
II.1. Grundelemente der Programmierung


- 1. Erste Schritte
- 2. Einfache Datentypen
- 3. Anweisungen und Kontrollstrukturen
- 4. Verifikation
- 5. Reihungen (Arrays)

4. Verifikation

■ Spezifikation: Angabe, *was* ein Programm tun soll

- natürliche Sprache
- grafische Sprachen (UML, ...)
- logische Sprachen (Z, VDM, ...)

■ Testen: Überprüfung für endlich viele Eingaben

 keine 100% Sicherheit

■ Verifikation: Mathematischer Beweis der Korrektheit

- Terminierung: Hält Programm immer an?
- Partielle Korrektheit: Falls Programm anhält, erfüllt es Spezifikation?
- Totale Korrektheit: Terminierung & Partielle Korrektheit

 Semantik der Programmiersprache

Fakultät

```
public static void main (String [] arguments) {
    int n = IO.Eingabe(), i, res;

    i = n;

    res = 1;

    while (i > 1) {

        res = res * i;

        i = i - 1;

    }

    System.out.println("Die Fakultät ist " + res);
}
```

Verifikation

Programm P

```
i = n;  
res = 1;  
while (i > 1) {  
    res = res * i;  
    i = i - 1;  
}
```

■ Spezifikation:

Programm berechnet (in `res`) Fakultät von `n`

■ Terminierung:

Programm hält an, weil `i` in jedem Schleifendurchlauf kleiner wird

■ Partielle Korrektheit:

Nach Ausführung ist `res = n!`

■ Totale Korrektheit

Wie beweist man so etwas ?

➡ Verifikation nötig bei sicherheitskritischen Anwendungen

➡ hilft für Programmwurf und Programmierstil

Partielle Korrektheit: Hoare-Kalkül

- Spezifikation (zur partiellen Korrektheit)

$$\langle \varphi \rangle \mathbf{P} \langle \psi \rangle$$

Wenn vor Ausführung von P **Vorbedingung** φ gilt
und Ausführung von P terminiert,
dann gilt hinterher **Nachbedingung** ψ .

- Bsp: $\langle \text{true} \rangle \mathbf{P} \langle \text{res} = \text{n!} \rangle$

- Partielle Korrektheit ist *semantische* Aussage

Hoare-Kalkül: 7 *syntaktische* Regeln zur Herleitung von
Korrektheitsaussagen

Zuweisungsregel

$$\frac{}{\langle \varphi [x/t] \rangle \quad x = t; \quad \langle \varphi \rangle}$$

x ist Variable, t ist Ausdruck (ohne Seiteneffekte),
 $\varphi [x/t]$ ist φ mit allen x ersetzt durch t

Bsp: $\langle 5 = 5 \rangle \quad x = 5; \quad \langle x = 5 \rangle$

$$\langle 5 = 5 \rangle$$
$$x = 5;$$
$$\langle x = 5 \rangle$$

Konsequenzregel 1 (Stärkere Vorbedingung)

$$\frac{\langle \varphi \rangle \text{ P } \langle \psi \rangle \qquad \alpha \Rightarrow \varphi}{\langle \alpha \rangle \text{ P } \langle \psi \rangle}$$

Bsp: $\langle \text{true} \rangle \quad \mathbf{x = 5}; \quad \langle \mathbf{x = 5} \rangle$, denn:

$$\frac{\langle 5 = 5 \rangle \quad \mathbf{x = 5}; \quad \langle \mathbf{x = 5} \rangle \qquad \text{true} \Rightarrow 5 = 5}{\langle \text{true} \rangle \quad \mathbf{x = 5}; \quad \langle \mathbf{x = 5} \rangle}$$

$\langle \text{true} \rangle$
 $\langle 5 = 5 \rangle$
 $\mathbf{x = 5};$
 $\langle \mathbf{x = 5} \rangle$

Konsequenzregel 2 (Schwächere Nachbedg.)

$$\frac{\langle \varphi \rangle \text{ P } \langle \psi \rangle \qquad \psi \Rightarrow \beta}{\langle \varphi \rangle \text{ P } \langle \beta \rangle}$$

Bsp: $\langle \text{true} \rangle \quad \mathbf{x = 5}; \quad \langle \mathbf{x \geq 5} \rangle$, denn:

$$\frac{\langle \text{true} \rangle \quad \mathbf{x = 5}; \quad \langle \mathbf{x = 5} \rangle \qquad \mathbf{x = 5} \Rightarrow \mathbf{x \geq 5}}{\langle \text{true} \rangle \quad \mathbf{x = 5}; \quad \langle \mathbf{x \geq 5} \rangle}$$

$$\begin{array}{l} \langle \text{true} \rangle \\ \langle \mathbf{5 = 5} \rangle \\ \mathbf{x = 5}; \\ \langle \mathbf{x = 5} \rangle \\ \langle \mathbf{x \geq 5} \rangle \end{array}$$

Sequenzregel

$$\frac{\langle \varphi \rangle \quad P \quad \langle \psi \rangle \qquad \langle \psi \rangle \quad Q \quad \langle \beta \rangle}{\langle \varphi \rangle \quad P \quad Q \quad \langle \beta \rangle}$$

Bsp: $\langle \text{true} \rangle$

$x = 5;$

$\text{res} = x * x + 6;$

$\langle \text{res} = 31 \rangle$

$\langle \text{true} \rangle$

$\langle 5 = 5 \rangle$

$x = 5;$

$\langle x = 5 \rangle$

$\langle x * x + 6 = 31 \rangle$

$\text{res} = x * x + 6;$

$\langle \text{res} = 31 \rangle$

Bedingungsregel 1

$$\frac{\langle \varphi \wedge B \rangle \text{ P } \langle \psi \rangle \quad \varphi \wedge \neg B \Rightarrow \psi}{\langle \varphi \rangle \text{ if } (B) \{P\} \langle \psi \rangle}$$

Bsp: $\langle \text{true} \rangle$

```
res = y;  
if (x > y) res = x;
```

$\langle \text{res} = \max(x, y) \rangle$

denn: $\langle \text{res} = y \wedge x > y \rangle$

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

```
res = x;
```

$\langle \text{res} = \max(x, y) \rangle$

und $\langle \text{res} = y \wedge \neg x > y \rangle$

$\Rightarrow \langle \text{res} = \max(x, y) \rangle$

$\langle \text{true} \rangle$

$\langle y = y \rangle$

```
res = y;
```

$\langle \text{res} = y \rangle$

```
if (x > y) res = x;
```

$\langle \text{res} = \max(x, y) \rangle$

Bedingungsregel 2

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

Bsp: $\langle \text{true} \rangle$

```
if (x < 0)
  res = -x;
else
  res = x;
```

$\langle \text{res} = |x| \rangle$

denn: $\langle x < 0 \rangle$

$\langle -x = |x| \rangle$

$\text{res} = -x;$

$\langle \text{res} = |x| \rangle$

und $\langle \neg x < 0 \rangle$

$\langle x = |x| \rangle$

$\text{res} = x;$

$\langle \text{res} = |x| \rangle$

Schleifenregel

$$\frac{\langle \varphi \wedge B \rangle \text{ P } \langle \varphi \rangle}{\langle \varphi \rangle \text{ while } (B) \{ P \} \langle \varphi \wedge \neg B \rangle}$$

```
<true>  
  i = n; res = 1;  
<i = n ∧ res = 1>  
<i! * res = n!>  
  while (i > 1) {res = res * i; i = i - 1; }  
<i! * res = n! ∧ ¬ i > 1>  
<res = n! >
```

φ ist Schleifen-
invariante

denn: **<i! * res = n! ∧ i > 1>**
<(i-1)! * (res * i) = n!>
 res = res * i;
 i = i - 1;
<i! * res = n!>

Hoare-Kalkül

■ Zuweisungsregel

$$\frac{}{\langle \varphi [x/t] \rangle \quad \mathbf{x} = \mathbf{t}; \quad \langle \varphi \rangle}$$

■ Konsequenzregeln

$$\frac{\langle \varphi \rangle \quad \mathbf{P} \quad \langle \psi \rangle \quad \alpha \Rightarrow \varphi}{\langle \alpha \rangle \quad \mathbf{P} \quad \langle \psi \rangle}$$

$$\frac{\langle \varphi \rangle \quad \mathbf{P} \quad \langle \psi \rangle \quad \psi \Rightarrow \beta}{\langle \varphi \rangle \quad \mathbf{P} \quad \langle \beta \rangle}$$

■ Sequenzregel

$$\frac{\langle \varphi \rangle \quad \mathbf{P} \quad \langle \psi \rangle \quad \langle \psi \rangle \quad \mathbf{Q} \quad \langle \beta \rangle}{\langle \varphi \rangle \quad \mathbf{P} \quad \mathbf{Q} \quad \langle \beta \rangle}$$

■ Bedingungsregeln

$$\frac{\langle \varphi \wedge \mathbf{B} \rangle \quad \mathbf{P} \quad \langle \psi \rangle \quad \varphi \wedge \neg \mathbf{B} \Rightarrow \psi}{\langle \varphi \rangle \quad \mathbf{if} \quad (\mathbf{B}) \quad \{\mathbf{P}\} \quad \langle \psi \rangle}$$

$$\frac{\langle \varphi \wedge \mathbf{B} \rangle \quad \mathbf{P} \quad \langle \psi \rangle \quad \langle \varphi \wedge \neg \mathbf{B} \rangle \quad \mathbf{Q} \quad \langle \psi \rangle}{\langle \varphi \rangle \quad \mathbf{if} \quad (\mathbf{B}) \quad \{\mathbf{P}\} \quad \mathbf{else} \quad \{\mathbf{Q}\} \quad \langle \psi \rangle}$$

■ Schleifenregel

$$\frac{\langle \varphi \wedge \mathbf{B} \rangle \quad \mathbf{P} \quad \langle \varphi \rangle}{\langle \varphi \rangle \quad \mathbf{while} \quad (\mathbf{B}) \quad \{\mathbf{P}\} \quad \langle \varphi \wedge \neg \mathbf{B} \rangle}$$

Terminierung

Für jede Schleife `while (B) {P}` finde einen `int`-Ausdruck V (*Variante* der Schleife), so dass:

$B \Rightarrow V \geq 0$ und $\langle V = m \wedge B \rangle P \langle V < m \rangle$

```
while (i > 1) {res = res * i; i = i - 1; }
```

Variante ist i ,

denn: $i > 1 \Rightarrow i \geq 0$

$\langle i = m \wedge i > 1 \rangle$

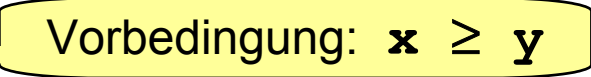
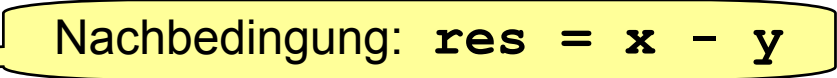
$\langle i-1 = m-1 \rangle$

`res = res * i; i = i - 1;`

$\langle i = m-1 \rangle$

$\langle i < m \rangle$

Verifikation der Subtraktion

```
public class Subtract {  
  
    public static void main (String [] args) {  
        System.out.print ("Gib 2 Zahlen ein: ");  
        int x = IO.Eingabe (), y = IO.Eingabe (), z, res;  
  
        z = y;   
        res = 0;  
  
        //Invariante:  $x \geq z \wedge res = z - y$   
        //Variante:  $x - z$   
  
        while (x > z) {  
            z = z + 1;  
            res = res + 1;  
        }   
  
        System.out.println (x + " - " + y + " = " + res);  
    }  
}
```

Verifikation eines Primzahl-Programms

```
public class Prim {
  public static void main (String [] args) {
    System.out.print ("Gib Zahl ein: ");
    int n = IO.Eingabe ();
    int Wurzel = (int) Math.sqrt (n),
        Teiler = 2;
    boolean istPrim = true;

    //Invariante:  $n \geq 2 \wedge \text{Wurzel} = \lfloor \text{sqrt}(n) \rfloor \wedge$ 
    //            $\text{istPrim} = \text{keine Zahl } i \text{ mit } 2 \leq i < \text{Teiler} \text{ teilt } n$ 
    //Variante:  $\text{Wurzel} - \text{Teiler}$ 

    while (Teiler <= Wurzel) {
      if (n % Teiler == 0)   istPrim = false;
      Teiler = Teiler + 1; }

    System.out.println (n + " prim: " + istPrim);
  }}

```

Vorbedingung: $n \geq 2$

Nachbedingung: $\text{istPrim} = \text{true}$ gdw. n ist Primzahl