Korrekte Software: Grundlagen und Methoden

Vorlesung 10 vom 15.06.21

Vorwärts mit Floyd und Hoare

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2021

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
- Modellierung
- Spezifikation von Funktionen
- 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, offen 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:

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

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

$$\frac{}{\vdash \{A\}\,\{\,\}\,\{A\}} \qquad \frac{\vdash \{A \land b\}\, c_0\, \{B\} \qquad \vdash \{A \land \neg b\}\, c_1\, \{B\}}{\vdash \{A\}\ \mathbf{if}\ (b)\ c_0\ \mathbf{else}\ c_1\, \{B\}}$$

$$\frac{\vdash \{A\}\, c_1\, \{B\} \qquad \vdash \{B\}\, c_2\, \{C\}}{\vdash \{A\}\, c_1; c_2\, \{C\}} \qquad \frac{\vdash \{A \land b\}\, c\, \{A\}}{\vdash \{A\}\ \mathbf{while}\ (b)\ c\, \{A \land \neg b\}}$$

$$\frac{A' \Longrightarrow A \qquad \vdash \{A\}\, c\, \{B\} \qquad B \Longrightarrow B'}{\vdash \{A'\}\, c\, \{B'\}}$$

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** 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 $\exists V.P$?
  - 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)

$$\exists x.\, x < 7 \qquad \exists x.\, x < 3 \wedge x > 7 \qquad \exists x.\, x < 7 \vee x < 3$$

$$\exists y \exists x.\, x + 3 = y \qquad \forall x \exists y.\, x \cdot y = 3 \qquad \exists x \forall y.\, x \cdot y = y$$

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;
// {∃V1. 0 ≤ V1 ∧ x = 2 · y}
x= x+1;
// {∃V2. (∃V1. 0 ≤ V1 ∧ x = 2 · y)[V2/x] ∧ x = (x + 1)[V2/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

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

```
// {0 ≤ x}
x= 2*y;
// {∃V1. 0 ≤ V1 ∧ x = 2 · y}
x= x+1;
// {∃V2. (∃V1. 0 ≤ V1 ∧ x = 2 · y)[V2/x] ∧ x = (x + 1)[V2/x]}
```

- **Vereinfachung** der letzten Nachbedingung:

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

Vorwärtsverkettung

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

```
// {0 ≤ x}
x= 2*y;
// {∃V1. 0 ≤ V1 ∧ x = 2 · y}
x= x+1;
// {∃V2. (∃V1. 0 ≤ V1 ∧ x = 2 · y)[V2/x] ∧ x = (x + 1)[V2/x]}
```

- **Vereinfachung** der letzten Nachbedingung:

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

Vorwärtsverkettung

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

```
// {0 ≤ x}
x= 2*y;
// {∃V1. 0 ≤ V1 ∧ x = 2 · y}
x= x+1;
// {∃V2. (∃V1. 0 ≤ V1 ∧ x = 2 · y)[V2/x] ∧ x = (x + 1)[V2/x]}
```

- **Vereinfachung** der letzten Nachbedingung:

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

Und jetzt...?

Regeln der Vorwärtsverkettung

Eigenschaften des Existenzquantors:

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

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

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

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

Regeln der Vorwärtsverkettung

Eigenschaften des Existenzquantors:

$$P(V) \land V = t \Longrightarrow P[t/V] \land V = t \qquad V \notin FV(t) \tag{1}$$

$$\exists V.\, P(V) \land V = t \Longrightarrow P[t/V] \qquad V \notin FV(t) \tag{2}$$

$$(\exists V.\, P) \land Q \Longleftrightarrow \exists V.\, P \land Q \qquad V \notin FV(Q) \tag{3}$$

$$\exists V.\, P \Longrightarrow P \qquad V \notin FV(P) \tag{4}$$

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 ≤ x}
x= 2*y;
// {∃V1. 0 ≤ V1 ∧ x = 2 · y}
x= x+1;
// {∃V2. (∃V1. 0 ≤ V1 ∧ x = 2 · y)[V2/x] ∧ x = (x + 1)[V2/x]}
```

- **Vereinfachung** der letzten Nachbedingung:

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

Vorwärtsverkettung

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

```
// {0 ≤ x}
x= 2*y;
// {∃V1. 0 ≤ V1 ∧ x = 2 · y}
x= x+1;
// {∃V2. (∃V1. 0 ≤ V1 ∧ x = 2 · y)[V2/x] ∧ x = (x + 1)[V2/x]}
```

- **Vereinfachung** der letzten Nachbedingung:

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

Vorwärtsverkettung bei der Arbeit

Vereinfachung erst am Ende:

$// \; \{i \cdot i \leq a \wedge t = 2 \cdot i + 1 \wedge s = i \cdot i + t \wedge s \leq a\}$

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 · 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}
```

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]}
```

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;
// {∃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}
```

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

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}
```

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}
```

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}
```

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]}
```

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}
```

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}
```

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;
// {∃I.(i·i ≤ a ∧ (i + 1)·(i + 1) ≤ a ∧ t = 2·i + 3 ∧ s = (i + 1)·(i + 1) + t)[I/i] ∧ i = (i + 1)[I/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;
// {∃I.(i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t)[I/i] ∧ i = (i + 1)[I/i]}
// {∃I.I · I ≤ a ∧ (I + 1) · (I + 1) ≤ a ∧ t = 2 · I + 3 ∧ s = (I + 1) · (I + 1) + t ∧ i = 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;
// {∃I.(i·i ≤ a ∧ (i+1)·(i+1) ≤ a ∧ t = 2·i+3 ∧ s = (i+1)·(i+1)+t)[I/i] ∧ i = (i+1)[I/i]}
// {∃I.I·I ≤ a ∧ (I+1)·(I+1) ≤ a ∧ t = 2·I+3 ∧ s = (I+1)·(I+1)+t ∧ i = I+1}
// {∃I.I·I ≤ a ∧ (I+1)·(I+1) ≤ a ∧ t = 2·I+3 ∧ s = (I+1)·(I+1)+t ∧ I = 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;
// {∃I.(i · i ≤ a ∧ (i + 1) · (i + 1) ≤ a ∧ t = 2 · i + 3 ∧ s = (i + 1) · (i + 1) + t)[I/i] ∧ i = (i + 1)[I/i]}
// {∃I.I · I ≤ a ∧ (I + 1) · (I + 1) ≤ a ∧ t = 2 · I + 3 ∧ s = (I + 1) · (I + 1) + t ∧ i = I + 1}
// {∃I.I · I ≤ a ∧ (I + 1) · (I + 1) ≤ a ∧ t = 2 · I + 3 ∧ s = (I + 1) · (I + 1) + t ∧ I = 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;
// {∃I.(i·i ≤ a ∧ (i+1)·(i+1) ≤ a ∧ t = 2·i+3 ∧ s = (i+1)·(i+1)+t)[I/i] ∧ i = (i+1)[I/i]}
// {∃I.I·I ≤ a ∧ (I+1)·(I+1) ≤ a ∧ t = 2·I+3 ∧ s = (I+1)·(I+1)+t ∧ i = I+1}
// {∃I.I·I ≤ a ∧ (I+1)·(I+1) ≤ a ∧ t = 2·I+3 ∧ s = (I+1)·(I+1)+t ∧ I = 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:

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

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**:

$$\sigma \models^I B[e/x] \Longleftrightarrow \sigma[x \mapsto [\![e]\!]_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

$$\begin{aligned} & \models \{P\}\, x = e\, \{\exists V.\, P[V/x] \wedge x = (e[V/x])\} \\ \Longleftrightarrow\ & \forall I.\, \forall \sigma.\, \sigma \models^I P \wedge \exists \sigma'.\, (\sigma, \sigma') \in [\![x = e]\!]_{\mathcal{C}} \Longrightarrow \sigma' \models^I \exists V.\, P[V/x] \wedge x = (e[V/x]) \end{aligned}$$

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \Longleftrightarrow \sigma[x \mapsto [\![e]\!]_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

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

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \Longleftrightarrow \sigma[x \mapsto [\![e]\!]_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

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

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \Longleftrightarrow \sigma[x \mapsto [\![e]\!]_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

$$\begin{aligned}
& \models \{P\}\, x = e\, \{\exists V.\, P[V/x] \wedge x = (e[V/x])\} \\
\Longleftrightarrow\; & \forall I.\, \forall \sigma.\, \sigma \models^I P \wedge \exists \sigma'.\, (\sigma, \sigma') \in [\![x = e]\!]_{\mathcal{C}} \Longrightarrow \sigma' \models^I \exists V.\, P[V/x] \wedge x = (e[V/x]) \\
\Longleftrightarrow\; & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma[x \mapsto [\![e]\!]_{\mathcal{A}}] \models^I \exists V.\, P[V/x] \wedge x = (e[V/x]) \\
\Longleftrightarrow\; & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma \models^I (\exists V.\, P[V/x] \wedge x = (e[V/x]))[e/x] \\
\Longleftrightarrow\; & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma \models^I (\exists V.\, P[V/x] \wedge e = (e[V/x]))
\end{aligned}$$

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \Longleftrightarrow \sigma[x \mapsto [\![e]\!]_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

$$\begin{aligned}
& \models \{P\}\, x = e\, \{\exists V.\, P[V/x] \land x = (e[V/x])\} \\
\Longleftrightarrow\; & \forall I.\, \forall \sigma.\, \sigma \models^I P \land \exists \sigma'.\, (\sigma, \sigma') \in [\![x = e]\!]_{\mathcal{C}} \Longrightarrow \sigma' \models^I \exists V.\, P[V/x] \land x = (e[V/x]) \\
\Longleftrightarrow\; & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma[x \mapsto [\![e]\!]_{\mathcal{A}}] \models^I \exists V.\, P[V/x] \land x = (e[V/x]) \\
\Longleftrightarrow\; & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma \models^I (\exists V.\, P[V/x] \land x = (e[V/x]))[e/x] \\
\Longleftrightarrow\; & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma \models^I (\exists V.\, P[V/x] \land e = (e[V/x])) \\
\Longleftrightarrow\; & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma \models^I (P[x/x] \land e = (e[x/x]))
\end{aligned}$$

Beweis der Zuweisungsregel Vorwärts

Erinnert Euch an das **Substitutionslemma**:

$$\sigma \models^I B[e/x] \Longleftrightarrow \sigma[x \mapsto [\![e]\!]_{\mathcal{A}}(\sigma)] \models^I B$$

Zu zeigen:

$$
\begin{aligned}
& \models \{P\}\, x = e\, \{\exists V.\, P[V/x] \land x = (e[V/x])\} \\
\Longleftrightarrow\ & \forall I.\, \forall \sigma.\, \sigma \models^I P \land \exists \sigma'.\, (\sigma, \sigma') \in [\![x = e]\!]_{\mathcal{C}} \Longrightarrow \sigma' \models^I \exists V.\, P[V/x] \land x = (e[V/x]) \\
\Longleftrightarrow\ & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma[x \mapsto [\![e]\!]_{\mathcal{A}}] \models^I \exists V.\, P[V/x] \land x = (e[V/x]) \\
\Longleftrightarrow\ & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma \models^I (\exists V.\, P[V/x] \land x = (e[V/x]))[e/x] \\
\Longleftrightarrow\ & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma \models^I (\exists V.\, P[V/x] \land e = (e[V/x])) \\
\Longleftrightarrow\ & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma \models^I (P[x/x] \land e = (e[x/x])) \\
\Longleftrightarrow\ & \forall I.\, \forall \sigma.\, \sigma \models^I P \Longrightarrow \sigma \models^I P \quad \square
\end{aligned}
$$

Vorwärtsverkettung

- Vorwärtsaxiom äquivalent zum Rückwärtsaxiom.
- Vorteil: Vorbedingung bleibt kleiner
- Nachteil: in der Anwendung **umständlicher**
- Die entstehende Nachbedingung beschreibt die **symbolische Auswärtung**
- 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
  - $\mathrm{sp}(P, c)$ 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

$$\models \{P\}\, c\, \{Q\} \Longleftrightarrow \mathrm{sp}(P, c) \Longrightarrow Q$$

- Wie können wir $\mathrm{sp}(P, c)$ 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

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

$$
\begin{aligned}
\mathrm{asp}(P, \{\,\}) &\stackrel{def}{=} P \\
\mathrm{asp}(P, x = e) &\stackrel{def}{=} \exists V.\, P[V/x] \land x = (e[V/x]) \\
\mathrm{asp}(P, c_1; c_2) &\stackrel{def}{=} \mathrm{asp}(\mathrm{asp}(P, c_1), c_2) \\
\mathrm{asp}(P, \textbf{if } (b)\ c_0 \textbf{ else } c_1) &\stackrel{def}{=} \mathrm{asp}(b \land P, c_0) \lor \mathrm{asp}(\neg b \land P, c_1) \\
\mathrm{asp}(P, /{*}{*}\ \{q\}\ {*}/) &\stackrel{def}{=} q \\
\mathrm{asp}(P, \textbf{while } (b)\ /{*}{*}\ \textbf{inv } i\ {*}/\ c) &\stackrel{def}{=} i \land \neg b
\end{aligned}
$$

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

$$
\begin{aligned}
\mathrm{asp}(P, \{\,\}) &\stackrel{def}{=} P \\
\mathrm{asp}(P, x = e) &\stackrel{def}{=} \exists V.\, P[V/x] \land x = (e[V/x]) \\
\mathrm{asp}(P, c_1; c_2) &\stackrel{def}{=} \mathrm{asp}(\mathrm{asp}(P, c_1), c_2) \\
\mathrm{asp}(P, \textbf{if } (b)\ c_0\ \textbf{ else }\ c_1) &\stackrel{def}{=} \mathrm{asp}(b \land P, c_0) \lor \mathrm{asp}(\neg b \land P, c_1) \\
\mathrm{asp}(P, /{*}{*}\ \{q\}\ {*}/) &\stackrel{def}{=} q \\
\mathrm{asp}(P, \textbf{while } (b)\ /{*}{*}\ \textbf{inv}\ \ i\ {*}/\ c) &\stackrel{def}{=} i \land \neg b
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{svc}(P, \{\,\}) &\stackrel{def}{=} \emptyset \\
\mathrm{svc}(P, x = e) &\stackrel{def}{=} \emptyset
\end{aligned}
$$

Überblick: Approximative stärkste Nachbedingung

$$
\begin{aligned}
\mathrm{asp}(P, \{\,\}) &\stackrel{def}{=} P \\
\mathrm{asp}(P, x = e) &\stackrel{def}{=} \exists V.\, P[V/x] \land x = (e[V/x]) \\
\mathrm{asp}(P, c_1; c_2) &\stackrel{def}{=} \mathrm{asp}(\mathrm{asp}(P, c_1), c_2) \\
\mathrm{asp}(P, \mathbf{if}\ (b)\ c_0\ \mathbf{else}\ c_1) &\stackrel{def}{=} \mathrm{asp}(b \land P, c_0) \lor \mathrm{asp}(\neg b \land P, c_1) \\
\mathrm{asp}(P, /{*}{*}\ \{q\}\ {*}/) &\stackrel{def}{=} q \\
\mathrm{asp}(P, \mathbf{while}\ (b)\ /{*}{*}\ \mathbf{inv}\ i\ {*}/\ c) &\stackrel{def}{=} i \land \neg b
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{svc}(P, \{\,\}) &\stackrel{def}{=} \emptyset \\
\mathrm{svc}(P, x = e) &\stackrel{def}{=} \emptyset \\
\mathrm{svc}(P, c_1; c_2) &\stackrel{def}{=} \mathrm{svc}(P, c_1) \cup \mathrm{svc}(\mathrm{asp}(P, c_1), c_2)
\end{aligned}
$$

Überblick: Approximative stärkste Nachbedingung

$$
\begin{aligned}
\mathrm{asp}(P, \{\,\}) &\stackrel{def}{=} P \\
\mathrm{asp}(P, x = e) &\stackrel{def}{=} \exists V.\, P[V/x] \land x = (e[V/x]) \\
\mathrm{asp}(P, c_1; c_2) &\stackrel{def}{=} \mathrm{asp}(\mathrm{asp}(P, c_1), c_2) \\
\mathrm{asp}(P, \textbf{if } (b)\ c_0\ \textbf{ else }\ c_1) &\stackrel{def}{=} \mathrm{asp}(b \land P, c_0) \lor \mathrm{asp}(\neg b \land P, c_1) \\
\mathrm{asp}(P, /{**}\ \{q\}\ {*}/) &\stackrel{def}{=} q \\
\mathrm{asp}(P, \textbf{while } (b)\ /{**}\ \textbf{inv }\ i\ {*}/\ c) &\stackrel{def}{=} i \land \neg b
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{svc}(P, \{\,\}) &\stackrel{def}{=} \emptyset \\
\mathrm{svc}(P, x = e) &\stackrel{def}{=} \emptyset \\
\mathrm{svc}(P, c_1; c_2) &\stackrel{def}{=} \mathrm{svc}(P, c_1) \cup \mathrm{svc}(\mathrm{asp}(P, c_1), c_2) \\
\mathrm{svc}(P, \textbf{if } (b)\ c_0\ \textbf{ else }\ c_1) &\stackrel{def}{=} \mathrm{svc}(P \land b, c_0) \cup \mathrm{svc}(P \land \neg b, c_1)
\end{aligned}
$$

Überblick: Approximative stärkste Nachbedingung

$$
\begin{aligned}
\mathrm{asp}(P, \{\,\}) &\stackrel{def}{=} P \\
\mathrm{asp}(P, x = e) &\stackrel{def}{=} \exists V.\, P[V/x] \land x = (e[V/x]) \\
\mathrm{asp}(P, c_1; c_2) &\stackrel{def}{=} \mathrm{asp}(\mathrm{asp}(P, c_1), c_2) \\
\mathrm{asp}(P, \mathbf{if}\ (b)\ c_0\ \mathbf{else}\ c_1) &\stackrel{def}{=} \mathrm{asp}(b \land P, c_0) \lor \mathrm{asp}(\neg b \land P, c_1) \\
\mathrm{asp}(P, /{*}{*}\ \{q\}\ {*}/) &\stackrel{def}{=} q \\
\mathrm{asp}(P, \mathbf{while}\ (b)\ /{*}{*}\ \mathbf{inv}\ i\ {*}/\ c) &\stackrel{def}{=} i \land \neg b
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{svc}(P, \{\,\}) &\stackrel{def}{=} \emptyset \\
\mathrm{svc}(P, x = e) &\stackrel{def}{=} \emptyset \\
\mathrm{svc}(P, c_1; c_2) &\stackrel{def}{=} \mathrm{svc}(P, c_1) \cup \mathrm{svc}(\mathrm{asp}(P, c_1), c_2) \\
\mathrm{svc}(P, \mathbf{if}\ (b)\ c_0\ \mathbf{else}\ c_1) &\stackrel{def}{=} \mathrm{svc}(P \land b, c_0) \cup \mathrm{svc}(P \land \neg b, c_1) \\
\mathrm{svc}(P, /{*}{*}\ \{q\}\ {*}/) &\stackrel{def}{=} \{P \longrightarrow q\}
\end{aligned}
$$

Überblick: Approximative stärkste Nachbedingung

$$
\begin{aligned}
\mathrm{asp}(P, \{\,\}) &\stackrel{\mathit{def}}{=} P \\
\mathrm{asp}(P, x = e) &\stackrel{\mathit{def}}{=} \exists V.\, P[V/x] \land x = (e[V/x]) \\
\mathrm{asp}(P, c_1; c_2) &\stackrel{\mathit{def}}{=} \mathrm{asp}(\mathrm{asp}(P, c_1), c_2) \\
\mathrm{asp}(P, \textbf{if } (b)\ c_0\ \textbf{ else }\ c_1) &\stackrel{\mathit{def}}{=} \mathrm{asp}(b \land P, c_0) \lor \mathrm{asp}(\neg b \land P, c_1) \\
\mathrm{asp}(P, /{*}{*}\ \{q\}\ {*}/) &\stackrel{\mathit{def}}{=} q \\
\mathrm{asp}(P, \textbf{while } (b)\ /{*}{*}\ \textbf{inv}\ \ i\ {*}/\ c) &\stackrel{\mathit{def}}{=} i \land \neg b
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{svc}(P, \{\,\}) &\stackrel{\mathit{def}}{=} \emptyset \\
\mathrm{svc}(P, x = e) &\stackrel{\mathit{def}}{=} \emptyset \\
\mathrm{svc}(P, c_1; c_2) &\stackrel{\mathit{def}}{=} \mathrm{svc}(P, c_1) \cup \mathrm{svc}(\mathrm{asp}(P, c_1), c_2) \\
\mathrm{svc}(P, \textbf{if } (b)\ c_0\ \textbf{ else }\ c_1) &\stackrel{\mathit{def}}{=} \mathrm{svc}(P \land b, c_0) \cup \mathrm{svc}(P \land \neg b, c_1) \\
\mathrm{svc}(P, /{*}{*}\ \{q\}\ {*}/) &\stackrel{\mathit{def}}{=} \{P \longrightarrow q\} \\
\mathrm{svc}(P, \textbf{while } (b)\ /{*}{*}\ \textbf{inv}\ \ i\ {*}/\ c) &\stackrel{\mathit{def}}{=} \mathrm{svc}(i \land b, c) \cup \{P \longrightarrow i\} \cup \{\mathrm{asp}(i \land b, c) \longrightarrow i\}
\end{aligned}
$$

Überblick: Approximative stärkste Nachbedingung

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

$$
\begin{aligned}
\mathrm{svc}(P, \{\,\}) &\stackrel{def}{=} \emptyset \\
\mathrm{svc}(P, x = e) &\stackrel{def}{=} \emptyset \\
\mathrm{svc}(P, c_1; c_2) &\stackrel{def}{=} \mathrm{svc}(P, c_1) \cup \mathrm{svc}(\mathrm{asp}(P, c_1), c_2) \\
\mathrm{svc}(P, \textbf{if } (b)\ c_0\ \textbf{ else }\ c_1) &\stackrel{def}{=} \mathrm{svc}(P \wedge b, c_0) \cup \mathrm{svc}(P \wedge \neg b, c_1) \\
\mathrm{svc}(P, /{*}{*}\ \{q\}\ {*}/) &\stackrel{def}{=} \{P \longrightarrow q\} \\
\mathrm{svc}(P, \textbf{while } (b)\ /{*}{*}\ \textbf{inv}\ i\ {*}/\ c) &\stackrel{def}{=} \mathrm{svc}(i \wedge b, c) \cup \{P \longrightarrow i\} \cup \{\mathrm{asp}(i \wedge b, c) \longrightarrow i\}
\end{aligned}
$$

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

Beispiel: Fakultät

```
// {0 ≤ n}
p= 1;
c= 1;
while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */
  p = p * c;
  c = c + 1;
  }
// {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!}
```

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

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf

```
// {0 ≤ n}
p= 1;
// {0 ≤ n ∧ p = 1}
c= 1;
// {0 ≤ n ∧ p = 1 ∧ c = 1}
while (c <= n) /** inv p = (c−1)! ∧ c−1 ≤ n; */ \{
 p = p * c;
 //
 //
 //
 c = c + 1;
 //
 //
 //
 }
// {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ n}
// {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf

```
// {0 ≤ n}
p= 1;
// {0 ≤ n ∧ p = 1}
c= 1;
// {0 ≤ n ∧ p = 1 ∧ c = 1}
while (c <= n) /** inv p = (c−1)! ∧ c−1 ≤ n; */ \{
 p = p * c;
 // {∃V_1.(p = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n)[V_1/p] ∧ p = (p · c)[V_1/p]}
 //
 //
 c = c + 1;
 //
 //
 //
 }
// {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ n}
// {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf

```
// {0 ≤ n}
p= 1;
// {0 ≤ n ∧ p = 1}
c= 1;
// {0 ≤ n ∧ p = 1 ∧ c = 1}
while (c <= n) /** inv p = (c−1)! ∧ c−1 ≤ n; */ \{
 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}
// {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf

```
// {0 ≤ n}
p= 1;
// {0 ≤ n ∧ p = 1}
c= 1;
// {0 ≤ n ∧ p = 1 ∧ c = 1}
while (c <= n) /** inv p = (c−1)! ∧ c−1 ≤ n; */ \{
  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}
  c = c + 1;
  //
  //
  //
  }
// {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ n}
// {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf

// $\{0 \leq n\}$
`p= 1;`
// $\{0 \leq n \land p = 1\}$
`c= 1;`
// $\{0 \leq n \land p = 1 \land c = 1\}$
`while` `(c <= n) /**` **`inv`** `p = (c−1)! ∧ c−1 ≤ n; */ \{`
  `p = p * c;`
  // $\{\exists V_1. (p = (c-1)! \land (c-1) \leq n \land c \leq n)[V_1/p] \land p = (p \cdot c)[V_1/p]\}$
  // $\{\exists V_1. (V_1 = (c-1)! \land (c-1) \leq n \land c \leq n) \land p = (V_1 \cdot c)\}$
  // $\{c - 1 \leq n \land c \leq n \land p = (c-1)! \cdot c\}$
  `c = c + 1;`
  // $\{\exists V_2. (c - 1 \leq n \land c \leq n \land p = (c-1)! \cdot c)[V_2/c] \land c = (c+1)[V_2/c]\}$
  //
  //
  }
// $\{\neg(c \leq n) \land p = (c-1)! \land c - 1 \leq n\}$
// $\{p = n!\}$

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf

```
// {0 ≤ n}
p= 1;
// {0 ≤ n ∧ p = 1}
c= 1;
// {0 ≤ n ∧ p = 1 ∧ c = 1}
while (c <= n) /** inv p = (c-1)! ∧ c-1 ≤ n; */ \{
  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}
  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}
// {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung (Schleifenrumpf

```
// {0 ≤ n}
p= 1;
// {0 ≤ n ∧ p = 1}
c= 1;
// {0 ≤ n ∧ p = 1 ∧ c = 1}
while (c <= n) /** inv p = (c−1)! ∧ c−1 ≤ n; */ \{
  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}
  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)}
  }
// {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ n}
// {p = n!}
```

Beispiel: Fakultät, Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  // {0 ≤ n}
2  p= 1;
   // svc₂ = ∅
   c= 1;
   // svc₃ = ∅
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5    p = p * c;
     // svc₅ = ∅
6    c = c + 1;
     // svc₆ = ∅
7    }
   // svc₄ = {asp₃ ⟹ (p = (c − 1)! ∧ c − 1 ≤ n),
   //               asp₆ ⟹ (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;
   // svc₂ = ∅
   c= 1;
   // svc₃ = ∅
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5    p = p * c;
     // svc₅ = ∅
6    c = c + 1;
     // svc₆ = ∅
7    }
   // svc₄ = {asp₃ ⟹ (p = (c − 1)! ∧ c − 1 ≤ n),
   //              asp₆ ⟹ (p = (c − 1)! ∧ c − 1 ≤ n)}
   // svc₄ = {(0 ≤ n ∧ p = 1 ∧ c = 1) ⟹ (p = (c − 1)! ∧ c − 1 ≤ n),
   //              (c − 2 ≤ n ∧ c − 1 ≤ n ∧ p = (c − 2)! · (c − 1))
   //                   ⟹ (p = (c − 1)! ∧ c − 1 ≤ n)}
8  // {p = n!}
```

Schließlich zu zeigen

$$
\begin{aligned}
svc_8 &= \{\{asp_8 \Longrightarrow p = n!\} \cup svc_4 \\
&= \{(p = (c-1)! \wedge c - 1 \leq n \,\&\&\, \neg(c <= n)) \Longrightarrow p = n!\}, \\
&\qquad (0 \leq n \wedge p = 1 \wedge c = 1) \Longrightarrow (p = (c-1)! \wedge c - 1 \leq n), \\
&\qquad (c - 2 \leq n \wedge c - 1 \leq n \wedge p = (c-2)! \cdot (c-1)) \\
&\qquad\qquad\qquad \Longrightarrow (p = (c-1)! \wedge c - 1 \leq n)\} \\
&\rightsquigarrow \{true\}
\end{aligned}
$$

Arbeitsblatt 10.4: Jetzt seid ihr dran!

Berechnet die stärkste Nachbedingung und Verifikationsbedingungen für die ganzzahlige Division:

```
/** {0 ≤ a} */
r= a;
q= 0;
while (b <= r) /** inv { a == b*q+r ∧ 0 <= r } */ {
  r= r-b;
  q= q+1;
  }
/** { a == b*q+ r ∧ 0 ≤ r ∧ r < b } */
```

Beispiel: Suche nach dem Maximalen Element

```
// {0 < n}
i= 0;
//
//
r= 0;
//
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
     r= i;
  }
  else {
     }
  i= i+1;
  }
//
// {(∀j. 0 ≤ j < n ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n}
```

Beispiel: Suche nach dem Maximalen Element

```
// {0 < n}
i= 0;
// {∃I₀.(0 < n)[I₀/i] ∧ i = 0[I₀/i]}
//
r= 0;
//
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
     r= i;
  }
  else {
     }
  i= i+1;
  }
//
// {(∀j. 0 ≤ j < n ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n}
```

Beispiel: Suche nach dem Maximalen Element

```
// {0 < n}
i= 0;
// {∃I0.(0 < n)[I0/i] ∧ i = 0[I0/i]}
// {0 < n ∧ i = 0}
r= 0;
//
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    r= i;
  }
  else {
    }
  i= i+1;
  }
//
// {(∀j. 0 ≤ j < n ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n}
```

Beispiel: Suche nach dem Maximalen Element

```
// {0 < n}
i= 0;
// {∃I₀.(0 < n)[I₀/i] ∧ i = 0[I₀/i]}
// {0 < n ∧ i = 0}
r= 0;
// {0 < n ∧ i = 0 ∧ r = 0}
while (i != n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    r= i;
  }
  else {
    }
  i= i+1;
  }
//
// {(∀j. 0 ≤ j < n ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n}
```

Beispiel: Suche nach dem Maximalen Element

```
// {0 < n}
i= 0;
// {∃l0.(0 < n)[l0/i] ∧ i = 0[l0/i]}
// {0 < n ∧ i = 0}
r= 0;
// {0 < n ∧ i = 0 ∧ r = 0}
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
     r= i;
  }
  else {
     }
  i= i+1;
  }
// {∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ ¬(i ≠ n)}
// {(∀j. 0 ≤ j < n ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n}
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    //
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    //
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    //
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    //
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    //
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    // {∃R0.((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i])[R0/r] ∧ r = i[R0/r]}
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    // {∃R0.((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i])[R0/r] ∧ r = i[R0/r]}
    // {∃R0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R0]) ∧ 0 ≤ R0 < n ∧ a[R0] < a[i] ∧ r = i}
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // $\{(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \land 0 \leq r < n \land a[r] < a[i]\}$
    r= i;
    // $\{\exists R_0.((\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \land 0 \leq r < n \land a[r] < a[i])[R_0/r] \land r = i[R_0/r]\}$
    // $\{\exists R_0.(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[R_0]) \land 0 \leq R_0 < n \land a[R_0] < a[i] \land r = i\}$
  }
  else {
    // $\{(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \land 0 \leq r < n \land \neg(a[r] < a[i])\}$
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    // {∃R0.((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i])[R0/r] ∧ r = i[R0/r]}
    // {∃R0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R0]) ∧ 0 ≤ R0 < n ∧ a[R0] < a[i] ∧ r = i}
  }
  else {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ ¬(a[r] < a[i])}
    }
  // {(∃R0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R0]) ∧ 0 ≤ R0 < n ∧ a[R0] < a[i] ∧ r = i) }
         ∨((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[i] ≤ a[r])
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    // {∃R_0.((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[r] < a[i])[R_0/r] ∧ r = i[R_0/r]}
    // {∃R_0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R_0]) ∧ 0 ≤ R_0 < n ∧ a[R_0] < a[i] ∧ r = i}
  }
  else {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ ¬(a[r] < a[i])}
    }
  // {(∃R_0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R_0]) ∧ 0 ≤ R_0 < n ∧ a[R_0] < a[i] ∧ r = i) }
        ∨((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[i] ≤ a[r])
  i= i+1;
  // {∃I_0.((∃R_0.(∀j.0 ≤ j < I_0 ⟶ a[j] ≤ a[R_0]) ∧ 0 ≤ R_0 < n ∧ a[R_0] < a[I_0] ∧ r = I_0) }
            ∨((∀j.0 ≤ j < I_0 ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n ∧ a[I_0] ≤ a[r])) ∧ i = I_0 + 1
  }
```

Verifikationsbedingungen

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

2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j.\, 0 \leq j < n$
$\longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n$

3 $(\exists I_0.((\exists R_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq R_0 < n \wedge a[R_0] < a[I_0] \wedge r = I_0)$
$\vee ((\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r])) \wedge i = I_0 + 1)$
$\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n$

Weitere Vereinfachungsregeln

Existenzquantoren und Disjunktionen können mit folgenden Regeln vereinfacht werden:

6. Der Gültigkeitsbereich des Existenzquantors kann verkleinert werden:
   - $(\exists x. P \vee Q) \rightsquigarrow (\exists x. P) \vee (\exists x. Q)$
7. Disjunktionen in der Prämisse ergeben eine Fallunterscheidung:
   - $A_1 \vee A_2 \longrightarrow B \rightsquigarrow A_1 \longrightarrow B, A_2 \longrightarrow B$
8. Konjunktion distributiert über Disjunktion:
   - $(A_1 \vee A_2) \wedge B \rightsquigarrow (A_1 \wedge B) \vee (A_2 \wedge B)$
9. … und andersherum:
   - $(A_1 \wedge A_2) \vee B \rightsquigarrow (A_1 \vee B) \wedge (A_2 \vee B)$

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 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$

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

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

3.1 $(\exists I_0.(\exists R_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq R_0 < n \wedge a[R_0] < a[I_0] \wedge r = I_0) \wedge i = I_0 +$
$\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$

3.2 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r] \wedge i = I_0 + 1)$
$\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$

3.3 $(\exists I_0.(\exists R_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq R_0 < n \wedge a[R_0] < a[I_0] \wedge r = I_0) \wedge i = I_0 +$
$\longrightarrow 0 \leq r < n$

3.4 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r] \wedge i = I_0 + 1)$
$\longrightarrow 0 \leq r < n$

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	$0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$
2.1	$(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$
2.2	$(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$
3.1	$(\exists R_0.((\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1)$ $\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
3.2	$(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r] \wedge i = I_0 + 1)$ $\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$
3.3	$(\exists R_0.(\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1$ $\longrightarrow 0 \leq r < n$
3.4	$(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r] \wedge i = I_0 + 1)$ $\longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

1.1 $0 < n \land i = 0 \land r = 0 \longrightarrow (\forall j.0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓

1.2 $0 < n \land i = 0 \land r = 0 \longrightarrow 0 \leq r < n$ ✓

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

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

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

3.2 $(\exists I_0.(\forall j.0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \land 0 \leq r < n \land a[I_0] \leq a[r] \land i = I_0 + 1)$
$\longrightarrow (\forall j.0 \leq j < i \longrightarrow a[j] \leq a[r])$

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

3.4 $(\exists I_0.(\forall j.0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \land 0 \leq r < n \land a[I_0] \leq a[r] \land i = I_0 + 1)$
$\longrightarrow 0 \leq r < n$

Vereinfachte Verifikationsbedingungen

1.1 $0 < n \land i = 0 \land r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓

1.2 $0 < n \land i = 0 \land r = 0 \longrightarrow 0 \leq r < n$ ✓

2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \land 0 \leq r < n \land \neg(i \neq n) \longrightarrow (\forall j.\, 0 \leq j < n \longrightarrow a[j] \leq a[r])$ ✓

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

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

3.2 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \land 0 \leq r < n \land a[I_0] \leq a[r] \land i = I_0 + 1)$
$\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$

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

3.4 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \land 0 \leq r < n \land a[I_0] \leq a[r] \land i = I_0 + 1)$
$\longrightarrow 0 \leq r < n$

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 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$ ✓

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

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

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

3.2 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r] \wedge i = I_0 + 1)$
$\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$

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

3.4 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r] \wedge i = I_0 + 1)$
$\longrightarrow 0 \leq r < n$ ✓

Vereinfachte Verifikationsbedingungen

1.1 $0 < n \land i = 0 \land r = 0 \longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓

1.2 $0 < n \land i = 0 \land r = 0 \longrightarrow 0 \leq r < n$ ✓

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

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

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

3.2 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \land 0 \leq r < n \land a[I_0] \leq a[r] \land i = I_0 + 1)$
$\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$

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

3.4 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \land 0 \leq r < n \land a[I_0] \leq a[r] \land i = I_0 + 1)$
$\longrightarrow 0 \leq r < n$ ✓

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 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$ ✓
2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$ ✓
2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$ ✓
3.1 $(\exists R_0.((\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1)$
$\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
3.2 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r] \wedge i = I_0 + 1)$
$\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
3.3 $(\exists R_0.(\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1$
$\longrightarrow 0 \leq r < n$
3.4 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r] \wedge i = I_0 + 1)$
$\longrightarrow 0 \leq r < n$ ✓

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 $0 < n \wedge i = 0 \wedge r = 0 \longrightarrow 0 \leq r < n$ ✓
2.1 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow (\forall j. 0 \leq j < n \longrightarrow a[j] \leq a[r])$ ✓
2.2 $(\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge \neg(i \neq n) \longrightarrow 0 \leq r < n$ ✓
3.1 $(\exists R_0.((\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1)$
$\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
3.2 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r] \wedge i = I_0 + 1)$
$\longrightarrow (\forall j. 0 \leq j < i \longrightarrow a[j] \leq a[r])$ ✓
3.3 $(\exists R_0.(\forall j. 0 \leq j < r \longrightarrow a[j] \leq a[R_0]) \wedge 0 \leq R_0 < n \wedge a[R_0] < a[r]) \wedge i = r + 1$
$\longrightarrow 0 \leq r < n$ ×
3.4 $(\exists I_0.(\forall j. 0 \leq j < I_0 \longrightarrow a[j] \leq a[r]) \wedge 0 \leq r < n \wedge a[I_0] \leq a[r] \wedge i = I_0 + 1)$
$\longrightarrow 0 \leq r < n$ ✓

Invariante muss verstärkt werden: $0 \leq i < n$

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶a[j]≤ a[r])∧ 0≤ i< n ∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    //
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ i< n ∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    //
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ i< n ∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    //
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶a[j]≤ a[r])∧ 0≤ i< n ∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    //
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶a[j]≤ a[r])∧ 0≤ i< n ∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    //
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶a[j]≤ a[r])∧ 0≤ i< n ∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    // {∃R_0.((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i])[R_0/r] ∧ r = i[R_0/r]}
    //
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ i< n ∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    // {∃R0.((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i])[R0/r] ∧ r = i[R0/r]}
    // {∃R0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R0]) ∧ 0 ≤ i < n ∧ 0 ≤ R0 < n ∧ a[R0] < a[i] ∧ r = i}
  }
  else {
    //
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶a[j]≤ a[r])∧ 0≤ i< n ∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    // {∃R0.((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i])[R0/r] ∧ r = i[R0/r]}
    // {∃R0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R0]) ∧ 0 ≤ i < n ∧ 0 ≤ R0 < n ∧ a[R0] < a[i] ∧ r = i}
  }
  else {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ ¬(a[r] < a[i])}
    }
  //
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶a[j]≤ a[r])∧ 0≤ i< n ∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    // {∃R0.((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i])[R0/r] ∧ r = i[R0/r]}
    // {∃R0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R0]) ∧ 0 ≤ i < n ∧ 0 ≤ R0 < n ∧ a[R0] < a[i] ∧ r = i}
  }
  else {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ ¬(a[r] < a[i])}
    }
  // {(∃R0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R0]) ∧ 0 ≤ i < n ∧ 0 ≤ R0 < n ∧ a[R0] < a[i] ∧ r = i) }
        ∨((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[i] ≤ a[r])
  i= i+1;
  //
  }
```

Beispiel: Suche nach dem Maximalen Element (Schleifenrumpf)

```
while (i!= n)
  /** inv (∀j. 0≤ j < i ⟶ a[j]≤ a[r])∧ 0≤ i< n ∧ 0≤ r< n */ {
  if (a[r] < a[i]) {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i]}
    r= i;
    // {∃R0.((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[r] < a[i])[R0/r] ∧ r = i[R0/r]}
    // {∃R0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R0]) ∧ 0 ≤ i < n ∧ 0 ≤ R0 < n ∧ a[R0] < a[i] ∧ r = i}
  }
  else {
    // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ ¬(a[r] < a[i])}
    }
  // {(∃R0.(∀j.0 ≤ j < i ⟶ a[j] ≤ a[R0]) ∧ 0 ≤ i < n ∧ 0 ≤ R0 < n ∧ a[R0] < a[i] ∧ r = i) }
        ∨((∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i < n ∧ 0 ≤ r < n ∧ a[i] ≤ a[r])
  i= i+1;
  // {∃I0.((∃R0.(∀j.0 ≤ j < I0 ⟶ a[j] ≤ a[R0]) ∧ 0 ≤ I0 < n ∧ 0 ≤ R0 < n ∧ a[R0] < a[I0] ∧ r = I0) }
            ∨((∀j.0 ≤ j < I0 ⟶ a[j] ≤ a[r]) ∧ 0 ≤ I0 < n ∧ 0 ≤ r < n ∧ a[I0] ≤ a[r])) ∧ i = I0 + 1
  }
```

Vereinfachte Verifikationsbedingungen

...

$$3.3 \quad (\exists I_0.(\exists R_0.(\forall j.0 \leq j < I_0 \longrightarrow a[j] \leq a[R_0]) \land 0 \leq I_0 < n \land 0 \leq R_0 < n \\ \land a[R_0] < a[I_0] \land r = I_0) \land i = I_0 + 1) \longrightarrow 0 \leq r < n$$

...

Läuft!

Vereinfachte Verifikationsbedingungen

...

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

...

Läuft!

Vereinfachte Verifikationsbedingungen

...

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

...

Läuft!

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 Verifkationsbedingungen 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.