Korrekte Software: Grundlagen und Methoden
Vorlesung 5 vom 03.05.18: Die Floyd-Hoare-Logik

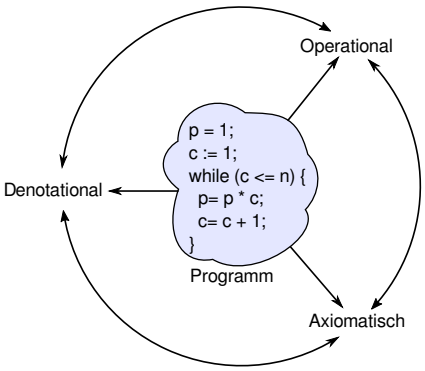
Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2018

Fahrplan

Drei Semantiken — Eine Sicht



Floyd-Hoare-Logik: Idee

- Was wird hier berechnet? $p = n!$
- Warum? Wie können wir das **beweisen**?
- Wir berechnen symbolisch, welche Werte Variablen über den Programmverlauf annehmen.

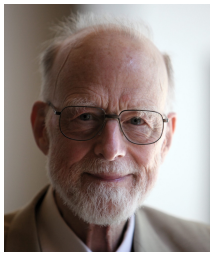
```
p= 1;
c= 1;
while (c <= n) {
  p = p * c;
  c = c + 1;
}
```

- Operationale/denotionale Semantik nicht für **Korrektheitsbeweise** geeignet: Ausdrücke werden zu groß, skaliert nicht.
- **Abstraktion** nötig.
- Grundidee: **Zusicherungen** über den Zustand an bestimmten Punkten im Programmablauf.

Bob Floyd und Tony Hoare

Bildquelle: Stanford University

Robert Floyd
1936 – 2001



Bildquelle: Wikipedia

Sir Anthony Charles Richard Hoare
* 1934

Grundbausteine der Floyd-Hoare-Logik

- **Zusicherungen** über den Zustand
- Beispiele:
  - (B): Hier gilt $p = c = 1$
  - (D): Hier ist c ist um eines größer als der Wert von c an Punkt (C)
- Gesamtaussage: Wenn am Punkt(A) der Wert von $n \geq 0$, dann ist am Punkt (E) $p = n!$.

```
// (A)
p= 1;
c= 1;
// (B)
while (c <= n) {
  p= p * c;
  // (C)
  c= c + 1;
  // (D)
}
// (E)
```


Grundbausteine der Floyd-Hoare-Logik

- **Logische Variablen** (zustandsfrei) und **Programmvariablen**
- **Zusicherungen** mit logischen und Programmvariablen
- **Floyd-Hoare-Tripel** $\{P\}\, c\, \{Q\}$
  - Vorbedingung P (Zusicherung)
  - Programm c
  - Nachbedingung Q (Zusicherung)
- Floyd-Hoare-Logik abstrahiert Programme durch logische Formeln.

Zusicherungen (Assertions)

- Erweiterung von **Aexp** and **Bexp** durch
  - **Logische** Variablen **Var** $v := N, M, L, U, V, X, Y, Z$
  - Definierte Funktionen und Prädikate über **Aexp** $n!, \sum_{i=1}^{n} i, \ldots$
  - Implikation und Quantoren $b_1 \Rightarrow b_2$, **forall** $v.\, b$, **exists** $v.\, b$
- Formal:

$$\begin{aligned} \textbf{Aexpv}\ a ::= {} & \mathbf{Z} \mid \mathbf{Idt} \mid \mathbf{Var} \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2 \\ & \mid f(e_1, \ldots, e_n) \\ \textbf{Assn}\ b ::= {} & \mathbf{1} \mid \mathbf{0} \mid a_1 == a_2 \mid a_1 ! = a_2 \mid a_1 <= a_2 \\ & \mid ! b \mid b_1 \,\&\&\, b2 \mid b_1 \,||\, b_2 \\ & \mid b_1 \Rightarrow b_2 \mid p(e_1, \ldots, e_n) \mid \textbf{forall}\ v;\ b \mid \textbf{exists}\ v;\ b \end{aligned}$$

Erfüllung von Zusicherungen

- Wann gilt eine Zusicherung $b \in \mathbf{Assn}$ in einem Zustand σ?
  - Auswertung (denotationale Semantik) ergibt *true*
  - **Aber:** was ist mit den logischen Variablen?
- **Belegung** der logischen Variablen: $I : \mathbf{Var} \to \mathbf{T}$
- Semantik von b unter der Belegung I: $\mathcal{B}_v[\![b]\!]^I, \mathcal{A}_v[\![a]\!]^I$

Erfülltheit von Zusicherungen

$b \in \mathbf{Assn}$ ist in Zustand σ mit Belegung I erfüllt ($\sigma \models^I b$), gdw

$$\mathcal{B}_v[\![b]\!]^I(\sigma) = \mathit{true}$$

Floyd-Hoare-Tripel

Partielle Korrektheit ($\models \{P\}\, c\, \{Q\}$)

c ist **partiell korrekt**, wenn für alle Zustände σ, die P erfüllen, gilt: **wenn** die Ausführung von c mit σ in σ' terminiert, **dann** erfüllt σ' Q.

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

- Gleiche Belegung der logischen Variablen für P und Q.

Totale Korrektheit ($\models [P]\, c\, [Q]$)

c ist **total korrekt**, wenn für alle Zustande σ, die P erfüllen, die Ausführung von c mit σ in σ' terminiert, und σ' erfüllt Q.

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

- Folgendes **gilt**: $\models \{\mathbf{1}\}\ \mathbf{while}(1)\{\ \}\ \{\mathbf{1}\}$
- Folgendes gilt **nicht**: $\models [\mathbf{1}]\ \mathbf{while}(1)\{\ \}\ [\mathbf{1}]$

Regeln der Floyd-Hoare-Logik

- Die Floyd-Hoare-Logik erlaubt es, Zusicherungen der Form $\vdash \{P\}\, c\, \{Q\}$ syntaktisch **herzuleiten**.
- Der **Kalkül** der Logik besteht aus sechs Regeln der Form

$$\frac{\vdash \{P_1\}\, c_1\, \{Q_1\} \ldots \vdash \{P_n\}\, c_n\, \{Q_n\}}{\vdash \{P\}\, c\, \{Q\}}$$

- Für jedes Konstrukt der Programmiersprache gibt es eine Regel.

Regeln des Floyd-Hoare-Kalküls: Zuweisung

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

- Eine Zuweisung x=e ändert den Zustand so dass an der Stelle x jetzt der Wert von e steht. Damit **nachher** das Prädikat P gilt, muss also **vorher** das Prädikat gelten, wenn wir x durch e ersetzen.
- Es ist völlig normal (aber dennoch falsch) zu denken, die Substitution gehöre eigentlich in die Nachbedingung.
- Beispiele:

$\{5 < 10 \iff (x < 10)[5/x]\}$
x = 5
$\{x < 10\}$

$\{x < 9 \iff x + 1 < 10\}$
x = x+ 1
$\{x < 10\}$

Regeln des Floyd-Hoare-Kalküls: Fallunterscheidung

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

- In der Vorbedingung des **if**-Zweiges gilt die Bedingung b, und im **else**-Zweig gilt die Negation $\neg b$.
- Beide Zweige müssem mit derselben Nachbedingung enden.

Regeln des Floyd-Hoare-Kalküls: Iteration

$$\frac{\vdash \{A \land b\}\, c\, \{A\}}{\vdash \{A\}\ \mathbf{while}(b)\ c\, \{A \land \neg b\}}$$

- Iteration korrespondiert zu **Induktion**.
- Bei (natürlicher) Induktion zeigen wir, dass die **gleiche** Eigenschaft P für 0 gilt, und dass wenn sie für $P(n)$ gilt, daraus folgt, dass sie für $P(n+1)$ gilt.
- Analog dazu benötigen wir hier eine **Invariante** A, die sowohl **vor** als auch **nach** dem Schleifenrumpf gilt.
- In der Vorbedingung des Schleifenrumpfes können wir die Schleifenbedingung b annehmen.
- Die **Vorbedingung** der **Schleife** ist die Invariante A, und die **Nachbedingung** der **Schleife** ist A und die Negation der Schleifenbedingung b.

Regeln des Floyd-Hoare-Kalküls: Sequenzierung

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

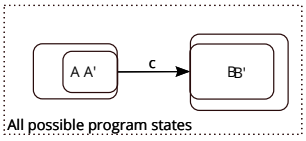
- Hier wird eine Zwischenzusicherung B benötigt.

$$\frac{}{\vdash \{A\}\, \{\}\, \{A\}}$$

- Trivial.

Regeln des Floyd-Hoare-Kalküls: Weakening

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



- $\models \{A\}\, c\, \{B\}$: Ausführung von c startet in Zustand, in dem A gilt, und endet (ggf) in Zustand, in dem B gilt.
- Zustandsprädikate beschreiben Mengen von Zuständen: $P \subseteq Q$ gdw. $P \implies Q$.
- Wir können A zu A' einschränken ($A' \subseteq A$ oder $A' \implies A$), oder B zu B' vergrößern ($B \subseteq B'$ oder $B \implies B'$), und erhalten $\models \{A'\}\, c\, \{B'\}$.

Überblick: die Regeln des Floyd-Hoare-Kalküls

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

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

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

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

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

Einfache Beispiele

Sei p:

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

Zu zeigen:

- p vertauscht x und y
- $\vdash \{x = X \wedge y = Y\}$ p $\{x = Y \wedge y = X\}$

Sei q:

```
if (x < y) {
  z = x;
  } else {
  z= y;
  }
```

Zu zeigen:

- q berechnet in z das Minimum von x und y
- $\vdash \{true\}$ q $\{z \leq x \wedge z \leq y\}$

Wie wir Floyd-Hoare-Beweise aufschreiben

```
// {P}
// {P1}
x= e;
// {P2}
// {P3}
while (x< n) {
    // {P3 ∧ x < n}
    // {P4}
    z= a;
    // {P3}
    }
// {P3 ∧ ¬(x < n)}
// {Q}
```

- Beispiel zeigt: $\vdash \{P\}\, c\, \{Q\}$
- Programm wird mit gültigen Zusicherungen annotiert.
- Vor einer Zeile steht die Vorbedingung, danach die Nachbedingung.
- Implizite Anwendung der Sequenzenregel.
- Weakening wird notiert durch mehrere Zusicherungen, und muss **bewiesen** werden.
  - Im Beispiel: $P \Longrightarrow P_1$, $P_2 \Longrightarrow P_3$, $P_3 \wedge x < n \Longrightarrow P_4$, $P_3 \wedge \neg(x < n) \Longrightarrow Q$.

Das einfache Beispiel in neuer Notation

```
// {x = X ∧ y = Y}
// {y = Y ∧ x = X}
z= x;
// {y = Y ∧ z = X}
x= y;
// {x = Y ∧ z = X}
y= z;
// {x = Y ∧ y = X}
```


Das Fakultätsbeispiel

```
// {1 = 0! ∧ ∧0 ≤ n}
// {1 = (1 − 1)! ∧ 1 ≤ 1 ∧ 1 − 1 ≤ n}
p= 1;
// {p = (1 − 1)! ∧ 1 ≤ 1 ∧ 1 − 1 ≤ n}
c= 1;
// {p = (c − 1)! ∧ 1 ≤ c ∧ c − 1 ≤ n}
while (c<= n) {
  // {p = (c − 1)! ∧ 1 ≤ c ∧ c − 1 ≤ n ∧ c ≤ n}
  // {p ∗ c = (c − 1)! ∗ c ∧ 1 ≤ c ∧ c ≤ n}
  // {p ∗ c = c! ∧ 1 ≤ c ∧ c ≤ n}
  // {p ∗ c = ((c + 1) − 1)! ∧ 1 ≤ c + 1 ∧ (c + 1) − 1 ≤ n}
  p= p∗c;
  // {p = ((c + 1) − 1)! ∧ 1 ≤ c + 1 ∧ (c + 1) − 1 ≤ n}
  c= c+1;
  // {p = (c − 1)! ∧ 1 ≤ c ∧ c − 1 ≤ n}
  }
// {p = (c − 1)! ∧ 1 ≤ c ∧ c − 1 ≤ n ∧ ¬(c ≤ n)}
// {p = (c − 1)! ∧ 1 ≤ c ∧ c − 1 ≤ n ∧ c > n}
// {p = (c − 1)! ∧ 1 ≤ c ∧ c − 1 = n}
// {p = n!}
```


Das Fakultätsbeispiel (Quellcode)

```
/** { 1 == factorial(0) && 0 <= n } */
/** { 1 == factorial(1- 1) && 1 <= 1 && 1-1 <= n } */
p= 1;
/** { p == factorial(1- 1) && 1 <= 1 && 1-1 <= n } */
c= 1;
/** { p == factorial(c- 1) && 1 <= c && c-1 <= n } */
while (c<= n) {
  /** { p == factorial(c-1) && 1<= c && c-1 <= n && c <= n } */
  /** { p*c == factorial(c-1)* c && 1 <= c && c <= n } */
  /** { p*c == factorial(c) && 1 <= c && c <= n } */
  /** { p*c == factorial((c+1)- 1) && 1 <= c+1 && (c+1)-1 <= n } */
  p= p*c;
  /** { p == factorial((c+1)-1) && 1<= c+1 && (c+1)-1 <= n } */
  c= c+1;
  /** { p == factorial(c-1) && 1 <= c && c-1 <= n } */
  }
/** { p == factorial(c-1) && 1<= c && c-1 <= n && factorial(c <= n) } *
/** { p == factorial(c-1) && 1<= c && c-1 <= n && c > n } */
/** { p == factorial(c-1) && 1<= c && c-1 == n } */
/** { p == factorial(n) } */
```


Zusammenfassung Floyd-Hoare-Logik

- Die Logik abstrahiert über konkrete Systemzustände durch Zusicherungen (Hoare-Tripel $\{P\}\, c\, \{Q\}$).
- Zusicherungen sind boolsche Ausdrücke, angereichert durch logische Variablen.
- Semantische **Gültigkeit** von Hoare-Tripeln: $\models \{P\}\, c\, \{Q\}$.
- Syntaktische **Herleitbarkeit** von Hoare-Tripeln: $\vdash \{P\}\, c\, \{Q\}$
- Zuweisungen werden durch Substitution modelliert, d.h. die Menge der gültigen Aussagen ändert sich.
- Für Iterationen wird eine **Invariante** benötigt (die **nicht** hergeleitet werden kann).