

Korrekte Software: Grundlagen und Methoden

Vorlesung 10 vom 21.06.22

Vorwärts mit Floyd und Hoare

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2022

Fahrplan

- ▶ Einführung
- ▶ Operationale Semantik
- ▶ Denotationale Semantik
- ▶ Äquivalenz der Operationalen und Denotationalen Semantik
- ▶ Der Floyd-Hoare-Kalkül I
- ▶ Der Floyd-Hoare-Kalkül II: Invarianten
- ▶ Korrektheit des Floyd-Hoare-Kalküls
- ▶ Strukturierte Datentypen
- ▶ Verifikationsbedingungen
- ▶ Vorwärts mit Floyd und Hoare
- ▶ Funktionen und Prozeduren I
- ▶ Funktionen und Prozeduren II
- ▶ Referenzen und Speichermodelle
- ▶ Ausblick und Rückblick

Idee

- Hier ist ein einfaches Programm:

```
// {X = x ∧ Y = y}  
z = y;  
//  
y = x;  
//  
x = z;  
// {X = y ∧ Y = x}
```

Idee

- Hier ist ein einfaches Programm:

```
// {X = x ∧ Y = y}  
z = y;  
//  
y = x;  
// {X = y ∧ Y = z}  
x = z;  
// {X = y ∧ Y = x}
```

Idee

- Hier ist ein einfaches Programm:

```
// {X = x ∧ Y = y}  
z = y;  
// {X = x ∧ Y = z}  
y = x;  
// {X = y ∧ Y = z}  
x = z;  
// {X = y ∧ Y = x}
```

Idee

- Hier ist ein einfaches Programm:

```
// {X = x ∧ Y = y}
z = y;
// {X = x ∧ Y = z}
y = x;
// {X = y ∧ Y = z}
x = z;
// {X = y ∧ Y = x}
```

- Wir haben gesehen:

- 1 Die Verifikation erfolgt **rückwärts** (von hinten nach vorne).
- 2 Die Verifikation kann **berechnet** werden.

Idee

- Hier ist ein einfaches Programm:

```
// {X = x ∧ Y = y}
z = y;
// {X = x ∧ Y = z}
y = x;
// {X = y ∧ Y = z}
x = z;
// {X = y ∧ Y = x}
```

- Wir haben gesehen:
 - 1 Die Verifikation erfolgt **rückwärts** (von hinten nach vorne).
 - 2 Die Verifikation kann **berechnet** werden.
- Muss das rückwärts sein? Warum nicht vorwärts? Was ist der Vorteil?

Nachteile der Rückwärtsberechnung

```
// {i ≠ 3}
.  
. // 400 Zeilen, die  
. // i nicht verändern  
.  
a[i] = 5;  
// {a[3] = 7}
```

Errechnete **Vorbedingung** (AWP)

$$(a[3] == 7)[5/a[i]] = ((i == 3 ? 5 : a[i]) == 7)$$

- ▶ Kann nicht vereinfacht werden, weil wir nicht wissen, ob $i \neq 3$
- ▶ AWP wird **sehr groß**.
- ▶ Das Problem wächst mit der Länge der Programme.

I. Der Floyd-Hoare-Kalkül Vorwärts

Regelanwendung rückwärts

► Um Regel **rückwärts** anwenden zu können:

- 1 **Nachbedingung** der Konklusion muss offene Variable sein
- 2 Alle **Vorbedingungen** der Prämissen müssen disjunkte, offene Variablen sein.
- 3 Gegenbeispiele: while-Regel, if-Regel

► Um Regeln **vorwärts** anwenden zu können:

- 1 **Vorbedingung** der Konklusion muss offene Variable sein
- 2 Alle **Nachbedingungen** der Prämissen müssen disjunkte, offene Variablen sein.
- 3 Gegenbeispiele: ...

Vorwärtsanwendung der Regeln

- Zuweisungsregel kann **nicht vorwärts** angewandt werden, weil die Vorbedingung keine offene Variable ist:

$$\overline{\vdash \{P[e/x]\} x = e \{P\}}$$

Vorwärtsanwendung der Regeln

- Zuweisungsregel kann **nicht vorwärts** angewandt werden, weil die Vorbedingung keine offene Variable ist:

$$\overline{\vdash \{P[e/x]\} x = e \{P\}}$$

- Andere Regeln passen bis auf if-Regel (keine **disjunkten** Variablen)

$$\frac{}{\vdash \{A\} \{ \} \{A\}} \quad \frac{\vdash \{A \wedge b\} c_0 \{B\} \quad \vdash \{A \wedge \neg b\} c_1 \{B\}}{\vdash \{A\} \text{ if } (b) c_0 \text{ else } c_1 \{B\}}$$

$$\frac{\vdash \{A\} c_1 \{B\} \quad \vdash \{B\} c_2 \{C\}}{\vdash \{A\} c_1; c_2 \{C\}}$$

$$\frac{\vdash \{A \wedge b\} c \{A\}}{\vdash \{A\} \text{ while } (b) c \{A \wedge \neg b\}}$$

$$\frac{A' \longrightarrow A \quad \vdash \{A\} c \{B\} \quad B \longrightarrow B'}{\vdash \{A'\} c \{B'\}}$$

Arbeitsblatt 10.1: If-Regel Vorwärts

- ▶ Wie kann die If-Regel vorwärts aussehen?

Arbeitsblatt 10.1: If-Regel Vorwärts

- ▶ Wie kann die If-Regel vorwärts aussehen?

Zuweisungsregel Vorwärts

- ▶ Alternative Zuweisungsregel (nach Floyd):

$$\frac{V \notin FV(P)}{\vdash \{P\} x = e \{ \exists V. P[V/x] \wedge x = (e[V/x]) \}}$$

- ▶ $FV(P)$ sind die **freien** logischen Variablen in P .
- ▶ Jetzt ist die Vorbedingung offen — Regel kann vorwärts angewandt werden
- ▶ Ist keine abgeleitete Regel — muss als korrekt **bewiesen** werden

Arbeitsblatt 10.2: Das Leben mit dem Quantor

- Was bedeutet $\exists V.P?$

Arbeitsblatt 10.2: Das Leben mit dem Quantor

- ▶ Was bedeutet $\exists V.P$?
 - ▶ Die Formel ist wahr, wenn es **irgendeinen** Wert t für V gibt, so dass $P[t/V]$ wahr ist.
- ▶ Was bedeutet $\forall V.P$?

Arbeitsblatt 10.2: Das Leben mit dem Quantor

► Was bedeutet $\exists V.P$?

► Die Formel ist wahr, wenn es **irgendeinen** Wert t für V gibt, so dass $P[t/V]$ wahr ist.

► Was bedeutet $\forall V.P$?

► Die Formel ist wahr, wenn für **alle** Werte t für V $P[t/V]$ wahr ist.

► Sind folgende Formeln wahr (für $x, y \in \mathbb{Z}$)? (Finde Gegenbeispiele oder Zeugen)

$$\exists x. x < 7$$

$$\exists x. x < 3 \wedge x > 7$$

$$\exists x. x < 7 \vee x < 3$$

$$\exists y \exists x. x + 3 = y$$

$$\forall x \exists y. x \cdot y = 3$$

$$\exists x \forall y. x \cdot y = y$$

Vorwärtsverkettung

$$\frac{V \notin FV(P)}{\vdash \{P\} x = e \{ \exists V. P[V/x] \wedge x = e[V/x] \}}$$

```
// {0 ≤ x}  
x = 2*y;  
//  
x = x+1;  
//
```

Vorwärtsverkettung

$$\frac{V \notin FV(P)}{\vdash \{P\} x = e \{ \exists V. P[V/x] \wedge x = e[V/x] \}}$$

```
// {0 ≤ x}  
x = 2*y;  
// {∃ V₁. 0 ≤ V₁ ∧ x = 2 · y}  
x = x+1;  
//
```

Vorwärtsverkettung

$$\frac{V \notin FV(P)}{\vdash \{P\} x = e \{ \exists V. P[V/x] \wedge x = e[V/x] \}}$$

// $\{0 \leq x\}$

x = 2 * y;

// $\{\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y\}$

x = x + 1;

// $\{\exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x + 1)[V_2/x]\}$

Vorwärtsverkettung

$$\frac{V \notin FV(P)}{\vdash \{P\} x = e \{ \exists V. P[V/x] \wedge x = e[V/x] \}}$$

// $\{0 \leq x\}$

x = 2*y;

// $\{\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y\}$

x = x+1;

// $\{\exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x+1)[V_2/x]\}$

► **Vereinfachung** der letzten Nachbedingung:

$$\exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x+1)[V_2/x]$$

Vorwärtsverkettung

$$\frac{V \notin FV(P)}{\vdash \{P\} x = e \{ \exists V. P[V/x] \wedge x = e[V/x] \}}$$

// $\{0 \leq x\}$

x = 2 * y ;

// $\{\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y\}$

x = x + 1 ;

// $\{\exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x + 1)[V_2/x]\}$

► **Vereinfachung** der letzten Nachbedingung:

$$\begin{aligned} & \exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x + 1)[V_2/x] \\ \iff & \exists V_2. (\exists V_1. 0 \leq V_1 \wedge V_2 = 2 \cdot y) \wedge x = V_2 + 1 \end{aligned}$$

Vorwärtsverkettung

$$\frac{V \notin FV(P)}{\vdash \{P\} x = e \{ \exists V. P[V/x] \wedge x = e[V/x] \}}$$

// $\{0 \leq x\}$

x = 2*y;

// $\{\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y\}$

x = x+1;

// $\{\exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x+1)[V_2/x]\}$

► **Vereinfachung** der letzten Nachbedingung:

$$\exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x+1)[V_2/x]$$

$$\iff \exists V_2. (\exists V_1. 0 \leq V_1 \wedge V_2 = 2 \cdot y) \wedge x = V_2 + 1$$

$$\iff \exists V_2. \exists V_1. 0 \leq V_1 \wedge x = V_2 + 1 \wedge V_2 = 2 \cdot y$$

Und jetzt...?

Regeln der Vorwärtsverkettung

Eigenschaften des Existenzquantors:

$$P(V) \wedge V = t \longrightarrow P[t/V] \wedge V = t$$

$$V \notin FV(t) \quad (1)$$

$$\exists V. P(V) \wedge V = t \longrightarrow P[t/V]$$

$$V \notin FV(t) \quad (2)$$

$$(\exists V. P) \wedge Q \iff \exists V. P \wedge Q$$

$$V \notin FV(Q) \quad (3)$$

$$\exists V. P \longrightarrow P$$

$$V \notin FV(P) \quad (4)$$

Regeln der Vorwärtsverkettung

Eigenschaften des Existenzquantors:

$$P(V) \wedge V = t \longrightarrow P[t/V] \wedge V = t \qquad V \notin FV(t) \qquad (1)$$

$$\exists V. P(V) \wedge V = t \longrightarrow P[t/V] \qquad V \notin FV(t) \qquad (2)$$

$$(\exists V. P) \wedge Q \iff \exists V. P \wedge Q \qquad V \notin FV(Q) \qquad (3)$$

$$\exists V. P \longrightarrow P \qquad V \notin FV(P) \qquad (4)$$

Damit gelten folgende Regeln bei der Vorwärtsverkettung:

- ① Wenn x nicht in Vorbedingung auftritt, dann $P[V/x] \equiv P$.
- ② Wenn x nicht in rechter Seite e auftritt, dann $e[V/x] \equiv e$.
- ③ Wenn beides der Fall ist, kann der Existenzquantor wegfallen (4)

Vorwärtsverkettung

$$\frac{V \notin FV(P)}{\vdash \{P\} x = e \{ \exists V. P[V/x] \wedge x = e[V/x] \}}$$

// $\{0 \leq x\}$

x = 2*y;

// $\{\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y\}$

x = x+1;

// $\{\exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x+1)[V_2/x]\}$

► **Vereinfachung** der letzten Nachbedingung:

$$\begin{aligned} & \exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x+1)[V_2/x] \\ \iff & \exists V_2. (\exists V_1. 0 \leq V_1 \wedge V_2 = 2 \cdot y) \wedge x = V_2 + 1 \\ \iff & \exists V_2. \exists V_1. 0 \leq V_1 \wedge x = V_2 + 1 \wedge V_2 = 2 \cdot y \end{aligned}$$

Vorwärtsverkettung

$$\frac{V \notin FV(P)}{\vdash \{P\} x = e \{ \exists V. P[V/x] \wedge x = e[V/x] \}}$$

// $\{0 \leq x\}$

$x = 2 * y;$

// $\{\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y\}$

$x = x + 1;$

// $\{\exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x + 1)[V_2/x]\}$

► **Vereinfachung** der letzten Nachbedingung:

$$\begin{aligned} & \exists V_2. (\exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y)[V_2/x] \wedge x = (x + 1)[V_2/x] \\ \iff & \exists V_2. (\exists V_1. 0 \leq V_1 \wedge V_2 = 2 \cdot y) \wedge x = V_2 + 1 \\ \iff & \exists V_2. \exists V_1. 0 \leq V_1 \wedge x = V_2 + 1 \wedge V_2 = 2 \cdot y \\ \iff & \exists V_1. 0 \leq V_1 \wedge x = 2 \cdot y + 1 \end{aligned}$$

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
//  $\{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$   
t = t + 2;  
//  $\{\exists T. (i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a)[T/t] \wedge t = (t + 2)[T/t]\}$ 
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
//  $\{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$   
t = t + 2;  
//  $\{\exists T. (i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a) [T/t] \wedge t = (t + 2) [T/t]\}$   
//  $\{\exists T. i \cdot i \leq a \wedge T = 2 \cdot i + 1 \wedge s = i \cdot i + T \wedge s \leq a \wedge t = T + 2\}$ 
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
//  $\{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$   
t = t + 2;  
//  $\{\exists T. (i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a) [T/t] \wedge t = (t + 2) [T/t]\}$   
//  $\{\exists T. i \cdot i \leq a \wedge T = 2 \cdot i + 1 \wedge s = i \cdot i + T \wedge s \leq a \wedge t = T + 2\}$   
s = s + t;  
//  $\{\exists S. (\exists T. i \cdot i \leq a \wedge T = 2 \cdot i + 1 \wedge s = i \cdot i + T \wedge s \leq a \wedge t = T + 2) [S/s] \wedge s = (s + t) [S/s]\}$ 
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2}
s = s + t;
// {∃ S. (∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t}
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2}
s = s + t;
// {∃ S. (∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t}
i = i + 1;
// {∃ I. (∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t) [I/i] ∧ i = (i + 1) [I/i]}
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2}
s = s + t;
// {∃ S. (∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t}
i = i + 1;
// {∃ l. (∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t) [l/i] ∧ i = (i + 1) [l/i]}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ T = 2 · l + 1 ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1}
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2}
s = s + t;
// {∃ S. (∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t}
i = i + 1;
// {∃ I. (∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t) [I/i] ∧ i = (i + 1) [I/i]}
// {∃ I. ∃ S. ∃ T. I · I ≤ a ∧ T = 2 · I + 1 ∧ S = I · I + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = I + 1}
// {∃ I. ∃ S. ∃ T. I · I ≤ a ∧ S = I · I + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = I + 1 ∧ T = 2 · I + 1}
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2}
s = s + t;
// {∃ S. (∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t}
i = i + 1;
// {∃ I. (∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t) [I/i] ∧ i = (i + 1) [I/i]}
// {∃ I. ∃ S. ∃ T. I · I ≤ a ∧ T = 2 · I + 1 ∧ S = I · I + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = I + 1}
// {∃ I. ∃ S. ∃ T. I · I ≤ a ∧ S = I · I + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = I + 1 ∧ T = 2 · I + 1}
// {∃ I. ∃ S. I · I ≤ a ∧ S = I · I + 2 · I + 1 ∧ S ≤ a ∧ t = 2 · I + 1 + 2 ∧ s = S + t ∧ i = I + 1}
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2}
s = s + t;
// {∃ S. (∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t}
i = i + 1;
// {∃ l. (∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t) [l/i] ∧ i = (i + 1) [l/i]}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ T = 2 · l + 1 ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1 ∧ T = 2 · l + 1}
// {∃ l. ∃ S. l · l ≤ a ∧ S = l · l + 2 · l + 1 ∧ S ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = S + t ∧ i = l + 1}
// {∃ l. ∃ S. l · l ≤ a ∧ S ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = S + t ∧ i = l + 1 ∧ S = (l + 1) · (l + 1)}
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2}
s = s + t;
// {∃ S. (∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t}
i = i + 1;
// {∃ l. (∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t) [l/i] ∧ i = (i + 1) [l/i]}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ T = 2 · l + 1 ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1 ∧ T = 2 · l + 1}
// {∃ l. ∃ S. l · l ≤ a ∧ S = l · l + 2 · l + 1 ∧ S ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = S + t ∧ i = l + 1}
// {∃ l. ∃ S. l · l ≤ a ∧ S ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = S + t ∧ i = l + 1 ∧ S = (l + 1) · (l + 1)}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = (l + 1) · (l + 1) + t ∧ i = l + 1}
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2}
s = s + t;
// {∃ S. (∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t}
i = i + 1;
// {∃ l. (∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t) [l/i] ∧ i = (i + 1) [l/i]}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ T = 2 · l + 1 ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1 ∧ T = 2 · l + 1}
// {∃ l. ∃ S. l · l ≤ a ∧ S = l · l + 2 · l + 1 ∧ S ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = S + t ∧ i = l + 1}
// {∃ l. ∃ S. l · l ≤ a ∧ S ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = S + t ∧ i = l + 1 ∧ S = (l + 1) · (l + 1)}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = (l + 1) · (l + 1) + t ∧ i = l + 1}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 3 ∧ s = (l + 1) · (l + 1) + t ∧ l = i - 1}
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2}
s = s + t;
// {∃ S. (∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t}
i = i + 1;
// {∃ l. (∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t) [l/i] ∧ i = (i + 1) [l/i]}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ T = 2 · l + 1 ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1 ∧ T = 2 · l + 1}
// {∃ l. ∃ S. l · l ≤ a ∧ S = l · l + 2 · l + 1 ∧ S ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = S + t ∧ i = l + 1}
// {∃ l. ∃ S. l · l ≤ a ∧ S ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = S + t ∧ i = l + 1 ∧ S = (l + 1) · (l + 1)}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = (l + 1) · (l + 1) + t ∧ i = l + 1}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 3 ∧ s = (l + 1) · (l + 1) + t ∧ l = i - 1}
// {(i - 1) · (i - 1) ≤ a ∧ ((i - 1) + 1) · ((i - 1) + 1) ≤ a ∧ t = 2 · (i - 1) + 3 ∧ s = ((i - 1) + 1) · ((i - 1) + 1) + t}
```

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2}
s = s + t;
// {∃ S. (∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t}
i = i + 1;
// {∃ l. (∃ S. ∃ T. i · i ≤ a ∧ T = 2 · i + 1 ∧ S = i · i + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t) [l/i] ∧ i = (i + 1) [l/i]}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ T = 2 · l + 1 ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1}
// {∃ l. ∃ S. ∃ T. l · l ≤ a ∧ S = l · l + T ∧ S ≤ a ∧ t = T + 2 ∧ s = S + t ∧ i = l + 1 ∧ T = 2 · l + 1}
// {∃ l. ∃ S. l · l ≤ a ∧ S = l · l + 2 · l + 1 ∧ S ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = S + t ∧ i = l + 1}
// {∃ l. ∃ S. l · l ≤ a ∧ S ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = S + t ∧ i = l + 1 ∧ S = (l + 1) · (l + 1)}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 1 + 2 ∧ s = (l + 1) · (l + 1) + t ∧ i = l + 1}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 3 ∧ s = (l + 1) · (l + 1) + t ∧ l = i - 1}
// {(i - 1) · (i - 1) ≤ a ∧ ((i - 1) + 1) · ((i - 1) + 1) ≤ a ∧ t = 2 · (i - 1) + 3 ∧ s = ((i - 1) + 1) · ((i - 1) + 1) + t}
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t}
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
//  $\{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$   
t = t + 2;  
//  $\{\exists T. (i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a)[T/t] \wedge t = (t + 2)[T/t]\}$ 
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
//  $\{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$   
t = t + 2;  
//  $\{\exists T. (i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a) [T/t] \wedge t = (t + 2) [T/t]\}$   
//  $\{\exists T. i \cdot i \leq a \wedge s = i \cdot i + T \wedge s \leq a \wedge t = T + 2 \wedge T = 2 \cdot i + 1\}$ 
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
//  $\{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$   
t = t + 2;  
//  $\{\exists T. (i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a) [T/t] \wedge t = (t + 2) [T/t]\}$   
//  $\{\exists T. i \cdot i \leq a \wedge s = i \cdot i + T \wedge s \leq a \wedge t = T + 2 \wedge T = 2 \cdot i + 1\}$   
//  $\{i \cdot i \leq a \wedge s = i \cdot i + 2 \cdot i + 1 \wedge s \leq a \wedge t = (2 \cdot i + 1) + 2\}$ 
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
//  $\{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$   
t = t + 2;  
//  $\{\exists T. (i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a) [T/t] \wedge t = (t + 2) [T/t]\}$   
//  $\{\exists T. i \cdot i \leq a \wedge s = i \cdot i + T \wedge s \leq a \wedge t = T + 2 \wedge T = 2 \cdot i + 1\}$   
//  $\{i \cdot i \leq a \wedge s = i \cdot i + 2 \cdot i + 1 \wedge s \leq a \wedge t = (2 \cdot i + 1) + 2\}$   
//  $\{i \cdot i \leq a \wedge s = (i + 1) \cdot (i + 1) \wedge s \leq a \wedge t = 2 \cdot i + 3\}$ 
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
//  $\{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$   
t = t + 2;  
//  $\{\exists T. (i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a) [T/t] \wedge t = (t + 2) [T/t]\}$   
//  $\{\exists T. i \cdot i \leq a \wedge s = i \cdot i + T \wedge s \leq a \wedge t = T + 2 \wedge T = 2 \cdot i + 1\}$   
//  $\{i \cdot i \leq a \wedge s = i \cdot i + 2 \cdot i + 1 \wedge s \leq a \wedge t = (2 \cdot i + 1) + 2\}$   
//  $\{i \cdot i \leq a \wedge s = (i + 1) \cdot (i + 1) \wedge s \leq a \wedge t = 2 \cdot i + 3\}$   
s = s + t;  
//  $\{\exists S. (i \cdot i \leq a \wedge s = (i + 1) \cdot (i + 1) \wedge s \leq a \wedge t = 2 \cdot i + 3) [S/s] \wedge s = (s + t) [S/s]\}$ 
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
//  $\{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$   
t = t + 2;  
//  $\{\exists T. (i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a) [T/t] \wedge t = (t + 2) [T/t]\}$   
//  $\{\exists T. i \cdot i \leq a \wedge s = i \cdot i + T \wedge s \leq a \wedge t = T + 2 \wedge T = 2 \cdot i + 1\}$   
//  $\{i \cdot i \leq a \wedge s = i \cdot i + 2 \cdot i + 1 \wedge s \leq a \wedge t = (2 \cdot i + 1) + 2\}$   
//  $\{i \cdot i \leq a \wedge s = (i + 1) \cdot (i + 1) \wedge s \leq a \wedge t = 2 \cdot i + 3\}$   
s = s + t;  
//  $\{\exists S. (i \cdot i \leq a \wedge s = (i + 1) \cdot (i + 1) \wedge s \leq a \wedge t = 2 \cdot i + 3) [S/s] \wedge s = (s + t) [S/s]\}$   
//  $\{\exists S. i \cdot i \leq a \wedge S = (i + 1) \cdot (i + 1) \wedge S \leq a \wedge t = 2 \cdot i + 3 \wedge s = S + t\}$ 
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
//  $\{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$   
t = t + 2;  
//  $\{\exists T. (i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a) [T/t] \wedge t = (t + 2) [T/t]\}$   
//  $\{\exists T. i \cdot i \leq a \wedge s = i \cdot i + T \wedge s \leq a \wedge t = T + 2 \wedge T = 2 \cdot i + 1\}$   
//  $\{i \cdot i \leq a \wedge s = i \cdot i + 2 \cdot i + 1 \wedge s \leq a \wedge t = (2 \cdot i + 1) + 2\}$   
//  $\{i \cdot i \leq a \wedge s = (i + 1) \cdot (i + 1) \wedge s \leq a \wedge t = 2 \cdot i + 3\}$   
s = s + t;  
//  $\{\exists S. (i \cdot i \leq a \wedge s = (i + 1) \cdot (i + 1) \wedge s \leq a \wedge t = 2 \cdot i + 3) [S/s] \wedge s = (s + t) [S/s]\}$   
//  $\{\exists S. i \cdot i \leq a \wedge S = (i + 1) \cdot (i + 1) \wedge S \leq a \wedge t = 2 \cdot i + 3 \wedge s = S + t\}$   
//  $\{i \cdot i \leq a \wedge (i + 1) \cdot (i + 1) \leq a \wedge t = 2 \cdot i + 3 \wedge s = (i + 1) \cdot (i + 1) + t\}$ 
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2 ∧ T = 2 · i + 1}
// {i · i ≤ a ∧ s = i · i + 2 · i + 1 ∧ s ≤ a ∧ t = (2 · i + 1) + 2}
// {i · i ≤ a ∧ s = (i + 1) · (i + 1) ∧ s ≤ a ∧ t = 2 · i + 3}
s = s + t;
// {∃ S. (i · i ≤ a ∧ s = (i + 1) · (i + 1) ∧ s ≤ a ∧ t = 2 · i + 3) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. i · i ≤ a ∧ S = (i + 1) · (i + 1) ∧ S ≤ a ∧ t = 2 · i + 3 ∧ s = S + t}
// {i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t}
i = i + 1;
// {∃ l. (i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t) [l/i] ∧ i = (i + 1) [l/i]}
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2 ∧ T = 2 · i + 1}
// {i · i ≤ a ∧ s = i · i + 2 · i + 1 ∧ s ≤ a ∧ t = (2 · i + 1) + 2}
// {i · i ≤ a ∧ s = (i + 1) · (i + 1) ∧ s ≤ a ∧ t = 2 · i + 3}
s = s + t;
// {∃ S. (i · i ≤ a ∧ s = (i + 1) · (i + 1) ∧ s ≤ a ∧ t = 2 · i + 3) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. i · i ≤ a ∧ S = (i + 1) · (i + 1) ∧ S ≤ a ∧ t = 2 · i + 3 ∧ s = S + t}
// {i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t}
i = i + 1;
// {∃ l. (i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t) [l/i] ∧ i = (i + 1) [l/i]}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 3 ∧ s = (l + 1) · (l + 1) + t ∧ i = l + 1}
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2 ∧ T = 2 · i + 1}
// {i · i ≤ a ∧ s = i · i + 2 · i + 1 ∧ s ≤ a ∧ t = (2 · i + 1) + 2}
// {i · i ≤ a ∧ s = (i + 1) · (i + 1) ∧ s ≤ a ∧ t = 2 · i + 3}
s = s + t;
// {∃ S. (i · i ≤ a ∧ s = (i + 1) · (i + 1) ∧ s ≤ a ∧ t = 2 · i + 3) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. i · i ≤ a ∧ S = (i + 1) · (i + 1) ∧ S ≤ a ∧ t = 2 · i + 3 ∧ s = S + t}
// {i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t}
i = i + 1;
// {∃ l. (i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t) [l/i] ∧ i = (i + 1) [l/i]}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 3 ∧ s = (l + 1) · (l + 1) + t ∧ i = l + 1}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 3 ∧ s = (l + 1) · (l + 1) + t ∧ l = i - 1}
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2 ∧ T = 2 · i + 1}
// {i · i ≤ a ∧ s = i · i + 2 · i + 1 ∧ s ≤ a ∧ t = (2 · i + 1) + 2}
// {i · i ≤ a ∧ s = (i + 1) · (i + 1) ∧ s ≤ a ∧ t = 2 · i + 3}
s = s + t;
// {∃ S. (i · i ≤ a ∧ s = (i + 1) · (i + 1) ∧ s ≤ a ∧ t = 2 · i + 3) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. i · i ≤ a ∧ S = (i + 1) · (i + 1) ∧ S ≤ a ∧ t = 2 · i + 3 ∧ s = S + t}
// {i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t}
i = i + 1;
// {∃ l. (i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t) [l/i] ∧ i = (i + 1) [l/i]}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 3 ∧ s = (l + 1) · (l + 1) + t ∧ i = l + 1}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 3 ∧ s = (l + 1) · (l + 1) + t ∧ l = i - 1}
// {(i - 1) · (i - 1) ≤ a ∧ ((i - 1) + 1) · ((i - 1) + 1) ≤ a ∧ t = 2 · (i - 1) + 3 ∧ s = ((i - 1) + 1) · ((i - 1) + 1) + t}
```

Vorwärtsverkettung bei der Arbeit II

Mit Vereinfachung on-the-fly:

```
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a}
t = t + 2;
// {∃ T. (i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t ∧ s ≤ a) [T/t] ∧ t = (t + 2) [T/t]}
// {∃ T. i · i ≤ a ∧ s = i · i + T ∧ s ≤ a ∧ t = T + 2 ∧ T = 2 · i + 1}
// {i · i ≤ a ∧ s = i · i + 2 · i + 1 ∧ s ≤ a ∧ t = (2 · i + 1) + 2}
// {i · i ≤ a ∧ s = (i + 1) · (i + 1) ∧ s ≤ a ∧ t = 2 · i + 3}
s = s + t;
// {∃ S. (i · i ≤ a ∧ s = (i + 1) · (i + 1) ∧ s ≤ a ∧ t = 2 · i + 3) [S/s] ∧ s = (s + t) [S/s]}
// {∃ S. i · i ≤ a ∧ S = (i + 1) · (i + 1) ∧ S ≤ a ∧ t = 2 · i + 3 ∧ s = S + t}
// {i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t}
i = i + 1;
// {∃ l. (i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t) [l/i] ∧ i = (i + 1) [l/i]}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 3 ∧ s = (l + 1) · (l + 1) + t ∧ i = l + 1}
// {∃ l. l · l ≤ a ∧ (l + 1) · (l + 1) ≤ a ∧ t = 2 · l + 3 ∧ s = (l + 1) · (l + 1) + t ∧ l = i - 1}
// {(i - 1) · (i - 1) ≤ a ∧ ((i - 1) + 1) · ((i - 1) + 1) ≤ a ∧ t = 2 · (i - 1) + 3 ∧ s = ((i - 1) + 1) · ((i - 1) + 1) + t}
// {i · i ≤ a ∧ t = 2 · i + 1 ∧ s = i · i + t}
```

Arbeitsblatt 10.3: Vorwärtsverkettung

Gegeben folgendes Programm. Berechnet die Vorwärtsverkettung der Vorbedingung mit Vereinfachung:

```
// {x = X ∧ y = Y}  
x= x+y;  
// {??}  
y= x-y;  
// {??}  
x= x-y;  
// {??}
```

Was bewirkt das Programm?

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \iff \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

$$\begin{aligned} & \models \{P\} x = e \{ \exists V. P[V/x] \wedge x = (e[V/x]) \} \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket x = e \rrbracket_{\mathcal{C}} \longrightarrow \sigma' \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \end{aligned}$$

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \iff \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

$$\begin{aligned} & \models \{P\} x = e \{ \exists V. P[V/x] \wedge x = (e[V/x]) \} \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket x = e \rrbracket_{\mathcal{C}} \longrightarrow \sigma' \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \end{aligned}$$

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \iff \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

$$\begin{aligned} & \models \{P\} x = e \{ \exists V. P[V/x] \wedge x = (e[V/x]) \} \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket x = e \rrbracket_{\mathcal{C}} \longrightarrow \sigma' \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma \models^I (\exists V. P[V/x] \wedge x = (e[V/x]))[e/x] \end{aligned}$$

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \iff \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

$$\begin{aligned} & \models \{P\} x = e \{ \exists V. P[V/x] \wedge x = (e[V/x]) \} \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket x = e \rrbracket_{\mathcal{C}} \longrightarrow \sigma' \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma \models^I (\exists V. P[V/x] \wedge x = (e[V/x]))[e/x] \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma \models^I (\exists V. P[V/x] \wedge e = (e[V/x])) \end{aligned}$$

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \iff \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

$$\begin{aligned} & \models \{P\} x = e \{ \exists V. P[V/x] \wedge x = (e[V/x]) \} \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket x = e \rrbracket_{\mathcal{C}} \longrightarrow \sigma' \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma \models^I (\exists V. P[V/x] \wedge x = (e[V/x]))[e/x] \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma \models^I (\exists V. P[V/x] \wedge e = (e[V/x])) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma \models^I (P[x/x] \wedge e = (e[x/x])) \end{aligned}$$

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \iff \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

$$\begin{aligned} & \models \{P\} x = e \{ \exists V. P[V/x] \wedge x = (e[V/x]) \} \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \wedge \exists \sigma'. (\sigma, \sigma') \in \llbracket x = e \rrbracket_{\mathcal{C}} \longrightarrow \sigma' \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)] \models^I \exists V. P[V/x] \wedge x = (e[V/x]) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma \models^I (\exists V. P[V/x] \wedge x = (e[V/x]))[e/x] \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma \models^I (\exists V. P[V/x] \wedge e = (e[V/x])) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma \models^I (P[x/x] \wedge e = (e[x/x])) \\ \iff & \forall I. \forall \sigma. \sigma \models^I P \longrightarrow \sigma \models^I P \quad \square \end{aligned}$$

Vorwärtsverkettung

- ▶ Vorwärtsregelaxiom äquivalent zum Rückwärtsregelaxiom.
- ▶ Vorteil: Vorbedingung bleibt kleiner
- ▶ Nachteil: in der Anwendung **umständlicher**
- ▶ Die entstehende Nachbedingung beschreibt die **symbolische Auswertung**
- ▶ Vereinfachung benötigt Rechnung mit Existenzquantor

Zwischenfazit: Der Floyd-Hoare-Kalkül ist **symmetrisch**

Es gibt zwei Zuweisungsregeln, eine für die **Rückwärtsanwendung** von Regeln, eine für die **Vorwärtsanwendung**.

II. Vorwärtsberechnung von Verifikationsbedingungen

Stärkste Nachbedingung

- ▶ Vorwärtsberechnung von Verifikationsbedingungen: Nachbedingung
- ▶ Gegeben C0-Programm c , Prädikat P , dann ist
 - ▶ $\text{sp}(P, c)$ die **stärkste Nachbedingung** Q so dass $\models \{P\} c \{Q\}$
 - ▶ Prädikat Q **stärker** als Q' wenn $Q \longrightarrow Q'$.
- ▶ Semantische Charakterisierung:

Stärkste Nachbedingung

Gegeben Zusicherung $P \in \mathbf{Assn}$ und Programm $c \in \mathbf{Stmt}$, dann

$$\models \{P\} c \{Q\} \iff \text{sp}(P, c) \longrightarrow Q$$

- ▶ Wie können wir $\text{sp}(P, c)$ berechnen?

Berechnung von Nachbedingungen

- ▶ Wir berechnen die **approximative** stärkste Nachbedingung.
- ▶ Viele Klauseln sind ähnlich der schwächsten Vorbedingung.
- ▶ Ausnahmen:
 - ▶ While-Schleife: andere Verifikationsbedingungen
 - ▶ If-Anweisung: Weakening eingebaut
 - ▶ **Zuweisung**: Vorwärtsregel
- ▶ Nach jeder Zuweisung Nachbedingung **vereinfachen**

Überblick: Approximative stärkste Nachbedingung

$$\text{asp}(P, \{\}) \stackrel{\text{def}}{=} P$$

Überblick: Approximative stärkste Nachbedingung

$$\begin{aligned}\text{asp}(P, \{\}) &\stackrel{\text{def}}{=} P \\ \text{asp}(P, x = e) &\stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x])\end{aligned}$$

Überblick: Approximative stärkste Nachbedingung

$$\begin{aligned}\text{asp}(P, \{\}) &\stackrel{\text{def}}{=} P \\ \text{asp}(P, x = e) &\stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x]) \\ \text{asp}(P, c_1; c_2) &\stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2)\end{aligned}$$

Überblick: Approximative stärkste Nachbedingung

$$\begin{aligned}\text{asp}(P, \{\}) &\stackrel{\text{def}}{=} P \\ \text{asp}(P, x = e) &\stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x]) \\ \text{asp}(P, c_1; c_2) &\stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2) \\ \text{asp}(P, \text{if } (b) \ c_0 \ \text{else } c_1) &\stackrel{\text{def}}{=} \text{asp}(b \wedge P, c_0) \vee \text{asp}(\neg b \wedge P, c_1)\end{aligned}$$

Überblick: Approximative stärkste Nachbedingung

$$\begin{aligned}\text{asp}(P, \{\}) &\stackrel{\text{def}}{=} P \\ \text{asp}(P, x = e) &\stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x]) \\ \text{asp}(P, c_1; c_2) &\stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2) \\ \text{asp}(P, \text{if } (b) \ c_0 \ \text{else } c_1) &\stackrel{\text{def}}{=} \text{asp}(b \wedge P, c_0) \vee \text{asp}(\neg b \wedge P, c_1) \\ \text{asp}(P, /** \{q\} */) &\stackrel{\text{def}}{=} q\end{aligned}$$

Überblick: Approximative stärkste Nachbedingung

$$\text{asp}(P, \{ \}) \stackrel{\text{def}}{=} P$$

$$\text{asp}(P, x = e) \stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x])$$

$$\text{asp}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2)$$

$$\text{asp}(P, \text{if } (b) \ c_0 \ \text{else} \ c_1) \stackrel{\text{def}}{=} \text{asp}(b \wedge P, c_0) \vee \text{asp}(\neg b \wedge P, c_1)$$

$$\text{asp}(P, /** \{q\} */) \stackrel{\text{def}}{=} q$$

$$\text{asp}(P, \text{while } (b) \ /** \ \text{inv } i \ */ \ c) \stackrel{\text{def}}{=} i \wedge \neg b$$

Überblick: Approximative stärkste Nachbedingung

$$\text{asp}(P, \{ \}) \stackrel{\text{def}}{=} P$$

$$\text{asp}(P, x = e) \stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x])$$

$$\text{asp}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2)$$

$$\text{asp}(P, \text{if } (b) \ c_0 \ \text{else } c_1) \stackrel{\text{def}}{=} \text{asp}(b \wedge P, c_0) \vee \text{asp}(\neg b \wedge P, c_1)$$

$$\text{asp}(P, /** \{q\} */) \stackrel{\text{def}}{=} q$$

$$\text{asp}(P, \text{while } (b) /** \text{inv } i */ c) \stackrel{\text{def}}{=} i \wedge \neg b$$

$$\text{svc}(P, \{ \}) \stackrel{\text{def}}{=} \emptyset$$

Überblick: Approximative stärkste Nachbedingung

$$\text{asp}(P, \{ \}) \stackrel{\text{def}}{=} P$$

$$\text{asp}(P, x = e) \stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x])$$

$$\text{asp}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2)$$

$$\text{asp}(P, \text{if } (b) \ c_0 \ \text{else} \ c_1) \stackrel{\text{def}}{=} \text{asp}(b \wedge P, c_0) \vee \text{asp}(\neg b \wedge P, c_1)$$

$$\text{asp}(P, /** \{q\} */) \stackrel{\text{def}}{=} q$$

$$\text{asp}(P, \text{while } (b) \ /** \ \text{inv } i \ */ \ c) \stackrel{\text{def}}{=} i \wedge \neg b$$

$$\text{svc}(P, \{ \}) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, x = e) \stackrel{\text{def}}{=} \emptyset$$

Überblick: Approximative stärkste Nachbedingung

$$\text{asp}(P, \{ \}) \stackrel{\text{def}}{=} P$$

$$\text{asp}(P, x = e) \stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x])$$

$$\text{asp}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2)$$

$$\text{asp}(P, \text{if } (b) \ c_0 \ \text{else } c_1) \stackrel{\text{def}}{=} \text{asp}(b \wedge P, c_0) \vee \text{asp}(\neg b \wedge P, c_1)$$

$$\text{asp}(P, /** \{q\} */) \stackrel{\text{def}}{=} q$$

$$\text{asp}(P, \text{while } (b) \ /** \text{inv } i \ */ \ c) \stackrel{\text{def}}{=} i \wedge \neg b$$

$$\text{svc}(P, \{ \}) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, x = e) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{svc}(P, c_1) \cup \text{svc}(\text{asp}(P, c_1), c_2)$$

Überblick: Approximative stärkste Nachbedingung

$$\text{asp}(P, \{ \}) \stackrel{\text{def}}{=} P$$

$$\text{asp}(P, x = e) \stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x])$$

$$\text{asp}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2)$$

$$\text{asp}(P, \text{if } (b) \ c_0 \ \text{else} \ c_1) \stackrel{\text{def}}{=} \text{asp}(b \wedge P, c_0) \vee \text{asp}(\neg b \wedge P, c_1)$$

$$\text{asp}(P, /** \{q\} */) \stackrel{\text{def}}{=} q$$

$$\text{asp}(P, \text{while } (b) \ /** \ \text{inv } i \ */ \ c) \stackrel{\text{def}}{=} i \wedge \neg b$$

$$\text{svc}(P, \{ \}) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, x = e) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{svc}(P, c_1) \cup \text{svc}(\text{asp}(P, c_1), c_2)$$

$$\text{svc}(P, \text{if } (b) \ c_0 \ \text{else} \ c_1) \stackrel{\text{def}}{=} \text{svc}(P \wedge b, c_0) \cup \text{svc}(P \wedge \neg b, c_1)$$

Überblick: Approximative stärkste Nachbedingung

$$\text{asp}(P, \{ \}) \stackrel{\text{def}}{=} P$$

$$\text{asp}(P, x = e) \stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x])$$

$$\text{asp}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2)$$

$$\text{asp}(P, \text{if } (b) \ c_0 \ \text{else} \ c_1) \stackrel{\text{def}}{=} \text{asp}(b \wedge P, c_0) \vee \text{asp}(\neg b \wedge P, c_1)$$

$$\text{asp}(P, /** \{q\} */) \stackrel{\text{def}}{=} q$$

$$\text{asp}(P, \text{while } (b) \ /** \text{inv } i \ */ \ c) \stackrel{\text{def}}{=} i \wedge \neg b$$

$$\text{svc}(P, \{ \}) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, x = e) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{svc}(P, c_1) \cup \text{svc}(\text{asp}(P, c_1), c_2)$$

$$\text{svc}(P, \text{if } (b) \ c_0 \ \text{else} \ c_1) \stackrel{\text{def}}{=} \text{svc}(P \wedge b, c_0) \cup \text{svc}(P \wedge \neg b, c_1)$$

$$\text{svc}(P, /** \{q\} */) \stackrel{\text{def}}{=} \{P \longrightarrow q\}$$

Überblick: Approximative stärkste Nachbedingung

$$\text{asp}(P, \{ \}) \stackrel{\text{def}}{=} P$$

$$\text{asp}(P, x = e) \stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x])$$

$$\text{asp}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2)$$

$$\text{asp}(P, \text{if } (b) \ c_0 \ \text{else} \ c_1) \stackrel{\text{def}}{=} \text{asp}(b \wedge P, c_0) \vee \text{asp}(\neg b \wedge P, c_1)$$

$$\text{asp}(P, /** \{q\} */) \stackrel{\text{def}}{=} q$$

$$\text{asp}(P, \text{while } (b) \ /** \text{inv } i \ */ \ c) \stackrel{\text{def}}{=} i \wedge \neg b$$

$$\text{svc}(P, \{ \}) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, x = e) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{svc}(P, c_1) \cup \text{svc}(\text{asp}(P, c_1), c_2)$$

$$\text{svc}(P, \text{if } (b) \ c_0 \ \text{else} \ c_1) \stackrel{\text{def}}{=} \text{svc}(P \wedge b, c_0) \cup \text{svc}(P \wedge \neg b, c_1)$$

$$\text{svc}(P, /** \{q\} */) \stackrel{\text{def}}{=} \{P \longrightarrow q\}$$

$$\text{svc}(P, \text{while } (b) \ /** \text{inv } i \ */ \ c) \stackrel{\text{def}}{=} \text{svc}(i \wedge b, c) \cup \{P \longrightarrow i\} \cup \{\text{asp}(i \wedge b, c) \longrightarrow i\}$$

Überblick: Approximative stärkste Nachbedingung

$$\text{asp}(P, \{ \}) \stackrel{\text{def}}{=} P$$

$$\text{asp}(P, x = e) \stackrel{\text{def}}{=} \exists V. P[V/x] \wedge x = (e[V/x])$$

$$\text{asp}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{asp}(\text{asp}(P, c_1), c_2)$$

$$\text{asp}(P, \text{if } (b) \ c_0 \ \text{else } c_1) \stackrel{\text{def}}{=} \text{asp}(b \wedge P, c_0) \vee \text{asp}(\neg b \wedge P, c_1)$$

$$\text{asp}(P, /** \{q\} */) \stackrel{\text{def}}{=} q$$

$$\text{asp}(P, \text{while } (b) \ /** \text{inv } i \ */ \ c) \stackrel{\text{def}}{=} i \wedge \neg b$$

$$\text{svc}(P, \{ \}) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, x = e) \stackrel{\text{def}}{=} \emptyset$$

$$\text{svc}(P, c_1; c_2) \stackrel{\text{def}}{=} \text{svc}(P, c_1) \cup \text{svc}(\text{asp}(P, c_1), c_2)$$

$$\text{svc}(P, \text{if } (b) \ c_0 \ \text{else } c_1) \stackrel{\text{def}}{=} \text{svc}(P \wedge b, c_0) \cup \text{svc}(P \wedge \neg b, c_1)$$

$$\text{svc}(P, /** \{q\} */) \stackrel{\text{def}}{=} \{P \longrightarrow q\}$$

$$\text{svc}(P, \text{while } (b) \ /** \text{inv } i \ */ \ c) \stackrel{\text{def}}{=} \text{svc}(i \wedge b, c) \cup \{P \longrightarrow i\} \cup \{\text{asp}(i \wedge b, c) \longrightarrow i\}$$

$$\text{svc}(\{P\} \ c \ \{Q\}) \stackrel{\text{def}}{=} \{\text{asp}(P, c) \longrightarrow Q\} \cup \text{svc}(P, c)$$

Beispiel: Fakultät

```
1 // {0 ≤ n}
2 p = 1;
3 c = 1;
4 while (c ≤ n) /** inv {p = (c - 1)! ∧ c - 1 ≤ n}; */
5     p = p * c;
6     c = c + 1;
7 }
8 // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung

Notation: asp_x = Stärkste Nachbedingung **nach** Zeile x .

```
1  // {0 ≤ n}
2  p = 1;
   //
   //
3  c = 1;
   //
   //
4  while (c ≤ n) /** inv {p = (c - 1)! ∧ c - 1 ≤ n}; */ {
5      p = p * c;
       //
6      c = c + 1;
       //
7  }
   //
8  // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung

Notation: asp_x = Stärkste Nachbedingung **nach** Zeile x .

```
1  // {0 ≤ n}
2  p = 1;
   // {∃ V. 0 ≤ n[V/p] ∧ p = (1[V/p])}
   //
3  c = 1;
   //
   //
4  while (c ≤ n) /** inv {p = (c - 1)! ∧ c - 1 ≤ n}; */ {
5      p = p * c;
       //
6      c = c + 1;
       //
7  }
   //
8  // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung

Notation: asp_x = Stärkste Nachbedingung **nach** Zeile x .

```
1  // {0 ≤ n}
2  p = 1;
   // {∃ V. 0 ≤ n[V/p] ∧ p = (1[V/p])}
   // {0 ≤ n ∧ p = 1}
3  c = 1;
   //
   //
4  while (c ≤ n) /** inv {p = (c - 1)! ∧ c - 1 ≤ n}; */ {
5      p = p * c;
       //
6      c = c + 1;
       //
7  }
   //
8  // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung

Notation: asp_x = Stärkste Nachbedingung **nach** Zeile x .

```
1  // {0 ≤ n}
2  p = 1;
   // {∃V. 0 ≤ n[V/p] ∧ p = (1[V/p])}
   // {0 ≤ n ∧ p = 1}
3  c = 1;
   // {∃V. (0 ≤ n ∧ p = 1)[V/c] ∧ c = (1[V/c])}
   //
4  while (c ≤ n) /** inv {p = (c - 1)! ∧ c - 1 ≤ n}; */ {
5      p = p * c;
       //
6      c = c + 1;
       //
7  }
   //
8  // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung

Notation: asp_x = Stärkste Nachbedingung **nach** Zeile x .

```
1 // {0 ≤ n}
2 p = 1;
  // {∃V. 0 ≤ n[V/p] ∧ p = (1[V/p])}
  // {0 ≤ n ∧ p = 1}
3 c = 1;
  // {∃V. (0 ≤ n ∧ p = 1)[V/c] ∧ c = (1[V/c])}
  // {0 ≤ n ∧ p = 1 ∧ c = 1}
4 while (c ≤ n) /** inv {p = (c - 1)! ∧ c - 1 ≤ n}; */ {
5   p = p * c;
  //
6   c = c + 1;
  //
7 }
  //
8 // {p = n!}
```

$$VC_1 = \{asp_3 \longrightarrow p = (c - 1)! \wedge c - 1 \leq n\}$$

Beispiel: Fakultät, stärkste Nachbedingung

Notation: asp_x = Stärkste Nachbedingung **nach** Zeile x .

```
1 // {0 ≤ n}
2 p = 1;
  // {∃V. 0 ≤ n[V/p] ∧ p = (1[V/p])}
  // {0 ≤ n ∧ p = 1}
3 c = 1;
  // {∃V. (0 ≤ n ∧ p = 1)[V/c] ∧ c = (1[V/c])}
  // {0 ≤ n ∧ p = 1 ∧ c = 1}
4 while (c ≤ n) /** inv {p = (c - 1)! ∧ c - 1 ≤ n}; */ {
5   p = p * c;
  //
6   c = c + 1;
  //
7 }
  // {¬(c ≤ n) ∧ p = (c - 1)! ∧ c - 1 ≤ n}
8 // {p = n!}
```

$$VC_1 = \{asp_3 \longrightarrow p = (c - 1)! \wedge c - 1 \leq n\}$$

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf)

```
1 // {0 ≤ n}
2 p = 1;
  // {0 ≤ n ∧ p = 1}
3 c = 1;
  // {0 ≤ n ∧ p = 1 ∧ c = 1}
4 while (c ≤ n) /** inv p = (c-1)! ∧ c-1 ≤ n; */ \{
5   p = p * c;
  //
  //
  //
  c = c + 1;
  //
  //
  //
  }
  // {¬(c ≤ n) ∧ p = (c-1)! ∧ c-1 ≤ n}
8 // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf)

```
1 // {0 ≤ n}
2 p = 1;
  // {0 ≤ n ∧ p = 1}
3 c = 1;
  // {0 ≤ n ∧ p = 1 ∧ c = 1}
4 while (c ≤ n) /** inv p = (c-1)! ∧ c-1 ≤ n; */ \{
5   p = p * c;
    // {∃V1. (p = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n)[V1/p] ∧ p = (p·c)[V1/p]}
    //
    //
    c = c + 1;
    //
    //
    //
  }
  // {¬(c ≤ n) ∧ p = (c-1)! ∧ c-1 ≤ n}
8 // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf)

```
1 // {0 ≤ n}
2 p = 1;
  // {0 ≤ n ∧ p = 1}
3 c = 1;
  // {0 ≤ n ∧ p = 1 ∧ c = 1}
4 while (c ≤ n) /** inv p = (c-1)! ∧ c-1 ≤ n; */ \{
5   p = p * c;
    // {∃ V1. (p = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n)[V1/p] ∧ p = (p · c)[V1/p]}
    // {∃ V1. (V1 = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n) ∧ p = (V1 · c)}
    //
    c = c + 1;
    //
    //
    //
  }
  // {¬(c ≤ n) ∧ p = (c-1)! ∧ c-1 ≤ n}
8 // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf)

```
1  // {0 ≤ n}
2  p = 1;
   // {0 ≤ n ∧ p = 1}
3  c = 1;
   // {0 ≤ n ∧ p = 1 ∧ c = 1}
4  while (c ≤ n) /** inv p = (c-1)! ∧ c-1 ≤ n; */ \{
5    p = p * c;
   // {∃ V1. (p = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n)[V1/p] ∧ p = (p · c)[V1/p]}
   // {∃ V1. (V1 = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n) ∧ p = (V1 · c)}
   // {c-1 ≤ n ∧ c ≤ n ∧ p = (c-1)! · c}
6    c = c + 1;
   //
   //
   //
   }
   // {¬(c ≤ n) ∧ p = (c-1)! ∧ c-1 ≤ n}
8  // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf)

```
1 // {0 ≤ n}
2 p = 1;
  // {0 ≤ n ∧ p = 1}
3 c = 1;
  // {0 ≤ n ∧ p = 1 ∧ c = 1}
4 while (c ≤ n) /** inv p = (c-1)! ∧ c-1 ≤ n; */ \{
5   p = p * c;
  // {∃ V1. (p = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n)[V1/p] ∧ p = (p · c)[V1/p]}
  // {∃ V1. (V1 = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n) ∧ p = (V1 · c)}
  // {c-1 ≤ n ∧ c ≤ n ∧ p = (c-1)! · c}
6   c = c + 1;
  // {∃ V2. (c-1 ≤ n ∧ c ≤ n ∧ p = (c-1)! · c)[V2/c] ∧ c = (c+1)[V2/c]}
  //
  //
  }
  // {¬(c ≤ n) ∧ p = (c-1)! ∧ c-1 ≤ n}
8 // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf)

```
1 // {0 ≤ n}
2 p = 1;
  // {0 ≤ n ∧ p = 1}
3 c = 1;
  // {0 ≤ n ∧ p = 1 ∧ c = 1}
4 while (c ≤ n) /** inv p = (c-1)! ∧ c-1 ≤ n; */ \{
5   p = p * c;
    // {∃ V1. (p = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n)[V1/p] ∧ p = (p · c)[V1/p]}
    // {∃ V1. (V1 = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n) ∧ p = (V1 · c)}
    // {c-1 ≤ n ∧ c ≤ n ∧ p = (c-1)! · c}
6   c = c + 1;
    // {∃ V2. (c-1 ≤ n ∧ c ≤ n ∧ p = (c-1)! · c)[V2/c] ∧ c = (c+1)[V2/c]}
    // {∃ V2. (V2-1 ≤ n ∧ V2 ≤ n ∧ p = (V2-1)! · V2) ∧ c = (V2+1)}
    //
  }
  // {¬(c ≤ n) ∧ p = (c-1)! ∧ c-1 ≤ n}
8 // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf)

```
1 // {0 ≤ n}
2 p = 1;
  // {0 ≤ n ∧ p = 1}
3 c = 1;
  // {0 ≤ n ∧ p = 1 ∧ c = 1}
4 while (c ≤ n) /** inv p = (c-1)! ∧ c-1 ≤ n; */ \{
5   p = p * c;
    // {∃ V1. (p = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n)[V1/p] ∧ p = (p · c)[V1/p]}
    // {∃ V1. (V1 = (c-1)! ∧ (c-1) ≤ n ∧ c ≤ n) ∧ p = (V1 · c)}
    // {c-1 ≤ n ∧ c ≤ n ∧ p = (c-1)! · c}
6   c = c + 1;
    // {∃ V2. (c-1 ≤ n ∧ c ≤ n ∧ p = (c-1)! · c)[V2/c] ∧ c = (c+1)[V2/c]}
    // {∃ V2. (V2-1 ≤ n ∧ V2 ≤ n ∧ p = (V2-1)! · V2) ∧ c = (V2+1)}
    // {c-2 ≤ n ∧ c-1 ≤ n ∧ p = (c-2)! · (c-1)}
7 }
  // {¬(c ≤ n) ∧ p = (c-1)! ∧ c-1 ≤ n}
8 // {p = n!}
```

Beispiel: Fakultät, Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1 // {0 ≤ n}
2 p = 1;
  //  $\text{svc}_2 = \emptyset$ 
  c = 1;
  //  $\text{svc}_3 = \emptyset$ 
4 while (c ≤ n) /** inv { $p = (c - 1)! \wedge c - 1 \leq n$ }; */ {
5   p = p * c;
  //  $\text{svc}_5 = \emptyset$ 
6   c = c + 1;
  //  $\text{svc}_6 = \emptyset$ 
7 }
  //  $\text{svc}_4 = \{ \text{asp}_3 \longrightarrow (p = (c - 1)! \wedge c - 1 \leq n),$ 
  //            $\text{asp}_6 \longrightarrow (p = (c - 1)! \wedge c - 1 \leq n) \}$ 
  //
  //
  //
  //
8 // { $p = n!$ }
```

Beispiel: Fakultät, Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1 // {0 ≤ n}
2 p = 1;
  //  $\text{svc}_2 = \emptyset$ 
  c = 1;
  //  $\text{svc}_3 = \emptyset$ 
4 while (c ≤ n) /** inv { $p = (c - 1)! \wedge c - 1 \leq n$ }; */ {
5   p = p * c;
    //  $\text{svc}_5 = \emptyset$ 
6   c = c + 1;
    //  $\text{svc}_6 = \emptyset$ 
7 }
  //  $\text{svc}_4 = \{ \text{asp}_3 \rightarrow (p = (c - 1)! \wedge c - 1 \leq n),$ 
  //            $\text{asp}_6 \rightarrow (p = (c - 1)! \wedge c - 1 \leq n) \}$ 
  //  $\text{svc}_4 = \{ (0 \leq n \wedge p = 1 \wedge c = 1) \rightarrow (p = (c - 1)! \wedge c - 1 \leq n),$ 
  //            $(c - 2 \leq n \wedge c - 1 \leq n \wedge p = (c - 2)! \cdot (c - 1))$ 
  //            $\rightarrow (p = (c - 1)! \wedge c - 1 \leq n) \}$ 
8 // { $p = n!$ }
```

Schließlich zu zeigen

$$\begin{aligned} \text{svc}_8 &= \{ \{ \text{asp}_8 \longrightarrow p = n! \} \cup \text{svc}_4 \\ &= \{ (p = (c - 1)! \wedge c - 1 \leq n \ \&\& \ \neg(c \leq n)) \longrightarrow p = n! \}, \\ &\quad (0 \leq n \wedge p = 1 \wedge c = 1) \longrightarrow (p = (c - 1)! \wedge c - 1 \leq n), \\ &\quad (c - 2 \leq n \wedge c - 1 \leq n \wedge p = (c - 2)! \cdot (c - 1)) \\ &\quad \longrightarrow (p = (c - 1)! \wedge c - 1 \leq n) \} \\ &\rightsquigarrow \{ \text{true} \} \end{aligned}$$

Arbeitsblatt 10.4: Jetzt seid ihr dran!

Berechnet die stärkste Nachbedingung und Verifikationsbedingungen für die ganzzahlige Division:

```
1  /** {  $0 \leq a$  } */
2  r = a;
3  q = 0;
4  while (b <= r) /** inv {  $a = b*q + r \wedge 0 \leq r$  } */ {
5      r = r - b;
6      q = q + 1;
7  }
8  /** {  $a = b*q + r \wedge 0 \leq r \wedge r < b$  } */
```

Beispiel: Suche nach dem Maximalen Element

```
1  // {0 < n}
2  i= 0;
3  //
4  //
5  r= 0;
6  //
7  while ( i!= n)
8      /** inv (∀j. 0 ≤ j < i → a[j] ≤ a[r]) ∧ 0 ≤ i ≤ n ∧ 0 ≤ r < n */ {
9      if (a[r] < a[i]) {
10         r= i;
11     }
12     else {
13     }
14     i= i+1;
15 }
16 //
17 // {(∀j. 0 ≤ j < n → a[j] ≤ a[r]) ∧ 0 ≤ r < n}
```

Beispiel: Suche nach dem Maximalen Element

```
1  // {0 < n}
2  i= 0;
3  // { $\exists l_0. (0 < n)[l_0/i] \wedge i = 0[l_0/i]$ }
4  //
5  r= 0;
6  //
7  while (i != n)
8      /** inv ( $\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]$ )  $\wedge 0 \leq i \leq n \wedge 0 \leq r < n$  */ {
9          if (a[r] < a[i]) {
10             r= i;
11         }
12         else {
13             }
14         i= i+1;
15     }
16 //
17 // {( $\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r]$ )  $\wedge 0 \leq r < n$ }
```

Beispiel: Suche nach dem Maximalen Element

```
1 // {0 < n}
2 i = 0;
3 // { $\exists l_0. (0 < n)[l_0/i] \wedge i = 0[l_0/i]$ }
4 // {0 < n  $\wedge$  i = 0}
5 r = 0;
6 //
7 while (i != n)
8   /** inv ( $\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]$ )  $\wedge 0 \leq i \leq n \wedge 0 \leq r < n$  */ {
9     if (a[r] < a[i]) {
10       r = i;
11     }
12     else {
13     }
14     i = i + 1;
15   }
16 //
17 // {( $\forall j. 0 \leq j < n \rightarrow a[j] \leq a[r]$ )  $\wedge 0 \leq r < n$ }
```

Beispiel: Suche nach dem Maximalen Element

```
1 // {0 < n}
2 i = 0;
3 // { $\exists l_0. (0 < n)[l_0/i] \wedge i = 0[l_0/i]$ }
4 // {0 < n  $\wedge$  i = 0}
5 r = 0;
6 // {0 < n  $\wedge$  i = 0  $\wedge$  r = 0}
7 while (i != n)
8   /** inv ( $\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]$ )  $\wedge$  0  $\leq$  i  $\leq$  n  $\wedge$  0  $\leq$  r < n */ {
9     if (a[r] < a[i]) {
10       r = i;
11     }
12     else {
13     }
14     i = i + 1;
15   }
16 //
17 // {( $\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r]$ )  $\wedge$  0  $\leq$  r < n}
```

Beispiel: Suche nach dem Maximalen Element

```
1  // {0 < n}
2  i = 0;
3  // { $\exists l_0. (0 < n)[l_0/i] \wedge i = 0[l_0/i]$ }
4  // {0 < n  $\wedge$  i = 0}
5  r = 0;
6  // {0 < n  $\wedge$  i = 0  $\wedge$  r = 0}
7  while (i != n)
8      /** inv ( $\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]$ )  $\wedge$  0  $\leq$  i  $\leq$  n  $\wedge$  0  $\leq$  r < n */ {
9          if (a[r] < a[i]) {
10             r = i;
11         }
12         else {
13             }
14         i = i + 1;
15     }
16 // { $\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]$ }  $\wedge$  0  $\leq$  i  $\leq$  n  $\wedge$  0  $\leq$  r < n  $\wedge$   $\neg(i \neq n)$ 
17 // { $(\forall j. 0 \leq j < n \rightarrow a[j] \leq a[r]) \wedge 0 \leq r < n$ }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
1  while (i != n)
2    /** inv ( $\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]$ )  $\wedge 0 \leq i \leq n \wedge 0 \leq r < n$  */ {
3    if (a[r] < a[i]) {
4      // {( $\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]$ )  $\wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i]$ }
5      r = i;
6      //
7      //
8    }
9    else {
10     //
11    }
12    //
13    i = i + 1;
14    //
15  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
1  while (i != n)
2    /** inv  $(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n$  */ {
3    if (a[r] < a[i]) {
4      //  $\{(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i]\}$ 
5      r = i;
6      //  $\{\exists R_0. ((\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i])[R_0/r] \wedge r = i[R_0/r]\}$ 
7      //
8    }
9    else {
10     //
11    }
12    //
13    i = i + 1;
14    //
15  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
1  while (i != n)
2    /** inv  $(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n$  */ {
3    if (a[r] < a[i]) {
4      //  $\{(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i]\}$ 
5      r = i;
6      //  $\{\exists R_0. ((\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i])[R_0/r] \wedge r = i[R_0/r]\}$ 
7      //  $\{\exists R_0. (\forall j. 0 \leq j < i \rightarrow a[j] \leq a[R_0]) \wedge 0 \leq i \leq n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[i] \wedge r = i\}$ 
8    }
9    else {
10     //
11    }
12    //
13    i = i + 1;
14    //
15  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
1  while (i != n)
2    /** inv  $(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n$  */ {
3    if (a[r] < a[i]) {
4      //  $\{(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i]\}$ 
5      r = i;
6      //  $\{\exists R_0. ((\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i])[R_0/r] \wedge r = i[R_0/r]\}$ 
7      //  $\{\exists R_0. (\forall j. 0 \leq j < i \rightarrow a[j] \leq a[R_0]) \wedge 0 \leq i \leq n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[i] \wedge r = i\}$ 
8    }
9    else {
10     //  $\{(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(a[r] < a[i])\}$ 
11    }
12    //
13    i = i + 1;
14    //
15  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
1  while (i != n)
2    /** inv  $(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n$  */ {
3    if (a[r] < a[i]) {
4      //  $\{(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i]\}$ 
5      r = i;
6      //  $\{\exists R_0. ((\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i])[R_0/r] \wedge r = i[R_0/r]\}$ 
7      //  $\{\exists R_0. (\forall j. 0 \leq j < i \rightarrow a[j] \leq a[R_0]) \wedge 0 \leq i \leq n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[i] \wedge r = i\}$ 
8    }
9    else {
10     //  $\{(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(a[r] < a[i])\}$ 
11    }
12    //  $\{(\exists R_0. (\forall j. 0 \leq j < i \rightarrow a[j] \leq a[R_0]) \wedge 0 \leq i \leq n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[i] \wedge r = i) \}$ 
        $\vee ((\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[i] \leq a[r])$ 
13   i = i + 1;
14   //
15 }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
1  while (i != n)
2    /** inv  $(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n$  */ {
3      if (a[r] < a[i]) {
4        //  $\{(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i]\}$ 
5        r = i;
6        //  $\{\exists R_0. ((\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[r] < a[i])[R_0/r] \wedge r = i[R_0/r]\}$ 
7        //  $\{\exists R_0. (\forall j. 0 \leq j < i \rightarrow a[j] \leq a[R_0]) \wedge 0 \leq i \leq n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[i] \wedge r = i\}$ 
8      }
9      else {
10       //  $\{(\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(a[r] < a[i])\}$ 
11     }
12     //  $\{(\exists R_0. (\forall j. 0 \leq j < i \rightarrow a[j] \leq a[R_0]) \wedge 0 \leq i \leq n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[i] \wedge r = i) \}$ 
13     //  $\vee ((\forall j. 0 \leq j < i \rightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge a[i] \leq a[r])$ 
14     i = i + 1;
15     //  $\{\exists l_0. ((\exists R_0. (\forall j. 0 \leq j < l_0 \rightarrow a[j] \leq a[R_0]) \wedge 0 \leq l_0 < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[l_0] \wedge r = l_0) \}$ 
16     //  $\vee ((\forall j. 0 \leq j < l_0 \rightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r])) \wedge i = l_0 + 1$ 
17   }
```

Verifikationsbedingungen

- 1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n$
- 2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow$
 $(\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n$
- 3 $(\exists l_0. ((\exists R_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq l_0 < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[l_0] \wedge r = l_0)$
 $\vee ((\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r])) \wedge i = l_0 + 1)$
 $\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n$

Weitere Vereinfachungsregeln

Existenzquantoren und Disjunktionen können mit folgenden Regeln vereinfacht werden:

⑥ Der Gültigkeitsbereich des Existenzquantors kann verkleinert werden:

$$\blacktriangleright (\exists x. P \vee Q) \rightsquigarrow (\exists x. P) \vee (\exists x. Q)$$

⑦ Disjunktionen in der Prämisse ergeben eine Fallunterscheidung:

$$\blacktriangleright A_1 \vee A_2 \longrightarrow B \rightsquigarrow A_1 \longrightarrow B, A_2 \longrightarrow B$$

⑧ Konjunktion distribuiert über Disjunktion:

$$\blacktriangleright (A_1 \vee A_2) \wedge B \rightsquigarrow (A_1 \wedge B) \vee (A_2 \wedge B)$$

⑨ ... und andersherum:

$$\blacktriangleright (A_1 \wedge A_2) \vee B \rightsquigarrow (A_1 \vee B) \wedge (A_2 \vee B)$$

Vereinfachte Verifikationsbedingungen

- 1.1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 1.2 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$
- 1.3 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n$
- 2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$
- 2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$
- 3.1 $(\exists l_0. (\exists R_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq l_0 < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[l_0] \wedge r = l_0) \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 3.2 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 3.3 $(\exists l_0. (\exists R_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq l_0 < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[l_0] \wedge r = l_0) \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$
- 3.4 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$
- 3.5 $(\exists l_0. (\exists R_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq l_0 < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[l_0] \wedge r = l_0) \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$
- 3.6 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

- 1.1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 1.2 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$
- 1.3 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n$
- 2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$
- 2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$
- 3.1 $(\exists l_0. (\exists R_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[R_0])) \wedge 0 \leq l_0 < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[l_0] \wedge r = l_0) \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 3.2 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 3.3 $(\exists l_0. (\exists R_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[R_0])) \wedge 0 \leq l_0 < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[l_0] \wedge r = l_0) \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$
- 3.4 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$
- 3.5 $(\exists l_0. (\exists R_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[R_0])) \wedge 0 \leq l_0 < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[l_0] \wedge r = l_0) \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$
- 3.6 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

- 1.1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 1.2 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$
- 1.3 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n$
- 2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$
- 2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$
- 3.1 $(\exists R_0. ((\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 3.2 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 3.3 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq i \leq n$
- 3.4 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$
- 3.5 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq r < n$
- 3.6 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

- 1.1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \checkmark$
- 1.2 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n \checkmark$
- 1.3 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n \checkmark$
- 2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$
- 2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$
- 3.1 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1$
 $\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 3.2 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1)$
 $\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 3.3 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1$
 $\longrightarrow 0 \leq i \leq n$
- 3.4 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1)$
 $\longrightarrow 0 \leq i \leq n$
- 3.5 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1$
 $\longrightarrow 0 \leq r < n$
- 3.6 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1)$
 $\longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

$$1.1 \quad 0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \quad \checkmark$$

$$1.2 \quad 0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n \quad \checkmark$$

$$1.3 \quad 0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n \quad \checkmark$$

$$2.1 \quad (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r]) \quad \checkmark$$

$$2.2 \quad (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n \quad \checkmark$$

$$3.1 \quad (\exists R_0. ((\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$$

$$3.2 \quad (\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$$

$$3.3 \quad (\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq i \leq n$$

$$3.4 \quad (\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$$

$$3.5 \quad (\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq r < n$$

$$3.6 \quad (\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$$

Vereinfachte Verifikationsbedingungen

- 1.1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 1.2 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$ ✓
- 1.3 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n$ ✓
- 2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$ ✓
- 2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$ ✓
- 3.1 $(\exists R_0. ((\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.2 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
- 3.3 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq i \leq n$
- 3.4 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$
- 3.5 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq r < n$
- 3.6 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

- 1.1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 1.2 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$ ✓
- 1.3 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n$ ✓
- 2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$ ✓
- 2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$ ✓
- 3.1 $(\exists R_0. ((\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.2 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.3 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq i \leq n$
- 3.4 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$
- 3.5 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq r < n$
- 3.6 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

- 1.1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 1.2 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$ ✓
- 1.3 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n$ ✓
- 2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$ ✓
- 2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$ ✓
- 3.1 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.2 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.3 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq i \leq n$ ✓
- 3.4 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$
- 3.5 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq r < n$
- 3.6 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

- 1.1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 1.2 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$ ✓
- 1.3 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n$ ✓
- 2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$ ✓
- 2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$ ✓
- 3.1 $(\exists R_0. ((\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.2 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.3 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq i \leq n$ ✓
- 3.4 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$ ✓
- 3.5 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq r < n$
- 3.6 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

- 1.1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 1.2 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$ ✓
- 1.3 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n$ ✓
- 2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$ ✓
- 2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$ ✓
- 3.1 $(\exists R_0. ((\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.2 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.3 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq i \leq n$ ✓
- 3.4 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$ ✓
- 3.5 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq r < n$ ✓
- 3.6 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

- 1.1 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 1.2 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$ ✓
- 1.3 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq i \leq n$ ✓
- 2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$ ✓
- 2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq i \leq n \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$ ✓
- 3.1 $(\exists R_0. ((\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.2 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
- 3.3 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq i \leq n$ ✓
- 3.4 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq i \leq n$ ✓
- 3.5 $(\exists R_0. (\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq r < n \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1 \longrightarrow 0 \leq r < n$ ✓
- 3.6 $(\exists l_0. (\forall j. 0 \leq j < l_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq l_0 < n \wedge 0 \leq r < n \wedge a[l_0] \leq a[r] \wedge i = l_0 + 1) \longrightarrow 0 \leq r < n$ ✓

Zusammenfassung

- ▶ Die Regeln des Floyd-Hoare-Kalküls sind **symmetrisch**: die Zuweisungsregel gibt es “rückwärts” und “vorwärts”.
- ▶ Dual zu Beweis und Verifikationsbedingung rückwärts gibt es Regel und Verifikationsbedingungen vorwärts.
- ▶ Bis auf die Invarianten an Schleifen können wir Korrektheit automatisch prüfen.
- ▶ Kern der Vorwärtsberechnung ist die Zuweisungsregel nach Floyd.
- ▶ Vorwärtsberechnung erzeugt kleinere Terme, ist aber umständlicher zu handhaben.
- ▶ Rückwärtsberechnung ist einfacher zu handhaben, erzeugt aber (tendenziell sehr) große Terme.