Korrekte Software: Grundlagen und Methoden
Vorlesung 10 vom 12.06.18: Vorwärts mit Floyd und Hoare

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2018

Fahrplan

- Einführung
- Operationale Semantik
- Denotationale Semantik
- Äquivalenz der Operationalen und Denotationalen Semantik
- Die Floyd-Hoare-Logik
- Invarianten und die Korrektheit des Floyd-Hoare-Kalküls
- Strukturierte Datentypen
- Modellierung und Spezifikation
- Verifikationsbedingungen
- Vorwärts mit Floyd und Hoare
- Funktionen und Prozeduren
- Referenzen
- Ausblick und Rückblick

Idee

- Hier ist ein einfaches Programm:

$// \{X = x \wedge Y = y\}$
z = y;

y = x;

x = z;
$// \{X = y \wedge Y = x\}$

Idee

- Hier ist ein einfaches Programm:

$// \{X = x \wedge Y = y\}$
z = y ;

y = x ;
$// \{X = y \wedge Y = z\}$
x = z ;
$// \{X = y \wedge Y = x\}$

Idee

- Hier ist ein einfaches Programm:

$// \{X = x \wedge Y = y\}$
z = y ;
$// \{X = x \wedge Y = z\}$
y = x ;
$// \{X = y \wedge Y = z\}$
x = z ;
$// \{X = y \wedge Y = x\}$

Idee

- Hier ist ein einfaches Programm:

  // $\{X = x \wedge Y = y\}$
  z = y ;
  // $\{X = x \wedge Y = z\}$
  y = x ;
  // $\{X = y \wedge Y = z\}$
  x = z ;
  // $\{X = y \wedge Y = x\}$

- Wir haben gesehen:

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

Idee

- Hier ist ein einfaches Programm:

  // $\{X = x \wedge Y = y\}$
  z = y ;
  // $\{X = x \wedge Y = z\}$
  y = x ;
  // $\{X = y \wedge Y = z\}$
  x = z ;
  // $\{X = y \wedge Y = x\}$

- 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:

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

Vorwärtsanwendung der Regeln

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

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

- Die anderen Regeln passen:

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

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

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

Zuweisungregel 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

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

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

- **Vereinfachung** der letzten Nachbedingung:

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

Vorwärtsverkettung

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

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

- **Vereinfachung** der letzten Nachbedingung:

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

Vorwärtsverkettung

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

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

- **Vereinfachung** der letzten Nachbedingung:

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

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:
$$V \notin FV(P) \Longrightarrow \exists V.\, P \equiv P$$
4. Wenn x vorher zugewiesen wurde, Vereinfachung mit
$$\exists V.\, P[V] \wedge V = t \Longrightarrow P[t/V]$$

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

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

Überblick: Approximative stärkste Nachbedingung

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

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!}
```

Fakultät: Stärkste Vorbedingung

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

$$
\begin{aligned}
asp_2 = \;& \exists V.\, 0 \leq n[V/p] \wedge p = (1[V/p]) \\
& \rightsquigarrow 0 \leq n \wedge p = 1 \\
asp_3 = \;& \exists V.\, (0 \leq n \wedge p = 1)[V/c] \wedge c = (1[V/c]) \\
& \rightsquigarrow 0 \leq n \wedge p = 1 \wedge c = 1 \\
asp_4 = \;& \neg(c \leq n) \wedge p = (c-1)! \wedge c - 1 \leq n \\
asp_5 = \;& \exists V_1.\, (p = (c-1)! \wedge (c-1) \leq n \wedge c \leq n)[V_1/p] \\
& \quad \wedge p = (p \cdot c)[V_1/p] \\
& \rightsquigarrow \exists V_1.\, (V_1 = (c-1)! \wedge (c-1) \leq n \wedge c \leq n) \wedge p = (V_1 \cdot c) \\
& \rightsquigarrow c - 1 \leq n \wedge c \leq n \wedge p = (c-1)! \cdot c \\
asp_6 = \;& \exists V_2.\, (c - 1 \leq n \wedge c \leq n \wedge p = (c-1)! \cdot c)[V_2/c] \\
& \quad \wedge c = (c+1)[V_2/c] \\
& \rightsquigarrow \exists V_2.\, (V_2 - 1 \leq n \wedge V_2 \leq n \wedge p = (V_2 - 1)! \cdot V_2) \wedge c = (V_2 + 1) \\
& \rightsquigarrow c - 2 \leq n \wedge c - 1 \leq n \wedge p = (c-2)! \cdot (c-1)
\end{aligned}
$$

Fakultät: Verifikationsbedingungen

Notation: vc_x = in Zeile x generierte Verifikationsbedingung

$$\begin{aligned} vc_4 = \quad & \{asp_3 \longrightarrow p = (c-1)! \wedge c - 1 \leq n, \\ & \quad asp_6 \longrightarrow p = (c-1)! \wedge c - 1 \leq n\} \\ vc_8 = \quad & asp_4 \longrightarrow p = n! \end{aligned}$$

Vereinfachung: $vc_4 \Longleftrightarrow vc_{4.1} \wedge vc_{4.2} \wedge vc_{4.3} \wedge vc_{4.4}$

$$\begin{aligned} vc_{4.1} \quad & = 0 \leq n \wedge p = 1 \wedge c = 1 \longrightarrow p = (c-1)! \\ & = 0 \leq n \longrightarrow 1 = (1-1)! \\ vc_{4.2} \quad & = 0 \leq n \wedge p = 1 \wedge c = 1 \longrightarrow c - 1 \leq n \\ & = 0 \leq n \longrightarrow 0 \leq n \\ vc_{4.3} \quad & = c - 2 \leq n \wedge c - 1 \leq n \wedge p = (c-2)! \cdot (c-1) \longrightarrow p = (c-1)! \\ & = c - 1 \leq n \wedge p = (c-2)! \cdot (c-1) \longrightarrow p = (c-1)! \\ vc_{4.4} \quad & = c - 2 \leq n \wedge c - 1 \leq n \wedge p = (c-2)! \cdot (c-1) \longrightarrow c - 1 \leq n \\ & = c - 1 \leq n \longrightarrow c - 1 \leq n \\ vc_8 \quad & = n \leq c - 1 \wedge c - 1 \leq n \wedge p = (c-1)! \longrightarrow p = n! \end{aligned}$$

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

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

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   {
6     if (a[r] < a[i]) {
7       r= i;
8       }
9     else {
10      }
11    i= i+1;
12    }
13  // {(∀j. 0 ≤ j < n ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n}
```

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

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. 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.