3.2 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 des Algorithmus, über Werte von Variablen nachweisen, ohne den Algorithmus auszuführen. Aussagen **gelten für alle Ausführungen** des Algorithmus. Im Gegensatz zum **dynamischen Testen**: Ausführen des Algorithmus für bestimmte Eingaben.

Durch logische Schlüsse über die Elemente von Algorithmen wird gezeigt, dass

- ein Algorithmus **aus jeder zulässigen Eingabe die geforderte Ausgabe** berechnet, z. B.

{ gegeben }	Algorithmus	{ gesucht }
{ a, b $\in \mathbb{N}$ }	Euklidischer Algorithmus	{ x ist größter gemeinsamer Teiler von a, b }

- eine Aussage an jeder Stelle des Algorithmus gilt, **Invariante**,
  z. B. Kellerinvariante in Mod-2.34
- an einer bestimmten **Stelle im Algorithmus eine bestimmte Aussage gilt**
  z. B. { $\neg$ empty (keller) } keller = pop (keller);
- **Schleifen terminieren.**

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

© 2001 bei Prof. Dr. Uwe Kastens

Verifikation: Algorithmus berechnet 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 ≠ b wiederhole
  { INV $\wedge$ a ≠ 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

© 2001 bei Prof. Dr. Uwe Kastens

Notation von Algorithmenelementen

Anweisungsform	Notation	Beispiel
Sequenz	Anweisung$_1$ **;** Anweisung$_2$	`a := x;` `b := y`
Zuweisung	Variable **:=** Ausdruck	`a := x`

Alternative, zweiseitig	**falls** Bedingung **:**
	Anweisung$_1$
	sonst Anweisung$_2$
		`falls a > b :`
		`  a := a - b`
		`sonst b := b - a`

bedingte Anweisung	**falls** Bedingung **:** Anweisung$_1$	`falls a < 0 :` `  a := - a`
Aufruf eines Unteralgorithmus ua	ua	`berechneGgT`

Schleife	**solange** Bedingung **wiederhole**
	Anweisung
		`solange a≠b wiederhole`
		`  falls a > b:`
		`  ...`

© 2001 bei Prof. Dr. Uwe Kastens

Notation von Aussagen zur Verifikation

Aussagen über den Algorithmenzustand, über Werte von Variablen
werden in den Algorithmus eingefügt:

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

bedeutet: Q gilt immer zwischen der Ausführung von A_1 und der Ausführung von A_2

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

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

{ Vorbedingung P } S { Nachbedingung Q }

nachweisen: Wenn vor der Ausführung des Schrittes S die 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:		gesucht:
Aussagen über die Eingabe		Aussagen über Zusammenhang zwischen Ein- und Ausgabe
{ Vorbedingung }	Algorithmus	{ Nachbedingung }

© 2001 bei Prof. Dr. Uwe Kastens

Schlussregeln für Sequenz

Hoarescher Kalkül definiert **für jede Anweisungsform** ein **Regelschema.**

Sequenzregel:

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

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:

$$\frac{\{\ x>0 \wedge y>0\ \}\ \texttt{a := x;}\ \{\ a>0 \wedge y>0\ \} \qquad \{\ a>0 \wedge y>0\ \}\ \texttt{b := y;}\ \{\ a>0 \wedge b>0\ \}}{\{\ x>0 \wedge y>0\ \}\ \texttt{a := x; b := y;}\ \{\ a>0 \wedge b>0\ \}}$$

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

© 2001 bei Prof. Dr. Uwe Kastens

Konsequenzregeln

Abschwächung der Nachbedingung

$$\frac{\{ P \}\ S\ \{ R \} \qquad \{ R \} \rightarrow \{ Q \}}{\{ P \}\ S\ \{ Q \}}$$

Verschärfung der Vorbedingung

$$\frac{\{ P \} \rightarrow \{ R \} \qquad \{ R \}\ S\ \{ Q \}}{\{ P \}\ S\ \{ Q \}}$$

Beispiel:

$$\frac{\{ a+b > 0 \}\ x := a+b\ \{ x > 0 \} \qquad \{ x > 0 \} \rightarrow \{ x \geq 0 \}}{\{ a+b > 0 \}\ x := a+b\ \{ x \geq 0 \}}$$

im Algorithmus können Implikationen in Ausführungsrichtung eingefügt werden:

$\{ a+b > 0 \}$
$x := a+b$
$\{ x > 0 \} \rightarrow \{ 2*x \geq 0 \}$
$y := 2*x$
$\{ y \geq 0 \}$

© 2001 bei Prof. Dr. Uwe Kastens

Zuweisungsregel

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

$$\{ P \begin{smallmatrix} x \\ e \end{smallmatrix} \} \; x := e \; \{P\}$$

$P \begin{smallmatrix} x \\ e \end{smallmatrix}$ steht für die **Substitution von x durch e in P**, d. h. P [x/e].

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

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 ∧ y>0 }
a := x;
{ a>0 ∧ y>0 }
b := y;
{ a>0 ∧ b>0 }

{ G ist ggT von a-b und b ∧ a-b>0 ∧ b>0 }
a := a – b
{ G ist ggT von a und b ∧ a>0 ∧ b>0 }

© 2001 bei Prof. Dr. Uwe Kastens

Beispiele für Zuweisungsregel

$\{ P^{x}_{e} \}$ x := e { P }

in P x durch e ersetzt

1.	{ a > 0 }	x := a	{ x > 0 }	
2.	{ a > 0 ∧ a > 0 }	x := a	{ x > 0 ∧ a > 0 }	x durch a ersetzen - nicht umgekehrt
3.	{ a > 0 ∧ x = 7 }	x := a	{ x > 0 ∧ x = 7 }	**falscher Schluss**! **alle** x durch a ersetzen!
4.	{ a > 0 ∧ z > 0 }	x := a	{ x > 0 ∧ z > 0 }	z > 0 ist nicht betroffen
5.	{ i + 1 > 0 }	i := i + 1	{ i > 0 }	
6.	{ i ≥ 0} ≡ { i + 1 > 0 }	i := i + 1	{ i > 0 }	passend umformen
7.	{ i = 2 } ≡ { i + 1 = 3 }	i := i + 1	{ i = 3 }	passend umformen
8.	{ wahr } ≡ { 1 = 1 }	x := 1	{ x = 1 }	passend umformen
9.	{ z = 5 } ≡ { z = 5 ∧ 1 = 1 }	x := 1	{ z = 5 ∧ x = 1 }	passend umformen

© 2000 bei Prof. Dr. Uwe Kastens

Aufrufregel

Aufruf eines Unteralgorithmus UA
mit der Spezifikation: Vorbedingung P, Nachbedingung Q:

$$\{ P \}\ UA\ \{ Q \}$$

Diese Regel gilt nur für Unteralgorithmen **ohne Parameter** und **ohne Funktionsergebnis**.
Sie lesen oder schreiben globale Variable. Sie können in P und Q vorkommen.

© 2000 bei Prof. Dr. Uwe Kastens

Regel für 2-seitige Alternative

$$\frac{\{ P \wedge B \}\ S_1\ \{ Q \} \qquad \{ P \wedge \neg B \}\ S_2\ \{ Q \}}{\{ P \}\ \text{falls } B\text{: } S_1 \text{ sonst } S_2\ \{ Q \}}$$

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

Beispiel:

$\{ a > 0 \}\ b := a\ \{ b > 0 \} \rightarrow \{ b \geq 0 \}$

$\{ \neg (a > 0) \} \rightarrow \{ -a \geq 0 \}\ b := -a\ \{ b \geq 0 \}$

$\{ \text{true} \}$ falls $a > 0$: $b := a$ sonst $b := -a\ \{ b \geq 0 \}$

im Algorithmus:

$\{ a>0 \wedge b>0 \wedge a \neq b \}$
`falls a > b :`
$\{ a>0 \wedge b>0 \wedge a>b \} \rightarrow$
$\{ a-b>0 \wedge b>0 \}$
`a := a - b`
$\{ a>0 \wedge b>0 \}$
`sonst`
$\{ a>0 \wedge b>0 \wedge b>a \} \rightarrow$
$\{ a>0 \wedge b-a>0 \}$
`b := b - a`
$\{ a>0 \wedge b>0 \}$
$\{ a>0 \wedge b>0 \}$

© 2001 bei Prof. Dr. Uwe Kastens

Regel für bedingte Anweisung

$$\frac{\{ P \wedge B \}\ \ S\ \{ Q \} \qquad P \wedge \neg B \rightarrow Q}{\{ P \}\ \text{falls } B : S\ \{ Q \}}$$

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

Beispiel:

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

im Algorithmus:

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

© 2001 bei Prof. Dr. Uwe Kastens

Schleifenregel

Wiederholung, Schleife:

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

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** $\}$

INV: $z \cdot a^b = x^y \wedge b \geq 0$

solange $b > 0$ wiederhole

$\{$ **INV** $\wedge\ b > 0\ \} \equiv \{\ z \cdot a \cdot a^{b-1} = x^y \wedge (b-1) \geq 0\ \}$

$b := b - 1;$

$\{\ z \cdot a \cdot a^b = x^y \wedge b \geq 0\ \}$

$z := z \cdot a$

$\{$ **INV** $\}$

$\{$ **INV** $\wedge\ b \leq 0\ \} \rightarrow \{\ z \cdot a^b = x^y \wedge b = 0\ \} \rightarrow \{\ z = x^y\ \}$

© 2001 bei Prof. Dr. Uwe Kastens

Terminierung von Schleifen

Die **Terminierung einer Schleife muss separat nachgewiesen werden**.

solange B wiederhole S

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**.

© 2001 bei Prof. Dr. Uwe Kastens

Beispiele zur Terminierung

1. terminiert

```
{ a > 0 ∧ b > 0}
Schleife1   solange a ≠ b wiederhole
Schleife2           solange a > b wiederhole
                            a := a - b;
Schleife3           solange a < b wiederhole
                            b := b - a
```

2. terminiert nicht immer

```
{ a > 0 ∧ b > 0}
Schleife1   solange a ≠ b wiederhole
Schleife2           solange a ≥ b wiederhole
                            a := a - b;
Schleife3           solange a < b wiederhole
                            b := b - a
```

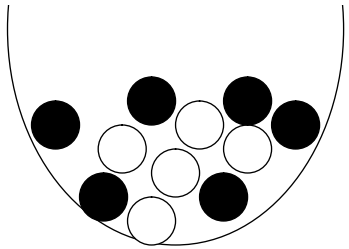
3. Terminierung ist unbewiesen

```
{ n ∈ ℕ ∧ n > 1 }
solange n > 1 wiederhole
        falls n gerade:
                    n := n / 2
        sonst   n := 3 * n + 1
```

© 2001 bei Prof. Dr. Uwe Kastens

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.

© 2000 bei Prof. Dr. Uwe Kastens

Schrittweise Konstruktion und Verifikation

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

Nachbedingung: $q = x^n$

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

Algorithmus:

$\{ n \geq 0 \} \rightarrow \{ n = n \wedge n \geq 0 \wedge x = x \wedge 1 = 1 \}$

```
a := x; q := 1; i := n;
```

$\{ i = n \wedge i \geq 0 \wedge a = x \wedge q = 1 \} \rightarrow \{ INV \}$

```
solange i > 0 wiederhole
```

$\{ INV \wedge i > 0 \}$

```
falls i ungerade:
```

$\{ INV \wedge i > 0 \wedge i \text{ ungerade} \} \rightarrow$

$\{ x^n = q * a * (a^2)^{i/2} \wedge i > 0 \}$ `q := q * a;` $\{ x^n = q * (a^2)^{i/2} \wedge i > 0 \}$

leere Alternative für i gerade:

$\{ INV \wedge i > 0 \wedge i \text{ gerade} \} \rightarrow \{ x^n = q * (a^2)^{i/2} \wedge i > 0 \}$

$\{ x^n = q * (a^2)^{i/2} \wedge i > 0 \}$

```
a := a * a;
```

$\{ x^n = q * a^{i/2} \wedge i > 0 \} \rightarrow \{ x^n = q * a^{i/2} \wedge i/2 \geq 0 \}$

```
i := i / 2
```

$\{ x^n = q * a^i \wedge i \geq 0 \} \equiv \{ INV \}$

$\{ INV \wedge i \leq 0 \} \rightarrow \{ q = x^n \}$

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

Schritte:

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

© 2000 bei Prof. Dr. Uwe Kastens