

## 4.3 Verifikation von Aussagen über Algorithmen

**Hoaresche Logik:** Kalkül zum Beweisen von **Aussagen über Algorithmen und Programme**, Programm-**Verifikation**, [C.A.R. Hoare, 1969].

**Statische Aussagen** über Zustände (Werte von Variablen), die der Algorithmus (das Programm) an bestimmten Stellen annehmen kann, z. B.  
 ...  $\{ \text{pegel} < \text{max} \}$  `pegel := pegel + 1;` ...  $\{ 0 < i \wedge i < 10 \}$  `a[i] := 42;` ...  $\{ x = \text{GGT} \}$ ;

Aussagen müssen beweisbar **für alle Ausführungen** des Algorithmus gelten. Im Gegensatz zum **dynamischen Testen**: Ausführen des Algorithmus für bestimmte Eingaben.

**Schlussregeln** für Anweisungsformen erlauben logische Schlüsse über Anweisungen hinweg:  
 $\{ \text{pegel} + 1 \leq \text{max} \}$  `pegel := pegel + 1;`  $\{ \text{pegel} \leq \text{max} \}$  wegen Schlussregel für Zuweisungen

**Verifikation** beweist, dass

- an einer bestimmten Programmstelle eine Aussage über Zustände gilt,
- vor und nach Ausführung eines Programmstückes eine **Invariante** gilt,
- ein Algorithmus **aus jeder zulässigen Eingabe die geforderte Ausgabe** berechnet, z. B.  
 $\{ a, b \in \mathbb{N} \}$  Euklidischer Algorithmus  $\{ x \text{ ist GGT von } a, b \}$
- eine **Schleife terminiert**.

Ein **Algorithmus und die Aussagen dazu sollen zusammen konstruiert** werden.

## Vorschau auf Konzepte

**Aussagen charakterisieren Zustände** der Ausführung

**Algorithmen in informeller Notation**

**Schlussregeln für Anweisungsformen** anwenden

**Invariante von Schleifen** (und anderen Konstrukten)

**Schlussketten über Anweisungen hinweg** verifizieren Aussagen

Nachweis der **Terminierung von Schleifen**

## Beispiel zur Vorschau: Verifikation des Algorithmus ggT

Vorbedingung:  $x, y \in \mathbb{N}$ , d. h.  $x > 0, y > 0$ ; sei G größter gemeinsame Teiler von x und y  
 Nachbedingung:  $a = G$

Algorithmus mit { Aussagen über Variable }:

```

{ G ist ggT von x und y  $\wedge$   $x > 0 \wedge y > 0$  }
a := x; b := y;
{ INV: G ist ggT von a und b  $\wedge$   $a > 0 \wedge b > 0$  }
solange a  $\neq$  b wiederhole
  { INV  $\wedge$   $a \neq b$  }
  falls a > b :
    { G ist ggT von a und b  $\wedge$   $a > 0 \wedge b > 0 \wedge a > b$  }  $\rightarrow$ 
    { G ist ggT von a-b und b  $\wedge$   $a-b > 0 \wedge b > 0$  }
    a := a - b
    { INV }
  sonst
    { G ist ggT von a und b  $\wedge$   $a > 0 \wedge b > 0 \wedge b > a$  }  $\rightarrow$ 
    { G ist ggT von a und b-a  $\wedge$   $a > 0 \wedge b-a > 0$  }
    b := b - a
    { INV }
  { INV }
{ INV  $\wedge$   $a = b$  }  $\rightarrow$ 
{ a = G }
  
```

Terminierung der Schleife:

- $a+b$  fällt monoton
- $a+b > 0$  ist Invariante

## Aussage charakterisiert Programmzustände

Eine **Aussage P** an einer **Stelle in einem Algorithmus** (Programm) vor oder nach einer Anweisung

...  $S_1 \{P\} S_2$  ...

**charakterisiert alle Zustände**, die das Programm an dieser Stelle **bei irgendeiner Ausführung** annehmen kann. P wird über **Variable des Algorithmus** formuliert.

Z. B. ...  $\{0 \leq i \wedge i < 10\} a[i] := 42;$  ...

Bei jeder Ausführung liegt der Wert von i im angegebenen Intervall.

Eine Aussage über andere Variablen wird hier nicht gemacht.

Nur die **gerade interessierende Eigenschaften der Zustände** werden beschrieben.

Aussagen können unterschiedlich scharf formuliert werden:

$\{f\}$	kein Zustand erfüllt P, Stelle nicht erreichbar
$\{0 \leq i \wedge i < 2 \wedge a[i] > 0\}$	schärfer; evtl. weniger Zustände; schwieriger zu verifizieren
$\{0 \leq i \wedge i < 2\}$	
$\{0 \leq i \wedge i < 10\}$	
$\{0 \leq i\}$	schwächer; evtl. mehr Zustände; leichter zu verifizieren
$\{w\}$	beliebige Zustände erfüllen P

## Notation von Algorithmentelementen

Anweisungsform	Notation	Beispiel
Sequenz	Anweisung <sub>1</sub> ; Anweisung <sub>2</sub>	<b>a := x;</b> <b>b := y</b>
Zuweisung	Variable := Ausdruck	<b>a := x</b>
Alternative, zweiseitig	<b>falls</b> Bedingung : Anweisung <sub>1</sub> <b>sonst</b> Anweisung <sub>2</sub>	<b>falls a &gt; b :</b> <b>a := a - b</b> <b>sonst b := b - a</b>
bedingte Anweisung	<b>falls</b> Bedingung : Anweisung <sub>1</sub>	<b>falls a &lt; 0 :</b> <b>a := - a</b>
Aufruf eines Unteralgorithmus ua	ua()	<b>berechneGgT()</b>
Schleife	<b>solange</b> Bedingung <b>wiederhole</b> Anweisung	<b>solange a≠b wiederhole</b> <b>falls a &gt; b :</b> ... ...

## Vor- und Nachbedingung von Anweisungen

Aussage Q charakterisiert die Zustände, die eine Ausführung zwischen den Anweisungen A<sub>1</sub> und A<sub>2</sub> annehmen kann:

$$\{ P \} A_1 \{ Q \} A_2 \{ R \}$$

Q ist **Nachbedingung** von A<sub>1</sub> und **Vorbedingung** von A<sub>2</sub>

Beispiel:  $\{ i+1 \geq 0 \} i := i + 1; \{ i \geq 0 \} a[i] := k; \{ \dots \}$

Zur Verifikation eines Algorithmus muss für jede Anweisung S ein Nachweis geführt werden:

$$\{ \text{Vorbedingung } P \} S \{ \text{Nachbedingung } Q \}$$

nachweisen: Wenn vor der Ausführung der Anweisung S die Aussage P gilt, dann gilt Q nach der Ausführung von S, falls S terminiert.

Beispiel:  $\{ i+1 \geq 0 \} i := i + 1; \{ i \geq 0 \}$  mit Zuweisungsregel nachweisen

Die Aussagen werden entsprechend der **Struktur von S verknüpft**.

Für jede Anweisungsform wird eine spezielle **Schlussregel** angewandt.

Eine **Spezifikation liefert Vorbedingung und Nachbedingung** des gesamten Algorithmus:

*gegeben:*

Aussagen über die Eingabe

*gesucht:*

Aussagen über Zusammenhang zwischen Ein- und Ausgabe

{ Vorbedingung }      Algorithmus      { Nachbedingung }

## Zuweisungsregel

Hoare'scher Kalkül definiert für jede Anweisungsform eine **Schlussregel**.

Eine **Zuweisung**  $x := e$  wertet den Ausdruck  $e$  aus und weist das Ergebnis der Variablen  $x$  zu.

$$\{ P_{[x/e]} \} x := e \{ P \}$$

Wenn vor der Ausführung  $P_{[x/e]}$  gilt ( $P$  wobei  $x$  durch  $e$  substituiert ist), gilt nach der Ausführung der Zuweisung  $P$ .

Beispiele:  $\{ a > 0 \} \quad x := a \quad \{ x > 0 \}$   
 $\{ i + 1 > 0 \} \quad i := i + 1 \quad \{ i > 0 \}$

Wenn man zeigen will, dass **nach der Zuweisung eine Aussage  $P$  für  $x$  gilt**, muss man zeigen, dass **vor der Zuweisung dieselbe Aussage  $P$  für  $e$  gilt**.

Beispiele im Algorithmus:

$\{ x > 0 \wedge y > 0 \}$

**a := x;**

$\{ a > 0 \wedge y > 0 \}$

**b := y;**

$\{ a > 0 \wedge b > 0 \}$

$\{ G \text{ ist ggT von } a-b \text{ und } b \wedge a-b > 0 \wedge b > 0 \}$

**a := a - b**

$\{ G \text{ ist ggT von } a \text{ und } b \wedge a > 0 \wedge b > 0 \}$

## Beispiele für Zuweisungsregel

$\{ P_{[x/e]} \} \quad x := e \quad \{ P \}$

1.  $\{ a > 0 \} \quad x := a \quad \{ x > 0 \}$

2.  $\{ a > 0 \wedge a > 0 \} \quad x := a \quad \{ x > 0 \wedge a > 0 \}$

3.  $\{ a > 0 \wedge x = 7 \} \quad x := a \quad \{ x > 0 \wedge x = 7 \}$

$x$  durch  $a$  ersetzen - nicht umgekehrt

**falscher Schluss!**  
**alle**  $x$  durch  $a$  ersetzen!

4.  $\{ a > 0 \wedge z > 0 \} \quad x := a \quad \{ x > 0 \wedge z > 0 \}$

$z > 0$  ist nicht betroffen

5.  $\{ i + 1 > 0 \} \quad i := i + 1 \quad \{ i > 0 \}$

6.  $\{ i \geq 0 \} \leftrightarrow \{ i + 1 > 0 \} \quad i := i + 1 \quad \{ i > 0 \}$

passend umformen

7.  $\{ i = 2 \} \leftrightarrow \{ i + 1 = 3 \} \quad i := i + 1 \quad \{ i = 3 \}$

passend umformen

8.  $\{ \text{wahr} \} \leftrightarrow \{ 1 = 1 \} \quad x := 1 \quad \{ x = 1 \}$

passend umformen

9.  $\{ z = 5 \} \leftrightarrow$   
 $\{ z = 5 \wedge 1 = 1 \} \quad x := 1 \quad \{ z = 5 \wedge x = 1 \}$

passend umformen

## Schlussregeln für Sequenz

### Sequenzregel:

$$\frac{\begin{array}{l} \{P\} \quad S_1 \quad \{Q\} \\ \{Q\} \quad S_2 \quad \{R\} \end{array}}{\{P\} \quad S_1; S_2 \quad \{R\}}$$

Bedeutung:

Wenn  $\{P\} S_1 \{Q\}$  und  $\{Q\} S_2 \{R\}$  korrekte Schlüsse sind, dann ist auch  $\{P\} S_1; S_2 \{R\}$  ein korrekter Schluss

**Beispiel:**

$$\{x>0 \wedge y>0\} \quad a := x; \quad \{a>0 \wedge y>0\}$$

$$\{a>0 \wedge y>0\} \quad b := y; \quad \{a>0 \wedge b>0\}$$

---


$$\{x>0 \wedge y>0\} \quad a := x; b := y; \{a>0 \wedge b>0\}$$

**im Algorithmus die Schritte**

$$\{x>0 \wedge y>0\}$$

$a := x;$

$$\{a>0 \wedge y>0\}$$

**und**

$$\{a>0 \wedge y>0\}$$

$b := y;$

$$\{a>0 \wedge b>0\}$$

**zusammensetzen:**

$$\{x>0 \wedge y>0\}$$

$a := x;$

$$\{a>0 \wedge y>0\}$$

$b := y;$

$$\{a>0 \wedge b>0\}$$

## Konsequenzregeln

Abschwächung der Nachbedingung

$$\frac{\begin{array}{l} \{P\} \quad S \quad \{R\} \\ \{R\} \rightarrow \{Q\} \end{array}}{\{P\} \quad S \quad \{Q\}}$$

Verschärfung der Vorbedingung

$$\frac{\begin{array}{l} \{P\} \rightarrow \{R\} \\ \{R\} \quad S \quad \{Q\} \end{array}}{\{P\} \quad S \quad \{Q\}}$$

**Beispiel:**

$$\{a+b > 0\} \quad x := a+b \quad \{x > 0\}$$

$$\{x > 0\} \quad \rightarrow \quad \{x \geq 0\}$$

---


$$\{a+b > 0\} \quad x := a+b \quad \{x \geq 0\}$$

im Algorithmus können Implikationen in Ausführungsrichtung eingefügt werden:

$$\{a+b > 0\}$$

$x := a+b$

$$\{x > 0\} \rightarrow \{2*x \geq 0\}$$

$y := 2*x$

$$\{y \geq 0\}$$

## Regel für 2-seitige Alternative

$$\frac{\begin{array}{l} \{P \wedge B\} \quad S_1 \{Q\} \\ \{P \wedge \neg B\} \quad S_2 \{Q\} \end{array}}{\{P\} \text{ falls } B : S_1 \text{ sonst } S_2 \{Q\}}$$

Aus der  
**gemeinsamen Vorbedingung P**  
führen beide Zweige auf  
**dieselbe Nachbedingung Q**

### Beispiel:

$$\frac{\begin{array}{l} \{\text{true} \wedge a > 0\} b := a \{b > 0\} \rightarrow \{b \geq 0\} \\ \{\text{true} \wedge \neg(a > 0)\} \rightarrow \{-a \geq 0\} b := -a \{b \geq 0\} \end{array}}{\{\text{true}\} \text{ falls } a > 0 : b := a \text{ sonst } b := -a \{b \geq 0\}}$$

### im Algorithmus:

```
{ a > 0 ∧ b > 0 ∧ a ≠ b }
falls a > b :
    { a > 0 ∧ b > 0 ∧ a > b } →
    { a - b > 0 ∧ b > 0 }
    a := a - b
    { a > 0 ∧ b > 0 }
sonst
    { a > 0 ∧ b > 0 ∧ b > a } →
    { a > 0 ∧ b - a > 0 }
    b := b - a
    { a > 0 ∧ b > 0 }
{ a > 0 ∧ b > 0 }
```

## Regel für bedingte Anweisung

$$\frac{\begin{array}{l} \{P \wedge B\} \quad S \{Q\} \\ P \wedge \neg B \quad \rightarrow Q \end{array}}{\{P\} \text{ falls } B : S \{Q\}}$$

Aus der  
**gemeinsamen Vorbedingung P** führen  
die Anweisung und die Implikation auf  
**dieselbe Nachbedingung Q**

### Beispiel:

$$\frac{\begin{array}{l} \{P \wedge a < 0\} \rightarrow \{-a \geq 0\} a := -a \{a \geq 0\} \\ P \wedge \neg(a < 0) \rightarrow a \geq 0 \end{array}}{\{P\} \text{ falls } a < 0 : a := -a \{a \geq 0\}}$$

### im Algorithmus:

```
{ P }
falls a < 0 :
    { P ∧ a < 0 } → { -a ≥ 0 }
    a := -a;
    { a ≥ 0 }
leere Alternative:
    { P ∧ ¬(a < 0) } → { a ≥ 0 }
{ a ≥ 0 }
```

## Aufrufregel

Der **Unteralgorithmus** UA habe **keine Parameter** und liefere **kein Ergebnis**. Seine **Wirkung auf globale Variable** sei spezifiziert durch die **Vorbedingung P** und die **Nachbedingung Q**.

Dann gilt für einen **Aufruf** von UA die Schlussregel

$$\{ P \} UA() \{ Q \}$$

(Ohne Parameter und Ergebnis ist diese Regel nur von sehr begrenztem Nutzen.)

## Schleifenregel

Wiederholung, Schleife:

$$\frac{\{ INV \wedge B \} S \{ INV \}}{\{ INV \} \text{ solange } B \text{ wiederhole } S \{ INV \wedge \neg B \}}$$

Eine Aussage P heißt **Schleifeninvariante**, wenn man zeigen kann, dass sie an folgenden Stellen gilt: **vor der Schleife**, **vor und nach jeder Ausführung von S** und **nach der Schleife**.

Beispiel: Algorithmus zum Potenzieren

a := x; b := y; z := 1;

{ INV }

solange b > 0 wiederhole

{ INV  $\wedge$  b > 0 }  $\leftrightarrow$  { z · a · a<sup>b-1</sup> = x<sup>y</sup>  $\wedge$  (b-1) ≥ 0 }

b := b - 1;

{ z · a · a<sup>b</sup> = x<sup>y</sup>  $\wedge$  b ≥ 0 }

z := z · a

{ INV }

{ INV  $\wedge$  b ≤ 0 }  $\leftrightarrow$  { z · a<sup>b</sup> = x<sup>y</sup>  $\wedge$  b = 0 }  $\rightarrow$  { z = x<sup>y</sup> }

**INV: z · a<sup>b</sup> = x<sup>y</sup>  $\wedge$  b ≥ 0**

## Terminierung von Schleifen

Die **Terminierung einer Schleife** solange B wiederhole S **muss separat nachgewiesen werden:**

1. Gib einen **ganzzahligen Ausdruck E** an über Variablen, die in der Schleife vorkommen, und zeige, dass E bei jeder Iteration durch S **verkleinert** wird.
2. Zeige, dass **E nach unten begrenzt** ist, z. B. dass  $0 \leq E$  eine Invariante der Schleife ist.

Es kann auch eine andere Grenze als 0 gewählt werden.

E kann auch monoton **vergrößert werden und nach oben begrenzt** sein.

**Nichtterminierung** wird bewiesen, indem man zeigt, dass  $R \wedge B$  eine Invariante der Schleife ist und dass es eine Eingabe gibt, so dass  $R \wedge B$  vor der Schleife gilt. R kann einen speziellen Zustand charakterisieren, in dem die Schleife nicht anhält.

Es gibt Schleifen, für die man **nicht entscheiden** kann, ob sie für jede Vorbedingung **terminieren**.

## Beispiele zur Terminierung (1)

1.

<i>Schleife1</i>	$\{ a > 0 \wedge b > 0 \}$ solange $a \neq b$ wiederhole
<i>Schleife2</i>	solange $a > b$ wiederhole $a := a - b;$
<i>Schleife3</i>	solange $a < b$ wiederhole $b := b - a$

terminiert weil:

**a.**  $INV = a > 0 \wedge b > 0$  ist Invariante für jede der 3 Schleifen, denn

	$\{INV\}$
<i>Schleife1</i>	solange $a \neq b$ wiederhole $\{INV \wedge a \neq b\}$
<i>Schleife2</i>	solange $a > b$ wiederhole $\{INV \wedge a > b\} \rightarrow$ $\{a - b > 0 \wedge b > 0\} a := a - b; \{INV\}$
	$\{INV\}$
<i>Schleife3</i>	solange $a < b$ wiederhole $\{INV \wedge a < b\} \rightarrow$ $\{a > 0 \wedge b - a > 0\} b := b - a \{INV\}$
	$\{INV\}$
	$\{INV\}$

**b.** *Schleife2*: a fällt monoton, weil  $b > 0$ ; a ist begrenzt, weil  $a > 0$ .

*Schleife3*: b fällt monoton, weil  $a > 0$ ; b ist begrenzt, weil  $b > 0$ .

*Schleife1*:  $a+b$  fällt monoton, weil wg.  $a \neq b$  Schl. 2 o. 3 mind. 1x iteriert wird;  $a+b$  begrenzt, wg. INV.

## Beispiele zur Terminierung (2)

2.

$\{ a > 0 \wedge b > 0 \}$   
*Schleife1* solange  $a \neq b$  wiederhole  
*Schleife2* solange  $a \geq b$  wiederhole  
 $a := a - b;$   
*Schleife3* solange  $a < b$  wiederhole  
 $b := b - a$

terminiert nicht immer:

$a > 0$  ist nicht invariant in den Schleifen.

Die Nachbedingung von Schleife 2 ist  $a < b \wedge a \geq 0$ .

Schleife 3 kann erreicht werden im Zustand R:  $a = 0$ , z.B. wenn initial  $a = 2 \cdot b$  gilt.

$a = 0 \wedge a < b$  ist invariant in Schleife 3 und  $a < b$  ist die **Schleifenbedingung**.

$$\{a = 0 \wedge a < b\} \rightarrow \{a = 0 \wedge a < b - a\} \quad b := b - a \quad \{a = 0 \wedge a < b\}$$

## Beispiele zur Terminierung (3)

3.

$\{ n \in \mathbb{N} \wedge n > 1 \}$   
 solange  $n > 1$  wiederhole  
 falls  $n$  gerade:  
 $n := n / 2$   
 sonst  $n := 3 * n + 1$

Terminierung / Nichtterminierung ist unbewiesen;  
 einige Ausführungen mit Anfangswerten  $n$ :

$n$	
2	1
3	10 5 16 8 4 2 1
4	2 1
5	16 8 4 2 1
6	3 10 5 16 8 4 2 1
7	22 11 34 17 52 26 13 50 25 76 38 19 ...

## Denksportaufgabe zu Invarianten

In einem Topf seien  $s$  schwarze und  $w$  weiße Kugeln,  $s + w > 0$

solange mindestens 2 Kugeln im Topf sind

nimm 2 beliebige Kugeln heraus

falls sie gleiche Farbe haben:

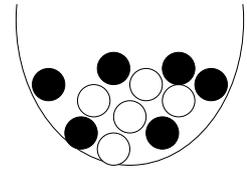
wirf beide weg und

lege eine neue schwarze Kugel in den Topf

falls sie verschiedene Farben haben:

lege die weiße Kugel zurück in den Topf und

wirf die schwarze Kugel weg



**Welche Farbe hat die letzte Kugel?**

**Finden Sie Invarianten, die die Frage beantworten.**

## Schrittweise Konstruktion und Verifikation

Vorbedingung:  $x \in \mathbb{R}$  und  $n \in \mathbb{N}_0$

Nachbedingung:  $q = x^n$

Algorithmus:

```

{ n ≥ 0 } → { n = n ∧ n ≥ 0 ∧ x = x ∧ 1 = 1 }
a := x; q := 1; i := n;
{ i = n ∧ i ≥ 0 ∧ a = x ∧ q = 1 } → { INV }
solange i > 0 wiederhole
  { INV ∧ i > 0 }
  falls i ungerade: { INV ∧ i > 0 ∧ i ungerade } →
    { x^n = q * a * (a^2)^{i/2} ∧ i > 0 } q := q * a; { x^n = q * (a^2)^{i/2} ∧ i > 0 }
  leere Alternative für i gerade:
    { INV ∧ i > 0 ∧ i gerade } → { x^n = q * (a^2)^{i/2} ∧ i > 0 }
  { x^n = q * (a^2)^{i/2} ∧ i > 0 }
  a := a * a;
  { x^n = q * a^{i/2} ∧ i > 0 } → { x^n = q * a^{i/2} ∧ i/2 ≥ 0 }
  i := i / 2
  { x^n = q * a^{i/2} ∧ i ≥ 0 } ↔ { INV }
{ INV ∧ i ≤ 0 } → { q = x^n }

```

Terminierung der Schleife:  $i$  fällt monoton und  $i \geq 0$  ist invariant.

Konstruktionsidee:

Invariante INV:  $x^n = q * a^i \wedge i \geq 0$

Zielbedingung:  $i \leq 0$

falls  $i$  gerade:  $x^n = q * (a^2)^{i/2}$

falls  $i$  ungerade:  $x^n = q * a * (a^2)^{i/2}$

**Schritte:**

1. Vor-, Nachbedingung
2. Schleifeninvariante
3. Schleife mit INV
4. Initialisierung
5. Idee für Schleifenrumpf
6. Alternative
7. Schleife komplett
8. Terminierung