Korrektheit und Hoare-Kalkül für Imperative Programme

Martin Wirsing

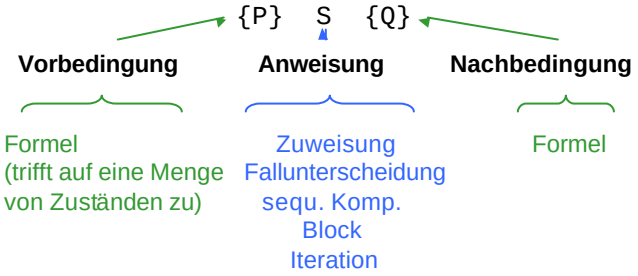
in Zusammenarbeit mit
Matthias Hölzl, Piotr Kosiuczenko, Dirk Pattinson

04/03

Ziele

- Wiederholung des Begriffs der partiellen und totalen Korrektheit
- Wiederholung der Regeln des Hoare-Kalkül für **`while`**-Programme
- Lernen einfache **`while`**-Programme als korrekt zu beweisen

Hoare-Formel



Bedeutung von Hoare-Formeln

2 Interpretationen:

- partielle Korrektheit
- totale Korrektheit

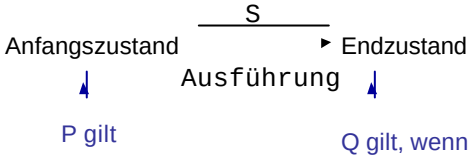
Partielle Korrektheit

`{P} S {Q}` ist gültig,

wenn S **partiell korrekt** ist bzgl.

Vorbedingung P und Nachbedingung Q,

d.h.



d.h. wenn folgendes gilt:

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

dann gilt Q nach Ausführung von S

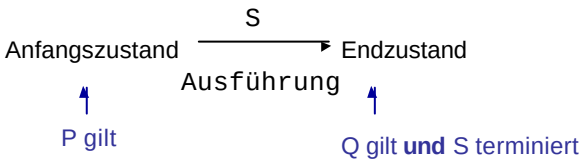
Totale Korrektheit

`{P}` `S` `{Q}` ist gültig,

wenn S **total korrekt** ist bzgl.

Vorbedingung P und Nachbedingung Q,

d.h.



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 Nichttermination

Die Hoare-Formel

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

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

Allgemein: `{P} S {false}` drückt Nichtterminierung aus, d.h

`{P} S {false}` partiell korrekt ⇒

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

Beispiele

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

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

Hoare-Regel Abschwächung

Abschwächungsregel

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

Beispiel

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

Hoare-Regeln

Fallunterscheidungsregel

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

Sequentielle Komposition

$$\frac{\{P\}\ S1\ \{R\}\quad \{R\}\ S2\ \{Q\}}{\{P\}\ S1\ S2\ \{Q\}}\ \text{(seq. Komp.)}$$

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|}
```


Hoare-Regel Block

Block-Regel

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

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

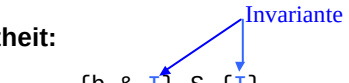
Beispiel

```
{max == 40}  final int C = 5; max = max – C;  {max == 35}
```

```
{max == 40} {final int C = 5; max = max – C;} {max == 35}
```

Iteration: Hoare-Regeln

Partielle Korrektheit:



$$\frac{\{b \;\&\; I\}\ S\ \{I\}}{\{I\}\ \texttt{while}\ (b)\ \ S\ \{(!b)\ \&\ I\}} \quad (\text{Iteration}_{partiell})$$

Totale Korrektheit:

$$\frac{\begin{array}{l}\{b \;\&\; I\}\ S\ \{I\} \\ \{b \;\&\; I \;\&\; t == z\}\ S\ \{t < z\}\ \texttt{//t wird echt kleiner} \\ I \Rightarrow t >= 0 \qquad \texttt{//t nie negativ}\end{array}}{\{I\}\ \texttt{while}\ (b)\ \ S\ \{(!b)\ \&\ I\}} \quad (\text{Iteration}_{total})$$

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)

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

Beispiel totale Korrektheit Sei t ≡ end+1-n

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

Beweisskizze

- Eine **Beweisskizze für partielle / totale Korrektheit** ist ein kommentiertes Programm S, bei dem **jede** Teilanweisung R von S mit Vor- und Nachbedingung der Form `{P} R {Q}` annotiert ist, so daß `{P} R {Q}` im Hoare-Kalkül ableitbar ist.
  Folgen zwei Zusicherungen `{P} {Q}` hintereinander, muss gelten `P ⇒ Q`

- Insbesondere sind annotiert

`while`- Anweisung	mit `{P} (while(b) {b&P} S {P}) {!b&P}`
`if`- Anweisung	mit `{P}(if(b) {b&P} S1 {Q} else {!b&P} S2{Q}){Q}`
Block	mit `{P} {{P} S {Q}} {Q}`
„Doppelformel“	mit `{P} ⇒ {P1} S {Q1} ⇒ {Q}` so daß `P ⇒ P1, Q1 ⇒ Q`

⇒ kann auch weggelassen werden

Beweisskizzen

Beispiele:

1. Zuweisung

```
{true}
   ⇓
{x * x >=0}
x = x * x;
{x >=0}
```

also ist auch

```
{true}
x = x * x;
{x >=0}
```

Beweisskizze

Beweisskizzen

2. if-then-else-Regel

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


Beweisskizzen

3. .... Beispiel

```
{true}
   ⇓
{1 <= 11}
int n = 1;
{n <= 11}
int end = 10;
{n <= end+1}  //Invariante
while  (n <= end)
{       {n <= end & n <= end+1}
                  ⇓
        {n+1 <= end+1}
            n = n+1;
        {n <= end+1}
}
{!(n <= end) & n <= end+1}
            ⇓
      {n == end+1}
```

Beweis der partiellen Korrektheit von

```
{true}
int n=1; int end=10;
while (n <= end) n = n+1;
{n == end+1}
```

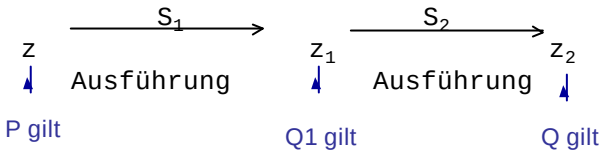
Annotierte Programme

- Ein annotiertes Programm ist ein Programm, das Zusicherungen als Kommentare enthält (aber möglicherweise nicht vollständig annotiert ist).
- Ein annotiertes Programm `{P} S {Q}` heißt partiell korrekt, wenn es
  1. partiell korrekt ist bzgl. `P,Q` und wenn
  2. für jede innere Zusicherung `Q1` von `S`
     `Q1` gültig ist in jedem Zustand, der erhalten wird durch Ausführung von `S` in einem Zustand, in dem `P` gilt.

Annotierte Programme

Beispiel

$S \equiv \{P\}\ S_1\ \{Q1\}\ S_2\ \{Q\}$



Sei `z` ein Zustand, in dem `P` gilt.

Dann müssen `Q1` in z_1 und `Q` in z_2 gelten

Annotierte Programme

Beispiele

- Jede Beweisskizze ist ein korrektes annotiertes Programm

  z.B. `int n = 1; {n==1} n = 2 * n; {n==2}`

- Jede gültige Hoare-Formel `{P} S {Q}` ist ein partiell korrektes annotiertes Programm

- Jedes Programm S kann als annotiertes Programm der Form `{true} S {true}` aufgefaßt werden

- `int n = 1; {n>0} n = 2 * n; {n==2}`

Beispiel: Division x/y und Rest

Sei $x \geq 0$, $y > 0$, $I \equiv$ `x == quo * y + rem & rem >=0`

Dann ist

```
int quo = 0, rem = x;
{I}   //Invariante
while (rem >= y)
{   rem = rem – y; quo = quo + 1;
}
{x == quo * y + rem  & 0 <= rem < y}
```

Ein partiell und total korrektes annotiertes Programm (Beweis Übung)

Realisierung in Java

- In Java kann man Zusicherungen der Form {P} durch die Anweisung

  `assert P`

  in das Programm einführen
- Durch > javac -source 1.4 <Dateiname>.java

  > java -ea <Dateiname>

  werden Programme mit Zusicherungen übersetzt und ausgeführt
- Ist bei der Programmausführung eine der Zusicherungen falsch, so bricht das Programm mit einem Fehler ab (und die Zeile mit der ungültigen Zusicherung wird angegeben)

Realisierung in Java

- Gelten alle Zusicherungen bei der Ausführung des Programms, so terminiert das Programm normal (und nur die print-Anweisungen werden ausgegeben).

Þ Java unterstützt das Testen von Programmen durch die Möglichkeit der dynamischen Überprüfung von Zusicherungen.

Testen liefert keinen Beweis der Korrektheit, sondern im Fehlerfall den Nachweis von Fehlern!

Beispiel: Division und Rest

```
int x = 1000, y = 37;
int quo = 0, rem = x;
assert x == quo * y + rem & rem >=0;
while (rem >= y)
{ rem = rem-y; quo = quo+1;
  assert x == quo * y + rem & rem >= 0;
}
assert x ==quo * y + rem & 0 <= rem & rem < y;
```


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.
- Beweisskizzen sind eine übersichtliche Präsentation von formalen Beweisen.
- Annotierte Programme können in Java mit der `assert`-Anweisung geschrieben und getestet werden.