Korrekte Software: Grundlagen und Methoden

Vorlesung 9 vom 16.06.20

Vorwärts mit Floyd und Hoare

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2020

Feedback Online-Lehre

- Was kann besser werden?
  - Aufgezeichnete Vorlesungen?
  - Lesematerial/“Flipped Classroom”?
  - Andere Formen der Gruppenarbeit?
- Was ist gut/schlecht an Zoom?
  - Technische Probleme?
  - Funktionalität?
  - Break-Out Rooms?
- Was wollen wir ändern?

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
- 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

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 seinM
  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:

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

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)

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

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

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

If-Regel Vorwärts

- Abgeleitete If-Regel:

$$\frac{\vdash \{A \wedge b\}\, c_0\, \{B_1\} \qquad \vdash \{A \wedge \neg b\}\, c_1\, \{B_2\}}{\vdash \{A\}\ \textbf{if}\ (b)\ c_0\ \textbf{else}\ c_1\, \{B_1 \vee B_2\}}$$

- Durch Verkettung der If-Regel mit Weakening: $B_1 \Longrightarrow B_1 \vee B_2$

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 9.1: Das Leben mit 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)

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

$$\exists y \exists x.\, x + 3 = y \qquad \forall x \exists y.\, x * y = 3 \qquad \exists x \forall y.\, x * y > 1$$

Vorwärtsverkettung

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

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

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

```
// {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:

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

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

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

```
// {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:

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

Regeln der Vorwärtsverkettung

Eigenschaften des Existenzquantors:

$$P[V] \wedge V = t \Longrightarrow P[t/V] \wedge V = t \tag{1}$$

$$\exists V.\, P[V] \wedge V = t \Longrightarrow P[t/V] \tag{2}$$

$$\text{wenn } V \notin FV(Q) \text{ dann } (\exists V.\, P) \wedge Q \Longleftrightarrow \exists V.\, P \wedge Q \tag{3}$$

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

Regeln der Vorwärtsverkettung

Eigenschaften des Existenzquantors:

$$P[V] \wedge V = t \Longrightarrow P[t/V] \wedge V = t \tag{1}$$

$$\exists V.\, P[V] \wedge V = t \Longrightarrow P[t/V] \tag{2}$$

$$\text{wenn } V \notin FV(Q) \text{ dann } (\exists V.\, P) \wedge Q \Longleftrightarrow \exists V.\, P \wedge Q \tag{3}$$

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

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)

Beispiel Vorwärtsverkettung

```
// {a < b}
a= b+a ;
// {∃a_1.(a < b)[a_1/a] ∧ a = (b + a)[a_1/a]}
```

Beispiel Vorwärtsverkettung

```
// {a < b}
a= b+a ;
// {∃a_1.(a < b)[a_1/a] ∧ a = (b + a)[a_1/a]}
// {∃a_1. a_1 < b ∧ a = b + a_1}
```

Beispiel Vorwärtsverkettung

```
// {a < b}
a= b+a ;
// {∃a_1. (a < b)[a_1/a] ∧ a = (b + a)[a_1/a]}
// {∃a_1. a_1 < b ∧ a = b + a_1}
b= 3*a+b ;
// {∃b_1. (∃a_1. a_1 < b ∧ a = b + a_1)[b_1/b] ∧ b = (3a + b)[b_1/b]}
```

Beispiel Vorwärtsverkettung

// $\{a < b\}$
a= b+a ;
// $\{\exists a_1 . (a < b)[a_1/a] \wedge a = (b + a)[a_1/a]\}$
// $\{\exists a_1 . a_1 < b \wedge a = b + a_1\}$
b= 3∗a+b ;
// $\{\exists b_1 . (\exists a_1 . a_1 < b \wedge a = b + a_1)[b_1/b] \wedge b = (3a + b)[b_1/b]\}$
// $\{\exists b_1 \exists a_1 . a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1\}$

Beispiel Vorwärtsverkettung

$// \ \{a < b\}$
a= b+a ;
$// \ \{\exists a_1 . (a < b)[a_1/a] \land a = (b + a)[a_1/a]\}$
$// \ \{\exists a_1 . \, a_1 < b \land a = b + a_1\}$
b= 3∗a+b ;
$// \ \{\exists b_1 . (\exists a_1 . \, a_1 < b \land a = b + a_1)[b_1/b] \land b = (3a + b)[b_1/b]\}$
$// \ \{\exists b_1 \exists a_1 . \, a_1 < b_1 \land a = b_1 + a_1 \land b = 3a + b_1\}$
a= b− 2∗a ;
$// \ \{\exists a_2 . (\exists b_1 \exists a_1 . \, a_1 < b_1 \land a = b_1 + a_1 \land b = 3a + b_1)[a_2/a] \land a = (b - 2a)[a_2/a]\}$
$// \ \{\exists a_2 \exists b_1 \exists a_1 . \, a_1 < b_1 \land a_2 = b_1 + a_1 \land b = 3a_2 + b_1 \land a = b - 2a_2\}$

Beispiel Vorwärtsverkettung

$// \ \{a < b\}$
a= b+a ;
$// \ \{\exists a_1 . (a < b)[a_1/a] \wedge a = (b + a)[a_1/a]\}$
$// \ \{\exists a_1 . \, a_1 < b \wedge a = b + a_1\}$
b= 3∗a+b ;
$// \ \{\exists b_1 . (\exists a_1 . \, a_1 < b \wedge a = b + a_1)[b_1/b] \wedge b = (3a + b)[b_1/b]\}$
$// \ \{\exists b_1 \exists a_1 . \, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1\}$
a= b− 2∗a ;
$// \ \{\exists a_2 . (\exists b_1 \exists a_1 . \, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1)[a_2/a] \wedge a = (b - 2a)[a_2/a]\}$
$// \ \{\exists a_2 \exists b_1 \exists a_1 . \, a_1 < b_1 \wedge a_2 = b_1 + a_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2\}$
$// \ \{\exists a_2 \exists b_1 \exists a_1 . \, a_1 < b_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2 \wedge a_2 = b_1 + a_1\}$

Beispiel Vorwärtsverkettung

$$
\begin{array}{l}
// \ \{a < b\} \\
\texttt{a= b+a ;} \\
// \ \{\exists a_1.\, (a < b)[a_1/a] \wedge a = (b + a)[a_1/a]\} \\
// \ \{\exists a_1.\, a_1 < b \wedge a = b + a_1\} \\
\texttt{b= 3*a+b ;} \\
// \ \{\exists b_1.\, (\exists a_1.\, a_1 < b \wedge a = b + a_1)[b_1/b] \wedge b = (3a + b)[b_1/b]\} \\
// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1\} \\
\texttt{a= b- 2*a ;} \\
// \ \{\exists a_2.\, (\exists b_1 \exists a_1.\, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1)[a_2/a] \wedge a = (b - 2a)[a_2/a]\} \\
// \ \{\exists a_2 \exists b_1 \exists a_1.\, a_1 < b_1 \wedge a_2 = b_1 + a_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2\} \\
// \ \{\exists a_2 \exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2 \wedge a_2 = b_1 + a_1\} \\
// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3(b_1 + a_1) + b_1 \wedge a = b - 2(b_1 + a_1)\}
\end{array}
$$

Beispiel Vorwärtsverkettung

$// \ \{a < b\}$
a= b+a ;
$// \ \{\exists a_1 . (a < b)[a_1/a] \wedge a = (b + a)[a_1/a]\}$
$// \ \{\exists a_1 . \, a_1 < b \wedge a = b + a_1\}$
b= 3∗a+b ;
$// \ \{\exists b_1 . (\exists a_1 . \, a_1 < b \wedge a = b + a_1)[b_1/b] \wedge b = (3a + b)[b_1/b]\}$
$// \ \{\exists b_1 \exists a_1 . \, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1\}$
a= b− 2∗a ;
$// \ \{\exists a_2 . (\exists b_1 \exists a_1 . \, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1)[a_2/a] \wedge a = (b - 2a)[a_2/a]\}$
$// \ \{\exists a_2 \exists b_1 \exists a_1 . \, a_1 < b_1 \wedge a_2 = b_1 + a_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2\}$
$// \ \{\exists a_2 \exists b_1 \exists a_1 . \, a_1 < b_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2 \wedge a_2 = b_1 + a_1\}$
$// \ \{\exists b_1 \exists a_1 . \, a_1 < b_1 \wedge b = 3(b_1 + a_1) + b_1 \wedge a = b - 2(b_1 + a_1)\}$
$// \ \{\exists b_1 \exists a_1 . \, a_1 < b_1 \wedge b = 3b_1 + 3a_1 + b_1 \wedge a = b - 2b_1 - 2a_1)\}$

Beispiel Vorwärtsverkettung

$$
\begin{array}{l}
// \ \{a < b\} \\
\texttt{a= b+a ;} \\
// \ \{\exists a_1.\, (a < b)[a_1/a] \wedge a = (b + a)[a_1/a]\} \\
// \ \{\exists a_1.\, a_1 < b \wedge a = b + a_1\} \\
\texttt{b= 3*a+b ;} \\
// \ \{\exists b_1.\, (\exists a_1.\, a_1 < b \wedge a = b + a_1)[b_1/b] \wedge b = (3a + b)[b_1/b]\} \\
// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1\} \\
\texttt{a= b- 2*a ;} \\
// \ \{\exists a_2.\, (\exists b_1 \exists a_1.\, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1)[a_2/a] \wedge a = (b - 2a)[a_2/a]\} \\
// \ \{\exists a_2 \exists b_1 \exists a_1.\, a_1 < b_1 \wedge a_2 = b_1 + a_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2\} \\
// \ \{\exists a_2 \exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2 \wedge a_2 = b_1 + a_1\} \\
// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3(b_1 + a_1) + b_1 \wedge a = b - 2(b_1 + a_1)\} \\
// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3b_1 + 3a_1 + b_1 \wedge a = b - 2b_1 - 2a_1)\} \\
// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 4b_1 + 3a_1 \wedge a = b - 2b_1 - 2a_1)\}
\end{array}
$$

Beispiel Vorwärtsverkettung

$$
\begin{aligned}
&// \ \{a < b\} \\
&\texttt{a= b+a ;} \\
&// \ \{\exists a_1.\, (a < b)[a_1/a] \wedge a = (b + a)[a_1/a]\} \\
&// \ \{\exists a_1.\, a_1 < b \wedge a = b + a_1\} \\
&\texttt{b= 3*a+b ;} \\
&// \ \{\exists b_1.\, (\exists a_1.\, a_1 < b \wedge a = b + a_1)[b_1/b] \wedge b = (3a + b)[b_1/b]\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1\} \\
&\texttt{a= b- 2*a ;} \\
&// \ \{\exists a_2.\, (\exists b_1 \exists a_1.\, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1)[a_2/a] \wedge a = (b - 2a)[a_2/a]\} \\
&// \ \{\exists a_2 \exists b_1 \exists a_1.\, a_1 < b_1 \wedge a_2 = b_1 + a_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2\} \\
&// \ \{\exists a_2 \exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2 \wedge a_2 = b_1 + a_1\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3(b_1 + a_1) + b_1 \wedge a = b - 2(b_1 + a_1)\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3b_1 + 3a_1 + b_1 \wedge a = b - 2b_1 - 2a_1)\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 4b_1 + 3a_1 \wedge a = b - 2b_1 - 2a_1)\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 4b_1 + 3a_1 \wedge a = (4b_1 + 3a_1) - 2b_1 - 2a_1)\}
\end{aligned}
$$

Beispiel Vorwärtsverkettung

$// \ \{a < b\}$
a= b+a ;
$// \ \{\exists a_1 . (a < b)[a_1/a] \wedge a = (b + a)[a_1/a]\}$
$// \ \{\exists a_1 . a_1 < b \wedge a = b + a_1\}$
b= 3∗a+b ;
$// \ \{\exists b_1 . (\exists a_1 . a_1 < b \wedge a = b + a_1)[b_1/b] \wedge b = (3a + b)[b_1/b]\}$
$// \ \{\exists b_1 \exists a_1 . a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1\}$
a= b− 2∗a ;
$// \ \{\exists a_2 . (\exists b_1 \exists a_1 . a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1)[a_2/a] \wedge a = (b - 2a)[a_2/a]\}$
$// \ \{\exists a_2 \exists b_1 \exists a_1 . a_1 < b_1 \wedge a_2 = b_1 + a_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2\}$
$// \ \{\exists a_2 \exists b_1 \exists a_1 . a_1 < b_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2 \wedge a_2 = b_1 + a_1\}$
$// \ \{\exists b_1 \exists a_1 . a_1 < b_1 \wedge b = 3(b_1 + a_1) + b_1 \wedge a = b - 2(b_1 + a_1)\}$
$// \ \{\exists b_1 \exists a_1 . a_1 < b_1 \wedge b = 3b_1 + 3a_1 + b_1 \wedge a = b - 2b_1 - 2a_1)\}$
$// \ \{\exists b_1 \exists a_1 . a_1 < b_1 \wedge b = 4b_1 + 3a_1 \wedge a = b - 2b_1 - 2a_1)\}$
$// \ \{\exists b_1 \exists a_1 . a_1 < b_1 \wedge b = 4b_1 + 3a_1 \wedge a = (4b_1 + 3a_1) - 2b_1 - 2a_1)\}$
$// \ \{\exists b_1 \exists a_1 . a_1 < b_1 \wedge b = 4b_1 + 3a_1 \wedge a = 2b_1 + a_1\}$

Beispiel Vorwärtsverkettung

$$
\begin{aligned}
&// \ \{a < b\} \\
&\texttt{a= b+a ;} \\
&// \ \{\exists a_1.\, (a < b)[a_1/a] \wedge a = (b + a)[a_1/a]\} \\
&// \ \{\exists a_1.\, a_1 < b \wedge a = b + a_1\} \\
&\texttt{b= 3*a+b ;} \\
&// \ \{\exists b_1.\, (\exists a_1.\, a_1 < b \wedge a = b + a_1)[b_1/b] \wedge b = (3a + b)[b_1/b]\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1\} \\
&\texttt{a= b- 2*a ;} \\
&// \ \{\exists a_2.\, (\exists b_1 \exists a_1.\, a_1 < b_1 \wedge a = b_1 + a_1 \wedge b = 3a + b_1)[a_2/a] \wedge a = (b - 2a)[a_2/a]\} \\
&// \ \{\exists a_2 \exists b_1 \exists a_1.\, a_1 < b_1 \wedge a_2 = b_1 + a_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2\} \\
&// \ \{\exists a_2 \exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3a_2 + b_1 \wedge a = b - 2a_2 \wedge a_2 = b_1 + a_1\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3(b_1 + a_1) + b_1 \wedge a = b - 2(b_1 + a_1)\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 3b_1 + 3a_1 + b_1 \wedge a = b - 2b_1 - 2a_1)\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 4b_1 + 3a_1 \wedge a = b - 2b_1 - 2a_1)\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 4b_1 + 3a_1 \wedge a = (4b_1 + 3a_1) - 2b_1 - 2a_1)\} \\
&// \ \{\exists b_1 \exists a_1.\, a_1 < b_1 \wedge b = 4b_1 + 3a_1 \wedge a = 2b_1 + a_1\}
\end{aligned}
$$

Arbeitsblatt 9.2: Vorwärtsverkettung

Gegeben folgendes Programm. Berechnet die Vorwärtsverkettung der Vorbedingung

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

Was bewirkt das Programm?

Vorwärtsverkettung

- Vorwärtsaxiom äquivalent zum Rückwärtsaxiom.
- Vorteil: Vorbedingung bleibt kleiner
- Nachteil: in der Anwendung **umständlicher**
- Vereinfachung benötigt Rechnungmit Existenzquantor

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
  - $\mathrm{sp}(P, c)$ 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

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

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

Überblick: Approximative stärkste Nachbedingung

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

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)! ∧ 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 ≤ n}
2  p= 1;
   // asp_2 = {∃V.0 ≤ n[V/p] ∧ p = (1[V/p])}
   //
3  c= 1;
   // asp_3 =
   //
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 ≤ n}
2  p= 1;
   // asp_2 = {∃V.0 ≤ n[V/p] ∧ p = (1[V/p])}
   // asp_2 = {0 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 =
   //
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 ≤ n}
2  p= 1;
   // asp_2 = {∃V.0 ≤ n[V/p] ∧ p = (1[V/p])}
   // asp_2 = {0 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 = {∃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   }
   // 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 = {∃V. 0 ≤ n[V/p] ∧ p = (1[V/p])}
   // asp_2 = {0 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 = {∃V. (0 ≤ n ∧ p = 1)[V/c] ∧ c = (1[V/c])}
   // asp_3 = {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   }
   // 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 = {∃V. 0 ≤ n[V/p] ∧ p = (1[V/p])}
   // asp_2 = {0 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 = {∃V. (0 ≤ n ∧ p = 1)[V/c] ∧ c = (1[V/c])}
   // asp_3 = {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   }
   // asp_4 = {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ n}
8  // {p = n!}
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  // {0 ≤ n}
2  p= 1;
   // svc2 =
3  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 =
8  // {p = n!}
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  // {0 ≤ n}
2  p= 1;
   // svc2 = ∅
3  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 =
8  // {p = n!}
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  // {0 ≤ n}
2  p= 1;
   // svc2 = ∅
3  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 =
8  // {p = n!}
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  // {0 ≤ n}
2  p= 1;
   // svc2 = ∅
3  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 =
8  // {p = n!}
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  // {0 ≤ n}
2  p= 1;
   // svc2 = ∅
3  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 =
8  // {p = n!}
```

Fakultät: Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  // {0 ≤ n}
2  p= 1;
   // svc_2 = ∅
3  c= 1;
   // svc_3 = ∅
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5   p = p * c;
    // svc_5 = ∅
6   c = c + 1;
    // svc_6 = ∅
7   }
    // svc_4 = {asp_3 ⟹ (p = (c−1)! ∧ c − 1 ≤ n), asp_6 ⟹ (p = (c − 1)! ∧
   c − 1 ≤ 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 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 = {0 ≤ n ∧ p = 1 ∧ c = 1}
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5   p = p * c;
    // asp_5 =
    //
    //
    c = c + 1;
    // asp_6 =
    //
    //
    }
   // asp_4 = {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ 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 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 = {0 ≤ n ∧ p = 1 ∧ c = 1}
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5   p = p * c;
    // asp_5 = {∃V_1.(p = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n)[V_1/p] ∧ p = (p · c)[V_1/p]}
    //
    //
    c = c + 1;
    // asp_6 =
    //
    //
    }
   // asp_4 = {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ 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 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 = {0 ≤ n ∧ p = 1 ∧ c = 1}
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5   p = p * c;
    // asp_5 = {∃V_1.(p = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n)[V_1/p] ∧ p = (p · c)[V_1/p]}
    // asp_5 = {∃V_1.(V_1 = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n) ∧ p = (V_1 · c)}
    //
    c = c + 1;
    // asp_6 =
    //
    //
    }
   // asp_4 = {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ 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 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 = {0 ≤ n ∧ p = 1 ∧ c = 1}
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5   p = p * c;
    // asp_5 = {∃V_1.(p = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n)[V_1/p] ∧ p = (p · c)[V_1/p]}
    // asp_5 = {∃V_1.(V_1 = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n) ∧ p = (V_1 · c)}
    // asp_5 = {c − 1 ≤ n ∧ c ≤ n ∧ p = (c − 1)! · c}
6   c = c + 1;
    // asp_6 =
    //
    //
    }
   // asp_4 = {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ 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 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 = {0 ≤ n ∧ p = 1 ∧ c = 1}
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5   p = p * c;
    // asp_5 = {∃V_1.(p = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n)[V_1/p] ∧ p = (p · c)[V_1/p]}
    // asp_5 = {∃V_1.(V_1 = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n) ∧ p = (V_1 · c)}
    // asp_5 = {c − 1 ≤ n ∧ c ≤ n ∧ p = (c − 1)! · c}
6   c = c + 1;
    // asp_6 = {∃V_2.(c − 1 ≤ n ∧ c ≤ n ∧ p = (c − 1)! · c)[V_2/c] ∧ c = (c + 1)[V_2/c]}
    //
    //
    }
   // asp_4 = {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ 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 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 = {0 ≤ n ∧ p = 1 ∧ c = 1}
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5   p = p * c;
    // asp_5 = {∃V_1.(p = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n)[V_1/p] ∧ p = (p · c)[V_1/p]}
    // asp_5 = {∃V_1.(V_1 = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n) ∧ p = (V_1 · c)}
    // asp_5 = {c − 1 ≤ n ∧ c ≤ n ∧ p = (c − 1)! · c}
6   c = c + 1;
    // asp_6 = {∃V_2.(c − 1 ≤ n ∧ c ≤ n ∧ p = (c − 1)! · c)[V_2/c] ∧ c = (c + 1)[V_2/c]}
    // asp_6 = {∃V_2.(V_2 − 1 ≤ n ∧ V_2 ≤ n ∧ p = (V_2 − 1)! · V_2) ∧ c = (V_2 + 1)}
    //
   }
   // asp_4 = {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ 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 ≤ n ∧ p = 1}
3  c= 1;
   // asp_3 = {0 ≤ n ∧ p = 1 ∧ c = 1}
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5   p = p * c;
    // asp_5 = {∃V_1.(p = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n)[V_1/p] ∧ p = (p · c)[V_1/p]}
    // asp_5 = {∃V_1.(V_1 = (c − 1)! ∧ (c − 1) ≤ n ∧ c ≤ n) ∧ p = (V_1 · c)}
    // asp_5 = {c − 1 ≤ n ∧ c ≤ n ∧ p = (c − 1)! · c}
6   c = c + 1;
    // asp_6 = {∃V_2.(c − 1 ≤ n ∧ c ≤ n ∧ p = (c − 1)! · c)[V_2/c] ∧ c = (c + 1)[V_2/c]}
    // asp_6 = {∃V_2.(V_2 − 1 ≤ n ∧ V_2 ≤ n ∧ p = (V_2 − 1)! · V_2) ∧ c = (V_2 + 1)}
    // asp_6 = {c − 2 ≤ n ∧ c − 1 ≤ n ∧ p = (c − 2)! · (c − 1)}
7   }
   // asp_4 = {¬(c ≤ n) ∧ p = (c − 1)! ∧ c − 1 ≤ n}
8  // {p = n!}
```

Beispiel: Fakultät, Verifikationsbedingungen

Notation: svc_x = in Zeile x generierte Verifikationsbedingung

```
1  // {0 ≤ n}
2  p= 1;
   // svc_2 = ∅
   c= 1;
   // svc_3 = ∅
4  while (c <= n) //** inv {p = (c − 1)! ∧ c − 1 ≤ n}; */ {
5    p = p * c;
     // svc_5 = ∅
6    c = c + 1;
     // svc_6 = ∅
7    }
   // svc_4 = {asp_3 ⟹ (p = (c − 1)! ∧ c − 1 ≤ n),
   //                  asp_6 ⟹ (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

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

Arbeitsblatt 9.3: Jetzt seid ihr dran!

Berechnet die stärkste Nachbedingung und Verifikationsbedingungen für die ganzzahlige Division:

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

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

$$(\exists V_1.\,(\forall j.\,0 \le j < i-1 \longrightarrow a[j] \le a[V_1]) \wedge i-1 \neq n \wedge a[V_1] < a[i-1] \wedge r = i-1) \longrightarrow 0 \le r < n$$

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 (∀j. 0 ≤ j < i ⟶ a[j] ≤ a[r])
                          ∧ 0 ≤ i ≤ n ∧ 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}
```

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

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 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.