Neues Kapitel

Überblick

3. Fehler finden und vermeiden
   - Contract Programming und der Hoare-Kalkül
   - Debugging
   - Testen

Hoare-Tripel

Ein *Hoare-Tripel* ist $\langle \phi \rangle\ P\ \langle \psi \rangle$, wobei

- ϕ ist die *Vorbedingung*,
- P ist ein Programm,
- ψ ist die *Nachbedingung*.

Falls immer, wenn die Vorbedingung vor dem Ausführen von P wahr ist, die Nachbedingung nach dem Ausführen von P wahr ist, dann ist das Tripel *korrekt*.

Hoare-Tripel

Beispiele

- $\langle x = 5 \rangle\ x = x * x;\ \langle x = 25 \rangle$
- $\langle x = 5 \wedge i \geq 4 \rangle$
  while$(i > 0)\ \{x = x - x * x; i = i - 1;\}$
  $\langle x \leq -176820 \rangle$
- $\langle x = 5 \rangle$ **if**$(y > 0)\ \{x = x + y;\}\ \langle x \geq 4 \rangle$
- $\langle x = 3 \rangle$ **while**$(x > 0)\ \{x = x + 1;\}\ \langle x = 10 \rangle$

Vorsicht: Nur *nach* dem Ausführen des Programms muß die Nachbedingung erfüllt sein.

Obige Hoare-Tripel sind alle korrekt.

Hoare-Tripel

Ein größeres Beispiel: Fibonacci-Zahlen

$\langle n \geq 0 \rangle$
int $a = 0;$
int $b = 1;$
int $k = n;$
while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}
$\langle a = F_n \rangle$

Wir können die Korrektheit durch Induktion beweisen.
Der *Hoare-Kalkül* dient zum mechanischen Herleiten von korrekten Hoare-Tripeln.
Er besteht aus verschiedenen *Schlußregeln*.

1. $\dfrac{}{\langle \phi \rangle\ \{\}\ \langle \phi \rangle}$ — leere Anweisung
2. $\dfrac{}{\langle \phi[x/t] \rangle\ x = t;\ \langle \phi \rangle}$ — Zuweisungsregel
3. $\dfrac{\langle \phi \rangle\ P\ \langle \psi \rangle}{\langle \alpha \rangle\ P\ \langle \psi \rangle}\ \alpha \rightarrow \phi$ — Konsequenzregel 1
4. $\dfrac{\langle \phi \rangle\ P\ \langle \psi \rangle}{\langle \phi \rangle\ P\ \langle \beta \rangle}\ \psi \rightarrow \beta$ — Konsequenzregel 2
5. $\dfrac{\langle \phi \rangle\ P\ \langle \psi \rangle \qquad \langle \psi \rangle\ Q\ \langle \beta \rangle}{\langle \phi \rangle\ P; Q\ \langle \beta \rangle}$ — Sequenzregel
6. $\dfrac{\langle \phi \wedge B \rangle\ P\ \langle \psi \rangle}{\langle \phi \rangle\ \textbf{if}(B)\ \{P\}\ \langle \psi \rangle}\ \phi \wedge \neg B \rightarrow \psi$ — Bedingungsregel 1
7. $\dfrac{\langle \phi \wedge B \rangle\ P\ \langle \psi \rangle \qquad \langle \phi \wedge \neg B \rangle\ Q\ \langle \psi \rangle}{\langle \phi \rangle\ \textbf{if}(B)\ \{P\}\ \textbf{else}\ \{Q\}\ \langle \psi \rangle}$ — Bedingungsregel 2
8. $\dfrac{\langle \phi \wedge B \rangle\ P\ \langle \phi \rangle}{\langle \phi \rangle\ \textbf{while}(B)\ \{P\}\ \langle \phi \wedge \neg B \rangle}$ — Schleifenregel

Herleitung des Fibonacci-Programms

Wir verwenden zweimal die Zuweisungsregel:

$$\frac{}{\langle 0 = 0 \wedge 1 = 1 \rangle \ \textbf{int}\ a = 0; \ \langle a = 0 \wedge 1 = 1 \rangle}$$

$$\frac{}{\langle a = 0 \wedge 1 = 1 \rangle \ \textbf{int}\ b = 1; \ \langle a = 0 \wedge b = 1 \rangle}$$

Und zweimal Konsequenzregeln (eigentlich unnötig):

$$\frac{\langle 0 = 0 \wedge 1 = 1 \rangle \ \textbf{int}\ a = 0; \ \langle a = 0 \wedge 1 = 1 \rangle}{\langle 0 = 0 \rangle \ \textbf{int}\ a = 0; \ \langle a = 0 \rangle}$$

$$\frac{\langle a = 0 \wedge 1 = 1 \rangle \ \textbf{int}\ b = 1; \ \langle a = 0 \wedge b = 1 \rangle}{\langle a = 0 \rangle \ \textbf{int}\ b = 1; \ \langle a = 0 \wedge b = 1 \rangle}$$

Und zweimal Konsequenzregeln (Wiederholung):

$$\frac{\langle 0 = 0 \land 1 = 1 \rangle \ \textbf{int}\ a = 0;\ \langle a = 0 \land 1 = 1 \rangle}{\langle 0 = 0 \rangle \ \textbf{int}\ a = 0;\ \langle a = 0 \rangle}$$

$$\frac{\langle a = 0 \land 1 = 1 \rangle \ \textbf{int}\ b = 1;\ \langle a = 0 \land b = 1 \rangle}{\langle a = 0 \rangle \ \textbf{int}\ b = 1;\ \langle a = 0 \land b = 1 \rangle}$$

Jetzt kommt die Sequenzregel:

$$\frac{\langle 0 = 0 \rangle \ \textbf{int}\ a = 0;\ \langle a = 0 \rangle \qquad \langle a = 0 \rangle \ \textbf{int}\ b = 1;\ \langle a = 0 \land b = 1 \rangle}{\langle 0 = 0 \rangle \ \textbf{int}\ a = 0; \textbf{int}\ b = 1;\ \langle a = 0 \land b = 1 \rangle}$$

Dies ergibt das erste Zwischenergebnis T_1:

$$T_1 \equiv \boxed{\langle 0 = 0 \rangle \ \textbf{int}\ a = 0; \textbf{int}\ b = 1;\ \langle a = 0 \land b = 1 \rangle}$$

Weiter mit Zuweisung und Konsequenz:

$$\frac{}{\langle a = 0 \wedge b = 1 \wedge n = n \rangle \ \textbf{int}\ k = n; \ \langle a = 0 \wedge b = 1 \wedge k = n \rangle}$$

$$\frac{\langle a = 0 \wedge b = 1 \wedge n = n \rangle \ \textbf{int}\ k = n; \ \langle a = 0 \wedge b = 1 \wedge k = n \rangle}{\langle a = 0 \wedge b = 1 \rangle \ \textbf{int}\ k = n; \ \langle a = 0 \wedge b = 1 \wedge k = n \rangle}$$

Wir verwenden T_1 in der Sequenzregel:

$$\frac{T_1 \qquad \langle a = 0 \wedge b = 1 \rangle \ \textbf{int}\ k = n; \ \langle a = 0 \wedge b = 1 \wedge k = n \rangle}{\langle 0 = 0 \rangle \ \textbf{int}\ a = 0; \textbf{int}\ b = 1; \textbf{int}\ k = n; \ \langle a = 0 \wedge b = 1 \wedge k = n \rangle}$$

Dies sei unser zweites Zwischenergebnis T_2:

$$\boxed{\langle 0 = 0 \rangle \ \textbf{int}\ a = 0; \textbf{int}\ b = 1; \textbf{int}\ k = n; \ \langle a = 0 \wedge b = 1 \wedge k = n \rangle}$$

Etwas übersichtlicher schreibt sich T_2 so:

$\langle 0 = 0 \rangle$
int $a = 0;$
int $b = 1;$
int $k = n;$
$\langle a = 0 \wedge b = 1 \wedge k = n \rangle$

Die Vorbedingung ist „leer“, sie gilt immer.

Die Nachbedingung garantiert, daß die Variablen die richtigen Inhalte haben.

Wir versuchen nun, folgendes Hoare-Tripel herzuleiten:

$\langle a = 0 \wedge b = 1 \wedge k = n \wedge n \geq 0 \rangle$
while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}
$\langle a = F_n \rangle$

Zusammen mit T_2 und der Sequenzregel können wir dann fast das ganze Fibonacci-Programm herleiten. T_2 lautete:

$\langle 0 = 0 \rangle$ **int** $a = 0;$ **int** $b = 1;$ **int** $k = n;$ $\langle a = 0 \wedge b = 1 \wedge k = n \rangle$

Leider passen die zwei Teile nicht genau zusammen.

Statt

$$\langle 0 = 0 \rangle \ \textbf{int}\ a = 0; \textbf{int}\ b = 1; \textbf{int}\ k = n; \ \langle a = 0 \wedge b = 1 \wedge k = n \rangle$$

benötigen wir

$$\langle n \geq 0 \rangle \textbf{int}\ a = 0; \textbf{int}\ b = 1; \textbf{int}\ k = n; \langle a = 0 \wedge b = 1 \wedge k = n \wedge n \geq 0 \rangle.$$

Wenn so etwas passiert, kann man es oft so reparieren:

Wir gehen die ganze Herleitung von T_2 noch einmal durch und fügen $n \geq 0$ sowohl bei allen Vor- als auch Nachbedingungen ein.

Hier geht das ohne Problem.

Auf diese Weise erhalten wir das Zwischenergebnis T_3:

$\langle n \geq 0 \rangle$
int $a = 0;$
int $b = 1;$
int $k = n;$
$\langle a = 0 \wedge b = 1 \wedge k = n \wedge n \geq 0 \rangle$

und wir versuchen jetzt dies herzuleiten:

$\langle a = 0 \wedge b = 1 \wedge k = n \wedge n \geq 0 \rangle$
while$(k > 0)$ $\{$
$\quad b = a + b;$
$\quad a = b - a;$
$\quad k = k - 1;$
$\}$
$\langle a = F_n \rangle$

Mit einer Konsequenzregel erhalten wir:

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \rangle \equiv: \phi$

while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}

$\langle a = F_n \rangle$

$\langle a = 0 \wedge b = 1 \wedge k = n \wedge n \geq 0 \rangle \equiv: \psi$

while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}

$\langle a = F_n \rangle$

Denn $\psi \rightarrow \phi$. (Vorbedingung „aufgeweicht“.)

Wir müssen also noch herleiten:

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \rangle$
while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}
$\langle a = F_n \rangle$

Um dies zu schaffen, müssen wir die Schleifenregel verwenden:

$$\frac{\langle \phi \wedge B \rangle \ P \ \langle \phi \rangle}{\langle \phi \rangle \ \textbf{while}(B) \ \{P\} \ \langle \phi \wedge \neg B \rangle}$$

Wir wählen $a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0$ als ϕ und $k > 0$ als B.

Die Schleifenregel wird so angewendet:

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \wedge k > 0 \rangle$
$b = a + b;$
$a = b - a;$
$k = k - 1;$
$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \rangle$

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \rangle$
while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}
$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \wedge \neg(k > 0) \rangle$

Mit einer Abschwächung der Nachbedingung erhalten wir:

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \rangle$
while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}
$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \wedge \neg(k > 0) \rangle$

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \rangle$
while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}
$\langle a = F_n \rangle$

Was müssen jetzt noch dies herleiten:

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k > 0 \rangle$

$b = a + b;$

$a = b - a;$

$k = k - 1;$

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \rangle$

Mit der Zuweisungsregel erhalten wir diese Tripel:

$\langle a = F_{n-k} \wedge a + b = F_{n-k+2} \wedge k > 0 \rangle$

$b = a + b;$

$\langle a = F_{n-k} \wedge b = F_{n-k+2} \wedge k > 0 \rangle$ und

$\langle b - a = F_{n-k+1} \wedge b = F_{n-k+2} \wedge k > 0 \rangle$

$a = b - a;$

$\langle a = F_{n-k+1} \wedge b = F_{n-k+2} \wedge k > 0 \rangle$ und

$\langle a = F_{n-k+1} \wedge b = F_{n-k+1} \wedge k > 0 \rangle$

$k = k - 1;$

$\langle a = F_{n-k} \wedge b = F_{n-k+2} \wedge k + 1 > 0 \rangle$

Stellt man alles zusammen, haben wir dieses Tripel erzeugt:

$\langle n \geq 0 \rangle$
int $a = 0;$
int $b = 1;$
int $k = n;$
while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}
$\langle a = F_n \rangle$

Dies war eine lange Herleitung.

In Praxis oft zu schwierig oder teuer.

Übersichtliche Zusammenfassung

(1) Zuweisungsregel

$\langle n \geq 0 \rangle$ **int** $a = 0;$ $\langle n \geq 0 \wedge a = 0 \rangle$

(2) Zuweisungsregel

$\langle n \geq 0 \wedge a = 0 \rangle$ **int** $b = 1;$ $\langle n \geq 0 \wedge a = 0 \wedge b = 1 \rangle$

(3) Sequenzregel aus (1), (2)

$\langle n \geq 0 \rangle$ **int** $a = 0;$ **int** $b = 1;$ $\langle n \geq 0 \wedge a = 0 \wedge b = 1 \rangle$

(4) Zuweisungsregel

$\langle n \geq 0 \wedge a = 0 \wedge b = 1 \rangle$ **int** $k = n;$ $\langle n \geq 0 \wedge a = 0 \wedge b = 1 \wedge k = n \rangle$

(5) Sequenzregel aus (3), (4)

$\langle n \geq 0 \rangle$ **int** $a = 0;$ **int** $b = 1;$ **int** $k = n;$ $\langle n \geq 0 \wedge a = 0 \wedge b = 1 \wedge k = n \rangle$

(6) Zuweisungsregel

$\langle a = F_{n-k} \wedge a + b = F_{n-k+2} \wedge k > 0 \rangle$

$b = a + b;$

$\langle a = F_{n-k} \wedge b = F_{n-k+2} \wedge k > 0 \rangle$

(7) Zuweisungsregel

$\langle b - a = F_{n-k+1} \wedge b = F_{n-k+2} \wedge k > 0 \rangle$

$a = b - a;$

$\langle a = F_{n-k+1} \wedge b = F_{n-k+2} \wedge k > 0 \rangle$

(8) Zuweisungsregel

$\langle a = F_{n-k+1} \wedge b = F_{n-k+2} \wedge k > 0 \rangle$

$k = k - 1;$

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k + 1 > 0 \rangle$

(9) Sequenzregel aus (6), (7)

$\langle a = F_{n-k} \wedge a + b = F_{n-k+2} \wedge k > 0 \rangle$
$b = a + b;$
$a = b - a;$
$\langle a = F_{n-k+1} \wedge b = F_{n-k+2} \wedge k > 0 \rangle$

(10) Sequenzregel aus (9), (8)

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \wedge k > 0 \rangle$
$b = a + b;$
$a = b - a;$
$k = k - 1;$
$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \rangle$

(11) Schleifenregel aus (10)

$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \rangle$
while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}
$\langle a = F_{n-k} \wedge b = F_{n-k+1} \wedge k \geq 0 \wedge \neg(k > 0) \rangle$

(12) Konsequenzregel aus (11)

$\langle a = 0 \wedge b = 1 \wedge k = n \wedge n \geq 0 \rangle$
while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}
$\langle a = F_n \rangle$

(13) Sequenzregel aus (5), (12)

$\langle n \geq 0 \rangle$
int $a = 0;$
int $b = 1;$
int $k = n;$
while$(k > 0)$ {
  $b = a + b;$
  $a = b - a;$
  $k = k - 1;$
}
$\langle a = F_n \rangle$

Damit ist das gesamte Hoare-Tripel in dreizehn Schritten hergeleitet.

Vorsicht: Der Hoare-Kalkül geht von echten ganzen Zahlen aus und nicht von beschränkter Arithmetik.

Das folgende Tripel ist korrekt:

$\langle k \geq 0 \rangle$
while$(k >= 0)$ {
  $k = k + 1;$
}
$\langle k = 25 \rangle$

Das Tripel sagt, daß *nach* Verlassen der **while**-Schleife die Variable k den Wert 25 enthält.

Das ist tatsächlich wahr! (Denn die **while**-Schleife wird nie verlassen.)