

Korrekte Software: Grundlagen und Methoden

Vorlesung 9 vom 06.06.19

Vorwärts mit Floyd und Hoare

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2019

Fahrplan

- ▶ Einführung
- ▶ Operationale Semantik
- ▶ Denotationale Semantik
- ▶ Äquivalenz der Operationalen und Denotationalen Semantik
- ▶ Der Floyd-Hoare-Kalkül
- ▶ Invarianten und die Korrektheit des Floyd-Hoare-Kalküls
- ▶ Strukturierte Datentypen
- ▶ Verifikationsbedingungen
- ▶ Vorwärts mit Floyd und Hoare
- ▶ Modellierung
- ▶ Spezifikation von Funktionen
- ▶ Referenzen und Speichermodelle
- ▶ Funktionsaufrufe und das Framing-Problem
- ▶ 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):

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

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

Der Floyd-Hoare-Kalkül

Vorwärts

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:

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

- Die anderen Regeln passen:

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

$$\frac{\vdash \{A\} c_1 \{B\} \quad \vdash \{B\} c_2 \{C\}}{\vdash \{A\} c_1; c_2 \{C\}} \quad \frac{\vdash \{A \wedge b\} c \{A\}}{\vdash \{A\} \text{ while } (b) c \{A \wedge \neg b\}}$$

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

Zuweisungsregel Vorwärts

- ▶ Alternative Zuweisungsregel (nach Floyd):

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

- ▶ $FV(P)$ sind die **freien** Variablen in P .
- ▶ Jetzt ist die Vorbedingung offen — Regel kann vorwärts angewandt werden
- ▶ Gilt auch für die anderen Regeln.

Vorwärtsverkettung

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

```
// {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

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

```
// {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:

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

Vorwärtsverkettung

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

```
// {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:

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

Vorwärtsverkettung

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

```
// {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:

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

Regeln 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:

$$V \notin FV(P) \implies \exists V. P \equiv P$$

- 4 Wenn x vorher zugewiesen wurde, Vereinfachung mit

$$\exists V. P[V] \wedge V = t \implies P[t/V]$$

Vorwärtsverkettung

- ▶ Vorwärtsaxiom äquivalent zum Rückwärtsaxiom.
- ▶ Vorteil: Vorbedingung bleibt kleiner
- ▶ Nachteil: in der Anwendung **umständlicher**
- ▶ Vereinfachung benötigt Lemma: $\exists x. P(x) \wedge x = t \iff P(t)$

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

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

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 $Q \implies Q'$.
- ▶ Semantische Charakterisierung:

Stärkste Nachbedingung

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

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

- ▶ 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

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

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

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

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

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

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

Überblick: Approximative stärkste Nachbedingung

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

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

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

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

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

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

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

Überblick: Approximative stärkste Nachbedingung

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

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

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

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

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

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

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

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

Überblick: Approximative stärkste Nachbedingung

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

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

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

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

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

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

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

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

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

Überblick: Approximative stärkste Nachbedingung

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

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

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

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

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

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

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

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

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

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

Überblick: Approximative stärkste Nachbedingung

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

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

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

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

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

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

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

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

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

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

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

Überblick: Approximative stärkste Nachbedingung

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

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

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

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

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

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

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

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

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

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

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

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

Überblick: Approximative stärkste Nachbedingung

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

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

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

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

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

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

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

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

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

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

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

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

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

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;
   //  $asp_2 =$ 
   //
3  c = 1;
   //  $asp_3 =$ 
   //
4  while (c ≤ n) /** inv { $p = (c - 1)! \wedge c - 1 \leq n$ }; */
5    p = p * c;
   //
6    c = c + 1;
   //
7  }
   //  $asp_4 =$ 
8  // { $p = n!$ }
```

Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  // {0 ≤ n}
2  p = 1;
   //  $asp_2 = \{\exists V. 0 \leq n[V/p] \wedge p = (1[V/p])\}$ 
   //
3  c = 1;
   //  $asp_3 =$ 
   //
4  while ( c ≤ n ) /** inv {  $p = (c - 1)! \wedge c - 1 \leq n$  }; */
5    p = p * c;
   //
6    c = c + 1;
   //
7  }
   //  $asp_4 =$ 
8  // {  $p = n!$  }
```

Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  // {0 ≤ n}
2  p = 1;
   //  $asp_2 = \{\exists V. 0 \leq n[V/p] \wedge p = (1[V/p])\}$ 
   //  $\rightsquigarrow asp_2 = \{0 \leq n \wedge p = 1\}$ 
3  c = 1;
   //  $asp_3 =$ 
   //
4  while (c ≤ n) /** inv { $p = (c - 1)! \wedge c - 1 \leq n$ }; */
5    p = p * c;
   //
6    c = c + 1;
   //
7  }
   //  $asp_4 =$ 
8  // { $p = n!$ }
```

Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  // {0 ≤ n}
2  p = 1;
   //  $asp_2 = \{\exists V. 0 \leq n[V/p] \wedge p = (1[V/p])\}$ 
   //  $\rightsquigarrow asp_2 = \{0 \leq n \wedge p = 1\}$ 
3  c = 1;
   //  $asp_3 = \{\exists V. (0 \leq n \wedge p = 1)[V/c] \wedge c = (1[V/c])\}$ 
   //
4  while ( c <= n ) /** inv {p = (c - 1)! ∧ c - 1 ≤ n}; */
5    p = p * c;
   //
6    c = c + 1;
   //
7  }
   //  $asp_4 =$ 
8  // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  //  $\{0 \leq n\}$ 
2  p = 1;
   //  $asp_2 = \{\exists V. 0 \leq n[V/p] \wedge p = (1[V/p])\}$ 
   //  $\rightsquigarrow asp_2 = \{0 \leq n \wedge p = 1\}$ 
3  c = 1;
   //  $asp_3 = \{\exists V. (0 \leq n \wedge p = 1)[V/c] \wedge c = (1[V/c])\}$ 
   //  $\rightsquigarrow asp_3 = \{0 \leq n \wedge p = 1 \wedge c = 1\}$ 
4  while (c <= n) /** inv  $\{p = (c - 1)! \wedge c - 1 \leq n\}$ ; */
5    p = p * c;
   //
6    c = c + 1;
   //
7  }
   //  $asp_4 =$ 
8  //  $\{p = n!\}$ 
```


Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  //  $\{0 \leq n\}$ 
2  p = 1;
   //  $asp_2 = \{\exists V. 0 \leq n[V/p] \wedge p = (1[V/p])\}$ 
   //  $\rightsquigarrow asp_2 = \{0 \leq n \wedge p = 1\}$ 
3  c = 1;
   //  $asp_3 = \{\exists V. (0 \leq n \wedge p = 1)[V/c] \wedge c = (1[V/c])\}$ 
   //  $\rightsquigarrow asp_3 = \{0 \leq n \wedge p = 1 \wedge c = 1\}$ 
4  while (c <= n) /** inv  $\{p = (c - 1)! \wedge c - 1 \leq n\}$ ; */
5     p = p * c;
   //
6     c = c + 1;
   //
7 }
   //  $asp_4 = \{\neg(c \leq n) \wedge p = (c - 1)! \wedge c - 1 \leq n\}$ 
8 //  $\{p = n!\}$ 
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  // {0 ≤ n}
2  p = 1;
   //  $\text{svc}_2 =$ 
   c = 1;
   //  $\text{svc}_3 =$ 
4  while (c ≤ n) /** inv { $p = (c - 1)! \wedge c - 1 \leq n$ }; */
5    p = p * c;
   //
7  SVCat5 =
6    c = c + 1;
   //  $\text{svc}_6 =$ 
7    }
   //  $\text{svc}_4 =$ 
8  // { $p = n!$ }
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  // {0 ≤ n}
2  p = 1;
   //  $\text{svc}_2 = \emptyset$ 
   c = 1;
   //  $\text{svc}_3 =$ 
4  while (c ≤ n) /** inv { $p = (c - 1)! \wedge c - 1 \leq n$ }; */
5    p = p * c;
   //
7  SVCat5 =
6    c = c + 1;
   //  $\text{svc}_6 =$ 
7    }
   //  $\text{svc}_4 =$ 
8  // { $p = n!$ }
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  //  $\{0 \leq n\}$ 
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;
   //
7  SVCat5 =
6    c = c + 1;
   //  $\text{svc}_6 =$ 
7    }
   //  $\text{svc}_4 =$ 
8  //  $\{p = n!\}$ 
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  //  $\{0 \leq n\}$ 
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;
   //
7  SVCat5 =  $\emptyset$ 
6    c = c + 1;
   //  $\text{svc}_6 =$ 
7    }
   //  $\text{svc}_4 =$ 
8  //  $\{p = n!\}$ 
```

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;
   //
7  SVCat5 =  $\emptyset$ 
6    c = c + 1;
   //  $\text{svc}_6 = \emptyset$ 
7  }
   //  $\text{svc}_4 =$ 
8  // { $p = n!$ }
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  //  $\{0 \leq n\}$ 
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;
   //
7  SVCat5 =  $\emptyset$ 
6    c = c + 1;
   //  $\text{svc}_6 = \emptyset$ 
7  }
   //  $\text{svc}_4 = \{asp_3 \implies (p = (c - 1)! \wedge c - 1 \leq n), asp_5 \implies (p = (c - 1)! \wedge$ 
   //  $c - 1 \leq n)\}$ 
8  //  $\{p = n!\}$ 
```

Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  //  $\{0 \leq n\}$ 
2  p = 1;
   //  $asp_2 = \{0 \leq n \wedge p = 1\}$ 
3  c = 1;
   //  $asp_3 = \{0 \leq n \wedge p = 1 \wedge c = 1\}$ 
4  while (c <= n) /** inv  $\{p = (c - 1)! \wedge c - 1 \leq n\}$ ; */
5    p = p * c;
   //  $asp_5 =$ 
   //
   //
   c = c + 1;
   //  $asp_6 =$ 
   //  $\rightsquigarrow$ 
   //  $\rightsquigarrow$ 
   }
   //  $asp_4 = \{\neg(c \leq n) \wedge p = (c - 1)! \wedge c - 1 \leq n\}$ 
8  //  $\{p = n!\}$ 
```


Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  //  $\{0 \leq n\}$ 
2  p = 1;
   //  $asp_2 = \{0 \leq n \wedge p = 1\}$ 
3  c = 1;
   //  $asp_3 = \{0 \leq n \wedge p = 1 \wedge c = 1\}$ 
4  while (c <= n) /** inv  $\{p = (c - 1)! \wedge c - 1 \leq n\}$ ; */
5    p = p * c;
   //  $asp_5 = \{\exists V_1. (p = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n)[V_1/p] \wedge p = (p \cdot c)[V_1/p]\}$ 
   //
   //
   c = c + 1;
   //  $asp_6 =$ 
   //  $\rightsquigarrow$ 
   //  $\rightsquigarrow$ 
   }
   //  $asp_4 = \{\neg(c \leq n) \wedge p = (c - 1)! \wedge c - 1 \leq n\}$ 
8  //  $\{p = n!\}$ 
```

Beispiel: Fakultät, stärkste Nachbedingung

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

```
1 // {0 ≤ n}
2 p = 1;
  //  $asp_2 = \{0 \leq n \wedge p = 1\}$ 
3 c = 1;
  //  $asp_3 = \{0 \leq n \wedge p = 1 \wedge c = 1\}$ 
4 while (c ≤ n) /** inv {p = (c - 1)! ∧ c - 1 ≤ n}; */
5   p = p * c;
    //  $asp_5 = \{\exists V_1. (p = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n)[V_1/p] \wedge p = (p \cdot c)[V_1/p]\}$ 
    //  $\rightsquigarrow asp_5 = \{\exists V_1. (V_1 = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n) \wedge p = (V_1 \cdot c)\}$ 
    //
    c = c + 1;
    //  $asp_6 =$ 
    //  $\rightsquigarrow$ 
    //  $\rightsquigarrow$ 
    }
  //  $asp_4 = \{\neg(c \leq n) \wedge p = (c - 1)! \wedge c - 1 \leq n\}$ 
8 // {p = n!}
```

Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  //  $\{0 \leq n\}$ 
2  p = 1;
   //  $asp_2 = \{0 \leq n \wedge p = 1\}$ 
3  c = 1;
   //  $asp_3 = \{0 \leq n \wedge p = 1 \wedge c = 1\}$ 
4  while (c <= n) /** inv  $\{p = (c - 1)! \wedge c - 1 \leq n\}$ ; */
5    p = p * c;
   //  $asp_5 = \{\exists V_1. (p = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n)[V_1/p] \wedge p = (p \cdot c)[V_1/p]\}$ 
   //  $\rightsquigarrow asp_5 = \{\exists V_1. (V_1 = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n) \wedge p = (V_1 \cdot c)\}$ 
   //  $\rightsquigarrow asp_5 = \{c - 1 \leq n \wedge c \leq n \wedge p = (c - 1)! \cdot c\}$ 
6    c = c + 1;
   //  $asp_6 =$ 
   //  $\rightsquigarrow$ 
   //  $\rightsquigarrow$ 
   }
   //  $asp_4 = \{\neg(c \leq n) \wedge p = (c - 1)! \wedge c - 1 \leq n\}$ 
8  //  $\{p = n!\}$ 
```

Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  // {0 ≤ n}
2  p = 1;
   //  $asp_2 = \{0 \leq n \wedge p = 1\}$ 
3  c = 1;
   //  $asp_3 = \{0 \leq n \wedge p = 1 \wedge c = 1\}$ 
4  while (c ≤ n) /** inv { $p = (c - 1)! \wedge c - 1 \leq n$ }; */
5    p = p * c;
   //  $asp_5 = \{\exists V_1. (p = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n)[V_1/p] \wedge p = (p \cdot c)[V_1/p]\}$ 
   //  $\rightsquigarrow asp_5 = \{\exists V_1. (V_1 = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n) \wedge p = (V_1 \cdot c)\}$ 
   //  $\rightsquigarrow asp_5 = \{c - 1 \leq n \wedge c \leq n \wedge p = (c - 1)! \cdot c\}$ 
6    c = c + 1;
   //  $asp_6 = \{\exists V_2. (c - 1 \leq n \wedge c \leq n \wedge p = (c - 1)! \cdot c)[V_2/c] \wedge c = (c + 1)[V_2/c]\}$ 
   //  $\rightsquigarrow$ 
   //  $\rightsquigarrow$ 
   }
   //  $asp_4 = \{\neg(c \leq n) \wedge p = (c - 1)! \wedge c - 1 \leq n\}$ 
8  // { $p = n!$ }
```

Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  //  $\{0 \leq n\}$ 
2  p = 1;
   //  $asp_2 = \{0 \leq n \wedge p = 1\}$ 
3  c = 1;
   //  $asp_3 = \{0 \leq n \wedge p = 1 \wedge c = 1\}$ 
4  while (c <= n) /** inv  $\{p = (c - 1)! \wedge c - 1 \leq n\}$ ; */
5    p = p * c;
   //  $asp_5 = \{\exists V_1. (p = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n)[V_1/p] \wedge p = (p \cdot c)[V_1/p]\}$ 
   //  $\rightsquigarrow asp_5 = \{\exists V_1. (V_1 = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n) \wedge p = (V_1 \cdot c)\}$ 
   //  $\rightsquigarrow asp_5 = \{c - 1 \leq n \wedge c \leq n \wedge p = (c - 1)! \cdot c\}$ 
6    c = c + 1;
   //  $asp_6 = \{\exists V_2. (c - 1 \leq n \wedge c \leq n \wedge p = (c - 1)! \cdot c)[V_2/c] \wedge c = (c + 1)[V_2/c]\}$ 
   //  $\rightsquigarrow asp_6 = \{\exists V_2. (V_2 - 1 \leq n \wedge V_2 \leq n \wedge p = (V_2 - 1)! \cdot V_2) \wedge c = (V_2 + 1)\}$ 
   //  $\rightsquigarrow$ 
   }
   //  $asp_4 = \{\neg(c \leq n) \wedge p = (c - 1)! \wedge c - 1 \leq n\}$ 
8  //  $\{p = n!\}$ 
```

Beispiel: Fakultät, stärkste Nachbedingung

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

```
1  //  $\{0 \leq n\}$ 
2  p = 1;
   //  $asp_2 = \{0 \leq n \wedge p = 1\}$ 
3  c = 1;
   //  $asp_3 = \{0 \leq n \wedge p = 1 \wedge c = 1\}$ 
4  while ( c <= n ) /** inv  $\{p = (c - 1)! \wedge c - 1 \leq n\}$ ; */
5    p = p * c;
   //  $asp_5 = \{\exists V_1. (p = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n)[V_1/p] \wedge p = (p \cdot c)[V_1/p]\}$ 
   //  $\rightsquigarrow asp_5 = \{\exists V_1. (V_1 = (c - 1)! \wedge (c - 1) \leq n \wedge c \leq n) \wedge p = (V_1 \cdot c)\}$ 
   //  $\rightsquigarrow asp_5 = \{c - 1 \leq n \wedge c \leq n \wedge p = (c - 1)! \cdot c\}$ 
6    c = c + 1;
   //  $asp_6 = \{\exists V_2. (c - 1 \leq n \wedge c \leq n \wedge p = (c - 1)! \cdot c)[V_2/c] \wedge c = (c + 1)[V_2/c]\}$ 
   //  $\rightsquigarrow asp_6 = \{\exists V_2. (V_2 - 1 \leq n \wedge V_2 \leq n \wedge p = (V_2 - 1)! \cdot V_2) \wedge c = (V_2 + 1)\}$ 
   //  $\rightsquigarrow asp_6 = \{c - 2 \leq n \wedge c - 1 \leq n \wedge p = (c - 2)! \cdot (c - 1)\}$ 
7  }
   //  $asp_4 = \{\neg(c \leq n) \wedge p = (c - 1)! \wedge c - 1 \leq n\}$ 
8  //  $\{p = n!\}$ 
```

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  }
   //  $\text{svc}_4 = \{ \text{asp}_3 \implies (p = (c - 1)! \wedge c - 1 \leq n),$ 
   //            $\text{asp}_5 \implies (p = (c - 1)! \wedge c - 1 \leq n) \}$ 
   //
   //
   //
   //
8  // { $p = n!$ }
```

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  }
   //  $\text{svc}_4 = \{ \text{asp}_3 \implies (p = (c - 1)! \wedge c - 1 \leq n),$ 
   //            $\text{asp}_5 \implies (p = (c - 1)! \wedge c - 1 \leq n) \}$ 
   //  $\rightsquigarrow \text{svc}_4 = \{ (0 \leq n \wedge p = 1 \wedge c = 1) \implies (p = (c - 1)! \wedge c - 1 \leq n),$ 
   //            $(c - 2 \leq n \wedge c - 1 \leq n \wedge p = (c - 2)! \cdot (c - 1))$ 
   //            $\implies (p = (c - 1)! \wedge c - 1 \leq n) \}$ 
8  // { $p = n!$ }
```


Schließlich zu zeigen

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

Beispiel: Suche nach dem Maximalen Element

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

► Problem: wir müssen u.a. zeigen

$$(\exists V_1. (\forall j. 0 \leq j < i - 1 \rightarrow a[j] \leq a[V_1]) \wedge$$

$$i - 1 \neq n \wedge a[V_1] < a[i - 1] \wedge r = i - 1) \rightarrow 0 \leq r < n$$

Deshalb: Invariante **verstärken!**

Beispiel: Suche nach dem Maximalen Element

Verstärkte Invariante (und Schleifenbedingung):

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

$$(\exists V_1. (\forall j. 0 \leq j < i - 1 \rightarrow a[j] \leq a[V_1]) \wedge \\ 0 \leq i - 1 < n \wedge a[V_1] < a[i - 1] \wedge r = i - 1) \rightarrow 0 \leq r < n$$

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 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.