Bemerkung zum Beweis:

- Wenn man Aussagen über Folgen von Befehlen machen möchte, ist es oft geschickt, von rechts nach links zu argumentieren.
- Warum haben wir $x \geq 0$ in die Invariante aufgenommen? Weil wir $x = 0$ beim Verlassen der Schleife brauchten. Man ist oft gezwungen, Schleifeninvarianten zu verstärken, um Invarianz zu erhalten. Insofern verhalten sich Schleifeninvarianten wie Induktionsvoraussetzungen.

3.2 Schwächste Vorbedingungen und Ausdrucksmächtigkeit

Ziel: Zeige, dass der Hoare-Kalkül

vollständig (complete): $\models \{A\}\, c\, \{B\}$ impliziert $\vdash \{A\}\, c\, \{B\}$

und korrekt (sound): $\vdash \{A\}\, c\, \{B\}$ impliziert $\models \{A\}\, c\, \{B\}$

ist.

Problem: Um Vollständigkeit zu zeigen, führe einen Beweis per Induktion nach der Struktur von Programmen. Im Induktionsschritt benötigt man für den Fall

$$\vdash \{A\}\, c_1; c_2\, \{B\}$$

eine Zwischenbedingung C, so dass

$$\vdash \{A\}\, c_1\, \{C\} \quad \text{und} \quad \vdash \{C\}\, c_2\, \{B\}.$$

Existiert ein solches C, lässt sich die Induktionsvoraussetzung aufrufen und liefert:

$$\vdash \{A\}\, c_1\, \{C\} \quad \text{und} \quad \vdash \{C\}\, c_2\, \{B\}.$$

Dann erhält man

$$\vdash \{A\}\, c_1; c_2\, \{B\} \quad \text{mittels (SEQ)}.$$

Wie können wir die Existenz eines solchen C nachweisen?

Lösung:

Wir zeigen, dass für jedes Programm c und jede Nachbedingung B
diejenige Menge an Zuständen
↳ von denen aus die Ausführung divergiert oder
↳ von denen aus die Ausführung in einem B-Zustand terminiert
mittels einer Assertion ausdrückbar ist.

Definition:
Sei c ein Programm und B eine Nachbedingung.
Die Weakest-Liberal-Precondition von B bzgl. c,
auch schwächste Vorbedingung genannt,
ist die Menge an Zuständen:

$$wlp(c, B) := \{\sigma \in State \mid \forall \sigma' \in State.\ (c, \sigma) \Downarrow \sigma' \Rightarrow \sigma' \in B\}.$$

- Der Begriff liberal bezieht sich auf die Tatsache,
dass wir uns für partielle Korrektheit interessieren.
Bei totaler Korrektheit sprech man von Weakest-Preconditions.
Da wir heute totale Korrektheit betrachten,
erlauben wir uns die Terminologie schwächste Vorbedingung.
- Es ist nicht klar, dass die von uns definierten Assertions
für jedes c und jedes B ein Element enthalten,
dass wlp (c, B) charakterisiert.

Definition:

Eine Assertion-Sprache $\mathcal{L}$ ist <u>ausdrucksmächtig</u>,
falls es für alle $c \in \mathcal{C}$ und alle $B \in \mathcal{L}$
eine Assertion $A \in \mathcal{L}$ gibt mit:

$$wlp(c, B) = [\![A]\!] := \{\sigma \in State \mid \sigma \vdash A\}$$

Lemma:

Sei $[\![A]\!] = wlp(c, B)$.
Dann gilt:

1) $\models \{A\}\, c\, \{B\}$
2) $\models \{A'\}\, c\, \{B\}$ impliziert $A' \Rightarrow A$.

- M.d. Aussage 1) ist A tatsächlich eine Vorbedingung,
die das Hoare-Tripel gültig macht.
- M.d. Aussage 2) ist A die <u>schwächste</u> Vorbedingung,
die das Hoare-Tripel gültig macht.
Schwach in der partiellen Ordnung $(\mathcal{L}/_{\Leftrightarrow}, \Rightarrow)$,
Assertions faktorisiert nach logischer Äquivalenz,
geordnet mittels Implikation.

<u>Sch (Dijkstra '76, Edsger Dijkstra, 1930–2002, Turing-Award 1972):</u>
Die obige Assertion-Sprache ist ausdrucksmächtig.

- Der Satz sagt, dass die schwächste Vorbedingung
in der Assertion-Sprache formbar ist.
- Außerdem zeigt er, wie man $wlp(-,-)$ berechnet.
- Wir beschränken uns hier auf <u>azyklische Programme</u>.

Zur Behandlung von while-Schleifen,
siehe Winskels Buch.

Beweis:

Wir bestimmen die schwächste Assertion
mittels Funktion $pred(c, B)$:

$$pred(skip, B) := B$$

$$pred(x := a, B) := B[^x/_a]$$

$$pred(c_1; c_2, B) := pred(c_1, pred(c_2, B))$$

$$pred(\underline{if}\ b\ \underline{then}\ c_1\ \underline{else}\ c_2\ \underline{fi}, B) := (b \wedge pred(c_1, B)) \vee (\neg b \wedge pred(c_2, B))$$

$$pred(\underline{assume}\ b, B) := b \to B.$$

Man kann zeigen, dass

$$\forall \sigma \in State: \quad \sigma \models pred(c, B) \iff \sigma \in wlp(c, B),$$

also $S[\![pred(c, B)]\!] = wlp(c, B)$.

Daher rechnen wir in der Regel $wlp(c, B)$ auch dann,
wenn wir das Prädikat $pred(c, B)$ meinen. □

3.3 Vollständigkeit und Korrektheit

Ziel: • Mit dem Begriff der Berechenbarkeit zur Hand
können wir Vollständigkeit der Hoare-Kalküls zeigen
• Korrektheit ist einfacher.

Die folgende Proposition ist dabei elementar.

Proposition:

$$\vdash \{pred(c, B)\}\ c\ \{B\}.$$

Beweis (Skizze):

Wir führen eine Induktion nach der Struktur von c und zeigen:

$\forall B \in \mathcal{L}. \quad \vdash \{pred(c, B)\}\, c\, \{B\}.$

IA: Für skip und $x := a$ folgt die Behauptung mittels (SKIP) und (ASSIGN).

IS: Fallunterscheidung über die zusammengesetzten Anweisungen, dann IV + Regeln des Hoare-Kalküls. □

Satz (Vollständigkeit, Cook '74, Stephen Cook, *1939, Turing-Award 1982):

$\models \{A\}\, c\, \{B\}$ impliziert $\vdash \{A\}\, c\, \{B\}$.

Beweis:

Mit obiger Proposition gilt $\vdash \{pred(c, B)\}\, c\, \{B\}$.

Der Satz von Dijkstra sagt $sll[pred(c, B)]\!] = wlp(c, B)$.

Mit obigem Lemma folgt $A \Rightarrow pred(c, B)$.

Regel (CONSEQUENCE) liefert $\vdash \{A\}\, c\, \{B\}$. □

- Korrektheit besagt, dass jedes ableitbare Theorem $\vdash \{A\}\, c\, \{B\}$ des Hoare-Kalküls ein gültiges Hoare-Tripel ist.
- Der Beweis ist eine standard Induktion nach der Höhe des Beweisbaums.

Satz (Korrektheit):

$\vdash \{A\}\, c\, \{B\}$ impliziert $\models \{A\}\, c\, \{B\}$.

3.4 Stärkste Nachbedingungen

Statt mit schwächsten Vorbedingungen kann man auch mit stärksten Nachbedingungen arbeiten.

Definition:
Sei c ein Programm und A eine Aussage.
Die stärkste Nachbedingung oder Strongest-Postcondition ist die Zustandsmenge

$$sp(A, c) := \{\sigma' \in State \mid \exists \sigma \in State.\ \sigma \models A \wedge (c, \sigma) \Downarrow \sigma'\}.$$

Lemma:
Sei $S[\![B]\!] = sp(A, c)$.
Dann gilt:

1) $\models \{A\}\, c\, \{B\}$.
2) $\models \{A\}\, c\, \{B'\}$ impliziert $B \Rightarrow B'$.

D.h. 1) ist B eine gültige Nachbedingung,
mit 2) die stärkste.

Bemerkung:
Auf azyklischen Programmen lässt sich die stärkste Nachbedingung $sp(A, c)$ als Pendant $post(A, c)$ charakterisieren:

$$\begin{aligned}
post(A, skip) &:= A \\
post(A, x := a) &:= \exists x'.\ A[x'/x] \wedge x = a[x'/x'],\ x' \text{ frisch} \\
post(A, c_1; c_2) &:= post(post(A, c_1), c_2) \\
post(A, \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ fi}) &:= post(A \wedge b, c_1) \vee post(A \wedge \neg b, c_2) \\
post(A, \text{assume } b) &:= A \wedge b.
\end{aligned}$$

Satz (Dijkstra '76):

$post(H, c) \Rightarrow B$ gdw. $H \Rightarrow pred(c, B)$.