DD1350 Logik för dataloger

Fö 13 – Hoare-logik

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

$$B ::= \texttt{true} \mid \texttt{false} \mid (E < E) \mid (!B) \mid (B \,\&\, B) \mid (B \,\|\, B)$$

- Instruktioner (eng: commands)

$$C ::= x = E \mid C\,\texttt{;}\,C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}$$

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

$$(|\ \phi\ |)\ \ P\ (|\ \psi\ |)$$

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

så slutar exekveringen i ett tillstånd där efter-villkoret ψ är sant.

Exempel

- Fakultet-programmet $Fac1$

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

kan specifieras med Hoare-tripletten

$$(|\ x \geq 0\ |)\ \ Fac1\ (|\ y = x!\ |)$$

Partiell korrekthet

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

om exekveringen av programmet P börjar i ett tillstånd där förvillkoret ϕ är sant,

och exekveringen terminerar,

så är eftervillkoret sant ψ i sluttillståndet

Total korrekthet

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

om exekveringen av programmet P börjar i ett tillstånd där förvillkoret ϕ är sant,

så terminerar exekveringen

och eftervillkoret ψ 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

$$(|\ x \geq 0\ |)\ \ Fac2\ \ (|\ y = x!\ |)$$

Logiska variabler

Vi vill utrycka hur **slut**värdena på variablerna relaterar till deras **start**värden

På så sätt kan vi specificera programmet från användarens sida

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

Logiska variabler

Fakultet-programmet $Fac2$

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

kan specifieras med Hoare-tripletten

$$(|\ x \geq 0 \wedge x = x_0\ |)\ \ Fac2\ \ (|\ y = x_0!\ |)$$

Programspecifikation

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

$$(|\, x \geq 0 \wedge x = x_0 \,|)\ \ Fac2\ \ (|\, y = x_0! \,|)$$

$x \geq 0$ 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 bevisträd

  …eller som s.k. ”tablåer”
- reducerar validiteten av Hoare-tripletter till validiteten av predikatlogiska formler över aritmetiken!

Regler: Tilldelning

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

Kom ihåg:

För en variabel x, en term t och en formel ϕ,

betecknar $\phi[t\ /\ x]$ formeln som är resultatet av substitutionen av alla fria förekomster av x i ϕ med t

Exempel:

$$\exists y\ (P(x,y) \wedge \exists x\ Q(x,y))\ [x+1/x]$$

$$= \exists y\ (P(x+1,y) \wedge \exists x\ Q(x,y))$$

Regler: ”Implied”

$$\frac{\vdash \phi' \rightarrow \phi \qquad (|\,\phi\,|)\ C\ (|\,\psi\,|)}{(|\,\phi'\,|)\ C\ (|\,\psi\,|)}$$

$$\frac{(|\,\phi\,|)\ C\ (|\,\psi\,|) \qquad \vdash \psi \rightarrow \psi'}{(|\,\phi\,|)\ C\ (|\,\psi'\,|)}$$

Regler: Sammansättning

$$\frac{(|\,\phi\,|)\ C_1\ (|\,\eta\,|) \qquad (|\,\eta\,|)\ C_2\ (|\,\psi\,|)}{(|\,\phi\,|)\ C_1\ ;\ C_2\ (|\,\psi\,|)}$$

inför ett mellanliggande påstående η i kontrollpunkten före C_2

Exempel

Programmet *Swap*

```
z = x;
x = y;
y = z;
```

kan specificeras med

$$(|\, x = x_0 \wedge y = y_0 \,|)\ \ Swap\ \ (|\, x = y_0 \wedge y = x_0 \,|)$$

Regler: If

$$\frac{(|\, \phi \wedge B \,|)\ C_1\ (|\, \psi \,|)\ \ (|\, \phi \wedge \neg B \,|)\ C_2\ (|\, \psi \,|)}{(|\, \phi \,|)\ \texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}\ (|\, \psi \,|)}$$

Exempel

Programmet Abs

```
if (x > 0) {
    y = x;
} else {
    y = -x;
}
```

kan specificeras med

$$(|\, x = x_0 \,|)\ Abs\ (|\, y = |\, x_0 \,| \,|)$$

Regler: Partial-while

$$\frac{(|\, \eta \wedge B \,|)\ C\ (|\, \eta \,|)}{(|\, \eta \,|)\ \texttt{while}\ B\ \{C\}\ (|\, \eta \wedge \neg B \,|)}$$

Slinginvariant (eng: loop invariant)