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

Vorwärts oder Rückwärts?

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.

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

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

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

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

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

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;
// {∃V.x = x + 1[V/x] ∧ (x = 1)[V/x]}
```

- Vereinfachung der Nachbedingung:

$$\begin{aligned} & \exists V.x = (x+1)[V/x] \wedge (x = 1)[V/x] \\ \Longleftrightarrow\ & \exists V.x = (V+1) \wedge (V = 1) \\ \Longleftrightarrow\ & x = 1 + 1 \\ \Longleftrightarrow\ & x = 2 \end{aligned}$$

Vorwärtsverkettung

- Vorwärtsaxiom äquivalent zum Rückwärtsaxiom.
- In der Anwendung umständlicher.
- Vereinfachung benötigt Lemma: $\exists x.P(x) \wedge x = t \Longleftrightarrow P(t)$
- 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
  - $\mathrm{wp}(c, P)$ die schwächste Vorbedingung Q so dass $\models \{Q\}\, c\, \{P\}$;
  - $\mathrm{sp}(P, c)$ die stärkste Nachbedingung Q so dass $\models \{P\}\, c\, \{Q\}$
- Prädikat P schwächer als Q wenn $Q \Longrightarrow P$ (stärker wenn $P \Longrightarrow Q$).
- Semantische Charakterisierung:

$$\begin{array}{rcl} \models \{P\}\, c\, \{Q\} & \iff & P \Longrightarrow \mathrm{wp}(c, Q) \\ \models \{P\}\, c\, \{Q\} & \iff & \mathrm{sp}(P, c) \Longrightarrow Q \end{array}$$

Berechnung von $\mathrm{wp}(c, Q)$

- Einfach für Programme ohne Schleifen:

$$\begin{array}{rcl} \mathrm{wp}(\{\,\}, P) & \stackrel{def}{=} & P \\ \mathrm{wp}(x = e, P) & \stackrel{def}{=} & P[e/x] \\ \mathrm{wp}(\{c\ c_s\}, P) & \stackrel{def}{=} & \mathrm{wp}(c, \mathrm{wp}(\{c_s\}, P)) \\ \mathrm{wp}(\textbf{if}\ (b)\ c_0\ \textbf{else}\ c_1, P) & \stackrel{def}{=} & (b \wedge \mathrm{wp}(c_0, P)) \vee (\neg b \wedge \mathrm{wp}(c_1, P)) \end{array}$$

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

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

  - 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 $\mathrm{awp}(c, Q)$ zusammen mit einer Menge von Verifikationsbedingungen $\mathrm{wvc}(c, Q)$
  - oder die approximative stärkste Nachbedingung $\mathrm{asp}(P, c)$ zusammen mit einer Menge von Verifikationsbedingungen $\mathrm{svc}(P, c)$
- Die Verifikationsbedingungen treten dort auf, wo die Weakening-Regel angewandt werden muss.
- Es gilt:

$$\begin{array}{rcl} \bigwedge \mathrm{wvc}(c, Q) & \Longrightarrow & \models \{\mathrm{awp}(c, Q)\}\, c\, \{Q\} \\ \bigwedge \mathrm{svc}(P, c) & \Longrightarrow & \models \{P\}\, c\, \{\mathrm{asp}(P, c)\} \end{array}$$

Approximative schwächste Vorbedingung

- Für die **while**-Schleife:

$$\begin{array}{rcl} \mathrm{awp}(\textbf{while}\ (b)\ /{**}\ \textbf{inv}\ i\ {*}/\ c, P) & \stackrel{def}{=} & i \\ \mathrm{wvc}(\textbf{while}\ (b)\ /{**}\ \textbf{inv}\ i\ {*}/\ c, P) & \stackrel{def}{=} & \mathrm{wvc}(c, i) \\ & & \cup \{i \wedge b \Longrightarrow \mathrm{awp}(c, i)\} \\ & & \cup \{i \wedge \neg b \Longrightarrow P\} \end{array}$$

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

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

$$\frac{A \wedge b \Longrightarrow C \qquad \vdash \{C\}\, c\, \{A\} \qquad A \wedge \neg b \Longrightarrow B}{\vdash \{A\}\ \textbf{while}(b)\ c\, \{B\}} \quad (2)$$

Überblick: Approximative schwächste Vorbedingung

$$\begin{array}{rcl} \mathrm{awp}(\{\,\}, P) & \stackrel{def}{=} & P \\ \mathrm{awp}(x = e, P) & \stackrel{def}{=} & P[e/x] \\ \mathrm{awp}(\{c\ c_s\}, P) & \stackrel{def}{=} & \mathrm{awp}(c, \mathrm{awp}(\{c_s\}, P)) \\ \mathrm{awp}(\textbf{if}\ (b)\ c_0\ \textbf{else}\ c_1, P) & \stackrel{def}{=} & (b \wedge \mathrm{awp}(c_0, P)) \vee (\neg b \wedge \mathrm{awp}(c_1, P)) \\ \mathrm{awp}(/{**}\ \{q\}\ {*}/, P) & \stackrel{def}{=} & q \\ \mathrm{awp}(\textbf{while}\ (b)\ /{**}\ \textbf{inv}\ i\ {*}/\ c, P) & \stackrel{def}{=} & i \end{array}$$

$$\begin{array}{rcl} \mathrm{wvc}(\{\,\}, P) & \stackrel{def}{=} & \emptyset \\ \mathrm{wvc}(x = e, P) & \stackrel{def}{=} & \emptyset \\ \mathrm{wvc}(\{c\ c_s\}, P) & \stackrel{def}{=} & \mathrm{wvc}(c, \mathrm{awp}(\{c_s\}, P)) \cup \mathrm{wvc}(\{c_s\}, P) \\ \mathrm{wvc}(\textbf{if}\ (b)\ c_0\ \textbf{else}\ c_1, P) & \stackrel{def}{=} & \mathrm{wvc}(c_0, P) \cup \mathrm{wvc}(c_1, P) \\ \mathrm{wvc}(/{**}\ \{q\}\ {*}/, P) & \stackrel{def}{=} & \{q \Longrightarrow P\} \\ \mathrm{wvc}(\textbf{while}\ (b)\ /{**}\ \textbf{inv}\ i\ {*}/\ c, P) & \stackrel{def}{=} & \mathrm{wvc}(c, i) \cup \{i \wedge b \Longrightarrow \mathrm{awp}(c, i)\} \\ & & \cup \{i \wedge \neg b \Longrightarrow P\} \end{array}$$

Alternative Schreibweise: AWP

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

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

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

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

Alternative Schreibweise: WVC

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

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

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

$$\frac{}{/{**}\ \{q\}\ {*}/, P \rightarrow_{wvc} \{q \Longrightarrow P\}}$$

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

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: $\mathrm{wvc}(F, p == \mathit{fac}(N))$

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.