

Programmverifikation

Proseminar

Ziel

1. Unrechtsbewusstsein
2. Theorie-Praxis
3. Interessante Algorithmen
4. Vortragen

Ablauf

- pro Nase 2 Themen
- pro Thema 30min Vortrag und 15 min Diskussion

Vortrag:

1. Gebiet vorstellen
2. Problem beschreiben
3. Verwendete Kniffe vorstellen
4. Algorithmische Lösung
5. Korrektheit

Diskussion:

1. inhaltlich
2. zur Präsentation

Hoare-Tripel

$$\{P\} \quad p \quad \{Q\}$$

Precondition

Postcondition

eine Assertion

ein Programm-
fragment

eine Assertion

Bedeutung von $\{P\} p \{Q\}$:

Wenn p bei einer Belegung der Programmvariablen gestartet wird, die P erfüllt und terminiert, dann

erfüllt die Belegung der Programmvariablen nach der Ausführung von p Q . “partielle Korrektheit”

Hoare-Kalkül

Skip-Anweisung Sequenz

$$\frac{}{\{P\} \text{ skip } \{P\}}$$

$$\frac{\{P\} p \{Q\} \quad \{Q\} q \{R\}}{\{P\} p ; q \{R\}}$$

Konsequenz

$$\frac{\{P\} p \{Q\} \quad P1 \rightarrow P \quad Q \rightarrow Q1}{\{P1\} p \{Q1\}}$$

Zuweisung

$$\frac{}{\{H[x \leftarrow E]\} \quad x := E \quad \{H\}}$$

fehlen noch: bedingte Anweisungen, Schleifen, (Prozeduren)

Beispiel 2

Wollen zeigen: $\{x = y\} \ x := x + 1; \ y := y + 1 \ \{x = y\}$

2. Zuweisung

$\{x = y + 1\} \ y := y + 1 \ \{x = y\}$

1. Zuweisung

$\{x + 1 = y + 1\} \ x := x + 1 \ \{x = y + 1\}$

Sequenz

$\{x + 1 = y + 1\} \ x := x + 1; \ y := y + 1 \ \{x = y\}$

Abschwächung der Precondition

$\{x = y\} \ x := x + 1; \ y := y + 1 \ \{x = y\}$

Bedingte Anweisungen

$$\frac{\{P \wedge B\} \quad p \{Q\} \quad \{P \wedge \neg B\} \quad q \{Q\}}{\{P\} \text{ if } B \text{ then } p \text{ else } q \text{ end } \{Q\}}$$

Beispiel: geg.: x, y , ges.: $\text{Maximum}(x, y)$

Ziel: $(\max = x \vee \max = y) \wedge \max \geq x \wedge \max \geq y$

```
{x = a ∧ y = b}
if x ≥ y then
    {x = a ∧ y = b ∧ x ≥ y}    max := x    {max = a ∧ y = b ∧ max ≥ y}
else
    {x = a ∧ y = b ∧ ¬x ≥ y}    max := y    {x = a ∧ max = b ∧ ¬x ≥ max}
end
{ (max = a ∨ max = b) ∧ max ≥ a ∧ max ≥ b }
```

While-Schleifen

entspricht Folge

if B then p; if B then p; if B then p; ...
unbekannter Länge

Regel für bedingte Anweisung liefert Vorschlag:

{P} if B then {P ∧ B} p; {P'} if B {P' ∧ B} then p; {P''} if B then ...

Kein Problem, falls $P = P' (= P'' = \dots)$

Also:
$$\frac{\{I \wedge B\} p \{I\}}{\{I\} \text{ while } B \text{ do } p \text{ end } \{I \wedge \neg B\}}$$

I heißt Schleifeninvariante

Bemerkungen zu Schleifeninvarianten

Die Schleifeninvarianten sind das Herzstück des Hoare-Kalküls

Zu überlegen, was sich in einer Schleife nicht ändert, ist zunächst ungewohnt, liefert aber meist interessante Einsichten in die zugrundeliegende algorithmische Idee

Auffinden geeigneter Invarianten ist anspruchsvoll und zurzeit nicht komplett automatisierbar

Beispiel 1: Lineare Suche

geg.: Feld $a[1 \dots n]$, b

Frage: Kommt b in a vor?

```
x := 1;
while x ≤ n ∧ a[x] ≠ b do
    x := x + 1;
end
if x ≤ n then
    answer = yes
else
    answer = no
end
```

Schleifeninvariante:

$$\forall i \ ((1 \leq i \leq n \wedge a[i]=b) \rightarrow x \leq i \leq n)$$

Zu Beginn des Programms

gilt der allg.gültige Ausdruck

$$\forall i \ ((1 \leq i \leq n \wedge a[i]=b) \rightarrow 1 \leq i \leq n)$$

Zuweisungsregel liefert
Schleifeninvariante als
Postcondition

Zu Beginn des Schleifenkörpers
haben wir also

$$\forall i \ ((1 \leq i \leq n \wedge a[i]=b) \rightarrow x \leq i \leq n) \\ \wedge x \leq n \wedge a[x] \neq b$$

Beispiel 1: Lineare Suche

geg.: Feld $a[1 \dots n]$, b

Frage: Kommt b in a vor?

```
x := 1;
while x ≤ n ∧ a[x] ≠ b do
    x := x + 1;
end
if x ≤ n then
    answer = yes
else
    answer = no
end
```

Schleifeninvariante:

$$\forall i ((1 \leq i \leq n \wedge a[i] = b) \rightarrow x \leq i \leq n)$$

Zu Beginn des Schleifenkörpers
haben wir also

$$\forall i ((1 \leq i \leq n \wedge a[i] = b) \rightarrow x \leq i \leq n) \\ \wedge x \leq n \wedge a[x] \neq b$$

Konsequenzregel:

$$\forall i ((1 \leq i \leq n \wedge a[i] = b) \rightarrow x + 1 \leq i \leq n)$$

Zuweisung im Schleifenkörper:

$$\forall i ((1 \leq i \leq n \wedge a[i] = b) \rightarrow x \leq i \leq n)$$

also die Invariante

Beispiel 1: Lineare Suche

geg.: Feld $a[1 \dots n]$, b

Frage: Kommt b in a vor?

```
x := 1;
while x ≤ n ∧ a[x] ≠ b do
    x := x + 1;
end
if x ≤ n then
    answer = yes
else
    answer = no
end
```

Schleifeninvariante:

$$\forall i ((1 \leq i \leq n \wedge a[i] = b) \rightarrow x \leq i \leq n)$$

Also nach der Schleife:

$$\forall i ((1 \leq i \leq n \wedge a[i] = b) \rightarrow x \leq i \leq n) \\ \wedge (x > n \vee a[x] = b)$$

Also im if-Zweig:

$$\forall i ((1 \leq i \leq n \wedge a[i] = b) \rightarrow x \leq i \leq n) \\ \wedge (x > n \vee a[x] = b) \wedge x \leq n \\ \text{also } a[x] = b \wedge x \leq n$$

und im else-Zweig

$$\forall i ((1 \leq i \leq n \wedge a[i] = b) \rightarrow x \leq i \leq n) \\ \wedge (x > n \vee a[x] = b) \wedge x > n,$$

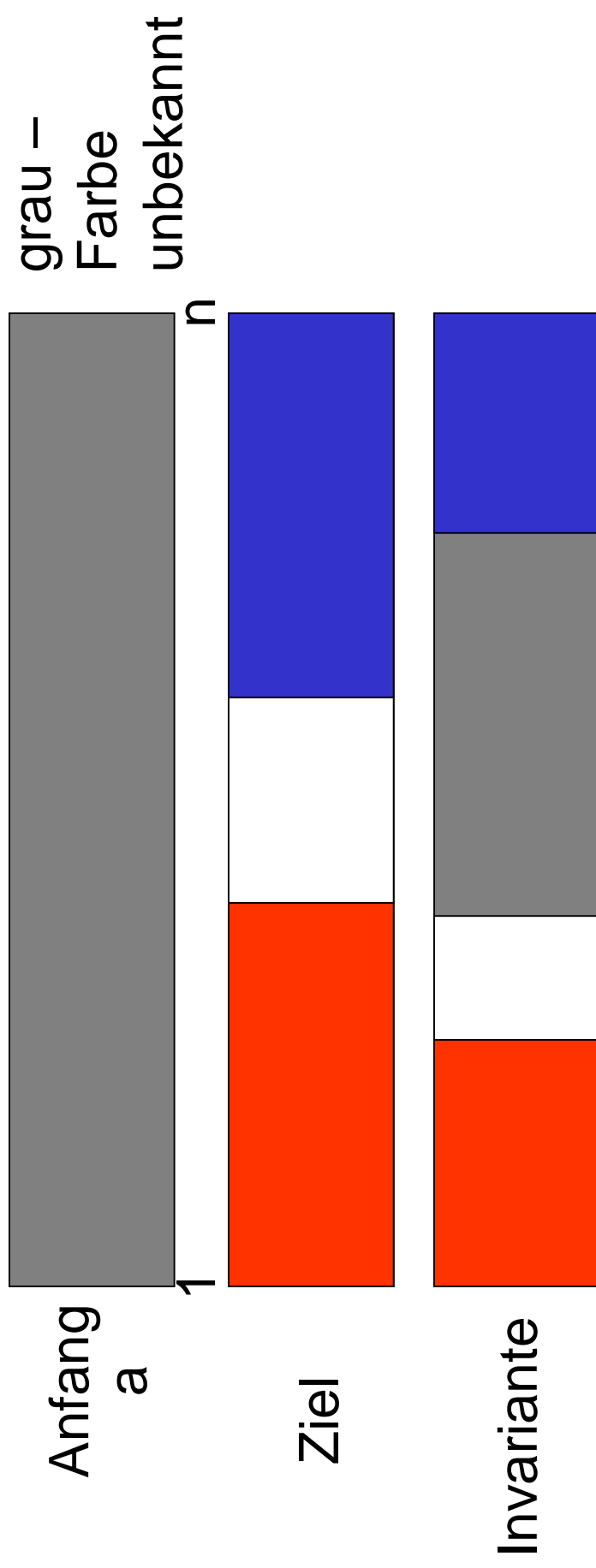
woraus folgt:

$$\forall i ((1 \leq i \leq n \wedge a[i] = b) \rightarrow \perp)$$

2. Beispiel: die holländische Flagge

geg.: Array $a[1..n]$, Inhalt einer der Werte {rot, weiss, blau},
durcheinander

Ziel: durch Nachfragen von Farben einzelner Elements und
Vertauschen je zweier Array-Elemente die Sortierung
rot – weiss – blau herstellen.



3. Beispiel: Euklidischer Algorithmus

geg.: x, y nat. Zahlen ungleich 0

ges: $d = \text{ggT}(x, y)$ ($d \mid x \wedge d \mid y \wedge \forall d' (d \mid x \wedge d \mid y \rightarrow d' \leq d)$)

```
while  $x \neq y$  do
  if  $x > y$  then  $x := x - y$ 
  else           $y := y - x$ 
  end
end
 $d := x$ ;
```

Invariante: $\text{ggT}(x, y) \mid x$
 $\text{ggT}(x, y) \mid x \wedge \text{ggT}(x, y) \mid y$

Totale Korrektheit

= partielle Korrektheit + Terminierung

gefährlich für Terminierung sind nur while-Schleifen

while B do p end

Verfahren: Gib “Abstiegswegfunktion” f in den Programmvariablen a_1, \dots, a_n . Zahlen a_1, \dots, a_n sind gegeben und folgendes leistet

1. $\forall a (a > 0): \{f(a_1, \dots, a_n) = a\} \quad p \quad \{f(a_1, \dots, a_n) < a\}$
2. $f(a_1, \dots, a_n) = 0 \rightarrow \neg B$

Beispiel 1: Lineare Suche

geg.: Feld $a[1 \dots n]$, b

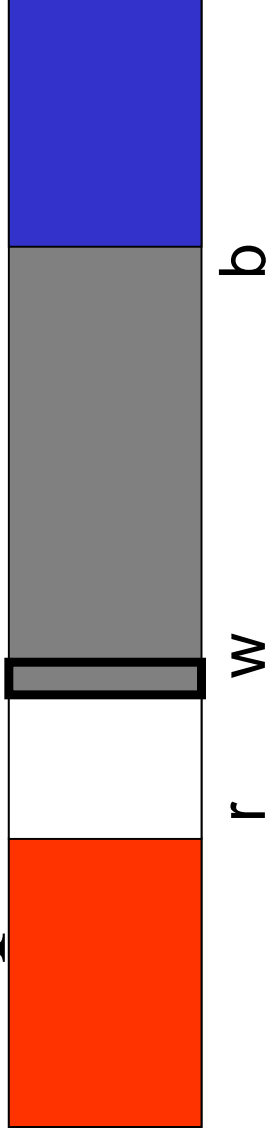
Frage: Kommt b in a vor?

```
x := 1;
while x ≤ n ∧ a[x] ≠ b do
  x := x + 1;
end
if x ≤ n then
  answer = yes
else
  answer = no
end
```

Abstiegssfunktion

$$f(x) = n - x + 1$$

2. Beispiel: holländische Flagge



```
r – das erste nicht rote Element      also Rot = [1...r-1]
w – das erste nicht weisse Element    also Weiss = [r ... w-1]
b – das erste nicht blaue Element     also Blau = [b+1...n]
                                       also Grau: [w...b]

r := 1; w := 1; b := n;
while w ≤ b do
  if rot(a[w]) then swap(a[r],a[w]); r := r+1; w := w+1;
  else if weiss(a[w]) then w := w + 1;
  else /* blau(a[w]) */ swap(a[w],a[b]); b := b - 1;
  end
end
end      Abstiegsfunktion: f(w,b) = b - w + 1
```

3. Beispiel: Euklidischer Algorithmus

geg.: x, y nat. Zahlen ungleich 0

ges: $d = \text{ggT}(x, y)$ $(d \mid x \wedge d \mid y \wedge \forall d' (d \mid x \wedge d \mid y \rightarrow d' \leq d)$

```
while  $x \neq y$  do
  if  $x > y$  then  $x := x - y$ 
  else           $y := y - x$ 
  end
end
 $d := x$ ;
```

Abstiegssfunktion: $f(x) = |x - y|$

Weakest Preconditions

$4 \mid x$ ist stärker als $2 \mid x$

Weakest Preconditions:

$\{?\}$ p $\{Q\}$

Was ist die schwächste Annahme, die ich zusichern muss, damit hinterher Q gilt?

Beispiel: $wp(x := y + 3, x = 28) = "y + 3 = 28"$

Erste Themen

Shell Sort
Heap Sort
Majority Voting ---
Largest Ascending Segment ---
Largest Prefix ---
Saddleback Search --
Maximum True Square ---
Shortest Path ----
Minimum Spanning Tree ---
Multiplication of Polynomials ---
Depth First Search ---
Convex Hull ---
String Matching ---