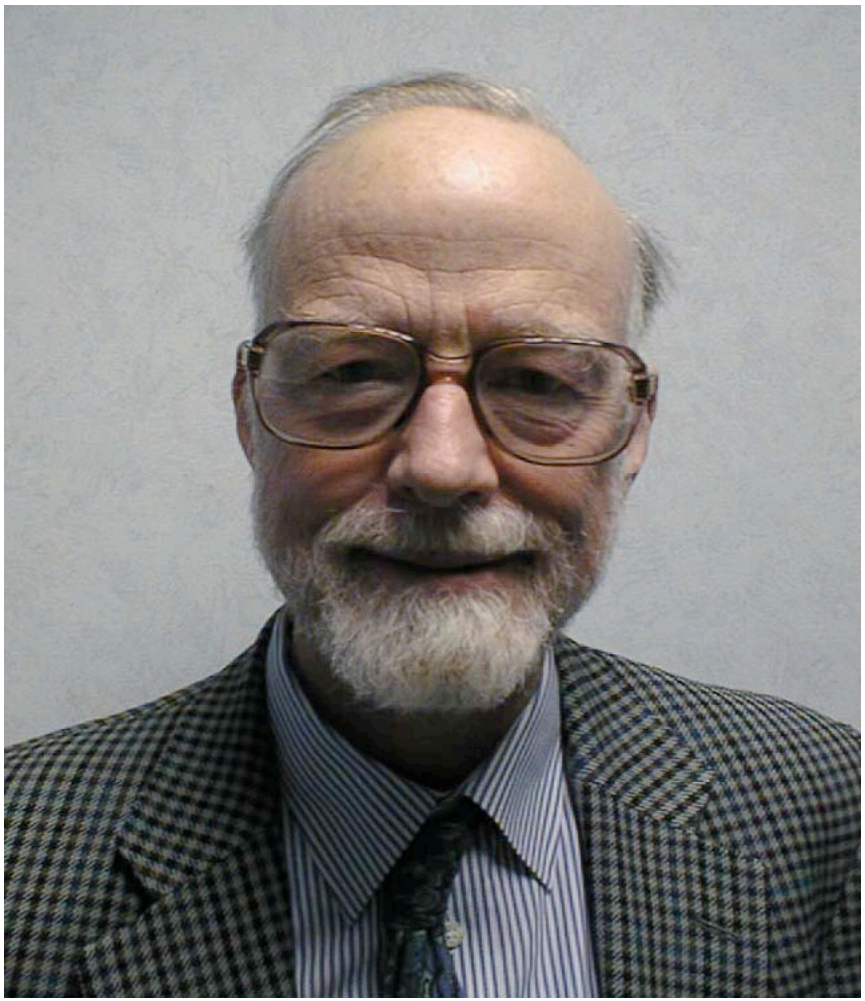
Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich



Programmverifikation

Sir Christopher Anthony R. Hoare

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Korrektheit von Programmen

Ein Programm ist teilweise (partiell) korrekt wenn es seine Spezifikation erfüllt.

Ein Programm ist vollständig (total) korrekt wenn es seine Spezifikation in einer endlichen Anzahl von Schritten erfüllt (terminiert).

Daraus folgt:

Ein Programm ist total korrekt wenn es partiell korrekt ist und gleichzeitig terminiert.

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Zusicherungen

Eine Zusicherung (assertion) ist eine Aussage, die an einer bestimmten Stelle des Programmablaufes gültig ist.

Die Precondition Q ist eine Zusicherung, die zu Beginn eines Programms gültig ist.

Die Postcondition R ist eine Zusicherung, die am Ende eines Programms gültig ist.

Eine Invariante P ist eine Zusicherung, die während eines ganzen Programmabschnittes gültig ist.

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Spezifikation von Programmen

Ein Programm kann durch seine Vorbedingung Q (Precondition) und seine Endbedingung R (Postcondition) spezifiziert werden:

$$\{Q\} \; . \; \{R\}$$

Ein Programm S ist korrekt, wenn es seine Precondition Q (in einer endlichen Anzahl von Schritten) in die Postcondition R überführt:

$$\{Q\} \; S \; \{R\}$$

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Hoare Logik

{Q} S {R}

ist ein prädikatenlogischer Ausdruck der genau dann den Wert true liefert, wenn das Programm S unter der Vorbedingung Q terminiert und danach die Endbedingung R erfüllt.

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Konsequenz-Regel

$$\frac{(Q \Rightarrow Q'),\ \{Q'\}\ S\ \{R'\},\ (R' \Rightarrow R)}{\{Q\}\ S\ \{R\}}$$

Das Programm {Q'} S {R'} erfüllt auch jede Spezifikation mit einer schärferen Voraussetzung Q und einer schwächeren Endbedingung R.

Die Bedingungen (Q ⇒ Q') und (R' ⇒ R) sind hinreichend dafür, dass das Programm {Q'} S {R'} die Spezifikation {Q} . {R} erfüllt.

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Zuweisungsaxiom

$$\frac{}{\{R(A)\}\ x=A\ \{R(x)\}}$$

Damit nach der Zuweisung des Ausdrucks A an die Variable x die Bedingung R(x) gilt muss vor dieser Zuweisung die Bedingung R(A) gelten.

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Sequenz-Regel

$$\frac{\{Q\}\ S_1\ \{P\},\ \{P\}\ S_2\ \{R\}}{\{Q\}\ S_1;S_2\ \{R\}}$$

Die Programmstücke S_1 und S_2 können zu einem Programmstück $S_1;S_2$ zusammengefügt werden wenn die Postcondition von S_1 mit der Precondition von S_2 identisch ist.

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Verallgemeinerte Sequenz-Regel

$$\frac{\{Q\}\ S_1\ \{R_1\},\ (R_1 \Rightarrow Q_2),\ \{Q_2\}\ S_2\ \{R\}}{\{Q\}\ S_1;S_2\ \{R\}}$$

Die Programmstücke S_1 und S_2 können zu einem Programmstück $S_1;S_2$ zusammengefügt werden wenn die Postcondition von S_1 die Precondition von S_2 sicherstellt (Konsequenz-Regel).

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

if-Regel

$$\frac{\{Q \text{ and } B\}\ S_1\ \{R\},\ \{Q \text{ and not } B\}\ S_2\ \{R\}}{\{Q\}\ \text{if } (B)\ S_1\ \text{else}\ S_2\ \{R\}}$$

Die Programmstücke S_1 und S_2 können zu einer if-Anweisung {Q} if (B) S_1 else S_2 {R} zusammengefügt werden wenn {Q and B} S_1 {R} und {Q and not B} S_2 {R} gilt.

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

while-Regel

$$\frac{\{P \text{ and } B\}\ S\ \{P\}}{\{P\}\ \text{while } (B)\ S\ \{P \text{ and not } B\}}$$

Für den Schleifenrumpf kann jedes Programmstück S eingesetzt werden, das unter der Bedingung B die Invariante P aufrecht erhält.

Gilt die Invariante P vor der Schleife so gilt sie auch nach Abbruch der Schleife. Zusätzlich gilt nach Abbruch der Schleife die Abbruchbedingung not B.

Achtung: die while-Regel garantiert nicht, dass die Schleife terminiert!

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Termination von Programmen

Ein Programm terminiert, wenn es sein Ergebnis in einer endlichen Anzahl von Schritten berechnet.

Die Termination eines Programms kann mittels einer ganzzahligen nichtnegativen Terminationsfunktion t ($t \geq 0$) gezeigt werden:

Ein Programm terminiert, wenn der Wert seiner Terminationsfunktion t durch jeden Programmschritt S kleiner wird:

$$\{t=T\}\ S\ \{t<T\}$$

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Beispiel: Euklid'scher Algorithmus(1)

geg: X,Y
ges: x=ggT(X,Y)

Spezifikation von ggT(x,y):
ggT(x,0) = x **für x>0**
ggT(x,y) = ggT(y,x%y) **für x,y>0**

Q: (X>0) and (Y>0)
R: x=ggT(X,Y)

Helmut Schauer
Educational Engineering Lab
Department for Information Technology
University of Zurich

Beispiel: Euklid'scher Algorithmus(2)

Q: (X>0) and (Y>0)
R: x=ggT(X,Y)
P: (ggT(x,y)= ggT(X,Y)) and (x>0) and (y≥0)

```
// Q
x=X; y=Y;
do {// P and (y≠0), t: y
    z=x%y; x=y; y=z; // P
} while (y≠0);
// P and (y=0) ⇒ R
```