Korrekte Software: Grundlagen und Methoden
Vorlesung 6 vom 15.05.18: Invarianten und die Korrektheit des Floyd-Hoare-Kalküls

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2018

Fahrplan

- Einführung
- Operationale Semantik
- Denotationale Semantik
- Äquivalenz der Operationalen und Denotationalen Semantik
- Die Floyd-Hoare-Logik
- Invarianten und die Korrektheit des Floyd-Hoare-Kalküls
- Strukturierte Datentypen
- Modellierung und Spezifikation
- Verifikationsbedingungen
- Vorwärts mit Floyd und Hoare
- Funktionen und Prozeduren
- Referenzen
- Ausblick und Rückblick

Invarianten

Invarianten Finden: die Fakultät

```
p= 1;
c= 1;
while (c <= n) {
   p = p * c;
   c = c + 1;
}
```

Invariante:

$$p = (c - 1)!$$

- Kern der Invariante: Fakultät bis $c - 1$ berechnet.

Invarianten Finden: die Fakultät

```
p= 1;
c= 1;
while (c <= n) {
   p = p * c;
   c = c + 1;
}
```

Invariante:

$$p = (c-1)! \wedge c - 1 \leq n$$

- Kern der Invariante: Fakultät bis $c-1$ berechnet.
- Invariante impliziert Nachbedingung

Invarianten Finden: die Fakultät

```
p= 1;
c= 1;
while (c <= n) {
  p = p * c;
  c = c + 1;
}
```

Invariante:

$$p = (c-1)! \wedge c - 1 \leq n \wedge c > 0$$

- Kern der Invariante: Fakultät bis $c-1$ berechnet.
- Invariante impliziert Nachbedingung
- Nebenbedingung für Weakening innerhalb der Schleife.

Invarianten finden

1. Initiale Invariante: momentaner Zustand der Berechnung

2. Invariante und negierte Schleifenbedingung muss Nachbedingung implizieren; ggf. Invariante verstärken.

3. Beweise innerhalb der Schleife benötigen ggf. weiter Nebenbedingungen; Invariante verstärken.

Zählende Schleifen

- Fakultät ist Beispiel für zählende Schleife (**for**).
- Für Nachbedigung ψ ist Invariante:

$$\psi[i-1/n] \wedge i-1 \leq n$$

- Ggf. weitere Nebenbedingungen erforderlich

```
for (i= 0; i<= n; i++) {
  ...
}
```

ist syntaktischer Zucker für

```
i= 0;
while (i<= n) {
  ...
  i= i+1;
}
```

Beispiel 1

```
1  // {0 ≤ y}
2  x= 1;
3  c= 1;
4  while (c <= y) {
5    x= 2*x;
6    c= c+1;
7  }
8  // {x = 2^y}
```

- Invariante:

Beispiel 1

```
1  // {0 ≤ y}
2  x= 1;
3  c= 1;
4  while (c <= y) {
5    x= 2*x;
6    c= c+1;
7  }
8  // {x = 2^y}
```

- Invariante:

$$x = 2^{c-1}$$

Beispiel 1

```
1  // {0 ≤ y}
2  x= 1;
3  c= 1;
4  while (c <= y) {
5    x= 2*x;
6    c= c+1;
7  }
8  // {x = 2^y}
```

- Invariante:

$$x = 2^{c-1} \wedge c - 1 \leq y$$

Beispiel 2

```
1  // {0 ≤ y}
2  x= 1;
3  c= 1;
4  while (c < y) {
5    c= c+1;
6    x= 2*x;
7  }
8  // {x = 2^y}
```

- Invariante:

Beispiel 2

```
1  // {0 ≤ y}
2  x= 1;
3  c= 1;
4  while (c < y) {
5    c= c+1;
6    x= 2*x;
7  }
8  // {x = 2^y}
```

- Invariante:

$$x = 2^c$$

Beispiel 2

```
1  // {0 ≤ y}
2  x= 1;
3  c= 1;
4  while (c < y) {
5    c= c+1;
6    x= 2*x;
7  }
8  // {x = 2^y}
```

- Invariante:

$$x = 2^c \wedge c \leq y$$

Beispiel 2

```
1  // {0 ≤ y}
2  x= 1;
3  c= 0;
4  while (c < y) {
5    c= c+1;
6    x= 2*x;
7  }
8  // {x = 2^y}
```

- Invariante:

$$x = 2^c \wedge c \leq y$$

Beispiel 3

```
1  // {y = Y ∧ 0 ≤ y}
2  x= 1;
3  while (y != 0) {
4    x= 2*x;
5    y= y-1;
6  }
7  // {x = 2^Y}
```

- Invariante:

$$x = 2^{Y-y}$$

Beispiel 4

```
1  // {0 ≤ a}
2  r= a;
3  q= 0;
4  while (b <= r) {
5      r= r-b;
6      q= q+1;
7      }
8  // {a = b * q + r ∧ 0 ≤ r ∧ r < b}
```

Invariante:

Beispiel 4

```
1  // {0 ≤ a}
2  r= a;
3  q= 0;
4  while (b <= r) {
5      r= r-b;
6      q= q+1;
7      }
8  // {a = b * q + r ∧ 0 ≤ r ∧ r < b}
```

Invariante:

$$a = b \cdot q + r \wedge 0 \leq r$$

- Spezieller Fall: letzter Teil der Nachbedingung ist genau negierte Schleifeninvariante

Beispiel 5

```
1  // {0 ≤ a}
2  t= 1;
3  s= 1;
4  i= 0;
5  while (s <= a) {
6     t= t+ 2;
7     s= s+ t;
8     i= i+ 1;
9     }
10 // {i² ≤ a ∧ a < (i + 1)²}
```

- Was berechnet das?
  Ganzzahlige Wurzel von a.
- Invariante:

  $s-t \leq a \wedge t = 2 \cdot i+1 \wedge s = i^2+t$

- Nachbedingung 1: aus
  $s - t \leq a, s = i^2 + t$ folgt
  $i^2 \leq a$.
- Nachbedingung 2: aus
  $s = i^2 + t, t = 2 \cdot i + 1$ und
  $a < s$ folgt $a < (i + 1)^2$.

Korrektheit des Floyd-Hoare-Kalküls

Floyd-Hoare-Tripel: Gültigkeit und Herleitbarkeit

- Definition von letzter Woche: $P, Q \in \mathbf{Assn}, c \in \mathbf{Stmt}$

  $\models \{P\}\, c\, \{Q\}$ "Hoare-Tripel gilt" (semantisch)

  $\vdash \{P\}\, c\, \{Q\}$ "Hoare-Tripel herleitbar" (syntaktisch)

- **Frage**: $\vdash \{P\}\, c\, \{Q\} \overset{?}{\rightsquigarrow} \models \{P\}\, c\, \{Q\}$

Floyd-Hoare-Tripel: Gültigkeit und Herleitbarkeit

- Definition von letzter Woche: $P, Q \in \mathbf{Assn}, c \in \mathbf{Stmt}$

  $\models \{P\}\, c\, \{Q\}$ "Hoare-Tripel gilt" (semantisch)

  $\vdash \{P\}\, c\, \{Q\}$ "Hoare-Tripel herleitbar" (syntaktisch)

- **Frage**: $\vdash \{P\}\, c\, \{Q\} \overset{?}{\rightsquigarrow} \models \{P\}\, c\, \{Q\}$
- **Korrektheit**: $\vdash \{P\}\, c\, \{Q\} \overset{?}{\Longrightarrow} \models \{P\}\, c\, \{Q\}$
  - Wir können nur gültige Eigenschaften von Programmen herleiten.
- **Vollständigkeit**: $\models \{P\}\, c\, \{Q\} \overset{?}{\Longrightarrow} \vdash \{P\}\, c\, \{Q\}$
  - Wir können alle gültigen Eigenschaften auch herleiten.

Korrektheit des Floyd-Hoare-Kalküls

Der Floyd-Hoare-Kalkül ist korrekt.

Wenn $\vdash \{P\}\, c\, \{Q\}$, dann $\models \{P\}\, c\, \{Q\}$.

Beweis:

- Definition von $\models \{P\}\, c\, \{Q\}$:

$$\models \{P\}\, c\, \{Q\} \iff \forall I.\, \forall \sigma.\, \sigma \models^I P \land \exists \sigma'.\, (\sigma, \sigma') \in \mathcal{C}[\![c]\!] \implies \sigma' \models^I Q$$

- Beweis durch **Regelinduktion** über der **Herleitung** von $\vdash \{P\}\, c\, \{Q\}$.
- Bsp: Zuweisung, Sequenz, Weakening, While.
  - Zuweisung benötigt Lemma: $\sigma \models^I B[e/x] \iff \sigma[\mathcal{A}[\![e]\!](\sigma)/x] \models^I B$

Vollständigkeit der Floyd-Hoare-Logik

Floyd-Hoare-Logik ist vollständig modulo weakening.

Wenn $\models \{P\}\, c\, \{Q\}$, dann $\vdash \{P\}\, c\, \{Q\}$ bis auf die Bedingungen der Weakening-Regel.

- Beweis durch Konstruktion einer schwächsten Vorbedingung $\mathrm{wp}(c, Q)$.
  - Problemfall: while-Schleife.
- Wenn wir eine gültige Zusicherung nicht herleiten können, liegt das nur daran, dass wir eine Beweisverpflichtung nicht beweisen können.
- Logik erster Stufe ist unvollständig, also **können** wir gar nicht besser werden.

Zusammenfassung

- Invarianten finden in **drei Schritten**,
- Floyd-Hoare-Logik ist **korrekt**, wir können nur gültige Zusicherungen herleiten.
- Floyd-Hoare-Logik ist **vollständig** bis auf das Weakening.