Hilfsvariablen

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

(1) Zuweisungsregel

$\langle n \geq 0 \rangle\ k = n;\ \langle n \geq 0 \wedge k = n \rangle$

(2) Zuweisungsregel

$\langle n \geq 0 \wedge k = n \rangle\ z = 0;\ \langle n \geq 0 \wedge k = n \wedge z = 0 \rangle$

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

$\langle n \geq 0 \rangle\ k = n; z = 0;\ \langle n \geq 0 \wedge k = n \wedge z = 0 \rangle$

(4) Zuweisungsregel

$\langle z + k = n \wedge k > 0 \rangle\ z = z + 1;\ \langle z - 1 + k = n \wedge k > 0 \rangle$

(5) Zuweisungsregel

$\langle z + k - 1 = n \wedge k > 0 \rangle\ k = k - 1;\ \langle z + k = n \wedge k \geq 0 \rangle$

(6) Sequenzregel aus (4), (5)

$\langle z + k = n \wedge k > 0 \rangle\ z = z + 1; k = k - 1;\ \langle z + k = n \wedge k \geq 0 \rangle$

(7) Schleifenregel aus (6)

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

(8) Konsequenzregel aus (7)

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

(9) Sequenzregel aus (3), (8)

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

Wir konnten beweisen, daß am Ende $z = n$ gilt.

Wie steht es mit diesem Programm:

```
z = 0;
while(n > 0) {
  z = z + 1;
  n = n - 1;
}
```

Wie beweisen wir, daß z am Ende denselben Wert hat wie n am Anfang?

Dies scheint nicht ausdrückbar zu sein.

Hilfsvariablen

Lösung: Verwende eine Hilfsvariable.

$\langle n' = n \rangle$
$z = 0;$
while$(n > 0)$ {
  $z = z + 1;$
  $n = n - 1;$
}
$\langle z = n' \rangle$

Die Hilfsvariable n' kommt im Programm gar nicht vor.

Die Herleitung im Hoare-Kalkül geschieht analog zum letzten Beispiel.

$\langle true \rangle$
if$(x > y)$ {
  $r = x;$
}
else {
  $r = y;$
}
$\langle r = \max(x, y) \rangle$

(1) Zuweisungsregel

$\langle x = \max(x, y) \rangle\ r = x;\ \langle r = \max(x, y) \rangle$

(2) Konsequenzregel aus (1)

$\langle x > y \rangle\ r = x;\ \langle r = \max(x, y) \rangle$

(3) Zuweisungsregel

$\langle y = \max(x, y) \rangle\ r = y;\ \langle r = \max(x, y) \rangle$

(4) Konsequenzregel aus (3)

$\langle \neg(x > y) \rangle\ r = y;\ \langle r = \max(x, y) \rangle$

Die 2. Bedingungsregel lautet:

$$\frac{\langle \phi \wedge B \rangle\ P\ \langle \psi \rangle \qquad \langle \phi \wedge \neg B \rangle\ Q\ \langle \psi \rangle}{\langle \phi \rangle\ \textbf{if}(B)\ \{P\}\ \textbf{else}\ \{Q\}\ \langle \psi \rangle}$$

(5) Bedingungsregel aus (2), (4)

$\langle true \rangle$
if$(x > y)$ {
  $r = x;$
}
else {
  $r = y;$
}
$\langle r = \max(x, y) \rangle$

Die Herleitung läßt sich gut „rückwärts“ finden.

Kompakte Herleitungen im Hoare-Kalkül

Die Sequenzregel

$$\frac{\langle\phi\rangle\ P\ \langle\psi\rangle \qquad \langle\psi\rangle\ Q\ \langle\beta\rangle}{\langle\phi\rangle\ P; Q\ \langle\beta\rangle}$$

können wir kompakt geschrieben so anwenden:

$\langle\phi\rangle$
P
$\langle\psi\rangle$
Q
$\langle\beta\rangle$

Die Herleitungen von $\langle\phi\rangle\ P\ \langle\psi\rangle$ und $\langle\psi\rangle\ Q\ \langle\beta\rangle$ werden einfach „aneinandergehängt“.

Die Konsequenzregel

$$\frac{\langle \phi \rangle \ P \ \langle \psi \rangle}{\langle \alpha \rangle \ P \ \langle \beta \rangle} \ \alpha \rightarrow \phi \wedge \beta \rightarrow \psi$$

wenden wir kompakt geschrieben so an:

$\langle \alpha \rangle$
$\langle \phi \rangle$
P
$\langle \psi \rangle$
$\langle \beta \rangle$

„Innen“ wird $\langle \phi \rangle \ P \ \langle \psi \rangle$ abgeleitet. Implikationen $\alpha \rightarrow \phi$ und $\psi \rightarrow \beta$ werden „von oben nach unter“ eingefügt.

Die Schleifenregel

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

wird kompakt so angewandt:

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

Beispiel

Wir leiten das folgende Hoare-Tripel durch die kompakte Schreibweise her:

$\langle n = n' \wedge n \geq 0 \rangle$
$k = 1;$
while$(n > 0)$ {
  $k = 2 * k;$
  $n = n - 1;$
}
$\langle k = 2^{n'} \rangle$

In der Herleitung wird keine Programmanweisung mehrfach geschrieben.

$\langle n = n' \wedge n \geq 0 \rangle$
$\langle n = n' \wedge 1 = 1 \wedge n \geq 0 \rangle$
$k = 1;$
$\langle n = n' \wedge k = 1 \wedge n \geq 0 \rangle$
$\langle k = 2^{n'-n} \wedge n \geq 0 \rangle$
while$(n > 0)$ {
  $\langle k = 2^{n'-n} \wedge n \geq 0 \wedge n > 0 \rangle$
  $\langle 2k = 2^{n'-n+1} \wedge n > 0 \rangle$
  $k = 2 * k;$
  $\langle k = 2^{n'-n+1} \wedge n > 0 \rangle$
  $\langle k = 2^{n'-(n-1)} \wedge n - 1 \geq 0 \rangle$
  $n = n - 1;$
  $\langle k = 2^{n'-n} \wedge n \geq 0 \rangle$
}
$\langle k = 2^{n'-n} \wedge n \geq 0 \wedge \neg(n > 0) \rangle$
$\langle k = 2^{n'} \rangle$

Die Bedingungsregel

$$\frac{\langle \phi \wedge B \rangle\ P\ \langle \psi \rangle \qquad \langle \phi \wedge \neg B \rangle\ Q\ \langle \psi \rangle}{\langle \phi \rangle\ \textbf{if}(B)\ \{P\}\ \textbf{else}\ \{Q\}\ \langle \psi \rangle}$$

wird zuletzt kompakt so angewandt:

$\langle \phi \rangle$
if(B) {
  $\langle \phi \wedge B \rangle$
  P
  $\langle \psi \rangle$
}
else {
  $\langle \phi \wedge \neg B \rangle$
  Q
  $\langle \psi \rangle$
}
$\langle \psi \rangle$

Terminierung von Programmen

Wir haben uns mit *partieller Korrektheit* beschäftigt:

Falls das Programm terminiert, *dann* ist das Ergebnis korrekt.

Wie zeigen wir, *daß* es terminiert?

Terminiert dieses Programm? Für alle n??

```
while(n > 1) {
  if(n % 2 == 1) { // ist n ungerade?
    n = 3 * n + 1;
  }
  else {
    n = n/2;
  }
}
```

Einfache Terminierungsbeweise

Nur **while**-Schleifen können nicht terminieren.

Für die totale Korrektheit eines Programms genügt es zu zeigen, daß alle **while**-Schleifen terminieren.

Die Schleife **while**(B) $\{P\}$ terminiert, falls
$\langle t = t' \wedge B \rangle\ P\ \langle t < t' \wedge t \geq 0 \rangle$ gilt.

Hier ist t eine ganzzahliger Ausdruck und t' eine Hilfsvariable.

Der Ausdruck t wird immer kleiner, kann aber nicht negativ werden. Daher muß die Schleife irgendwann enden. Wir nennen t die *Variante* der Schleife.

Einfache Terminierungsbeweise

Beispiel:

```
while(k > 0) {
  k = k - 1;
}
```

Wir wählen $t := k$. Dieser Ausdruck wird immer kleiner, kann aber nicht negativ werden.

Wir können zeigen, daß
$\langle k = k' \wedge k > 0 \rangle\ k = k - 1;\ \langle k < k' \wedge k \geq 0 \rangle$ gilt.

Damit terminiert diese Schleife.

Schwierigere Situationen

Für die totale Korrektheit muß (und kann) man manchmal nicht von *allen* **while**-Schleifen isoliert die Terminierung beweisen.

Der Kontext ist wichtig.

Beispiel:

```
if(x > 0) {
  while(y > 0) {
    y = y - x;
  }
}
```

Dieses Programm terminiert immer.

Die Variante ist einfach y.

Contract Programming

Contract Programming ist durch die Hoare-Tripel motiviert.

Wenn wir ein Programm P schreiben, machen wir einen „Vertrag“:

Wir verlangen im Vertrag, daß unser Programm nur bestimmte Eingaben erhält (Vorbedingungen).

Wir garantieren im Vertrag, daß die Ausgabe unseres Programms bestimmte Eigenschaften hat (Nachbedingungen).

Wir versprechen also, daß ein Hoare-Tripel $\langle \phi \rangle\ P\ \langle \psi \rangle$ gültig ist.

Geht etwas schief, dann ist unser Programm „schuldig“, falls ϕ gilt, aber ψ nicht.

Die Vor- und Nachbedingungen lassen sich unter Umständen im Programm leicht testen.