Korrektheit und Hoare-Kalkül für Imperative Programme

Martin Wirsing

in Zusammenarbeit mit

Moritz Hammer und Axel Rauschmayer

SS 06

Ziele

- Partielle und totale Korrektheit kennen lernen
- Die Regeln des Hoare-Kalkül für **`while`**-Programme verstehen lernen
- Lernen einfache **`while`**-Programme als korrekt zu beweisen

Was berechnet das folgende Programm?

```
int i = 7; int j = 4;
int p = 1;
int k = 0;
while (k < j) {
      p *= i;    // p = p*i;       entspricht p = i*i*...*i (k-mal)
      k++;
}

// k == j und p = i^j
System.out.println(i + "^" + j + " = " + p);
```

- Das Programm berechnet in `p` die `j`-fache Potenz von `i`
- Genauer: Am Ende des Programms gilt:

  $p = i^j$

Wie kann man das testen? Oder sogar beweisen?

Testen: Zusicherungen (Assertions) in Java

```
int i = ...; // i > 0
int j = ...; // j >= 0
int p = 1;
int k = 0;



while (k < j) {
      p *= i;
      k++;
}

assert k == j && p == Math.pow(i,k) :
      "p = " + p + ", i = " + i + ", k = " + k;

System.out.println(i + "^" + j + " = " + p);
```

Formale Zusicherung; Nachbedingung (für while)

Testen: Zusicherungen (Assertions) in Java

- Mit Zusicherungen kann man testen, ob an einer Stelle des Programms eine Eigenschaft (d.h. ein Boolescher Ausdruck) gültig ist.
- Die **Zusicherung** (Überprüfung/Test) einer Bedingung erfolgt mit

```
assert <boolean expression>;
```

  Bedeutung:

  - Hat der Ausdruck den Wert `true`, wird das Programm ordnungsgemäß fortgesetzt.
  - Hat der Ausdruck den Wert `false`, wird ein Fehler vom Typ `AssertionError` ausgelöst.

- Eine **Fehlermeldung** kann nach dem „:“ angegeben werden:

```
assert <boolean expression> : <String expression> ;
```

- Durch `> java -ea <Dateiname>` werden Programme mit Zusicherungen übersetzt und ausgeführt.

Testen: Zusicherungen (Assertions) in Java

Vorbedingung (für while)

```
int i = ...;   // i > 0
int j = ...;   // j >= 0
int p = 1;
int k = 0;

assert k <= j && p == Math.pow(i,k) :
       "p = " + p + ", i = " + i + ", k = " + k;

while (k < j) {
       p *= i;
       k++;
}

assert k == j && p == Math.pow(i,k) :
       "p = " + p + ", i = " + i + ", k = " + k;

System.out.println(i + "^" + j + " = " + p);
```

Nachbedingung (für while)

Beweis von Zusicherungen

- Die Gültigkeit von Zusicherungen kann man mit dem

  Hoare-Kalkül

  beweisen.
- Informell nennt man ein Programm

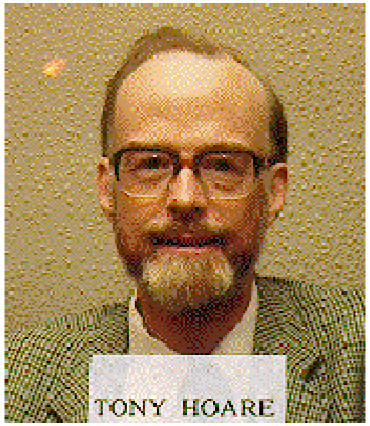
  korrekt bzgl. seiner Vor- und Nachbedingung,

  wenn die Nachbedingung bei jeder Ausführung

  des Programms gilt,

  bei der auch die Vorbedingung gegolten hat.

(Genauer auf den nächsten Folien)



C.A.R Hoare , *1934
Erfinder von Quicksort,
Hoare Logik,
Strukt. Programmierung,
CSP, Occam
Turing-Preis 1980

Hoare-Formel

{P} S {Q}

Vorbedingung **Anweisung** **Nachbedingung**

Vorbedingung	Anweisung	Nachbedingung
Formel	Zuweisung Fallunterscheidung sequ. Komp. Block Iteration	Formel

Trifft auf eine Menge von Zuständen zu;
Enthält Boolesche Operatoren &, | ,!,
lok. Variablen, „Logische" Variablen

Bedeutung von Hoare-Formeln

2 Interpretationen:

- partielle Korrektheit
- totale Korrektheit

Partielle Korrektheit

`{P} S {Q}` ist gültig,

wenn S **partiell korrekt** ist bzgl.

Vorbedingung P und Nachbedingung Q,

d.h.

$$\text{Anfangszustand} \xrightarrow[\texttt{Ausführung}]{\texttt{S}} \text{Endzustand}$$

P gilt

Q gilt, wenn S terminiert

d.h. wenn folgendes gilt:

wenn P im Anfangszustand von S gilt und wenn S terminiert,

dann gilt Q nach Ausführung von S

Totale Korrektheit

`{P}  S  {Q}` ist gültig,

wenn S **total korrekt** ist bzgl.

Vorbedingung P und Nachbedingung Q,

d.h.

$$\text{Anfangszustand} \xrightarrow[\texttt{Ausführung}]{\texttt{S}} \text{Endzustand}$$

P gilt

Q gilt **und** S terminiert

d.h. wenn folgendes gilt:

wenn P im Anfangszustand von S gilt,

dann terminiert S und Q gilt nach Ausführung von S

Folgerung

- Totale Korrektheit = Partielle Korrektheit + Terminierung

- Für eine Anweisung S ohne Iteration stimmen totale und partielle Korrektheit überein.

Partielle und totale Korrektheit

Beispiele

- Totale Korrektheit (und partielle Korrektheit):

```
{true} if(y>0) x=y; else x=-y; {x == |y|}

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

{x>1}  x=x+1; y=x; {y>2 & x>2}

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

- Partielle Korrektheit (aber nicht totale Korrektheit):

```
{true}  while(x!=0) x=x-1; {x==0}

      (terminiert nicht für x<0!)
```

Beschreibung von Nicht-Terminierung

Die Hoare-Formel

`{x>0}` **`while`** `(x>0) x=x+1; {`**`false`**`}`

terminiert nie! Sie ist partiell korrekt, aber nicht total korrekt.

<u>Allgemein:</u> Die Gültigkeit von `{P} S {`**`false`**`}` drückt Nichtterminierung aus, d.h

`{P} S {`**`false`**`}` partiell korrekt $\Rightarrow$

`S` terminiert **nicht** für alle Anfangszustände, die P erfüllen.

Partielle und totale Korrektheit: Hoare-Kalkül

- Der Hoare-Kalkül dient zum (konstruktiven) Beweisen von partieller und totaler Korrektheit

- Idee von Hoare:

  Leite (rückwärts schreitend)

  ausgehend von der (gewünschten) Nachbedingung die

  Vorbedingung ab

Hoare-Regel Zuweisung

Zuweisungsaxiom

```
{P[exp/x]} x = exp; {P}
```

Ersetze x in P durch exp

Deklaration

```
{P[exp/x]} type x = exp; {P}
```

Beispiele

```
{max? - C == 35} max = max - C; {max == 35}

{max - 5? == 35}    int C = 5; {max - C == 35}
```

Hoare-Regel Abschwächung

Abschwächungsregel

$$\frac{\texttt{P1} \Rightarrow \texttt{P},\ \{\texttt{P}\}\ \texttt{S}\ \{\texttt{Q}\},\ \texttt{Q} \Rightarrow \texttt{Q1}}{\{\texttt{P1}\}\ \texttt{S}\ \{\texttt{Q1}\}}$$

Beispiel

$$\frac{\texttt{n==3} \Rightarrow \texttt{2n>=6},\ \{\texttt{2n>=6}\}\ \texttt{n = 2*n;}\ \{\texttt{n>=6}\},\ \texttt{n>=6} \Rightarrow \texttt{n>5}}{\{\ \texttt{n==3}\}\ \texttt{n = 2*n;}\ \{\texttt{n>5}\}}\ \text{(Abschwchg.)}$$

Hoare-Regeln

Fallunterscheidungsregel

$$\frac{\{b \;\&\; P\}\; S1\; \{Q\} \qquad \{(!b) \;\&\; P\}\; S2\; \{Q\}}{\{P\}\; \textbf{if}\; (b)\; S1\; \textbf{else}\; S2\; \{Q\}}$$

`(if)` falls b ohne Seiteneffekt ist (d.h. den Zustand nicht ändert)

Sequentielle Komposition

$$\frac{\{P\}\; S1\; \{R\} \qquad \{R\}\; S2\; \{Q\}}{\{P\}\; S1\; S2\; \{Q\}}$$

`(seq. Komp.)`

Hoare-Regel Fallunterschiedung

Beispiel

```
{x>=0 & x == A}      y = x;        {x == A & y ==|A|} *
{x< 0 & x == A}      y = -x;       {x == A & y ==|A|} **
---------------------------------------------------------------
{x == A}   if (x>=0) y = x; else y = -x; {x == A & y ==|A|}
```

* wegen

```
  x>=0 & x == A  ⇒  x == A & x ==|A|
 {x == A & x ==|A|}      y = x;    {x == A & y ==|A|}
--------------------------------------------------------- (Abschwchg.)
 {x>=0 & x == A}         y = x;  {x == A & y ==|A|}
```

** wegen

```
  x<0 & x == A  ⇒  x == A & -x ==|A|
 {x == A & -x ==|A|}     y = -x;    {x == A & y ==|A|}
--------------------------------------------------------- (Abschwchg.)
 {x<0 &  x == A}         y = -x; {x == A & y ==|A|}
```

Zuweisgsaxiom

Hoare-Regel Block

Block-Regel

$$\frac{\{P\}\ S\ \{Q\}}{\{P\}\ \{S\}\ \{Q\}}$$

(falls P und Q keine in S deklarierten lokalen Variablen enthalten)

Beispiel

$$\frac{\{\texttt{max == 40}\}\ \ \textbf{final int}\ \texttt{C = 5; max = max - C;}\ \ \{\texttt{max == 35}\}}{\{\texttt{max == 40}\}\ \{\textbf{final int}\ \texttt{C = 5; max = max - C;}\}\ \{\texttt{max == 35}\}}$$

Iteration: Hoare-Regeln

Partielle Korrektheit:

Invariante

$$\frac{\texttt{\{b \& I\} S \{I\}}}{\texttt{\{I\} }\mathbf{while}\texttt{ (b) S \{(!b) \& I\}}}$$

(Iteration$_{partiell}$)

falls b seiteneffektfrei

Totale Korrektheit:

$$\frac{\begin{array}{ll}\texttt{\{b \& I\} S \{I\}} & \\ \texttt{\{b \& I \& t == z\} S \{t < z\}} & \texttt{//t wird echt kleiner} \\ \texttt{I} \Rightarrow \texttt{t >= 0} & \texttt{//t nie negativ}\end{array}}{\texttt{\{I\} }\mathbf{while}\texttt{ (b) S \{(!b) \& I\}}}$$

(Iteration$_{total}$)

falls b seiteneffektfrei

t – ein Integer-Ausdruck für die Terminierung der while-Schleife

z – eine „logische“ Variable, die nicht in `I`, `b`, `S` oder `t` vorkommt, also durch S nicht verändert wird.

Hoare-Regel While

Beispiel partielle Korrektheit

Sei I ≡ (n <= end+1) (Invariante)

$$\frac{\{n \le end \ \&\ n \le end+1\}\quad n = n+1; \quad \{n \le end+1\}}{\{n \le end+1\}\ \textbf{while}\ (n \le end)\ n = n+1;\ \{n \le end+1\ \&\ !(n \le end)\}}$$

Beispiel totale Korrektheit

Sei t ≡ end+1-n

$$\frac{\begin{array}{lll} \{n \le end \ \&\ n \le end+1\} & n = n+1; & \{n \le end+1\} \\ \{n \le end \ \&\ n \le end+1 \ \&\ (end+1-n)==z\} & n = n+1; & \{(end+1-n)<z\} \\ \quad n \le end+1 \ \Rightarrow\ (end+1-n) >= 0 & & \end{array}}{\{n \le end+1\}\ \textbf{while}\ (n \le end)\ n = n+1;\ \{n \le end+1\ \&\ !(n \le end)\}}$$

Zusammenfassung

- Partielle und totale Korrektheit sind wichtige Begriffe zur Beschreibung des Ein- / Ausgabe-Verhaltens eines Programms.
- Der Hoare-Kalkül erlaubt den Beweis der partiellen und totalen Korrektheit (kleiner) Programme.