Korrekte Software: Grundlagen und Methoden

Vorlesung 6 vom 28.05.20

Invarianten und die Korrektheit des Floyd-Hoare-Kalküls

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2020

Fahrplan

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

Überblick: die Regeln des Floyd-Hoare-Kalküls

$$\frac{}{\vdash \{P[e/x]\}\, x = e\, \{P\}}$$

$$\frac{\vdash \{A \wedge b\}\, c_0\, \{B\} \qquad \vdash \{A \wedge \neg b\}\, c_1\, \{B\}}{\vdash \{A\}\ \textbf{if}\ (b)\ c_0\ \textbf{else}\ c_1\, \{B\}}$$

$$\frac{\vdash \{A \wedge b\}\, c\, \{A\}}{\vdash \{A\}\ \textbf{while}(b)\ c\, \{A \wedge \neg b\}}$$

$$\frac{}{\vdash \{A\}\, \{\,\}\, \{A\}} \qquad \frac{\vdash \{A\}\, c_1\, \{B\} \qquad \vdash \{B\}\, c_2\, \{C\}}{\vdash \{A\}\, c_1; c_2\, \{C\}}$$

$$\frac{A' \Longrightarrow A \qquad \vdash \{A\}\, c\, \{B\} \qquad B \Longrightarrow B'}{\vdash \{A'\}\, c\, \{B'\}}$$

Invarianten

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 $p = n! = (c-1)!$
- Nebenbedingung für Weakening innerhalb der Schleife.
  - $c! = c * (c-1)!$ gilt nur für $c > 0$.

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 Nachbedingung $\psi[n]$ 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: Zählende Schleife

```
// {0 ≤ n}
x= 0;
c= 1;
while (c <= n) {
  x= x+c;
  c= c+1;
}
// {x = ∑_0^n}
```

- Invariante:

$$x = \sum_0^{c-1} \wedge c - 1 \leq n$$

Hierbei ist $\sum_a^b$ die Summe der Zahlen von a bis b, mit folgenden Eigenschaften:

$$\sum_0^0 = 0$$

$$a > 0 \Longrightarrow \sum_0^a = \sum_0^{a+1} + a$$

Beispiel 2: Variante der zählenden Schleife

```
// {0 ≤ y}
x= 0;
c= 0;
while (c < y) {
  c= c+1;
  x= x+c;
}
// {x = ∑_0^n}
```

- Invariante:

$$x = \sum_{0}^{c} \wedge 0 \leq c$$

- Kein C-Idiom
  - Startwert 0 wird ausgelassen

Beispiel 3: Andere Variante der zählenden Schleife

```
// {n = N ∧ 0 ≤ n}
x= 0;
while (n != 0) {
  x= x+n;
  n= n-1;
}
// {x = ∑_0^N}
```

- Invariante:

$$x = \sum_{n}^{N} \wedge n \leq N$$

Arbeitsblatt 6.1: Fakultät Revisited

Dieses Programm berechnet die Fakultat von n:

```
// {0 ≤ n ∧ n = N}
p= 1;
while (0< n) {
  p= p*n;
  n= n-1;
}
// {p = N!}
```

- Finden Sie eine Invariante.
- Beweisen Sie die Korrektheit.

Für die Invariante benötigen sie ein indiziertes Produkt (analog zur Summenfunktion):

$$\prod_{a}^{b} = a \cdot (a+1) \cdot \ldots \cdot b$$

Für das Produkt gelten folgende Eigenschaften:

$$a! = \prod_{1}^{a} \qquad a > b \Longrightarrow \prod_{a}^{b} = 1$$

$$a \leq b \Longrightarrow \prod_{a}^{b} = a \cdot \prod_{a+1}^{b}$$

Beispiel 4: Nicht-zählend (rekursiv)

```
// {0 ≤ a}
r= a;
q= 0;
while (b <= r) {
  r= r-b;
  q= q+1;
  }
// {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: Jetzt wird's kompliziert...

```
// {0 ≤ a}
t= 1;
s= 1;
i= 0;
while (s <= a) {
  t= t+ 2;
  s= s+ t;
  i= i+ 1;
  }
// ?{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:
  - $s{-}t \leq a, s = i^2{+}t \Longrightarrow i^2 \leq a$.
- Nachbedingung 2:
  - $s = i^2 + t, t = 2 \cdot i + 1 \Longrightarrow s = (i+1)^2$
  - $a < s, s = (i+1)^2 \Longrightarrow a < (i+1)^2$

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**: $\vdash \{P\}\,c\,\{Q\} \quad \stackrel{?}{\leadsto} \quad \models \{P\}\,c\,\{Q\}$
- **Korrektheit**: $\vdash \{P\}\,c\,\{Q\} \stackrel{?}{\Longrightarrow} \models \{P\}\,c\,\{Q\}$
  - Wir können nur gültige Eigenschaften von Programmen herleiten.
- **Vollständigkeit**: $\models \{P\}\,c\,\{Q\} \stackrel{?}{\Longrightarrow} \vdash \{P\}\,c\,\{Q\}$
  - 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 $\models \{P\}\,c\,\{Q\}$:

$$\models \{P\}\,c\,\{Q\} \Longleftrightarrow \forall I.\, \forall \sigma.\, \sigma \models^I P \wedge \exists \sigma'.\, (\sigma, \sigma') \in [\![c]\!]_{\mathcal{C}} \Longrightarrow \sigma' \models^I Q$$

- Beweis durch **Regelinduktion** über der **Herleitung** von $\vdash \{P\}\,c\,\{Q\}$.
- Bsp: Zuweisung, Sequenz, Weakening, While.
  - While-Schleife erfordert Induktion über Fixpunkt-Konstruktion

Arbeitsblatt 6.2: Korrektheit der Zuweisung

Beweisen Sie die Korrektheit der **Zuweisungsregel**:

$$\overline{\vdash \{P[e/x]\}\, x = e\, \{P\}}$$

1. Was genau ist zu zeigen?

2. Wir benötigen folgendes **Lemma**:

$$\sigma \models^I B[e/x] \Longleftrightarrow \sigma[[\![e]\!]_{\mathcal{A}}(\sigma)/x] \models^I B$$

Wie zeigen wir damit die Behauptung?

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 $wp(c, Q)$.
  - Problemfall: while-Schleife.

Vollständigkeitsbeweis

- Zu Zeigen:

$$\forall c \in \textbf{Stmt}. \forall Q \in \textbf{Assn}. \exists\, wp(c, Q). \forall I. \forall \sigma. \sigma \models^I wp(c, Q) \Rightarrow [\![c]\!]_{\mathcal{C}}\sigma \models^I Q$$

- Beweis per struktureller Induktion über c:
  - $c \equiv \{\}$: Wähle $wp(\{\}, Q) := Q$
  - $c \equiv X = a$: wähle $wp(X = a, Q) := Q[a/x]$
  - $c \equiv c_0; c_1$: Wähle $wp(c_0; c_1, Q) := wp(c_0, wp(c_1, Q))$
  - $c \equiv$ **if** $b\ c_0$ **else** c_1: Wähle $wp(c, Q) := (b \wedge wp(c_0, Q)) \vee (\neg b \wedge wp(c_1, Q))$
  - $c \equiv$ **while** $(b)\ c_0$: ??

Vollständigkeitsbeweis: while

- $c \equiv$ **while** $(b)\ c_0$:
  Wie müssen eine Formel finden ($wp($**while** $(b)\ c_0, Q)$) die alle σ charakterisiert, so dass

$$\begin{aligned} &\sigma \models^I wp(\textbf{while}\ (b)\ c_0, Q) \\ &\longleftrightarrow \forall k \geq 0 \forall \sigma_0, \ldots, \sigma_k. \quad \sigma = \sigma_0 \\ &\qquad \forall 0 \leq i < k.(\sigma_i \models^I b \wedge \underbrace{[\![c_0]\!]_{\mathcal{C}}\sigma_i = \sigma_{i+1}}_{c_0\ \text{terminiert auf}\ \sigma_i\ \text{in}\ \sigma_{i+1}} \\ &\qquad \sigma_k \models^I b \vee Q \end{aligned}$$

- Es gibt so eine Formel ausdrückbar in **Assn**, die im Wesentlichen darauf aufbaut, dass
  1. jede Sequenz an Werten, die die Programmvariablen $\overline{X}$ in b und c_0 annehmen, mittels einer Formel beschrieben werden kann (β-Prädikat)
  2. $wp(c_0, \overline{X} = \overline{\sigma_{i+1}(X)})$ die Formel beschreibt, was vor c_0 gelten muss, damit hinterher die Programmvariablen $\overline{X}$ die Werte $\overline{\sigma_{i+1}(X)}$ haben
  3. $\neg\, wp(c_0, \mathit{false})$ beschreibt was vor c_0 nicht gelten darf, damit c_0 nicht terminiert.

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 $wp(c, Q)$.
  - Problemfall: while-Schleife.
- Vollständigkeit (relativ):

$$\models \{P\}\, c\, \{Q\} \Leftrightarrow P \Rightarrow wp(c, Q)$$

- 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.