtia
- nykypäivän haasteet
  - ohjelmat ovat suuria
  - rinnakkaisuus, vuorovaikutteisuus, jne...

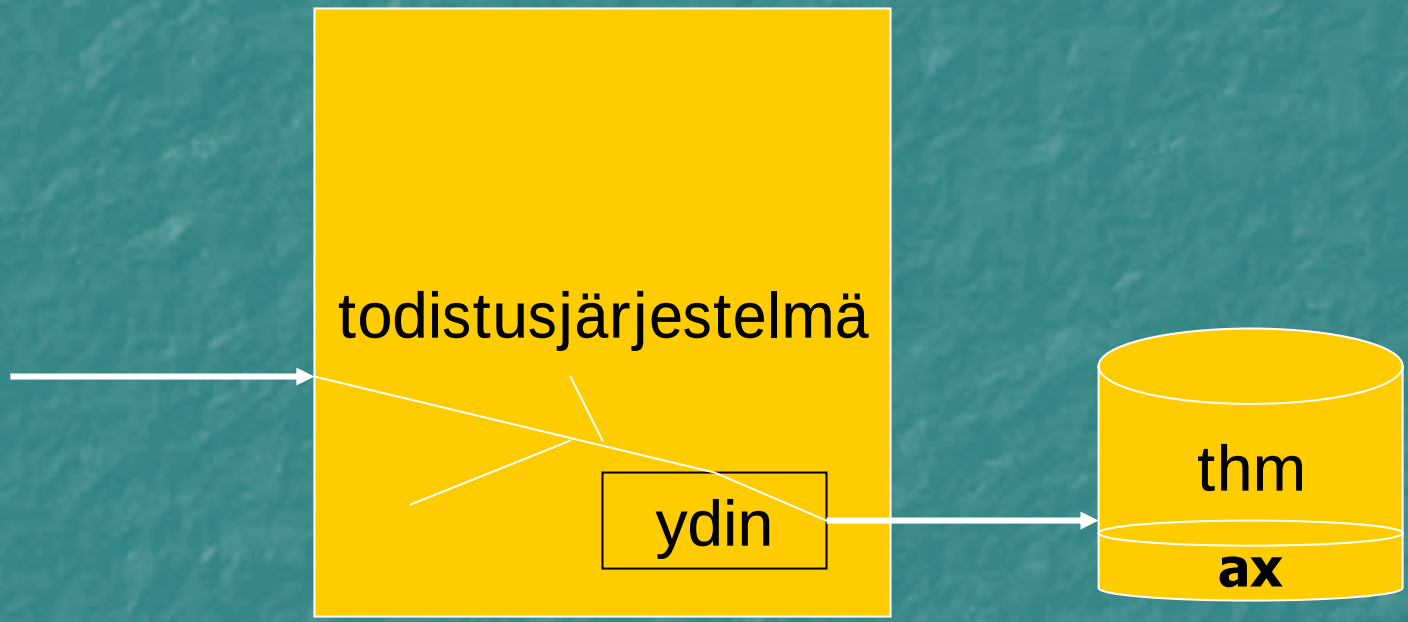
Todistaminen tietokoneella

- miten hallitaan suuret todistukset?
  - tarvitaan tietokoneen apua
- tietokoneen käyttö todistustyössä:
  - kertakäyttöohjelma (esim 4-väriteoreema, 1976)
  - todistusjärjestelmä (mekanisoitu logiikka)

Mikä on todistusjärjestelmä?

- interaktiivinen ohjelma joka manipuloi kaavoja
  - tietyt kaavat saavat statuksen "teoreema"
- pieni ydin: aksiomat + päättelysäännöt
- teoreema todistetaan aina formaalisti
- todistusstrategioita voi ohjelmoida
- teoreemoja voi säästää, käyttää, ...
- mahdollistaa todistusten teollista tuotantoa

Todistusjäärjestelmän rakenne



Mekanisoitu ohjelmointilogiikka

- HOL-järjestelmä sisältää peruslogiikan
  - $\forall x \bullet \exists y \bullet x = y$
- ohjelmointikäsitteet formalisoidaan
  - ; = ...
  - while = ...
- todustussäännöt todistetaan oikeiksi
- todistusstrategiat ohjelmoidaan

Mekanisoitu oikeellisuustodistus

- syötetään todistettava (goal)
  - HOL hyväksyy ja odottaa
- valitaan sääntö
  - HOL näyttää mitä pitää viellä todistaa (subgoals)
- tarvittaessa annetaan säännön argumentit
  - invariantti
- tämä interaktio vaatii osaamista

Kohti automatisoitua todistusta

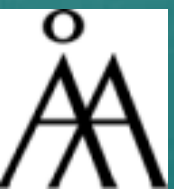
- tarkennuskalkyyli on mekanisoitu (HOL)
- inkrementaalinen ohjelmointi: osatodistukset sopivan kokoisia
- todistusta ohjaava tieto sijoitetaan ohjelmatekstiin: esim. invariantit
- HOL todistaa sen minkä voi ja näyttää mitä vielä tarvitsee todistaa

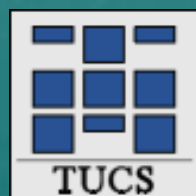
Nykyajan ohjelmistot

- rinnakkaisuus
  - useita pankiiautomaattia käytössä yhtaikaa
- vuorovaikutteisuus
  - käyttäjä ohjaa automaattia, valitsee toimintoja
- olio-ohjelmointii
- jne...

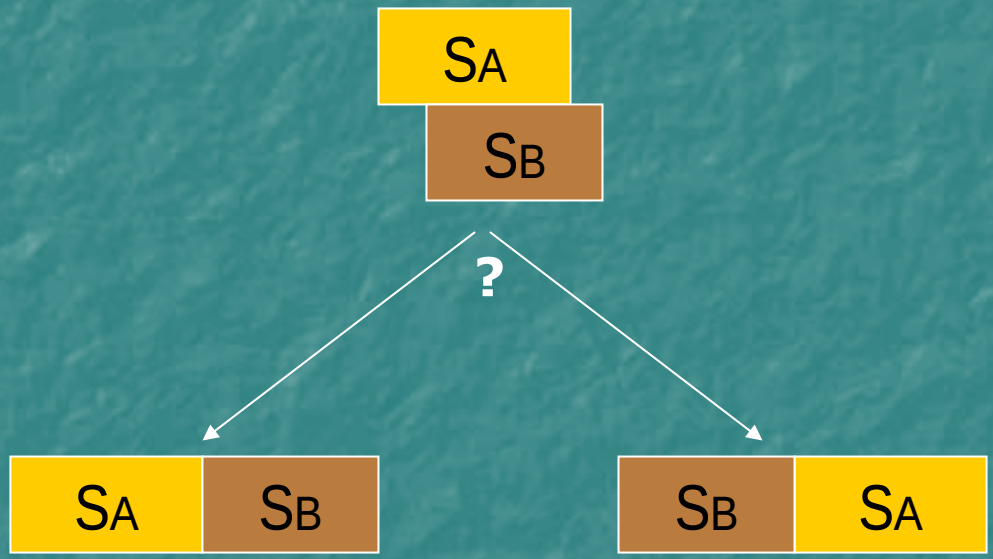
Rinnakkaisuus

- samanaikainen ohjelmansuoritus
- rinnakkaiset toiminnot tulkitaan toisensa jälkeen tapahtuviksi (jossain järjestyksessä)
- uusi käsite: "epämääräisyys" (nondeterminism)





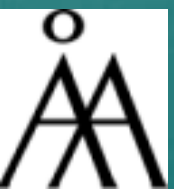
Rinnakkaisuus

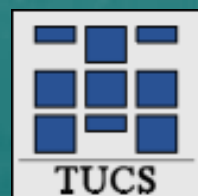


Rinnakkaisuus ohjelmointilogiikassa

- epämääräinen valintaoperaattori | (“tai”)
- rinnakkaisuutta ilmaistaan valinnalla

  $S_A;S_B \mid S_B;S_A$
- uudentyyppisiä todistusehtoja
- sopivasti laajennettu ohjelmointilogiikka toimii yhä





Vuorovaikutteisuus

- perinteinen ohjelma laskee tuloksen
  - suorituksen lopputulos riippuu vain lähtöarvoista
- nykypäivän ohjelma palvelee käyttäjiä
  - käyttäja voi olla ihminen, kone, toinen ohjelma...
- ohjelmointilogiikan täytyy huomioida eri osapuolia ja niiden tavoitteet, oikeudet, jne
- lisätään käsite "peli"

Vuorovaikutteisuus ohj.logiikassa

- oikeellisuusväittämä {pre} prog {post}
- klassinen tulkinta: ohjelma saavuttaa aina post
- pelitulkinta: tekemällä oikeat valinnat käyttäjä voi varmistaa sen että post saavutetaan
- sopivasti laajennettu ohjelmointilogiikka toimii yhä

Päätelmät

- virheettömiä ohjelmia voi rakentaa
- formaaleilla menetelmillä on tulevaisuus käytännön ohjelmistotyössä
- tavoitteena ohjelmointiympäristö jonka yhdessä valikossa on vaihtoehto “todista”