Kapitel 8

Programmverifikation

Eine *Spezifikation* sagt, *was* gemacht werden soll. Eine *Implementierung* sagt, *wie* etwas gemacht werden soll. Unter einer *Verifikation* versteht man eine Argumentation, die zeigt, dass eine Implementierung macht was eine Spezifikation vorschreibt.

Diese grundlegende Idee, die auf Hardware und Software angewendet werden kann, machen wir jetzt am Beispiel der Programmiersprache IMP konkret. Spezifikationen formulieren wir mit den Formeln von ASSN, als Implementierungen verwenden wir die Kommandos von IMP. Auf dieser Grundlage entwickeln wir Verifikationstechniken, mit denen wir zeigen können, dass ein Kommando eine Spezifikation implementiert.

Da wesentliche Ideen dieser Konstruktion von C.A.R. Hoare entwickelt wurden (um 1970), bezeichnet man sie auch als Hoare-Logik.

Lesematerial

[Winskel, Kapitel 6 und 7]

8.1 Spezifikationen und Korrektheit

Für IMP verwenden wir Spezifikationen, die aus einer *Vorbedingung* und einer *Nachbedingung* bestehen. Hier ist ein Beispiel:

Vorbedingung: $X \geq 1 \wedge Y \geq 0$
Nachbedingung: $Z = X + Y$

Sei das Kommando c ein Kandidat für die Implementierung dieser Spezifikation. Die Vorbedingung legt fest, auf welche Zustände c angewendet werden soll. Die

Nachbedingung legt fest, welche Eigenschaften die Endzustände haben sollen, die die Ausführung von c mit einem Anfangszustand gemäß der Vorbedingung liefert. Wenn wir annehmen, dass c die Lokationen X und Y nicht ändert, legt die obige Spezifikation gerade fest, dass c die Summe $X+Y$ berechnen und in der Lokation Z ablegen soll.

Es ist zweckmäßig, zwischen Termination und partieller Korrektheit zu trennen. Partielle Korrektheit läßt zu, dass ein Kommando für zulässige Anfangszustände divergiert. Nur für den Fall, dass der Anfangszustand die Vorbedingung erfüllt und das Kommando für diesen Zustand terminiert, muss der gelieferte Endzustand die Nachbedingung erfüllen. Unter totaler Korrektheit versteht man partielle Korrektheit plus Termination für alle Anfangszustände, die die Vorbedingung erfüllen. Hoare-Logik behandelt partielle Korrektheit und ignoriert Termination.

Im Kontext von Verifikation bezeichnet man die Formeln von ASSN auch als *Zusicherungen* oder *Assertionen*. Für Zusicherungen werden wir die folgenden Metavariablen verwenden:

$$A, B, C, I \in Assn$$

Aus technischen Gründen ist es sinnvoll, darauf zu bestehen, dass die folgenden Beziehungen gelten:

1. $Var = Loc$.
2. $Aexp \subseteq Ter$ und $\forall a \in Ter\colon\ \mathcal{A}[\![a]\!] = \mathcal{T}[\![a]\!]$.
3. $Bexp \subseteq Assn$ und $\forall b \in Bexp\colon\ \mathcal{B}[\![b]\!] = \mathcal{D}[\![b]\!]$.

Wir haben IMP und ASSN mit Bedacht so definiert, dass die Erfüllung der Bedingung (1) die Erfüllung der Bedingungen (2) und (3) nach sich zieht.

In der Praxis ist es sinnvoll, für Zusicherungen weitere arithmetische Operationen (zum Beispiel Exponentiation n^k) zur Verfügung zu stellen. Es ist einfach, ASSN entsprechend zu erweitern. Alle Ergebnisse dieses Kapitels gelten auch für entsprechende Erweiterungen von *ASSN*.

Eine *Spezifikation* ist ein Paar $\langle A, B\rangle$ aus zwei Zusicherungen $A, B \in Assn$.

Definition 8.1.1 *Ein Kommando $c \in Com$* erfüllt *eine Spezifikation $\langle A, B\rangle$ genau dann, wenn:*

$$\forall \sigma, \sigma' \in \Sigma\colon\ (\sigma \models A \ \wedge\ \mathcal{C}[\![c]\!]\sigma = \sigma') \ \Rightarrow\ \sigma' \models B$$

Die Tripel aus $Assn \times Com \times Assn$ nennen wir *Hoare-Tripel* und schreiben sie als $\{A\}c\{B\}$. Wir definieren:

$$\models \{A\}c\{B\} \overset{\text{def}}{\iff} \{A\}c\{B\} \textit{ ist gültig } \overset{\text{def}}{\iff} c \textit{ erfüllt } \langle A, B\rangle$$

Um die Gültigkeit von Hoare-Tripeln kompakter charakterisieren zu können, definieren wir eine Relation auf $\Sigma_\perp \times Assn$ wie folgt:

$$\begin{aligned}&\forall s \in \Sigma_\perp\ \forall A \in Assn:\\ &\qquad s \models A \overset{\text{def}}{\iff} (s \neq \perp \Rightarrow \mathcal{D}[\![A]\!]s = 1)\end{aligned}$$

Diese Relation stimmt auf $\Sigma \times Assn$ mit der für *ASSN* eingeführten Relation „$\sigma \models A$“ überein. Jetzt können wir die Gültigkeit von Hoare-Tripeln wie folgt charakterisieren:

$$\begin{aligned}&\forall A \in Assn\ \forall c \in Com\ \forall B \in Assn:\\ &\qquad \models \{A\}c\{B\} \iff (\forall \sigma \in \Sigma:\ \sigma \models A \Rightarrow \mathcal{C}[\![c]\!]\sigma \models B)\end{aligned}$$

Partielle und totale Korrektheit

Ein Kommando c heißt *partiell korrekt* für eine Spezifikation $\langle A, B\rangle$ genau dann, wenn $\models \{A\}c\{B\}$. Ein Kommando c heißt *total korrekt* für eine Spezifikation $\langle A, B\rangle$ genau dann, wenn $\models \{A\}c\{B\}$ und

$$\forall \sigma \in \Sigma:\ \sigma \models A \ \Rightarrow\ \mathcal{C}[\![c]\!]\sigma \neq \perp$$

Expressivität von Spezifikationen

Sei c ein Kommando mit

$$\models \{\texttt{true}\}c\{\texttt{false}\}$$

Da die Nachbedingung `false` von keinem Zustand erfüllt wird, bedeutet dies, dass c für alle Zustände divergiert. Wir haben also

$$\forall c \in Com:\ (\forall \sigma \in \Sigma:\ \mathcal{C}[\![c]\!]\sigma = \perp) \iff \models \{\texttt{true}\}c\{\texttt{false}\}$$

Nehmen Sie jetzt für einen Augenblick an, dass die Menge *Loc* endlich ist. Dann können wir für jeden Zustand σ eine Zusicherung A_σ angeben, so dass σ der einzige Zustand ist, der A erfüllt. Sei beispielsweise $Loc = \{X, Y\}$ und

$$\sigma = \{X \mapsto 1,\ Y \mapsto 2\}$$

Dann können wir

$$A_\sigma = (X = 1 \wedge Y = 2)$$

wählen. Damit gilt für $c \in Com$ und $\sigma, \sigma' \in \Sigma$:

$$\mathcal{C}[\![c]\!]\sigma = \bot \iff \models \{A_\sigma\}c\{\texttt{false}\}$$
$$\mathcal{C}[\![c]\!]\sigma = \sigma' \iff \models \{A_\sigma\}c\{A_{\sigma'}\} \;\wedge\; \neg \models \{A_\sigma\}c\{\texttt{false}\}$$

Da Kommandos immer nur endlich viele Lokationen enthalten, bekommen wir diese Äquivalenzen auch bei unendlichem *Loc*. Allerdings müssen wir A_σ dann relativ zu der endlichen Menge von Lokationen konstruieren, die in c vorkommen.

8.2 Hoare-Regeln

Wir definieren jetzt mithilfe von Inferenzregeln, die als Hoare-Regeln bezeichnet werden, eine Menge von Hoare-Tripeln. Wir werden zeigen, dass diese Menge genau die Menge der gültigen Hoare-Tripel ist. An dieser Definition ist interessant, dass sie unabhängig von den bisherigen Semantiken für IMP erfogt. Da man mit der Menge der gültigen Hoare-Tripeln die Semantik von IMP beschreiben kann (siehe oben), kann man die Hoare-Regeln als eine weitere Semantik für IMP ansehen. Man spricht von einer *axiomatischen Semantik*.

Wir definieren also eine Menge *AS*

$$\vdash \{A\}c\{B\} \quad \stackrel{\text{def}}{\iff} \quad \langle A, c, B\rangle \in AS \subseteq Assn \times Com \times Assn$$

rekursiv durch die folgenden Inferenzregeln (Hoare-Regeln):

$$\frac{}{\vdash \{A\}\texttt{skip}\{A\}} \qquad \frac{}{\vdash \{B[a/X]\}X := a\{B\}}$$

$$\frac{\vdash \{A\}c_1\{C\} \qquad \vdash \{C\}c_2\{B\}}{\vdash \{A\}c_1; c_2\{B\}} \qquad \frac{\vdash \{A \wedge b\}c_1\{B\} \qquad \vdash \{A \wedge \neg b\}c_2\{B\}}{\vdash \{A\}\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\{B\}}$$

$$\frac{\vdash \{I \wedge b\}c\{I\}}{\vdash \{I\}\texttt{while}\ b\ \texttt{do}\ c\{I \wedge \neg b\}} \qquad \frac{A \models A' \qquad \vdash \{A'\}c\{B'\} \qquad B' \models B}{\vdash \{A\}c\{B\}}$$

Da wir *AS* als Teilmenge der Grundmenge $Assn \times Com \times Assn$ definieren, verzichten wir auf Prämissen der Bauart $A \in Assn$.

Die Zusicherung I in der Regel für Schleifen nennt man *Invariante*. Die zuletzt angegebene Hoare-Regel heißt *Abschwächungsregel*.

Satz 8.2.1 (Korrektheit der Hoare-Regeln)

$$\forall c \in Com\ \forall A, B \in Assn\colon\ \vdash \{A\}c\{B\} \Rightarrow \models \{A\}c\{B\}$$

Beweis Wir zeigen

$$\forall c \in Com\ \forall A, B \in Assn\colon$$
$$\vdash \{A\}c\{B\} \Rightarrow (\forall \sigma, \sigma' \in \Sigma\colon\ \sigma \models A \land \mathcal{C}[\![c]\!]\sigma = \sigma' \Rightarrow \sigma' \models B)$$

durch Regelinduktion über $\vdash$.

1. *Regel für Sequentialisierung*. Sei $\models \{A\}c_1\{C\}$ und $\models \{C\}c_2\{B\}$ (Induktionsannahmen). Weiter sei $\mathcal{C}[\![c_1; c_2]\!]\sigma = \sigma'$ und $\sigma \models A$. Wir müssen $\sigma' \models B$ zeigen. Mit der Definition von $\mathcal{C}$ haben wir ein σ'' mit $\mathcal{C}[\![c_1]\!]\sigma = \sigma''$ und $\mathcal{C}[\![c_2]\!]\sigma'' = \sigma'$. Mit der ersten Induktionsannahme folgt $\sigma'' \models C$. Mit der zweiten Induktionsannahme folgt $\sigma' \models B$.

2. *Regel für Schleifen*. Sei $\models \{I \land b\}c\{I\}$ (Induktionsannahme). Weiter sei $\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma = \sigma'$ und $\sigma \models I$. Wir müssen $\sigma' \models I \land \neg b$ zeigen. Nach Definition von $\mathcal{C}$ genügt es zu zeigen:

   $$\forall n \in \mathbb{N}\ \forall \sigma \in \Sigma\colon$$
   $$\sigma' = (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))^n(\lambda\, \sigma\,.\, \bot))\ \sigma \ \land\ \sigma \models I \Rightarrow \sigma' \models I \land \neg b$$

   Dies Behauptung zeigt man leicht durch Induktion über $n \in \mathbb{N}$. Übung!

3. Die Beweisteile für die restlichen Regeln sind leicht. Übung! □

8.3 Verifikationsbedingungen

Wir entwickeln jetzt eine Methode, mit der man Kommandos verifizieren kann. Wir geben einen Algorithmus an, der zu einem Kommando c und einer Spezifikation $\langle A, B\rangle$ eine endliche Menge V von Zusicherungen berechnet, so dass

$$\models V \Rightarrow \vdash \{A\}c\{B\}$$

gilt. Die Zusicherungen in V nennt man *Verifikationsbedingungen*. Den Algorithmus nennt man *Verifikationsbedingungsgenerator*. Der Algorithmus funktioniert allerdings nur, wenn man jede Schleife von Hand mit einer Invariante annotiert. Annotierte Schleifen schreiben wir als

$$\{I\}\texttt{while}\ b\ \texttt{do}\ c$$

$$
\begin{array}{lcl}
\multicolumn{3}{l}{VC' \in Com \times Assn \to Assn \times \mathcal{P}_{fin}(Assn)} \\
VC'(\texttt{skip},\ B) & = & \langle B, \emptyset \rangle \\
VC'(X := a,\ B) & = & \langle B[a/X], \emptyset \rangle \\
VC'(c_1; c_2,\ B) & = & \text{let } \langle A_2, V_2 \rangle = VC'(c_2, B) \\
 & & \quad\ \langle A_1, V_1 \rangle = VC'(c_1, A_2) \\
 & & \text{in } \langle A_1, V_1 \cup V_2 \rangle \\
VC'(\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2,\ B) & = & \text{let } \langle A_1, V_1 \rangle = VC'(c_1, B) \\
 & & \quad\ \langle A_2, V_2 \rangle = VC'(c_2, B) \\
 & & \text{in } \langle b \wedge A_1 \vee \neg b \wedge A_2,\ V_1 \cup V_2 \rangle \\
VC'(\{I\}\texttt{while } b \texttt{ do } c,\ B) & = & \text{let } \langle A, V \rangle = VC'(c, I) \\
 & & \text{in } \langle I,\ V \cup \{I \wedge b \Rightarrow A,\ I \wedge \neg b \Rightarrow B\} \rangle
\end{array}
$$

$$
\begin{array}{l}
VC \in Assn \times Com \times Assn \to \mathcal{P}_{fin}(Assn) \\
VC(A, c, B) \ = \ \text{let } \langle A', V \rangle = VC'(c, B) \text{ in } V \cup \{A \Rightarrow A'\}
\end{array}
$$

Abbildung 8.1: Verifikationsbedingungsgenerator für IMP

Zu einem Kommando c mit n Schleifen berechnet der Algorithmus $2n + 1$ Verifikationsbedingungen. Für jede Schleife bekommt man genau zwei Verifikationsbedingungen.

Sei *ComA* die Menge der annotierten Kommandos. Um unsere Notation einfach zu halten, benutzen wir die Metavariable c sowohl für einfache wie für annotierte Kommandos. Zudem lassen wir annotierte Kommandos auch überall da zu, wo einfache Kommandos verlangt werden. Dabei werden die annotierten Invarianten automatisch gelöscht. Dieser notationale Trick spart uns viel Schreibarbeit.

Abbildung 8.1 zeigt den Verifikationsbedingungsgenerator für IMP und *Assn*.

Satz 8.3.1 (Korrektheit von VC') *Sei* $VC'(c, B) = \langle A, V \rangle$ *und* $\models V$. *Dann* $\vdash \{A\}c\{B\}$.

Beweis Durch strukturelle Induktion über $c \in Com$. Sei $VC'(c, B) = \langle A, V \rangle$ und $\models V$. Wir geben nur die Beweisteile für Konditionale und Schleifen an.

1. Sei $c = \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2$. Sei $VC'(c_1, B) = \langle A_1, V_1 \rangle$ und $VC'(c_2, B) = \langle A_2, V_2 \rangle$. Dann $A = (b \wedge A_1 \vee \neg b \wedge A_2)$ und $V = V_1 \cup V_2$

(Definition von VC'). Also:

$\vdash \{A_1\}c_1\{B\}$	Induktionsannahme
$\vdash \{A_2\}c_2\{B\}$	Induktionsannahme
$\vdash \{A \wedge b\}c_1\{B\}$	Abschwächungsregel
$\vdash \{A \wedge \neg b\}c_2\{B\}$	Abschwächungsregel
$\vdash \{A\}c\{B\}$	Regel für Konditionale

2. Sei $c = \{I\}\texttt{while}\ b\ \texttt{do}\ c'$. Sei $VC'(c', I) = \langle A', V'\rangle$. Dann $A = I$ und $V = V' \cup \{I \wedge b \Rightarrow A',\ I \wedge \neg b \Rightarrow B\}$ (Definition von VC'). Also:

$\vdash \{A'\}c'\{I\}$	Induktionsannahme
$\vdash \{I \wedge b\}c'\{I\}$	Abschwächungsregel
$\vdash \{I\}c\{I \wedge \neg b\}$	Schleifenregel
$\vdash \{A\}c\{B\}$	Abschwächungsregel

□

Korollar 8.3.2 (Korrektheit von VC) *Sei* $\models VC(A, c, B)$. *Dann* $\vdash \{A\}c\{B\}$.

Beweis Satz 8.3.1. □

8.4 VC-Methode

Wir können jetzt die Gültigkeit eines Hoare-Tripels wie folgt zeigen:

1. Annotiere alle Schleifen des Kommandos mit Invarianten.
2. Zeige, dass alle Verifikationsbedingungen gültig sind.

Wir bezeichnen dieses Vorgehen als *VC-Methode*. Als Beispiel verifizieren wir das Hoare-Tripel:

$$\begin{array}{ll}
A = & \{Y \geq 0\} \\
 & N := Y; \\
 & Z := 1; \\
 & \texttt{while}\ N \geq 1\ \texttt{do}\ (N; = N - 1;\ \ Z := Z * X) \\
B = & \{Z = X^Y\}
\end{array}$$

Für dieses Beispiel haben wir die Terme von ASSN um Exponentiation erweitert. Als Invariante wählen wir:

$$I = (X^Y = Z * X^N \wedge N \geq 0)$$

Jetzt zeigen wir die Gültigkeit der drei Verifikationsbedingungen:

1. $I \wedge \neg(N \geq 1) \models B$. Gilt, da:

$$I \wedge \neg(N \geq 1) \;\;\Dashv\models\;\; X^Y = Z * X^N \wedge N = 0 \;\;\models\;\; B$$

2. $I \wedge N \geq 1 \models I[Z * X/Z][N - 1/N]$. Gilt, da:

$$\begin{aligned} I[Z * X/Z][N-1/N] &\;\Dashv\models\; (X^Y = (Z * X) * X^N \wedge N \geq 0)[N - 1/N] \\ &\;\Dashv\models\; X^Y = (Z * X) * X^{N-1} \wedge N - 1 \geq 0 \\ &\;\Dashv\models\; X^Y = Z * X^N \wedge N \geq 1 \end{aligned}$$

3. $A \models I[1/Z][Y/N]$. Gilt, da:

$$\begin{aligned} I[1/Z][Y/N] &\;\Dashv\models\; (X^Y = 1 * X^N \wedge N \geq 0)[Y/N] \\ &\;\Dashv\models\; X^Y = 1 * X^Y \wedge Y \geq 0 \\ &\;\Dashv\models\; Y \geq 0 \end{aligned}$$

8.5 Vollständigkeit der VC-Methode

Satz 8.5.1 (Monotonie von VC') *Sei $VC'(c, B_1) = \langle A_1, V_1\rangle$, $VC'(c, B_2) = \langle A_2, V_2\rangle$, und $B_1 \models B_2$. Dann $A_1 \models A_2$ und $V_1 \models V_2$.*

Beweis Durch strukturelle Induktion über $c \in Com$. Sei $VC'(c, B_1) = \langle A_1, V_1\rangle$, $VC'(c, B_2) = \langle A_2, V_2\rangle$, und $B_1 \models B_2$. Wir geben nur die Beweisteile für Konditionale und Schleifen an.

1. Sei $c = \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2$. Sei

$$\begin{aligned} (A_{11}, V_{11}) &= VC'(c_1, B_1) \\ (A_{12}, V_{12}) &= VC'(c_1, B_2) \\ (A_{21}, V_{21}) &= VC'(c_2, B_1) \\ (A_{22}, V_{22}) &= VC'(c_2, B_2) \end{aligned}$$

Nach Induktionsannahme gilt $A_{11} \models A_{12}$ und $V_{11} \models V_{12}$ sowie $A_{21} \models A_{22}$ und $V_{21} \models V_{22}$. Außerdem liefert die Definition von VC':

$$\begin{aligned} (A_1, V_1) &= (b \wedge A_{11} \vee \neg b \wedge A_{21},\;\; V_{11} \cup V_{21}) \\ (A_2, V_2) &= (b \wedge A_{12} \vee \neg b \wedge A_{22},\;\; V_{12} \cup V_{22}) \end{aligned}$$

Also $A_1 \models A_2$ und $V_1 \models V_2$.

2. Sei $c = \{I\}\texttt{while}\ b\ \texttt{do}\ c'$. Sei $VC'(c', I) = \langle A, V\rangle$. Die Definition von VC' liefert:

$$(A_1, V_1) = (I,\ V \cup \{I \wedge b \Rightarrow A,\ I \wedge \neg b \Rightarrow B_1\})$$
$$(A_2, V_2) = (I,\ V \cup \{I \wedge b \Rightarrow A,\ I \wedge \neg b \Rightarrow B_2\})$$

Also $A_1 \models A_2$ und $V_1 \models V_2$.

□

Satz 8.5.2 (Vollständigkeit von VC') *Sei* $\vdash \{A\}c\{B\}$. *Dann kann man c so mit Invarianten notieren, dass gilt:*

$$\forall A'\ \forall V:\ VC'(c, B) = \langle A', V\rangle\ \Rightarrow\ (A \models A' \wedge \models V)$$

Beweis Durch Regelinduktion über die Hoare-Regeln. Wir geben nur die Beweisteile für die Regeln für Sequentialisierung, Schleifen und Abschwächung an. Dabei benötigen wir die gerade gezeigte Monotonie-Eigenschaft von VC'.

1. Regel für $c_1; c_2$. Sei $\vdash \{A\}c_1\{C\}$ und $\vdash \{C\}c_2\{B\}$. Wir annotieren c_1 und c_2 gemäß der Induktionsannahme. Sei $VC'(c_1, C) = \langle A_1, V_1\rangle$ und $VC'(c_2, B) = \langle A_2, V_2\rangle$. Nach Induktionsannahme gilt $\models V_1 \cup V_2$ sowie $A \models A_1$ und $C \models A_2$. Sei $VC'(c_1, A_2) = \langle A_1', V_1'\rangle$. Dann

$$VC'(c_1; c_2, B) = \langle A_1', V_1' \cup V_2\rangle$$

Nach Satz 8.5.1 gilt $A_1 \models A_1'$ und $V_1 \models V_1'$. Damit folgt $A \models A_1'$ und $\models V_1' \cup V_2$.

2. Regel für $\texttt{while}\ b\ \texttt{do}\ c$. Sei $\vdash \{I \wedge b\}c\{I\}$. Wir annotieren c gemäß der Induktionsannahme. Sei $VC'(c, I) = \langle A, V\rangle$. Dann

$$VC'(\{I\}\texttt{while}\ b\ \texttt{do}\ c,\ I \wedge \neg b) = (I, V \cup \{I \wedge b \Rightarrow A,\ I \wedge \neg b \Rightarrow I \wedge \neg b\})$$

Damit gilt die Behauptung, da die Induktionsannahme für $VC'(c, I) = \langle A, V\rangle$ liefert, dass $I \wedge b \models A$ und $\models V$.

3. Abschwächungsregel. Sei $A \models A'$, $\vdash \{A'\}c\{B'\}$ und $B' \models B$. Wir annotieren c gemäß der Induktionsannahme. Sei $VC'(c, B') = (A'', V)$ und $VC'(c, B) = (A''', V')$. Nach Satz 8.5.1 gilt $A'' \models A'''$ und $V \models V'$. Damit gilt die Behauptung, da die Induktionsannahme für $VC'(c, B') = (A'', V)$ liefert, dass $A' \models A''$ und $\models V$.

□

Korollar 8.5.3 (Vollständigkeit von VC) *Sei* $\vdash \{A\}c\{B\}$. *Dann kann man c so mit Invarianten annotieren, dass* $\models VC(A, c, B)$.

Wir wissen jetzt, dass wir der VC-Methode genau die Hoare-Tripel verifizieren können, die wir mit den Hoare-Regeln ableiten können.

8.6 Schwächste Vorbedingungen

Ein $W \in Assn$ heißt *schwächste Vorbedingung (SVB)* für $c \in Com$ und $B \in Assn$ genau dann, wenn:

$$\forall \sigma \in \Sigma\colon\ (\sigma \models W \iff \mathcal{C}[\![c]\!]\sigma \models B)$$

Eine *SVB-Funktion* ist eine Funktion

$$w \in Com \to Assn \to Assn$$

für die gilt:

$$\forall c \in Com\ \forall B \in Assn\ \forall \sigma \in \Sigma\colon\ (\sigma \models wcB \iff \mathcal{C}[\![c]\!]\sigma \models B)$$

Proposition 8.6.1 *Sei w eine SVB-Funktion. Dann gilt für alle $c, c' \in Com$ und $A, B, B' \in Assn$:*

1. $\models \{A\}c\{B\} \iff A \models wcB$.
2. $(\mathcal{C}[\![c]\!] = \mathcal{C}[\![c']\!] \wedge \mathcal{D}[\![B]\!] = \mathcal{D}[\![B']\!]) \Rightarrow \mathcal{D}[\![wcB]\!] = \mathcal{D}[\![wc'B']\!]$.

Die Behauptungen der Proposition folgen direkt aus den Definitionen.

Wir werden zeigen, dass eine SVB-Funktion existiert. Der Beweis dieser Tatsache ist nicht einfach. Ein entsprechendes Resultat wurde erstmals 1978 von Stephen A. Cook publiziert. Cook verwendete dabei Techniken, die der Logiker Kurt Gödel um 1930 entwickelt hat.

Proposition 8.6.2 *Seien w und w' zwei SVB-Funktionen. Dann:*

$$\forall c \in Com\ \forall B \in Assn\colon\ wcB \models\!\!\!\!\!\dashv w'cB$$

Proposition 8.6.3 *Sei w eine SVB-Funktion. Dann gilt für alle $c, c_1, c_2 \in Com$, $X \in Loc$, $a \in Aexp$ und $B \in Assn$:*

1. $w(\mathtt{skip})B \models\!\!\!\!\!\dashv B$.
2. $w(X := a)B \models\!\!\!\!\!\dashv B[a/X]$.
3. $w(c_1; c_2)B \models\!\!\!\!\!\dashv w(c_1)(wc_2B)$.
4. $w(\mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2)B \models\!\!\!\!\!\dashv b \wedge wc_1B \vee \neg b \wedge wc_2B$.
5. $w(\mathtt{while}\ b\ \mathtt{do}\ c)B \models\!\!\!\!\!\dashv b \wedge wc(w(\mathtt{while}\ b\ \mathtt{do}\ c)B) \vee \neg b \wedge B$.

Beweis Jede Behauptung folgt durch einfaches Rechnen. Wir zeigen hier nur die Teilbehauptungen (2), (3) und (5). Sei $\sigma \in \Sigma$.

Behauptung (2)

$$\begin{array}{llll} \sigma \models w(X := a)B & \iff & \mathcal{C}[\![X := a]\!]\sigma \models B & \text{Definition von } w \\ & \iff & \sigma[\mathcal{A}[\![a]\!]\sigma/X] \models B & \text{Definition von } \mathcal{C} \\ & \iff & \sigma \models B[a/X] & \text{Substitutions-Lemma} \end{array}$$

Behauptung (3)

$$\begin{array}{lll} & \sigma \models w(c_1; c_2)B & \\ \iff & \mathcal{C}[\![c_1; c_2]\!]\sigma \models B & \text{Def. von } w \\ \iff & \mathcal{C}[\![c_1]\!]\sigma \neq \bot \Rightarrow \mathcal{C}[\![c_2]\!](\mathcal{C}[\![c_1]\!]\sigma) \models B & \text{Def. von } \mathcal{C} \\ \iff & \mathcal{C}[\![c_1]\!]\sigma \neq \bot \Rightarrow \mathcal{C}[\![c_1]\!]\sigma \models wc_2B & \text{Def. von } w \\ \iff & \mathcal{C}[\![c_1]\!]\sigma \models wc_2B & \\ \iff & \sigma \models wc_1(wc_2B) & \text{Def. von } w \end{array}$$

Behauptung (5)

$$\begin{array}{lll} & \sigma \models w(\texttt{while}\ b\ \texttt{do}\ c)B & \\ \iff & \sigma \models w(\texttt{if}\ b\ \texttt{then}\ c; \texttt{while}\ b\ \texttt{do}\ c\ \texttt{else}\ \texttt{skip})B & \text{Prop 8.6.1 und 6.1.2} \\ \iff & \sigma \models b \wedge w(c; \texttt{while}\ b\ \texttt{do}\ c)B \vee \neg b \wedge w(\texttt{skip})B & \text{Teil (4)} \\ \iff & \sigma \models b \wedge w(c; \texttt{while}\ b\ \texttt{do}\ c)B \vee \neg b \wedge B & \text{Teil (1)} \end{array}$$

□

Satz 8.6.4 *Sei w eine SVB-Funktion. Dann:*

$$\forall c \in \mathit{Com}\ \forall B \in \mathit{Assn}\colon\ \vdash \{wcB\}c\{B\}$$

Beweis Durch Induktion über $c \in \mathit{Com}$. Wir zeigen die Beweisteile für Zuweisung, Sequentialisierung und Schleifen.

$c = (x := a)$

$$\begin{array}{ll} \vdash \{B[a/X]\}c\{B\} & \text{Regel für :=} \\ \vdash \{w(X := a)B\}c\{B\} & \text{Prop. 8.6.3} \end{array}$$

$c = (c_1; c_2)$

$$\begin{array}{ll} \vdash \{wc_2B\}c_2\{B\} & \text{Induktionsannahme} \\ \vdash \{wc_1(wc_2B)\}c_1\{wc_2B\} & \text{Induktionsannahme} \\ \vdash \{wc_1(wc_2B)\}c_1; c_2\{B\} & \text{Regel für ;} \\ \vdash \{w(c_1; c_2)B)\}c_1; c_2\{B\} & \text{Prop 8.6.3 und Abschwächungsregel} \end{array}$$

$c = (\texttt{while}\ b\ \texttt{do}\ c')$

$$
\begin{array}{ll}
I \stackrel{\text{def}}{=} wcB & \\
b \wedge I \models\!\!\!\mid b \wedge wc'I & \text{Prop 8.6.3} \\
\neg b \wedge I \models\!\!\!\mid \neg b \wedge B & \text{Prop 8.6.3} \\
\vdash \{wc'I\}c'\{I\} & \text{Induktionsannahme} \\
\vdash \{I \wedge b\}c'\{I\} & \text{Abschwächungsregel} \\
\vdash \{I\}c\{I \wedge \neg b\} & \text{Regel für } \texttt{while} \\
\vdash \{wcB\}c\{B\} & \text{Abschwächungsregel}
\end{array}
$$

□

Korollar 8.6.5 *Wenn eine SVB-Funktion existiert, dann:*

$$\forall A \in \mathit{Assn}\ \forall c \in \mathit{Com}\ \forall B \in \mathit{Assn}\colon \quad \models \{A\}c\{B\} \ \Rightarrow \vdash \{A\}c\{B\}$$

Beweis Sei w eine SVB-Funktion und $\models \{A\}c\{B\}$. Dann $A \models wcB$ mit Proposition 8.6.1. Mit dem obigen Satz folgt $\vdash \{wcB\}c\{B\}$. Also folgt $\vdash \{A\}c\{B\}$ mit der Abschwächungsregel. □

Im Folgenden verwenden wir die Funktion

$$LC \in \mathit{Com} \to \mathcal{P}_{\mathit{fin}}(\mathit{Loc})$$

die uns zu jedem Kommando c die Menge aller Lokationen liefert, die in c vorkommen.

Ein *Transformator* für ein Kommando $c \in \mathit{Com}$ ist eine Funktion

$$\tau \in \mathit{Assn} \to \mathit{Assn}$$

für die gilt:

1. $\forall B \in \mathit{Assn}\colon\ \tau B$ ist SVB für c, B.
2. $\forall B \in \mathit{Assn}\colon\ FV(\tau B) \subseteq FV(B) \cup LC(c)$.

Ein *Schleifenkombinator* ist eine Funktion

$$\omega \in \mathit{Bexp} \times (\mathit{Assn} \to \mathit{Assn}) \to (\mathit{Assn} \to \mathit{Assn})$$

für die gilt:

$$\begin{array}{l}\forall c \in \mathit{Com}\ \forall \tau \in \mathit{Assn} \to \mathit{Assn}\ \forall b \in \mathit{Bexp}\colon \\ \qquad \tau \text{ Transformator für } c \ \Rightarrow\ \omega(b, \tau) \text{ Transformator für } \texttt{while}\ b\ \texttt{do}\ c\end{array}$$

Satz 8.6.6 *Sei ω ein Schleifenkombinator und sei w wie folgt definiert:*

$$
\begin{aligned}
&w \in Com \to Assn \to Assn \\
&\qquad\qquad w(\mathtt{skip})B &&= B \\
&\qquad\qquad w(X := a)B &&= B[a/X] \\
&\qquad\qquad w(c_1; c_2)B &&= wc_1(wc_2B) \\
&w(\mathtt{if}\, b\ \mathtt{then}\, c_1\ \mathtt{else}\, c_2)B &&= b \wedge wc_1B \vee \neg b \wedge wc_2B \\
&\qquad w(\mathtt{while}\, b\ \mathtt{do}\, c)B &&= \omega(b,\ wc)B
\end{aligned}
$$

Dann ist ωc für alle $c \in Com$ ein Transformator für c.

Beweis Machen Sie sich zuerst klar, dass w ordnungsgemäß mittels struktureller Rekursion über $c \in Com$ definiert ist. Wir beweisen durch strukturelle Induktion über $c \in Com$, dass

$$\forall c \in Com\ \forall B \in Assn\ \forall \sigma \in \Sigma\colon\ (\sigma \models wcB \iff \mathcal{C}[\![c]\!]\sigma \models B)$$

Wir beschränken uns auf die Teilbeweise für Zuweisungen, Sequentialisierung und Schleifen:

$c = (X := a)$

$$
\begin{aligned}
\sigma \models wcB &\iff \sigma \models B[a/X] && \text{Definition von } w \\
&\iff \sigma[\mathcal{A}[\![a]\!]\sigma/X] \models B && \text{Substitutionseigenschaft} \\
&\iff \mathcal{C}[\![c]\!]\sigma \models B && \text{Definition von } \mathcal{C}
\end{aligned}
$$

$c = (c_1; c_2)$

$$
\begin{aligned}
&\sigma \models wcB \\
&\iff \sigma \models wc_1(wc_2B) && \text{Def. von } w \\
&\iff \mathcal{C}[\![c_1]\!]\sigma \models wc_2B && \text{Induktionsannahme für } c_1 \\
&\iff \mathcal{C}[\![c_1]\!]\sigma \neq \bot \Rightarrow \mathcal{C}[\![c_1]\!]\sigma \models wc_2B \\
&\iff \mathcal{C}[\![c_1]\!]\sigma \neq \bot \Rightarrow \mathcal{C}[\![c_2]\!](\mathcal{C}[\![c_1]\!]\sigma) \models B && \text{Induktionsannahme für } c_2 \\
&\iff \mathcal{C}[\![c]\!]\sigma \models B && \text{Definition von } \mathcal{C}
\end{aligned}
$$

$c = (\mathtt{while}\ b\ \mathtt{do}\ c')$

$$
\begin{aligned}
\sigma \models wcB &\iff \sigma \models \omega(b, wc')B && \text{Def. von } w \\
&\iff \mathcal{C}[\![c]\!]\sigma \models B && \text{Induktionsannahme für } c'
\end{aligned}
$$

□

8.7 Konstruktion eines Schleifenkombinators

Im Folgenden nehmen wir an, dass eine endliche Menge

$$L \in \mathcal{P}_{fin}(Loc)$$

von Lokationen und eine bijektive Funktion

$$\lambda \in L \to \{1, \dots, |L|\}$$

gegeben sind. Dabei ist $|L|$ die Anzahl der Lokationen in L. Wir benötigen die folgende Notationen:

$$\begin{aligned}
\Sigma_L &\stackrel{\text{def}}{=} \{\sigma \in \Sigma \mid \forall X \in Loc - L\colon\ \sigma X = 0\} \\
Com_L &\stackrel{\text{def}}{=} \{c \in Com \mid c \text{ enthält nur Lokationen in } L\} \\
Assn_L &\stackrel{\text{def}}{=} \{A \in Assn \mid FV(A) \subseteq L\} \\
Bexp_L &\stackrel{\text{def}}{=} \{b \in Bexp \mid FV(b) \subseteq L\}
\end{aligned}$$

Wir definieren eine Funktion

$$\begin{aligned}
&Z \in \Sigma_L \to Assn_L \\
&Z\sigma = (\bigwedge_{X \in L} X = \sigma X)
\end{aligned}$$

Diese Definition ist etwas schlampig, da wir die Glieder der Konjunktion in einer bestimmten Reihenfolge anordnen müssen. Das ist aber kein Problem, da λ eine totale Ordnung auf L induziert, die wir für die Anordnung benützen können.

Proposition 8.7.1 $\forall \sigma, \sigma' \in \Sigma_L\colon\ \sigma = \sigma' \iff \sigma' \models Z\sigma$.

Proposition 8.7.2 *Sei τ ein Transformator für $c \in Com_L$. Dann:*

$$\forall \sigma, \sigma' \in \Sigma_L\colon\ \mathcal{C}[\![c]\!]\sigma = \sigma' \iff\ \models (Z\sigma \Rightarrow (\tau(Z\sigma') \wedge \neg\tau(\texttt{false})))$$

Lemma 8.7.3 (Generatorfunktion) *Es gibt eine Funktion $g \in \mathbb{Z}^3 \to \mathbb{Z}$ so dass:*

$$\forall k \geq 1\ \forall \langle x_1, \dots, x_k \rangle \in \mathbb{Z}^k\ \exists u, v \in \mathbb{Z}\ \forall i \in \{1, \dots, k\}\colon\ x_i = g(u, v, i)$$

Mithilfe einer Generatorfunktion kann jedes Tupel von Zahlen durch nur zwei Zahlen u und v repräsentiert werden (*** eine Zahl würde reichen ***). Die Konstruktion einer Generatorfunktion findet man in [Winskel, Kapitel 7.2]. Sie basiert auf Gödels β-Prädikat. Im Folgenden nehmen wir an, dass eine Generatorfunktion g gegeben ist.

Jetzt definieren wir eine Funktion

$$\begin{aligned}
&S \in \mathbb{Z}^3 \to \Sigma_L \\
&S(u, v, i)X = \text{if } X \in L \text{ then } g(u, v, (i-1)|L| + \lambda X) \text{ else } 0
\end{aligned}$$

Proposition 8.7.4 *Sei $k \geq 1$ und $\langle \sigma_1, \ldots, \sigma_k \rangle \in {\Sigma_L}^k$. Dann existieren $u, v \in \mathbb{Z}$ so dass $\sigma_i = S(u, v, i)$ für alle $i \in \{1, \ldots, k\}$.*

Die Proposition sagt, dass sich jedes Tupel von Zuständen in Σ_L durch zwei Zahlen repräsentieren lässt.

Wir definieren eine Funktion

$$T \in \mathbb{Z}^3 \to Assn_L$$
$$T(u, v, i) = \bigwedge_{X \in L} X = g(u, v, (i-1)|L| + \lambda X)$$

Proposition 8.7.5 *Seien $u, v, i \in \mathbb{Z}$ und $\sigma \in \Sigma_L$. Dann:*

1. $\sigma = S(u, v, i) \iff \sigma \models T(u, v, i)$.
2. $Z(S(u, v, i)) \models\!\!\!\dashv T(u, v, i)$.
3. $\forall A \in Assn_L\colon\ S(u, v, i) \models A \iff \models (T(u, v, i) \Rightarrow A)$.

Der Ausgangspunkt für die Konstruktion des Schleifenkombinators ist:

Lemma 8.7.6 *Seien $b \in Bexp_L$, $c \in Com_L$, $B \in Assn_L$ und $\sigma \in \Sigma$. Dann*

$$\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma \models B$$

genau dann, wenn

$$\begin{aligned}
&\forall k \geq 1\ \forall \langle \sigma_1, \ldots, \sigma_k \rangle \in \Sigma_L^k\colon \\
&\quad (\ \sigma \models Z\sigma_1 \ \wedge \\
&\quad\ \ \forall i \in \{1, \ldots, k-1\}\colon \\
&\quad\quad\ \sigma_i \models b \ \wedge\ \mathcal{C}[\![c]\!]\sigma_i = \sigma_{i+1}\ ) \\
&\quad \Rightarrow\ \ \sigma_k \models b \vee B
\end{aligned}$$

Beweis Die zwei Richtungen der Äquivalenz müssen getrennt gezeigt werden. Die Richtung „⇒“ zeigt man durch Induktion über die Fixpunktkonstruktion (siehe den Beweisteil für Schleifen bei Proposition 6.3.2). Die andere Richtung zeigt man durch Induktion über k. □

Wir formen jetzt die Aussage auf der rechten Seite der Äquivalenz des Lemmas schrittweise um. Schließlich werden wir eine Zusicherung $A \in Assn_L$ konstruieren, so dass die Aussage genau dann gilt, wenn $\sigma \models A$ gilt. Damit ist A eine schwächste Vorbedingung für `while` b `do` c und B.

Als erstes machen wir Gebrauch von der Generatorfunktion und eliminieren damit die Quantifizierung über die Zustände σ_i.

$$\begin{array}{l}
\forall k \geq 1\ \forall u, v \in \mathbb{Z}: \\
\quad [\ \sigma \models Z(S(u, v, 1)) \wedge \\
\qquad \forall i \in \{1, \ldots, k-1\}: \\
\qquad\quad S(u, v, i) \models b\ \wedge\ \mathcal{C}[\![c]\!](S(u, v, i)) = S(u, v, i+1)\ ] \\
\quad \Rightarrow\ S(u, v, k) \models b \vee B
\end{array}$$

Um die Aussage besser lesbar zu machen, verwenden wir auch eckige Klammern. Sei τ ein Transformator für c. Mithilfe der Propositionen 8.7.5 und 8.7.2 bekommen wir:

$$\begin{array}{l}
\forall k \geq 1\ \forall u, v \in \mathbb{Z}: \\
\quad [\ \sigma \models T(u, v, 1) \wedge \\
\qquad \forall i \in \{1, \ldots, k-1\}: \\
\qquad\quad \models T(u, v, i) \Rightarrow b \wedge\ \models (T(u, v, i) \Rightarrow (\tau(T(u, v, i+1)) \wedge \neg\tau(\texttt{false})))\ ] \\
\quad \Rightarrow\ \models T(u, v, k) \Rightarrow (b \vee B)
\end{array}$$

Diese Aussage lässt sich noch etwas vereinfachen:

$$\begin{array}{l}
\forall k \geq 1\ \forall u, v \in \mathbb{Z}: \\
\quad [\ \sigma \models T(u, v, 1) \wedge \\
\qquad \forall i \in \{1, \ldots, k-1\}: \\
\qquad\quad \models (T(u, v, i) \Rightarrow (b \wedge \tau(T(u, v, i+1)) \wedge \neg\tau(\texttt{false})))\ ] \\
\quad \Rightarrow\ \models T(u, v, k) \Rightarrow (b \vee B)
\end{array}$$

Als nächstes wollen wir die Metaquantifizierung für i durch eine Objektquantifizierung ersetzen. Dazu nehmen wir an, dass wir eine Funktion

$$\tilde{T} \in \mathit{Ter}^3 \to \mathit{Assn}_L$$

konstruieren können, für die gilt:

$$\forall a_1, a_2, a_3 \in \mathit{Ter}:\ \ T(\mathcal{A}[\![a_1]\!]\sigma, \mathcal{A}[\![a_2]\!]\sigma, \mathcal{A}[\![a_3]\!]\sigma) \models\!\!\!\dashv \tilde{T}(a_1, a_2, a_3)$$

Die Konstruktion von $\tilde{T}$ gelingt mit Gödels β-Prädikat (siehe [Winskel, Kapitel 7.2]).

Jetzt wählen wir eine Variable $X_1 \in Loc - L$ und eliminieren damit die Metaquantifizierung für i:

$$
\begin{aligned}
&\forall k \geq 1\ \forall u, v \in \mathbb{Z}: \\
&\quad [\ \sigma \models T(u, v, 1) \land \\
&\quad\ \models \forall X_1\ ((1 \leq X_1 \land X_1 \leq k-1 \land \tilde{T}(u, v, X_1)) \\
&\qquad\qquad\qquad \Rightarrow (b \land \tau(\tilde{T}(u, v, X_1 + 1)) \land \neg\tau(\texttt{false})))\ ] \\
&\quad \Rightarrow\ \models T(u, v, k) \Rightarrow (b \lor B)
\end{aligned}
$$

Die in den Objektformeln frei auftretetenden Variablen sind alle in L. Wir schreiben $\forall L$ als Abkürzung für

$$\forall Y_1\ \dots\ \forall Y_m$$

wobei $L = \{Y_1, \dots, Y_m\}$ gelten soll. Damit können wir die obige Aussage wie folgt umschreiben:

$$
\begin{aligned}
&\forall k \geq 1\ \forall u, v \in \mathbb{Z}: \\
&\quad \sigma \models (\ [\ T(u, v, 1) \land \\
&\qquad\qquad \forall L\ \forall X_1\ (\ (1 \leq X_1 \land X_1 \leq k-1 \land \tilde{T}(u, v, X_1)) \\
&\qquad\qquad\qquad\qquad \Rightarrow (b \land \tau(\tilde{T}(u, v, X_1 + 1)) \land \neg\tau(\texttt{false}))\ )\ ] \\
&\qquad \Rightarrow\ \forall L\ T(u, v, k) \Rightarrow (b \lor B)\ )
\end{aligned}
$$

Wenn wir jetzt noch die Metaquantifizierungen für k, u, v wie vorher für i auf die Objektebene schieben, haben wir eine schwächste Vorbedingung $A \in Assn_L$ für `while` b `do` c und B konstruiert. Also haben wir das folgende Lemma bewiesen:

Lemma 8.7.7 *Seien $b \in Bexp_L$, $c \in Com_L$, $B \in Assn_L$ und τ ein Transformator für c. Dann kann man eine Formel $A \in Assn_L$ konstruieren, so dass A eine schwächste Vorbedingung für* `while` *b* `do` *c und B ist.*

Satz 8.7.8 *Man kann einen Schleifenkombinator konstruieren.*

8.8 Zusammenfassung

Wir haben gesehen, das wir mit den Formeln von ASSN Spezifikationen für die Kommandos von IMP schreiben können. Spezifikationen bestehen dabei aus einer Vor- und Nachbedingung. Den Begriff der partiellen Korrektheit haben wir basierend auf der denotationalen Semantik von IMP wie folgt definiert:

$$\models \{A\}c\{B\} \iff (\forall \sigma:\ \sigma \models A \Rightarrow \mathcal{C}[\![c]\!]\sigma \models B)$$

Mit den Hoare-Regeln haben wir unabhängig von den beiden Semantiken für IMP eine Relation

$$\vdash \{A\}c\{B\}$$

definiert. Wir haben gezeigt, dass

$$\models \{A\}c\{B\} \iff \vdash \{A\}c\{B\}$$

gilt. Die Richtung von rechts nach links heißt Korrektheit und ist einfach zu zeigen. Die andere Richtung heißt Vollständigkeit und ist nicht einfach zu zeigen. Die Schwierigkeit kommt daher, dass wir die Semantik von Schleifen (ein Rekursionskonstrukt) mithilfe einer Sprache (ASSN) charakterisieren müssen, die kein Rekursionskonstrukt hat. Die Charakterisierung der Semantik von Schleifen gelingt mithilfe von Quantifizierung.

Für die Vollständigkeit der Hoare-Regeln haben wir gezeigt, dass man zu jedem Kommando c und jeder Nachbedingung B eine schwächste Vorbedingung A konstruieren kann, die wie folgt charakterisiert ist:

$$\forall \sigma : \ \sigma \models A \iff \mathcal{C}[\![c]\!]\sigma \models B$$

Mithilfe von schwächsten Vorbedingungen kann man zu c, σ, σ' stets eine Formel A von IMP konstruieren (Proposition 8.7.2), für die gilt:

$$\mathcal{C}[\![c]\!]\sigma = \sigma' \iff \models A$$

Man kann also die Sematik von IMP mithilfe der Formeln von ASSN beschreiben.

Um die Verifikation der Gültigkeit von Hoare-Tripeln praktikabel zu machen, haben wir einen Verifikationsbedingungsgenerator *VC* angegeben, der zu einem mit Schleifeninvarianten annotierten Kommando c und zu Formel A, B endlich viele Formeln berechnet, so dass gilt:

$$\models \{A\}c\{B\} \iff \exists \text{Annotierung von } c : \quad \models VC(A, c, B)$$