

Korrekte Software: Grundlagen und Methoden

Vorlesung 7 vom 22.05.24

Korrektheit des Floyd-Hoare-Kalküls

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2024

11:49-01 2024-07-04

1 [18]

erik U

Fahrplan

- Einführung
- Operationale Semantik
- Denotationale Semantik
- Äquivalenz der Operationalen und Denotationalen Semantik
- Der Floyd-Hoare-Kalkül
- Invarianten im Floyd-Hoare-Kalkül
- **Korrektheit des Floyd-Hoare-Kalküls**
- Strukturierte Datentypen
- Verifikationsbedingungen
- Funktionen und Prozeduren I
- Funktionen und Prozeduren II
- Referenzen
- Ausblick und Rückblick

Korrekte Software

2 [18]

erik U

Floyd-Hoare-Tripel: Gültigkeit und Herleitbarkeit

- Definition von letzter Woche: $P, Q \in \text{Assn}, c \in \text{Stmt}$
- $\models \{P\} c \{Q\}$ "Hoare-Tripel gilt" (semantisch)
- $\vdash \{P\} c \{Q\}$ "Hoare-Tripel herleitbar" (syntaktisch)
- **Frage:** $\vdash \{P\} c \{Q\} \stackrel{?}{\iff} \models \{P\} c \{Q\}$
- **Korrektheit:** $\vdash \{P\} c \{Q\} \stackrel{?}{\implies} \models \{P\} c \{Q\}$
 - Wir können nur gültige Eigenschaften von Programmen herleiten.
- **Vollständigkeit:** $\models \{P\} c \{Q\} \stackrel{?}{\implies} \vdash \{P\} c \{Q\}$
 - Wir können alle gültigen Eigenschaften auch herleiten.

Korrekte Software

3 [18]

erik U

Überblick: die Regeln des Floyd-Hoare-Kalküls

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

$$\frac{\vdash \{A \wedge b\} c_0 \{B\} \quad \vdash \{A \wedge \neg b\} c_1 \{B\}}{\vdash \{A\} \text{if } (b) \text{ c}_0 \text{ else } c_1 \{B\}}$$

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

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

$$\frac{A' \implies A \quad \vdash \{A\} c \{B\} \quad B \implies B'}{\vdash \{A'\} c \{B' \}}$$

Korrekte Software

4 [18]

erik U

Korrektheit des Floyd-Hoare-Kalküls

Der Floyd-Hoare-Kalkül ist korrekt.

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

Beweis:

- Definition von $\models \{P\} c \{Q\}$:
$$\models \{P\} c \{Q\} \iff \forall I. \forall \sigma. \sigma \models^I P \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c \rrbracket_c \implies \sigma' \models^I Q$$
- Beweis durch **Regelinduktion** über der **Herleitung** von $\vdash \{P\} c \{Q\}$.
- Bsp: Zuweisung, Sequenz, Weakening, While.
- While-Schleife erfordert Induktion über Fixpunkt-Konstruktion

Korrekte Software

5 [18]

erik U

Korrektheit der Zuweisung

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

Zu zeigen: $\models \{P[e/x]\} x = e \{P\}$

$$\iff \forall I. \forall \sigma. \sigma \models^I P[e/x] \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket x = e \rrbracket_c \implies \sigma' \models^I P$$

$$\iff \forall I. \forall \sigma. \sigma \models^I P[e/x] \implies \sigma([x \mapsto \llbracket e \rrbracket_A(\sigma)]) \models^I P$$

with $(\sigma, \sigma([x \mapsto \llbracket e \rrbracket_A(\sigma)])) \in \llbracket x = e \rrbracket_c$

Wir benötigen folgende **Lemmata** (Beweis durch strukturelle Induktion über B und a):

$$\sigma \models^I B[e/x] \iff \sigma[x \mapsto \llbracket e \rrbracket_A(\sigma)] \models^I B \quad (1)$$

$$\llbracket a[e/x] \rrbracket'_A(\sigma) = \llbracket a \rrbracket'_A(\sigma[x \mapsto \llbracket e \rrbracket_A(\sigma)]) \quad (2)$$

Korrekte Software

6 [18]

erik U

Arbeitsblatt 7.1: Substitution und Zustands-Update

$$\sigma \models^I B[e/x] \iff \sigma[x \mapsto \llbracket e \rrbracket_A(\sigma)] \models^I B \quad (3)$$

Beweis per struktureller Induktion über B . Zeigt die folgenden Fälle des Beweises:

- 1 Induktionsanfang: B ist $a_0 = a_1$
- 2 Induktionsschritt: B ist der Form $B_1 \wedge B_2$

Anmerkung:

- $\sigma \models^I B \iff \llbracket B \rrbracket'_B(\sigma) = \text{true}$
- $\llbracket a[e/y] \rrbracket'_A(\sigma) = \llbracket a \rrbracket'_A(\sigma[y \mapsto \llbracket e \rrbracket_A(\sigma)])$
- $\llbracket \cdot \rrbracket'_A$ ist strikt
- Falls für einen Ausdruck a $\llbracket a \rrbracket'_A$ undefiniert ist ($\llbracket a \rrbracket'_A = \perp$), dann ist $\sigma[x \mapsto \llbracket a \rrbracket'_A]$ auch undefiniert.
- $\llbracket \cdot \rrbracket'_B$ ist nicht strikt.

Korrekte Software

7 [18]

erik U

Korrektheit der Sequenzenregel

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

Induktions-Annahmen:

$$(A1) \vdash \{A\} c_1 \{B\} \iff \forall I. \forall \sigma. \sigma \models^I A \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c_1 \rrbracket'_c \implies \sigma' \models^I B$$

$$(A2) \vdash \{B\} c_2 \{C\} \iff \forall I. \forall \sigma. \sigma \models^I B \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c_2 \rrbracket'_c \implies \sigma' \models^I C$$

Zu zeigen:

$$\vdash \{A\} c_1; c_2 \{C\} \iff \forall I. \forall \sigma. \sigma \models^I A \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c_1; c_2 \rrbracket'_c \implies \sigma' \models^I C$$

$$(\sigma, \sigma') \in \llbracket c_1; c_2 \rrbracket'_c \iff (\sigma, \sigma') \in \llbracket c_1 \rrbracket'_c \circ \llbracket c_2 \rrbracket'_c$$

$$\iff \exists \rho. (\sigma, \rho) \in \llbracket c_1 \rrbracket'_c \wedge (\rho, \sigma') \in \llbracket c_2 \rrbracket'_c$$

Aus $\sigma \models^I A$ und $\exists \rho. (\sigma, \rho) \in \llbracket c_1 \rrbracket'_c$ folgt mit (A1) $\rho \models^I B$

Aus $\rho \models^I B$ und $\exists \sigma'. (\rho, \sigma') \in \llbracket c_2 \rrbracket'_c$ folgt mit (A2) $\sigma' \models^I C$ $\square$

Korrekte Software

8 [18]

erik U

Korrektheit der If-Then-Else-Regel

$$\frac{\vdash \{A \wedge b\} c_1 \{B\} \quad \vdash \{A \wedge \neg b\} c_2 \{B\}}{\vdash \{A\} \text{if } (b) \text{ c}_1 \text{ else c}_2 \{B\}}$$

Induktions-Annahmen:

$$\begin{aligned} (A1) \quad & \vdash \{A \wedge b\} c_1 \{B\} \iff \forall I. \forall \sigma. \sigma \models^I (A \wedge b) \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c_1 \rrbracket_c^I \implies \sigma' \models^I B \\ & \iff \forall I. \forall \sigma. (\sigma, true) \in \llbracket A \wedge b \rrbracket_B^I \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c_1 \rrbracket_c^I \implies \sigma' \models^I B \\ (A2) \quad & \vdash \{A \wedge \neg b\} c_2 \{B\} \iff \forall I. \forall \sigma. \sigma \models^I (A \wedge \neg b) \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c_2 \rrbracket_c^I \implies \sigma' \models^I B \\ & \iff \forall I. \forall \sigma. (\sigma, true) \in \llbracket (A \wedge \neg b) \rrbracket_B^I \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c_2 \rrbracket_c^I \implies \sigma' \models^I B \end{aligned}$$

Zu zeigen:

$$\vdash \{A\} \text{if } (b) \text{ c}_1 \text{ else c}_2 \{B\} \\ \iff \forall I. \forall \sigma. \sigma \models^I A \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket \text{if } (b) \text{ c}_1 \text{ else c}_2 \rrbracket_c^I \implies \sigma' \models^I B$$

Korrekte Software

9 [18]

etk U

Korrektheit der If-Then-Else-Regel

Zu zeigen:

$$\begin{aligned} & \vdash \{A\} \text{if } (b) \text{ c}_1 \text{ else c}_2 \{B\} \\ & \iff \forall I. \forall \sigma. \sigma \models^I A \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket \text{if } (b) \text{ c}_1 \text{ else c}_2 \rrbracket_c^I \implies \sigma' \models^I B \\ & \iff \forall I. \forall \sigma. (\sigma, true) \in \llbracket A \rrbracket_A^I \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket \text{if } (b) \text{ c}_1 \text{ else c}_2 \rrbracket_c^I \implies \sigma' \models^I B \end{aligned}$$

Folgt aus Definition

$$\begin{aligned} \llbracket \text{if } (b) \text{ c}_1 \text{ else c}_2 \rrbracket_c^I = & \{(\sigma, \sigma') \mid (\sigma, true) \in \llbracket b \rrbracket_B^I \wedge (\sigma, \sigma') \in \llbracket c_1 \rrbracket_c^I\} \\ & \cup \{(\sigma, \sigma') \mid (\sigma, false) \in \llbracket b \rrbracket_B^I \wedge (\sigma, \sigma') \in \llbracket c_2 \rrbracket_c^I\} \end{aligned}$$

mit (A1) und (A2)

$$\begin{aligned} (A1) \quad & \vdash \{A \wedge b\} c_1 \{B\} \iff \forall I. \forall \sigma. (\sigma, true) \in \llbracket A \wedge b \rrbracket_B^I \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c_1 \rrbracket_c^I \implies \sigma' \models^I B \\ (A2) \quad & \vdash \{A \wedge \neg b\} c_2 \{B\} \iff \forall I. \forall \sigma. (\sigma, true) \in \llbracket (A \wedge \neg b) \rrbracket_B^I \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c_2 \rrbracket_c^I \implies \sigma' \models^I B \end{aligned}$$

Korrekte Software

10 [18]

etk U

Korrektheit der While-Regel

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

Induktionsannahme:

$$\vdash \{A \wedge b\} c \{A\} \iff \forall I. \forall \sigma. \sigma \models^I (A \wedge b) \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c \rrbracket_c^I \implies \sigma' \models^I A$$

Zu zeigen:

$$\begin{aligned} \vdash \{A\} \text{while}(b) c \{A \wedge \neg b\} & \iff \forall I. \forall \sigma. \sigma \models^I A \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket \text{while}(b) c \rrbracket_c^I \\ & \implies \sigma' \models^I A \wedge \neg b \\ & \iff \forall I. \forall \sigma. \sigma \models^I A \wedge \exists \sigma'. (\sigma, \sigma') \in \bigcup_{n \geq 0} \Gamma^n(\emptyset) \\ & \implies \sigma' \models^I A \wedge \neg b \end{aligned}$$

Beweis per Induktion über n .

Korrekte Software

11 [18]

etk U

Arbeitsblatt 7.2: Unterbeweis Korrektheit der While-Schleife

$$\begin{aligned} \Gamma^0(\emptyset) &:= \emptyset \\ \Gamma^{n+1}(\emptyset) &:= \{(\sigma, \sigma') \mid \llbracket b \rrbracket_B \sigma = true \wedge (\sigma, \sigma') \in \llbracket c \rrbracket_c \circ \Gamma^n(\emptyset)\} \\ &\quad \cup \{(\sigma, \sigma') \mid \llbracket b \rrbracket_B \sigma = false\} \end{aligned}$$

Gegeben Induktionsannahme aus Regelinduktionsbeweis:

$$\vdash \{A \wedge b\} c \{A\} \iff \forall I. \forall \sigma. \sigma \models^I (A \wedge b) \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket c \rrbracket_c^I \implies \sigma' \models^I A$$

Zu zeigen: für alle $n \geq 0$:

$$\forall \sigma, \sigma'. \sigma \models^I A \wedge (\sigma, \sigma') \in \Gamma^n(\emptyset) \implies \sigma' \models^I A \wedge \neg b$$

Korrekte Software

12 [18]

etk U

Sequenzenregel

$$\frac{A' \implies A \quad \vdash \{A\} c \{B\} \quad B \implies B'}{\vdash \{A'\} c \{B'\}}$$

Induktionsannahme:

$$\begin{aligned} & \vdash \{A\} c \{B\} \iff \forall I. \forall \sigma. \sigma \models^I A \wedge (\sigma, \sigma') \in \llbracket c \rrbracket_c \implies \sigma' \models^I B \\ & \forall I. \forall \sigma. \sigma \models^I (A' \implies A) \iff \forall I. \forall \sigma. \sigma \models^I A' \implies \sigma \models^I A \\ & \forall I. \forall \sigma. \sigma \models^I (B \implies B') \iff \forall I. \forall \sigma. \sigma \models^I B \implies \sigma \models^I B' \end{aligned}$$

Zu zeigen:

$$\vdash \{A'\} c \{B'\} \iff \forall I. \forall \sigma. \sigma \models^I A' \wedge (\sigma, \sigma') \in \llbracket c \rrbracket_c \implies \sigma' \models^I B'$$

Korrekte Software

13 [18]

etk U

Vollständigkeit der Floyd-Hoare-Logik

Floyd-Hoare-Logik ist vollständig modulo weakening.

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

► Beweis durch Konstruktion einer schwächsten Vorbedingung $\text{wp}(c, Q)$.

► Problemfall: while-Schleife.

Korrekte Software

14 [18]

etk U

Vollständigkeitsbeweis

► Zu Zeigen:

$$\forall c \in \text{Stmt}. \forall Q \in \text{Assn}. \exists \text{wp}(c, Q). \forall I. \forall \sigma. \sigma \models^I \text{wp}(c, Q) \Rightarrow \llbracket c \rrbracket_c \sigma \models^I Q$$

► Beweis per struktureller Induktion über c :

- $c \equiv \{\}$: Wähle $\text{wp}(\{\}, Q) := Q$
- $c \equiv X = a$: Wähle $\text{wp}(X = a, Q) := Q[a/x]$
- $c \equiv c_0; c_1$: Wähle $\text{wp}(c_0; c_1, Q) := \text{wp}(c_0, \text{wp}(c_1, Q))$
- $c \equiv \text{if } b \text{ c}_0 \text{ else c}_1$: Wähle $\text{wp}(c, Q) := (b \wedge \text{wp}(c_0, Q)) \vee (\neg b \wedge \text{wp}(c_1, Q))$
- $c \equiv \text{while } (b) \text{ c}_0$: ??

Korrekte Software

15 [18]

etk U

Vollständigkeitsbeweis: while

- $c \equiv \text{while } (b) \text{ c}_0$:
Wie müssen eine Formel finden ($\text{wp}(\text{while } (b) \text{ c}_0, Q)$) die alle σ charakterisiert, so dass $\sigma \models^I \text{wp}(\text{while } (b) \text{ c}_0, Q)$
 $\iff \forall k \geq 0 \forall \sigma_0, \dots, \sigma_k. \quad \sigma = \sigma_0$

$$\forall 0 \leq i < k. (\sigma_i \models^I b \wedge \underbrace{\llbracket c_0 \rrbracket_c \sigma_i = \sigma_{i+1}}_{c_0 \text{ terminiert auf } \sigma_i \text{ in } \sigma_{i+1}})$$

$$\sigma_k \models^I b \vee Q$$

- Es gibt so eine Formel ausdrückbar in **Assn**, die im Wesentlichen darauf aufbaut, dass
 - 1 jede Sequenz an Werten, die die Programmvariablen $\bar{X} = \text{Vars}(b) \cup \text{Vars}(c_0)$ in jeder Iteration $(\sigma_0, \dots)$ beim Test b haben, mittels einer Formel beschrieben werden kann (β -Prädikat)
 - 2 $\text{wp}(c_0, \bar{X} = \overline{\sigma_{i+1}(X)})$ die Formel beschreibt, was vor c_0 gelten muss, damit hinterher die Programmvariablen $\bar{X}$ die Werte $\sigma_{i+1}(X)$ haben
 - 3 $\neg \text{wp}(c_0, false)$ beschreibt was vor c_0 nicht gelten darf, damit c_0 nicht terminiert.

Korrekte Software

16 [18]

etk U

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 $\text{wp}(c, Q)$.
 - ▶ Problemfall: while-Schleife.
- ▶ Vollständigkeit (relativ):

$$\models \{P\} c \{Q\} \iff (P \implies \text{wp}(c, Q))$$

- ▶ 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

- ▶ Die Floyd-Hoare-Logik ist **korrekt**, wir können nur gültige Zusicherungen herleiten.
- ▶ Beweis durch Struktur über der Ableitung: wir beweisen jede Regel als korrekt.
- ▶ Die Floyd-Hoare-Logik ist **vollständig** bis auf das Weakening.