

# DD1350 Logik för dataloger

## Fö 13 – Hoare-logik

1

## Specifikation och verifiering

---

### Programspecifikation

- **formaliserar** programmets korrekthet
- relaterar tillståndsegenskaper
  - före och efter exekveringen

### Programverifiering

- **bevisar** programmets korrekthet relativt en specifikation

## Ett enkelt programspråk

---

Ett enkelt program:

```

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

```

## Programspråk: Syntax

---

- Aritmetiska uttryck

$$E ::= n \mid x \mid (E + E) \mid (E - E) \mid (E * E)$$

- Boolska uttryck

$$B ::= \mathbf{true} \mid \mathbf{false} \mid (E < E) \mid$$

$$(!B) \mid (B \& B) \mid (B \parallel B)$$

- Instruktioner (eng: **commands**)

$$C ::= x = E \mid C ; C \mid \mathbf{if} B \{C\} \mathbf{else} \{C\} \mid$$

$$\mathbf{while} B \{C\}$$

## Tillstånd och konfiguration

---

### Tillstånd (i denna kontext)

- beskriver värdena på variablerna i programmet
- jämför *omgivning* i predikatlogiken!

### Kontrollpunkt

- beskriver var vi är i exekveringen

Konfiguration = kontrollpunkt + tillstånd

## Tillståndsegenskaper

---

- Tillståndsegenskaper
  - kan beskrivas med predikatlogiska formler
  - kallas *påstående* (eng: assertion)  
t.ex.  $x > y$  eller  $\exists z (x = 2 * z)$
- Vi binder påståenden till kontrollpunkter i programmet

## Programspecifikation

---

- Med två påståenden:
  - ett förvillkor (eng: pre-condition)
  - ett eftervillkor (eng: post-condition)
- Formell notation: Hoare-tripletter

$$(\mid \phi \mid) P (\mid \psi \mid)$$

läs: **om** exekveringen av programmet  $P$  börjar i ett tillstånd där förvillkoret  $\phi$  är sant,

**så** slutar exekveringen i ett tillstånd där eftervillkoret  $\psi$  är sant.

## Exempel

---

- Fakultet-programmet *Fac1*

```

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

```

kan specificeras med Hoare-tripletten

$$(\mid x \geq 0 \mid) \text{Fac1} (\mid y = x! \mid)$$

## Partiell korrekthet

---

$\models_{\text{par}} (\phi) P (\psi)$  gäller om:

*om* exekveringen av programmet  $P$  börjar i ett tillstånd där förvillkoret  $\phi$  är sant,  
*och* exekveringen terminerar,  
*så* är eftervillkoret sant  $\psi$  i sluttillståndet

## Total korrekthet

---

$\models_{\text{tot}} (\phi) P (\psi)$  gäller om:

*om* exekveringen av programmet  $P$  börjar i ett tillstånd där förvillkoret  $\phi$  är sant,  
*så* terminerar exekveringen  
*och* eftervillkoret  $\psi$  är sant i sluttillståndet

## Programspecifikation

---

- Ska i ideella fallet:
  - fånga **vad** koden gör, inte **hur**
  - dvs separera **användningen** av koden (interface) från dess **implementation**
  - kunna fungera som ett *kontrakt* mellan användare (uppdragsgivare) och programmere

## Programspecifikation

---

Kan fakultet-programmet *Fac2*

```

y = 1;
while (x != 0) {
    y = y * x;
    x = x - 1;
}

```

specifieras med Hoare-tripletten

$$(\mid x \geq 0 \mid) \textit{Fac2} (\mid y = x! \mid)$$

## Logiska variabler

---

Vi vill uttrycka hur **slutvärdena** på variablerna relaterar till deras **startvärden**

På så sätt kan vi specificera programmet från användarens sida

Ett sätt är att införa extra variabler, s.k. **logiska variabler**, som betraktas som allkvantifierade

## Logiska variabler

---

Fakultet-programmet *Fac2*

```

y = 1;
while (x != 0) {
    y = y * x;
    x = x - 1;
}

```

kan specificeras med Hoare-tripletten

$$(\mid x \geq 0 \wedge x = x_0 \mid) \text{ Fac2 } (\mid y = x_0! \mid)$$

## Programspecifikation

---

Specifikationen beskriver hur programmet kan användas **utan** att känna till den konkreta implementationen!

$$(\mid x \geq 0 \wedge x = x_0 \mid) \text{ Fac2 } (\mid y = x_0! \mid)$$

- $x \geq 0$  fakultetsfunktionen är bara definierad för positiva tal
- $x = x_0$  fångar startvärdet på  $x$
- $y = x_0!$  relaterar slutvärdet på  $y$  till startvärdet på  $x$

## Programverifiering

---

### Med regler

- över Hoare-trippletter
- vi fokuserar på partiell korrekthet

### Bevis

- som bevisråd  
...eller som s.k. "tablåer"
- reducerar validiteten av Hoare-trippletter till validiteten av predikatlogiska formler över aritmetiken!



### Regler: Tilldelning

---

$$\frac{-}{(\vdash \psi[E/x] \vdash) \quad x = E \quad (\vdash \psi \vdash)}$$

### Regler: "Implied"

---

$$\frac{\vdash \phi' \rightarrow \phi \quad (\vdash \phi \vdash) \quad C \quad (\vdash \psi \vdash)}{(\vdash \phi' \vdash) \quad C \quad (\vdash \psi \vdash)}$$

$$\frac{(\vdash \phi \vdash) \quad C \quad (\vdash \psi \vdash) \quad \vdash \psi \rightarrow \psi'}{(\vdash \phi \vdash) \quad C \quad (\vdash \psi' \vdash)}$$

## Regler: Sammansättning

---

$$\frac{(\phi) C_1 (\eta) \quad (\eta) C_2 (\psi)}{(\phi) C_1 ; C_2 (\psi)}$$

inför ett mellanliggande påstående  $\eta$  i kontrollpunkten före  $C_2$

## Exempel

---

Programmet *Swap*

$\mathbf{z} = \mathbf{x};$

$\mathbf{x} = \mathbf{y};$

$\mathbf{y} = \mathbf{z};$

kan specificeras med

$$(\mathbf{x} = \mathbf{x}_0 \wedge \mathbf{y} = \mathbf{y}_0) \text{ Swap } (\mathbf{x} = \mathbf{y}_0 \wedge \mathbf{y} = \mathbf{x}_0)$$