

Korrekte Software: Grundlagen und Methoden
Vorlesung 7 vom 18.05.17: Vorwärts und Rückwärts mit Floyd und
Hoare

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2017

Fahrplan

- ▶ Einführung
- ▶ Die Floyd-Hoare-Logik
- ▶ Operationale Semantik
- ▶ Denotationale Semantik
- ▶ Äquivalenz der Operationalen und Denotationalen Semantik
- ▶ Korrektheit des Hoare-Kalküls
- ▶ Vorwärts und Rückwärts mit Floyd und Hoare
- ▶ Funktionen und Prozeduren
- ▶ Referenzen und Speichermodelle
- ▶ Verifikationsbedingungen Revisited
- ▶ Vorwärtsrechnung Revisited
- ▶ Programmsicherheit und Frame Conditions
- ▶ Ausblick und Rückblick

Vorwärts oder Rückwärts?

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 sehen:

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 sehen:

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

- ▶ Muss das so sein? Ist das immer so?

Rückwärtsanwendung der Regeln

- ▶ Zuweisungsregel kann rückwärts angewandt werden, weil die Nachbedingung eine offene Variable ist — P passt auf jede beliebige Nachbedingung.

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

Rückwärtsanwendung der Regeln

- ▶ Zuweisungsregel kann **rückwärts** angewandt werden, weil die Nachbedingung eine offene Variable ist — P passt auf jede beliebige Nachbedingung.

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

- ▶ Was ist mit den anderen Regeln?

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

$$\frac{\vdash \{A\} c \{B\} \quad \vdash \{B\} \{c_s\} \{C\}}{\vdash \{A\} \{c c_s\} \{C\}} \quad \frac{\vdash \{A \wedge b\} c \{A\}}{\vdash \{A\} \mathbf{while}(b) c \{A \wedge \neg b\}}$$

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

Rückwärtsanwendung der Regeln

- ▶ Zuweisungsregel kann **rückwärts** angewandt werden, weil die Nachbedingung eine offene Variable ist — P passt auf jede beliebige Nachbedingung.

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

- ▶ Was ist mit den anderen Regeln? Nur **while** macht Probleme!

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

$$\frac{\vdash \{A\} c \{B\} \quad \vdash \{B\} \{c_s\} \{C\}}{\vdash \{A\} \{c c_s\} \{C\}}$$

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

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

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

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

- ▶ Ein einfaches Beispiel (nach Mike Gordon):

// {x = 1}

x = x + 1;

// { $\exists V. x = x + 1[V/x] \wedge (x = 1)[V/x]$ }

- ▶ Vereinfachung der Nachbedingung:

$$\exists V. x = (x + 1)[V/x] \wedge (x = 1)[V/x]$$

$$\iff \exists V. x = (V + 1) \wedge (V = 1)$$

$$\iff x = 1 + 1$$

$$\iff x = 2$$

Vorwärtsverkettung

- ▶ Vorwärtsaxiom äquivalent zum Rückwärtsaxiom.
- ▶ In der Anwendung **umständlicher**.
- ▶ Vereinfachung benötigt Lemma: $\exists x.P(x) \wedge x = t \iff P(t)$
- ▶ Vorteile?
 - ▶ Wir wollten doch sowieso die Anwendung automatisieren. . .
 - ▶ Wir stellen die Frage erstmal zurück

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

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

Schwächste Vorbedingungen

Berechnung von Vorbedingungen

- ▶ Die Rückwärtsrechnung von einer gegebenen Nachbedingung entspricht der Berechnung einer Vorbedingung
- ▶ Gegeben C0-Programm c , Prädikat P , dann ist
 - ▶ $wp(c, P)$ die **schwächste Vorbedingung** Q so dass $\models \{Q\} c \{P\}$;
- ▶ Prädikat P **schwächer** als Q wenn $Q \implies P$ (**stärker** wenn $P \implies Q$).

Berechnung von Vorbedingungen

- ▶ Die Rückwärtsrechnung von einer gegebenen Nachbedingung entspricht der Berechnung einer Vorbedingung
- ▶ Gegeben C0-Programm c , Prädikat P , dann ist
 - ▶ $\text{wp}(c, P)$ die **schwächste Vorbedingung** Q so dass $\models \{Q\} c \{P\}$;
- ▶ Prädikat P **schwächer** als Q wenn $Q \implies P$ (**stärker** wenn $P \implies Q$).
- ▶ Semantische Charakterisierung:

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

Berechnung von Vorbedingungen

- ▶ Die Rückwärtsrechnung von einer gegebenen Nachbedingung entspricht der Berechnung einer Vorbedingung
- ▶ Gegeben C0-Programm c , Prädikat P , dann ist
 - ▶ $\text{wp}(c, P)$ die **schwächste Vorbedingung** Q so dass $\models \{Q\} c \{P\}$;
 - ▶ $\text{sp}(P, c)$ die **stärkste Nachbedingung** Q so dass $\models \{P\} c \{Q\}$
- ▶ Prädikat P **schwächer** als Q wenn $Q \implies P$ (**stärker** wenn $P \implies Q$).
- ▶ Semantische Charakterisierung:

$$\begin{aligned}\models \{P\} c \{Q\} &\iff P \implies \text{wp}(c, Q) \\ \models \{P\} c \{Q\} &\iff \text{sp}(P, c) \implies Q\end{aligned}$$

Berechnung von $wp(c, Q)$

- ▶ Einfach für Programme ohne Schleifen:

$$wp(\{\}, P) \stackrel{def}{=} P$$

$$wp(x = e, P) \stackrel{def}{=} P[e/x]$$

$$wp(\{c \ c_s\}, P) \stackrel{def}{=} wp(c, wp(\{c_s\}, P))$$

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

- ▶ Für Schleifen: nicht entscheidbar.
 - ▶ “Cannot in general compute a **finite** formula” (Gordon)
- ▶ Wir können rekursive Formulierung angeben:

$$wp(\mathbf{while} (b) \ \{c\}, P) \stackrel{def}{=} (\neg b \wedge P) \vee (b \wedge wp(c, wp(\mathbf{while} (b) \ \{c\}, P)))$$

- ▶ Hilft auch nicht weiter...

Lösung: Annotierte Programme

- ▶ Wir helfen dem Rechner weiter und **annotieren** die Schleifeninvariante am Programm.
- ▶ Damit berechnen wir:
 - ▶ die **approximative** schwächste Vorbedingung $\text{awp}(c, Q)$ zusammen mit einer Menge von **Verifikationsbedingungen** $\text{wvc}(c, Q)$
- ▶ Die Verifikationsbedingungen treten dort auf, wo die Weakening-Regel angewandt werden muss.
- ▶ Es gilt:

$$\bigwedge \text{wvc}(c, Q) \implies \models \{\text{awp}(c, Q)\} c \{Q\}$$

Lösung: Annotierte Programme

- ▶ Wir helfen dem Rechner weiter und **annotieren** die Schleifeninvariante am Programm.
- ▶ Damit berechnen wir:
 - ▶ die **approximative** schwächste Vorbedingung $\text{awp}(c, Q)$ zusammen mit einer Menge von **Verifikationsbedingungen** $\text{wvc}(c, Q)$
 - ▶ oder die **approximative** stärkste Nachbedingung $\text{asp}(P, c)$ zusammen mit einer Menge von **Verifikationsbedingungen** $\text{svc}(P, c)$
- ▶ Die Verifikationsbedingungen treten dort auf, wo die Weakening-Regel angewandt werden muss.
- ▶ Es gilt:

$$\begin{aligned}\bigwedge \text{wvc}(c, Q) &\implies \models \{ \text{awp}(c, Q) \} c \{ Q \} \\ \bigwedge \text{svc}(P, c) &\implies \models \{ P \} c \{ \text{asp}(P, c) \}\end{aligned}$$

Approximative schwächste Vorbedingung

- Für die **while**-Schleife:

$$\text{awp}(\mathbf{while}(b) \ / ** \ \mathbf{inv} \ i \ */ \ c, P) \stackrel{\text{def}}{=} i$$

$$\begin{aligned} \text{wvc}(\mathbf{while}(b) \ / ** \ \mathbf{inv} \ i \ */ \ c, P) &\stackrel{\text{def}}{=} \text{wvc}(c, i) \\ &\cup \{i \wedge b \implies \text{awp}(c, i)\} \\ &\cup \{i \wedge \neg b \implies P\} \end{aligned}$$

- Entspricht der **while**-Regel (1) mit Weakening (2):

$$\frac{\vdash \{A \wedge b\} c \{A\}}{\vdash \{A\} \mathbf{while}(b) c \{A \wedge \neg b\}} \quad (1)$$

$$\frac{A \wedge b \implies C \quad \vdash \{C\} c \{A\} \quad A \wedge \neg b \implies B}{\vdash \{A\} \mathbf{while}(b) c \{B\}} \quad (2)$$

Überblick: Approximative schwächste Vorbedingung

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

$$\text{awp}(x = e, P) \stackrel{\text{def}}{=} P[e/x]$$

$$\text{awp}(\{c \ c_s\}, P) \stackrel{\text{def}}{=} \text{awp}(c, \text{awp}(\{c_s\}, P))$$

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

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

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

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

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

$$\text{wvc}(\{c \ c_s\}, P) \stackrel{\text{def}}{=} \text{wvc}(c, \text{awp}(\{c_s\}, P)) \cup \text{wvc}(\{c_s\}, P)$$

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

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

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

Alternative Schreibweise: AWP

$$\overline{P \leftarrow_{awp} \{ \}, P}$$

$$\overline{P[e/x] \leftarrow_{awp} x = e, P}$$

$$\frac{P_c \leftarrow_{awp} c, P_{c_s} \quad P_{c_s} \leftarrow_{awp} \{c_s\}, P}{P_c \leftarrow_{awp} \{c \ c_s\}, P}$$

$$\frac{P_{c_0} \leftarrow_{awp} c_0, P \quad P_{c_1} \leftarrow_{awp} c_1, P}{(b \wedge P_{c_0}) \vee (\neg b \wedge P_{c_1}) \leftarrow_{awp} \mathbf{if} (b) \ c_0 \ \mathbf{else} \ c_1, P}$$

$$\overline{i \leftarrow_{awp} \mathbf{while} (b) \ /** \ \mathbf{inv} \ i \ */ \ c, P}$$

Alternative Schreibweise: WVC

$$\overline{\{\}, P \rightarrow_{wvc} \emptyset}$$

$$\overline{x = e, P \rightarrow_{wvc} \emptyset}$$

$$\frac{P_{c_s} \leftarrow_{awp} c_s, P \quad c, P_{c_s} \rightarrow_{wvc} VC_c \quad \{c_s\}, P \rightarrow_{wvc} VC_{c_s}}{\{c \ c_s\}, P \rightarrow_{wvc} VC_c \cup VC_{c_s}}$$

$$\frac{c_0, P \rightarrow_{wvc} VC_{c_0} \quad c_1, P \rightarrow_{wvc} VC_{c_1}}{\text{if } (b) \ c_0 \ \text{else } c_1, P \rightarrow_{wvc} VC_{c_0} \cup VC_{c_1}}$$

$$\overline{/** \{q\} */ , P \rightarrow_{wvc} \{q \implies P\}}$$

$$\frac{c, i \rightarrow_{wvc} VC_c \quad P_c \leftarrow_{awp} c, i}{\text{while } (b) \ /** \ \text{inv } i \ */ \ c, P \rightarrow_{wvc} VC_c \cup \{i \wedge b \implies P_c, i \wedge \neg b \implies P\}}$$

Beispiel: das Fakultätsprogramm

- ▶ In der Praxis sind Vor- und Nachbedingung gegeben, und nur die Verifikationsbedingungen relevant.
- ▶ Sei F das annotierte Fakultätsprogramm:

```
int c, n, p;
```

```
/** { 0 <= n } */
```

```
p = 1;
```

```
c = 1;
```

```
while (c <= n) /** inv p == fac(c-1) && c-1 <= n; */ {  
    p = p * c;  
    c = c + 1;  
}
```

```
/** { p == fac(n) } */
```

- ▶ Berechnung der Verifikationsbedingungen zur Nachbedingung:
 $wvc(F, p == fac(N))$

Zusammenfassung

- ▶ Die Regeln des Floyd-Hoare-Kalküls sind **symmetrisch**: Die Zuweisungsregel gibt es “rückwärts” und “vorwärts”.
- ▶ Bis auf die Invarianten an Schleifen können wir Korrektheit automatisch prüfen.
- ▶ Wir **annotieren** daher die Invarianten an Schleifen, und können dann die schwächste Vorbedingung und Verifikationsbedingungen automatisch berechnen.
 - ▶ Davon sind die **Verifikationsbedingungen** das interessante.
- ▶ Die Generierung von Verifikationsbedingungen korrespondiert zur relativen Vollständigkeit der Floyd-Hoare-Logik.