Korrekte Software: Grundlagen und Methoden
Vorlesung 2 vom 10.04.16: Die Floyd-Hoare-Logik

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2016

Fahrplan

- Einführung
- Die Floyd-Hoare-Logik
- Operationale Semantik
- Denotationale Semantik
- Äquivalenz der Semantiken
- Verifikation: Vorwärts oder Rückwärts?
- Korrektheit des Hoare-Kalküls
- Einführung in Isabelle/HOL
- Weitere Datentypen: Strukturen und Felder
- Funktionen und Prozeduren
- Referenzen und Zeiger
- Frame Conditions & Modification Clauses
- Ausblick und Rückblick

Idee

- Was wird hier berechnet?

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

Idee

- Was wird hier berechnet? $p = n!$
- Wie können wir das beweisen?

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

Idee

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

$\{1 \leq n\}$

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

$\{p = n!\}$

Idee

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

$\{1 \leq n\}$

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

$\{p = n!\}$

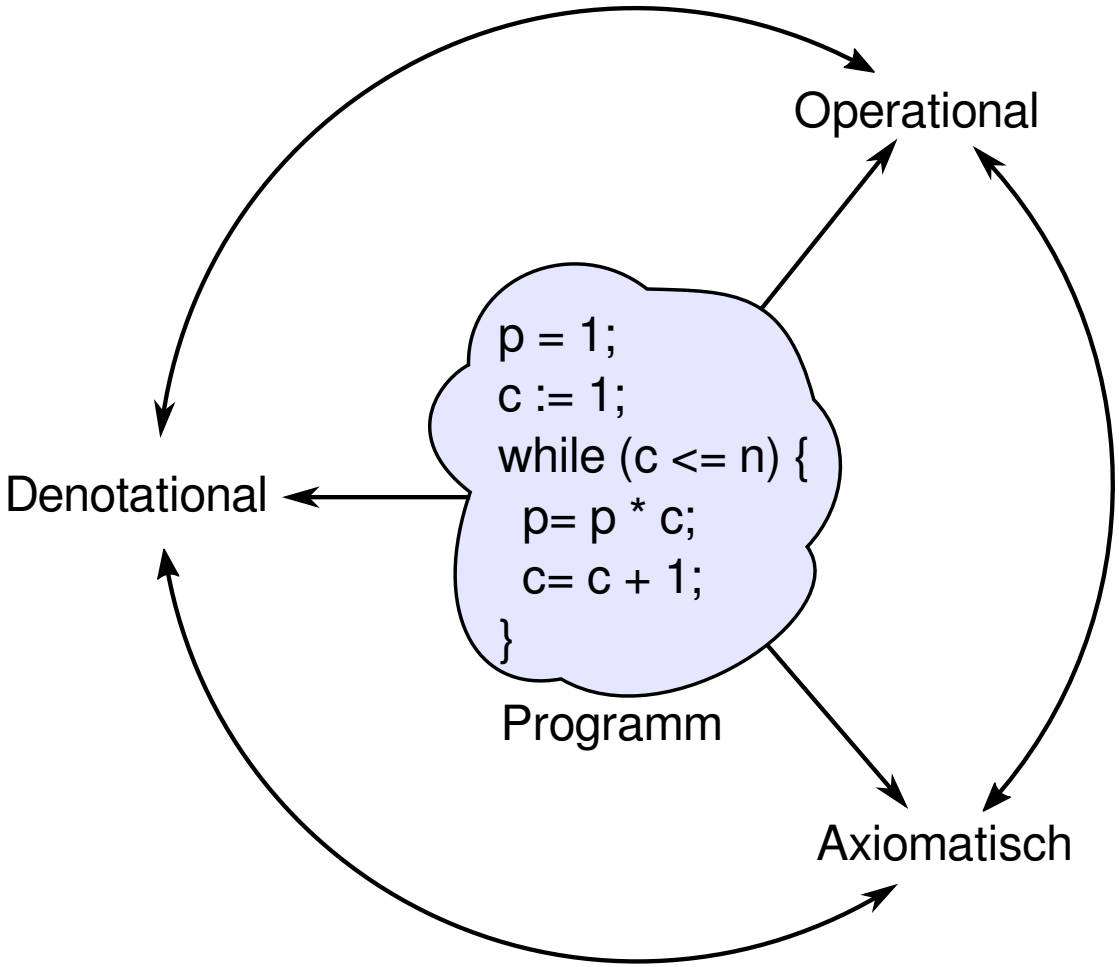
- Um Aussagen über ein Program zu beweisen, benötigen wir einen Formalismus (eine Logik), die es erlaubt, Zusicherungen über Werte von Variablen zu bestimmten Ausführungszeitpunkten (im Programm) aufzuschreiben und zu beweisen.
- Dazu müssen wir auch die Bedeutung (Semantik) des Programmes definieren — die Frage "Was tut das Programm" mathematisch exakt beantworten.

Semantik von Programmiersprachen

Drei wesentliche Möglichkeiten:

- Operationale Semantik beschreibt die Bedeutung eines Programmes, indem die Ausführung auf einer abstrakten Maschine beschrieben wird.
- Denotationale Semantik bildet jedes Programm auf ein mathematisches Objekt (meist ein partielle Funktion zwischen Systemzuständen) ab.
- Axiomatische Semantik beschreibt die Bedeutung eines Programmes durch Beweisregeln, mit welchem sich gültige Eigenschaften herleiten lassen. Das prominenteste Beispiel hierzu ist die Floyd-Hoare-Logik.

Drei Semantiken — Eine Sicht



- Jede Semantik ist eine Sicht auf das Program.
- Diese Semantiken sollten alle äquivalent sein. Wir müssen sie also in Beziehung setzen, und zeigen dass sie die gleiche Sicht ergeben.
- Für die axiomatische Semantik (die Floyd-Hoare-Logik) ist das die Frage der Korrektheit der Regeln.

Floyd-Hoare-Logik

- Grundbaustein der Floyd-Hoare-Logik sind Zusicherungen der Form $\{P\}\, c\, \{Q\}$ (Floyd-Hoare-Tripel), wobei P die Vorbedingung ist, c das Programm, und Q die Nachbedingung.
- Die Logik hat sowohl logische Variablen (zustandsfrei), und Programmvariablen (deren Wert sich über die Programmausführung ändert).
- Die Floyd-Hoare-Logik hat eine wesentliches Prinzip and einen Trick.
- Das Prinzip ist die Abstraktion vom Programmzustand durch eine logische Sprache; insbesondere wird die Zuweisung durch Substitution modelliert.
- Der Trick behandelt Schleifen: Iteration im Programm entspricht Rekursion in der Logik. Ein Beweis ist daher induktiv, und benötig eine Induktionsannahme — eine Invariante.

Unsere Programmiersprache

Wir betrachten einen Ausschnitt der Programmiersprache C (C0).

Ausbaustufe 1 kennt folgende Konstrukte:

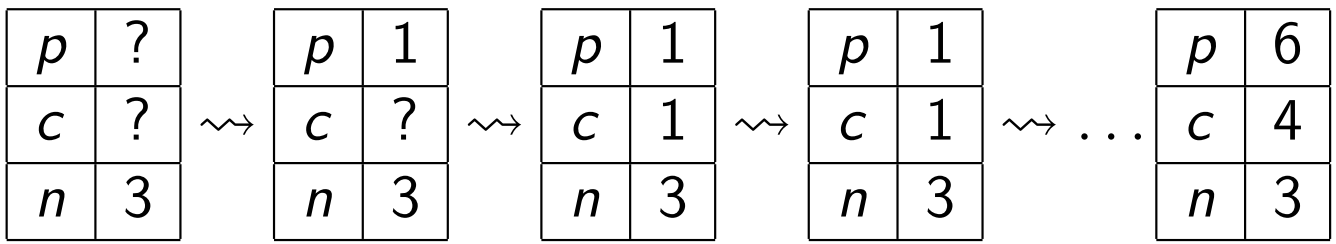
- Typen: **int**;
- Ausdrücke: Variablen, Literale (für ganze Zahlen), arithmetische Operatoren (für ganze Zahlen), Relationen (==, !=, <=, ...), boolsche Operatoren (&&, ||);
- Anweisungen:
  - Fallunterscheidung (**if**...**else**...), Iteration (**while**), Zuweisung, Blöcke;
  - Sequenzierung und leere Anweisung sind implizit

C0: Ausdrücke und Anweisungen

$$
\begin{array}{rl}
\text{Aexp} & a ::= \mathbf{N} \mid \mathbf{Loc} \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \mid a_1 / a_2 \\
\text{Bexp} & b ::= \mathbf{0} \mid \mathbf{1} \mid a_1 == a_2 \mid a_1 != a_2 \\
 & \quad \mid a_1 <= a_2 \mid !b \mid b_1 \,\&\&\, b2 \mid b_1 \,||\, b_2 \\
\text{Exp} & e := \mathbf{Aexp} \mid \mathbf{Bexp} \\
\text{Stmt} & c ::= \mathbf{Loc} = \mathbf{Exp}; \\
 & \quad \mid \textbf{if} \ (\ b\ )\ c_1\ \textbf{else}\ c_2 \\
 & \quad \mid \textbf{while}\ (\ b\ )\ c \\
 & \quad \mid \{c^*\}
\end{array}
$$

Semantik von C0

- Die (operationale) Semantik einer imperativen Sprache wie C0 ist ein Zustandsübergang: das System hat einen impliziten Zustand, der durch Zuweisung von Werten an Adressen geändert werden kann.
- Konkretes Beispiel: $n = 3$



Systemzustände

- Ausdrücke werten zu Werten **Val** (hier ganze Zahlen) aus.
- Adressen **Loc** sind hier Programmvariablen (Namen)
- Ein Systemzustand bildet Adressen auf Werte ab: $\Sigma = \mathbf{Loc} \rightharpoonup \mathbf{Val}$
- Ein Programm bildet einen Anfangszustand möglicherweise auf einen Endzustand ab (wenn es terminiert).
- Zusicherungen sind Prädikate über dem Systemzustand.

Floyd-Hoare-Tripel

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

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

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

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 der Floyd-Hoare-Logik: 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:

x = 5
$\{x < 10\}$

x = x+ 1

Regeln der Floyd-Hoare-Logik: 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:

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

x = x+ 1

Regeln der Floyd-Hoare-Logik: 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:

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

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

Regeln der Floyd-Hoare-Logik: 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:

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

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

Regeln der Floyd-Hoare-Logik: Fallunterscheidung und Sequenzierung

$$\frac{\vdash \{A \,\&\&\, [\![b]\!]\}\, c_0\, \{B\} \qquad \vdash \{A \,\&\&\, \neg[\![b]\!]\}\, c_1\, \{B\}}{\vdash \{A\}\ \textbf{if}\ (b)\ c_0\ \textbf{else}\ c_1\, \{B\}}$$

- 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.

$$\frac{\vdash \{A\}\, c\, \{B\} \qquad \vdash \{B\}\, \{c_s\}\, \{C\}}{\vdash \{A\}\, \{c\ c_s\}\, \{C\}}$$

- Hier wird ein Zwischenzustand B benötigt.

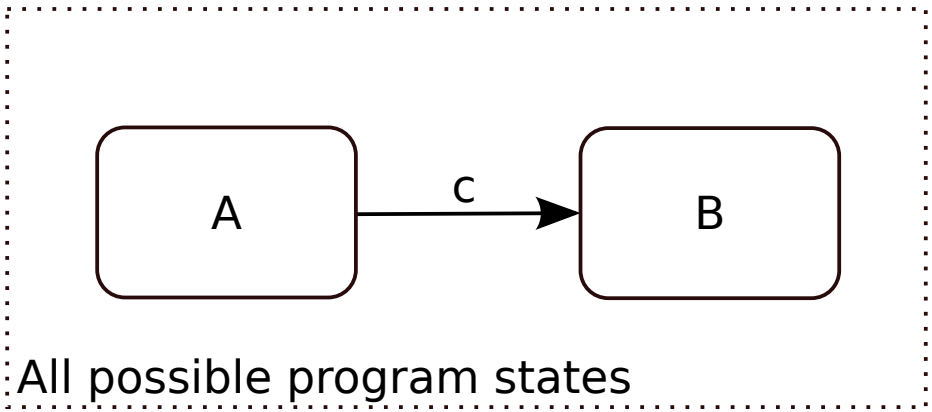
Regeln der Floyd-Hoare-Logik: Iteration

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

- 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 der Floyd-Hoare-Logik: Weakening

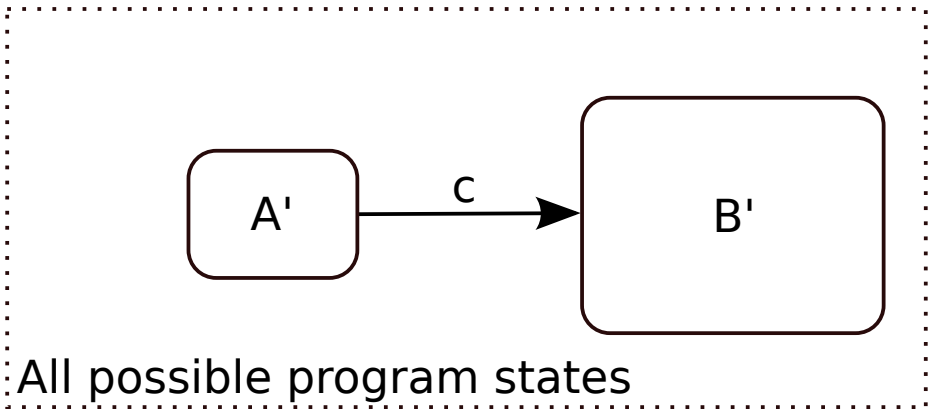
$$\frac{A' \longrightarrow A \qquad \vdash \{A\}\, c\, \{B\} \qquad B \longrightarrow B'}{\vdash \{A'\}\, c\, \{B'\}}$$



- $\models \{A\}\, c\, \{B\}$: 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. $P \longrightarrow Q$.

Regeln der Floyd-Hoare-Logik: Weakening

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



- $\models \{A\}\, c\, \{B\}$: 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. $P \longrightarrow Q$.
- Wir können A zu A' einschränken ($A' \subseteq A$ oder $A' \longrightarrow A$), oder B zu B' vergrößern ($B \subseteq B'$ oder $B \longrightarrow B'$), und erhalten $\models \{A'\}\, c\, \{B'\}$.

Überblick: die Regeln des Floyd-Hoare-Kalküls

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

$$\frac{}{\vdash \{A\}\, \{\}\, \{A\}} \qquad \frac{\vdash \{A\}\, c\, \{B\} \qquad \vdash \{B\}\, \{c_s\}\, \{C\}}{\vdash \{A\}\, \{c\ c_s\}\, \{C\}}$$

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

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

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

Eigenschaften der Floyd-Hoare-Logik

Korrektheit

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

- Wenn wir eine Korrektheitsaussage herleiten können, dann gilt sie auch.
- Wird gezeigt, indem wir $\models \{P\}\, c\, \{Q\}$ durch die anderen Semantiken definieren, und zeigen, dass alle Regeln diese Gültigkeit erhalten.

Relative Vollständigkeit

Wenn $\models \{P\}\, c\, \{Q\}$, dann $\vdash \{P\}\, c\, \{Q\}$ (bis auf Weakening)

- Wenn eine Korrektheitsaussage nicht beweisen werden kann (aber sie stimmt), dann liegt das immer daran, dass eine logische Aussage (in einer Anwendung der Weakening-Regelx) nicht bewiesen werden kann.
- Das ist zu erwarten: alle interessanten Logiken sind unvollständig.

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: $P \longrightarrow P_1$, $P_2 \longrightarrow P_3$, $P_3 \wedge x < n \longrightarrow P_4$, $P_3 \wedge \neg(x < n) \longrightarrow Q$.

Warum Verifikation?

Hier sind Varianten des Fakultätsbeispiels.
Welche sind korrekt?

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

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

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

Eine Handvoll Beispiele

```
// {y = Y}
x= 1;
while (y != 0) {
 y= y-1;
 x= 2*x;
}
// {x = 2^Y}
```

```
// {a ≥ 0 ∧ b ≥ 0}
r= b;
q= 0;
while (b <= r) {
   r= r-y;
   q= q+1;
   }
// {a = b * q + r ∧ r < b}
```

```
// {0 ≤ a}
t= 1;
s= 1;
i= 0;
while (s <= a) {
   t= t+ 2;
   s= s+ t;
   i= i+ 1;
   }
// {i^2 ≤ a ∧ a < (i + 1)^2}
```

Eine Handvoll Beispiele

```
// {y = Y ∧ y ≥ 0}
x= 1;
while (y != 0) {
 y= y-1;
 x= 2*x;
}
// {x = 2^Y}
```

```
// {a ≥ 0 ∧ b ≥ 0}
r= b;
q= 0;
while (b <= r) {
   r= r-y;
   q= q+1;
   }
// {a = b * q + r ∧ r < b}
```

```
// {0 ≤ a}
t= 1;
s= 1;
i= 0;
while (s <= a) {
   t= t+ 2;
   s= s+ t;
   i= i+ 1;
   }
// {i^2 ≤ a ∧ a < (i + 1)^2}
```

Zusammenfassung

- Floyd-Hoare-Logik zusammengefasst:
  - Die Logik abstrahiert über konkrete Systemzustände durch Zusicherungen (Hoare-Tripel $\models \{P\}\, c\, \{Q\}$).
  - Zusicherungen sind boolsche Ausdrücke, angereichert durch logische Variablen und Programmvariablen.
  - Wir können partielle Korrektheitsaussgen der Form $\models \{P\}\, c\, \{Q\}$ herleiten (oder totale, $\models [P]\, c\, [Q]$).
  - 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).
- Die Korrektheit hängt sehr davon ab, wie exakt wir die Semantik der Programmiersprache beschreiben können.