7.5 Vollständigkeit der axiomatischen Semantik

Korrektheit (Thm. 49) sagt aus, dass sich mit den Regeln der axiomatischen Semantik nur Eigenschaften beweisen lassen, die auch in der operationalen gelten. Umgekehrt bedeutet Vollständigkeit eines Kalküls, dass sich alle richtigen Aussagen auch mit den Regeln des Kalküls beweisen lassen. Für die axiomatische Semantik bedeutet dies, dass wenn $\vDash \{ P \}\, c \,\{ Q \}$, dann auch $\vdash \{ P \}\, c \,\{ Q \}$. Diese Vollständigkeit wollen wir in diesem Teil untersuchen.

Definition 48 (Schwächste freie Vorbedingung). Die *schwächste freie Vorbedingung* (*weakest liberal precondition*) $\mathrm{wlp}\,(c,\ Q)$ zu einer Anweisung c und einer Nachbedingung Q ist definiert als

$$\mathrm{wlp}\,(c,\ Q) = \lambda\sigma.\ \forall\sigma'.\ \langle c,\ \sigma\rangle \Downarrow \sigma' \implies Q(\sigma')$$

Sie beschreibt also gerade die Menge von Zuständen, die als Anfangszustand aller terminierenden Ausführungen von c nur zu Endzuständen in Q führen.

Beispiel 41. Die schwächste freie Vorbedingung für $Q = \lambda\sigma.\ \mathbf{ff}$ ist die Menge der Anfangszustände (als Prädikat betrachtet), für die c nicht terminiert. Konkret:

$$\mathrm{wlp}\,(\texttt{while (true) do skip},\ \mathbf{ff})\,\sigma = \mathbf{tt}$$
$$\mathrm{wlp}\,(\texttt{y := 1; while (not (x == 1)) do (y := y * x; x := x - 1)},\ \mathbf{ff})\,\sigma = \sigma(\mathtt{x}) \leq 0$$

Lemma 50. Für alle c und Q ist $\mathrm{wlp}\,(c,\ Q)$ die schwächste mögliche Vorbedingung:

$$\vDash \{ \mathrm{wlp}\,(c,\ Q) \}\, c \,\{ Q \} \qquad \text{und} \qquad \text{wenn, } \vDash \{ P \}\, c \,\{ Q \} \text{ dann } P \implies \mathrm{wlp}\,(c,\ Q)$$

Beweis. Zum Beweis von $\vDash \{ \mathrm{wlp}\,(c,\ Q) \}\, c \,\{ Q \}$ seien σ, σ' beliebig mit $\mathrm{wlp}\,(c,\ Q)\,(\sigma)$ und $\langle c,\ \sigma\rangle \Downarrow \sigma'$. Nach Definition von $\mathrm{wlp}\,(c,\ Q)$ gilt $Q(\sigma')$, was zu zeigen ist.

Sei nun $\vDash \{ P \}\, c \,\{ Q \}$. Zu zeigen: Für alle σ mit $P(\sigma)$ gilt $\mathrm{wlp}\,(c,\ Q)\,(\sigma)$.
Sei also σ' beliebig mit $\langle c,\ \sigma\rangle \Downarrow \sigma'$. Wegen $P(\sigma)$ gilt dann nach $\vDash \{ P \}\, c \,\{ Q \}$ auch $Q(\sigma')$, was zu zeigen ist. □

Lemma 51. Für alle c und Q gilt $\vdash \{ \mathrm{wlp}\,(c,\ Q) \}\, c \,\{ Q \}$.

Beweis. Beweis durch Induktion über c (Q beliebig).

- Fall `skip`: Zu zeigen: $\vdash \{ \mathrm{wlp}\,(\texttt{skip},\ Q) \}\, \texttt{skip} \,\{ Q \}$.

  Es gilt $\mathrm{wlp}\,(\texttt{skip},\ Q)\,(\sigma) = (\forall\sigma'.\, \langle \texttt{skip},\ \sigma\rangle \Downarrow \sigma' \implies Q(\sigma')) = Q(\sigma)$. Damit folgt die Behauptung aus der Regel $\textsc{Skip}_\mathrm{P}$.

- Fall x `:=` a: Zu zeigen: $\vdash \{ \mathrm{wlp}\,(x \texttt{ := } a,\ Q) \}\, x \texttt{ := } a \,\{ Q \}$.

  Es gilt $\mathrm{wlp}\,(x \texttt{ := } a,\ Q) = Q[x \mapsto \mathcal{A}\,[\![a]\!]]$, da

  $$\begin{aligned}\mathrm{wlp}\,(x \texttt{ := } a,\ Q)\,(\sigma) &= (\forall\sigma'.\, \langle x \texttt{ := } a,\ \sigma\rangle \Downarrow \sigma' \implies Q(\sigma')) \\ &= (\forall\sigma'.\ \sigma' = \sigma[x \mapsto \mathcal{A}\,[\![a]\!]\,\sigma] \implies Q(\sigma')) = Q(\sigma[x \mapsto \mathcal{A}\,[\![a]\!]\,\sigma])\end{aligned}$$

  Damit folgt die Behauptung nach der Regel $\textsc{Ass}_\mathrm{P}$.

- Fall $c_1\texttt{;}\ c_2$:
  Induktionsannahmen: Für alle Q gelten $\vdash \{ \mathrm{wlp}\,(c_1,\ Q) \}\, c_1 \,\{ Q \}$ und $\vdash \{ \mathrm{wlp}\,(c_2,\ Q) \}\, c_2 \,\{ Q \}$.
  Zu zeigen: $\vdash \{ \mathrm{wlp}\,(c_1\texttt{;}\ c_2,\ Q) \}\, c_1\texttt{;}\ c_2 \,\{ Q \}$.

  Aus den Induktionsannahmen folgen $\vdash \{ \mathrm{wlp}\,(c_1,\ \mathrm{wlp}\,(c_2,\ Q)) \}\, c_1 \,\{ \mathrm{wlp}\,(c_2,\ Q) \}$ und
  $\vdash \{ \mathrm{wlp}\,(c_2,\ Q) \}\, c_2 \,\{ Q \}$. Damit gilt auch $\vdash \{ \mathrm{wlp}\,(c_1,\ \mathrm{wlp}\,(c_2,\ Q)) \}\, c_1\texttt{;}\ c_2 \,\{ Q \}$ nach Regel $\textsc{Seq}_\mathrm{P}$.
  Daraus folgt die Behauptung nach Regel $\textsc{Cons}_\mathrm{P}$, falls $\mathrm{wlp}\,(c_1\texttt{;}\ c_2,\ Q) \implies \mathrm{wlp}\,(c_1,\ \mathrm{wlp}\,(c_2,\ Q))$.

Für diese Implikation ist für alle σ zu zeigen, dass wenn $\text{wlp}\,(c_1\,;\ c_2,\ Q)\,\sigma$ gilt, dann gilt auch $\text{wlp}\,(c_1,\ \text{wlp}\,(c_2,\ Q))\,\sigma$.
Sei also – nach Definition von $\text{wlp}\,(\_,\ \_)$ – σ' beliebig mit $\langle c_1,\ \sigma\rangle \Downarrow \sigma'$. Zu zeigen: $\text{wlp}\,(c_2,\ Q)\,\sigma'$.
Sei also – wieder nach Definition – σ'' beliebig mit $\langle c_2,\ \sigma'\rangle \Downarrow \sigma''$. Nun bleibt zu zeigen: $Q(\sigma'')$.
Aus den Annahmen $\langle c_1,\ \sigma\rangle \Downarrow \sigma'$ und $\langle c_2,\ \sigma'\rangle \Downarrow \sigma''$ folgt, dass $\langle c_1\,;\ c_2,\ \sigma\rangle \Downarrow \sigma''$. Da $\text{wlp}\,(c_1\,;\ c_2,\ Q)\,\sigma$, folgt die Behauptung $Q(\sigma')$ nach Definition von $\text{wlp}\,(\_,\ \_)$.

- Fall `if (`b`) then` c_1 `else` c_2:
  Induktionsannahmen: Für alle Q gelten $\vdash \{\,\text{wlp}\,(c_1,\ Q)\,\}\,c_1\,\{\,Q\,\}$ und $\vdash \{\,\text{wlp}\,(c_2,\ Q)\,\}\,c_2\,\{\,Q\,\}$.
  Zu zeigen: $\vdash \{\,\text{wlp}\,(\texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2,\ Q)\,\}\,\texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2\,\{\,Q\,\}$.

  Sei P definiert als

  $$P(\sigma) = (\mathcal{B}\,[\![b]\!]\,\sigma \wedge \text{wlp}\,(c_1,\ Q)\,\sigma) \vee (\neg\mathcal{B}\,[\![b]\!]\,\sigma \wedge \text{wlp}\,(c_2,\ Q)\,\sigma)$$

  Dann gilt mit $c = \texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2$:

  $$\frac{\text{wlp}\,(c,\ Q) \implies P \qquad \dfrac{A \qquad B}{\vdash \{\,P\,\}\,c\,\{\,Q\,\}}\ \textsc{If}_\text{P} \qquad Q \implies Q}{\vdash \{\,\text{wlp}\,(c,\ Q)\,\}\,c\,\{\,Q\,\}}\ \textsc{Cons}_\text{P}$$

  wobei mit den Induktionsannahmen gilt:

  $$\text{A: } \frac{\mathcal{B}\,[\![b]\!] \wedge P \implies \text{wlp}\,(c_1,\ Q) \qquad \vdash \{\,\text{wlp}\,(c_1,\ Q)\,\}\,c_1\,\{\,Q\,\} \qquad Q \implies Q}{\vdash \{\,\mathcal{B}\,[\![b]\!] \wedge P\,\}\,c_1\,\{\,Q\,\}}\ \textsc{Cons}_\text{P}$$

  $$\text{B: } \frac{\neg\mathcal{B}\,[\![b]\!] \wedge P \implies \text{wlp}\,(c_2,\ Q) \qquad \vdash \{\,\text{wlp}\,(c_2,\ Q)\,\}\,c_2\,\{\,Q\,\} \qquad Q \implies Q}{\vdash \{\,\neg\mathcal{B}\,[\![b]\!] \wedge P\,\}\,c_2\,\{\,Q\,\}}\ \textsc{Cons}_\text{P}$$

  Die Implikation $\text{wlp}\,(c,\ Q) \implies P$ wird wie im Fall $c_1\,;\ c_2$ gezeigt.

- Fall `while (`b`) do` c: Induktionsannahme: $\vdash \{\,\text{wlp}\,(c,\ Q)\,\}\,c\,\{\,Q\,\}$ für alle Q
  Zu zeigen: $\vdash \{\,\text{wlp}\,(\texttt{while (}b\texttt{) do}\ c,\ Q)\,\}\,\texttt{while (}b\texttt{) do}\ c\,\{\,Q\,\}$.

  Sei $P = \text{wlp}\,(\texttt{while (}b\texttt{) do}\ c,\ Q)$. Wir wollen zeigen, dass P eine Schleifeninvariante ist. Mit der Induktionsannahme, spezialisiert auf $Q = P$, gilt dann:

  $$\frac{\dfrac{\dfrac{\mathcal{B}\,[\![b]\!] \wedge P \implies \text{wlp}\,(c,\ P) \quad \vdash \{\,\text{wlp}\,(c,\ P)\,\}\,c\,\{\,P\,\}}{\vdash \{\,\mathcal{B}\,[\![b]\!] \wedge P\,\}\,c\,\{\,P\,\}}\ \textsc{Cons}_\text{P}}{\vdash \{\,P\,\}\,\texttt{while (}b\texttt{) do}\ c\,\{\,\neg\mathcal{B}\,[\![b]\!] \wedge P\,\}}\ \textsc{While}_\text{P} \qquad \neg\mathcal{B}\,[\![b]\!] \wedge P \implies Q}{\vdash \{\,P\,\}\,\texttt{while (}b\texttt{) do}\ c\,\{\,Q\,\}}\ \textsc{Cons}_\text{P}$$

  Wir müssen dazu aber noch die Implikationen der $\textsc{Cons}_\text{P}$-Anwendungen nachweisen:
  - $\neg\mathcal{B}\,[\![b]\!] \wedge P \implies Q$: Sei σ beliebig mit $\neg\mathcal{B}\,[\![b]\!]\,\sigma \wedge P(\sigma)$, also insbesondere $\mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{ff}$. Dann gilt $\langle\texttt{while (}b\texttt{) do}\ c,\ \sigma\rangle \Downarrow \sigma$ nach Regel $\textsc{WhileFF}_\text{BS}$. Wegen $P(\sigma)$ folgt damit $Q(\sigma)$ nach Definition.
  - $\mathcal{B}\,[\![b]\!] \wedge P \implies \text{wlp}\,(c,\ P)$: Sei also σ beliebig mit $\mathcal{B}\,[\![b]\!]\,\sigma \wedge P(\sigma)$. Zu zeigen: $\text{wlp}\,(c,\ P)\,\sigma$.
    Sei also σ' beliebig mit $\langle c,\ \sigma\rangle \Downarrow \sigma'$. Zu zeigen: $P(\sigma')$.
    Da $P = \text{wlp}\,(\texttt{while (}b\texttt{) do}\ c,\ Q)$, sei also σ'' beliebig mit $\langle\texttt{while (}b\texttt{) do}\ c,\ \sigma'\rangle \Downarrow \sigma''$. Zu zeigen: $Q(\sigma'')$.
    Wegen $\langle c,\ \sigma\rangle \Downarrow \sigma'$ und $\langle\texttt{while (}b\texttt{) do}\ c,\ \sigma'\rangle \Downarrow \sigma''$ gilt auch $\langle\texttt{while (}b\texttt{) do}\ c,\ \sigma\rangle \Downarrow \sigma''$ nach Regel $\textsc{WhileTT}_\text{BS}$. Mit $\text{wlp}\,(\texttt{while (}b\texttt{) do}\ c,\ Q)\,\sigma$ $(= P(\sigma))$, folgt $Q(\sigma'')$. □

Theorem 52 (Vollständigkeit der axiomatischen Semantik).
Wenn $\vDash \{\,P\,\}\,c\,\{\,Q\,\}$, dann $\vdash \{\,P\,\}\,c\,\{\,Q\,\}$.

Beweis. Nach Lem. 50 und 51 gilt:

$$\frac{P \implies \mathrm{wlp}\,(c,\ Q) \qquad \vdash \{\,\mathrm{wlp}\,(c,\ Q)\,\}\,c\,\{\,Q\,\} \qquad Q \implies Q}{\vdash \{\,P\,\}\,c\,\{\,Q\,\}}\ \textsc{Cons}_{\mathrm{P}} \qquad \square$$

Korollar 53. $\vDash \{\,P\,\}\,c\,\{\,Q\,\}$ gdw. $\vdash \{\,P\,\}\,c\,\{\,Q\,\}$.

7.6 Semantische Prädikate und syntaktische Bedingungen

Beispiel 41 suggeriert bereits, dass es keinen Algorithmus geben kann, der die Ableitbarkeit von $\vdash \{\,P\,\}\,c\,\{\,Q\,\}$ entscheiden kann – sonst wäre auch das Halteproblem lösbar: Seien $P = \lambda\sigma.\ \mathbf{tt}$ und $Q = \lambda\sigma.\ \mathbf{ff}$. Die Zusicherung $\vdash \{\,P\,\}\,c\,\{\,Q\,\}$ charakterisiert dann all die Programme c, die niemals anhalten. Ebensowenig ist $\mathrm{wlp}\,(c,\ Q)$ berechenbar. Dies liegt an der Regel $\textsc{Cons}_{\mathrm{P}}$, da Implikationen zwischen beliebigen Prädikaten P und P' nicht entscheidbar sind, man aber auf diese auch nicht verzichten kann. Ein automatisches Verifikationssystem, das alle Programmeigenschaften beweisen kann, ist also auch mit axiomatischer Semantik nicht möglich.

Damit ein solches System überhaupt arbeiten kann, braucht man eine symbolische Darstellung der Prädikate. Dies ist aber nichts anderes als eine Unterscheidung zwischen Syntax und Semantik! Die Menge der Zustandsprädikate ist die Menge der semantischen Bedeutungsobjekte für Vor- und Nachbedingungen – wir haben also bisher nur mit den semantischen Objekten gearbeitet. Jetzt fehlt uns noch die Syntax und die Interpretationsfunktion $\mathcal{I}\,[\![\_]\!]$, die aus den syntaktischen Ausdrücken wieder das semantische Prädikat gewinnt. Genau genommen sind unsere notationellen Vereinfachungen der Zusicherungen aus Kap. 7.2 bereits ein halbherziger Versuch, Syntax für Bedingungen einzuführen.

Wechselt man von semantischen Prädikaten in den Vor- und Nachbedingungen auf syntaktische Bedingungen innerhalb einer solchen Logik, übertragen sich nicht alle Eigenschaften, die wir in diesem Kapitel untersucht haben. Korrektheit (Thm. 49) ist in jedem Fall gewährleistet, sofern die Operationen $\_[\_ \mapsto \_]$, $\lambda\sigma.\ \_(\sigma) \wedge \_(\sigma)$ und $\mathcal{B}\,[\![b]\!]$, die in den Regeln vorkommen, auch in der Syntax semantisch korrekt umgesetzt werden.

Vollständigkeit (Thm. 52) lässt sich dagegen nicht automatisch übertragen: Es ist nicht klar, dass für alle *syntaktischen* Bedingungen Q die schwächste Vorbedingung $\mathrm{wlp}\,(c,\ [\![Q]\!])$ und alle Zwischenbedingungen (z.B. die Schleifeninvarianten), die für die Ableitbarkeit von $\vdash \{\,\mathrm{wlp}\,(c,\ [\![Q]\!])\,\}\,c\,\{\,[\![Q]\!]\,\}$ benötigt werden, in der Logik *syntaktisch ausdrückbar* sind.

Beispiel 42. Sei $c = c_1\,;\ c_2$ mit

$$c_1 = \texttt{while (1 <= x) do (x := x - 1; z := z + y)}$$
$$c_2 = \texttt{while (1 <= z) do z := z} - \texttt{y}$$

Dann gilt $\vDash \{\,\mathtt{z} = 0 \wedge \mathtt{x} \geq 0 \wedge \mathtt{y} \geq 0\,\}\,c\,\{\,\mathtt{z} = 0\,\}$. Die schwächste Vorbedingung zu c_2 und Nachbedingung $\{\,\mathtt{z} = 0\,\}$ ist, dass $\mathtt{z}$ ein Vielfaches von $\mathtt{y}$ ist, d.h., $\exists n.\ \mathtt{z} = \mathtt{y} \cdot n$. Nimmt man Presburger-Arithmetik[6] als Sprache der Zusicherungen, so lässt sich dieser Zusammenhang zwischen $\mathtt{y}$ und $\mathtt{z}$ nicht ausdrücken und damit auch nicht mit dem Kalkül $\vdash \{\,\_\,\}\,\_\,\{\,\_\,\}$ beweisen. Bemerkenswert ist dabei, dass das Programm c selbst gar keine Multiplikation verwendet, sondern diese aus der einfacheren Addition synthetisiert.

Für die mächtigere Sprache „Logik erster Stufe mit Addition und Multiplikation“ (FOL + $(\mathbb{N}, +, \cdot)$) gilt, dass sich alle schwächsten freien Vorbedingungen ausdrücken lassen – diese Eigenschaft heißt *Expressivität*. Damit überträgt sich dann auch die Vollständigkeit (Thm. 52).

[6] Presburger-Arithmetik ist die Logik erster Stufe mit Addition auf den ganzen Zahlen, aber ohne Multiplikation.

Theorem 54. Sei Q eine (syntaktische) Formel aus FOL + $(\mathbb{N}, +, \cdot)$. Dann ist auch $\mathrm{wlp}\,(c,\ Q)$ in FOL + $(\mathbb{N}, +, \cdot)$ ausdrückbar.

Beweisidee.

1. Die Menge V der in Q und c vorkommenden Variablen ist endlich und $\mathrm{wlp}\,(c,\ Q)$ ignoriert die Belegungen aller anderen Variablen. Deswegen lässt sich der relevante Teil eines Zustands als endliches Tupel von Zahlen schreiben. Dieses Tupel lässt sich wiederum in einer einzigen natürlichen Zahl, der Gödelnummer des Zustands, kodieren.
2. Programmausführungen lassen sich als FOL-Formel kodieren. Für ein Programm c mit Variablen aus V gibt es eine FOL-Formel $R_c(\sigma, \sigma')$, so dass $R_c(\sigma, \sigma')$ gilt gdw. $\langle c,\ \sigma\rangle \Downarrow \sigma'$.
3. Dann ist $P(\sigma) = \exists\sigma'.\ R_c(\sigma, \sigma') \wedge Q(\sigma')$ die gesuchte Formel für $\mathrm{wlp}\,(c,\ Q)$. □

Trotz dieses Theorems ist $\vdash \{\_\}\,\_\,\{\_\}$ ist nicht rekursiv aufzählbar, d.h., es gibt keinen Algorithmus, der alle wahren Aussagen über alle Programme ausgeben könnte. Ansonsten wäre das Halteproblem entscheidbar, da dann auch $\{\, c \mid \vdash \{\,\mathtt{true}\,\}\, c\, \{\,\mathtt{false}\,\}\,\}$ rekursiv aufzählbar wäre. Diese Menge besteht aber genau aus den Programmen, die niemals terminieren, und von der ist bekannt, dass sie nicht rekursiv aufzählbar ist. Dies mag zuerst überraschen, weil Thm. 54 eigentlich einen Konstruktionsalgorithmus für die schwächste Vorbedingung für alle Programme liefert. Allerdings darf man mit Regel $\textsc{Cons}_{\mathrm{P}}$ jederzeit Vorbedingungen verschärfen und Implikationen in FOL sind nicht rekursiv aufzählbar.

Deswegen nennt man das Hoare-Kalkül *relativ vollständig*: Es ist vollständig (im Sinne rekursiver Aufzählbarkeit aller Ableitungen) relativ zur Vollständigkeit (d.h., rekursiven Aufzählbarkeit) der verwendeten Formelsprache.

7.7 Verifikationsbedingungen

Axiomatische Semantik eignet sich für den Nachweis konkreter Eigenschaften eines konkreten Programms besser als operationale und denotationale Ansätze, weil die Regeln der axiomatischen Semantik die Programmsyntax in eine Formelsprache überführen, in der die Programmsyntax nicht mehr vorkommt. Der Beweis der Vollständigkeit (Thm. 52) hat gezeigt, dass letztendlich nur die Schleifeninvarianten gefunden und die Anwendungen der Regel $\textsc{Cons}_{\mathrm{P}}$ nachgerechnet werden müssen, der Rest ergibt sich automatisch aus der Programmstruktur. Dies lässt sich automatisieren und liefert einen *Verifikationsbedingungsgenerator* vc.

Im Folgenden bäuchten wir eigentlich eine Formelsyntax für Invarianten, Vor- und Nachbedingungen, aber die konkrete Syntax spielt keine wesentliche Rolle. Es genügt vorläufig, wenn alle booleschen Ausdrücke aus Bexp enthalten sind. Außerdem brauchen wir noch

1. die Interpretationsfunktion $\mathcal{I}\,[\![\_]\!]$ auf Formeln – d.h., $\mathcal{I}\,[\![P]\!]\,\sigma = \mathbf{tt}$ gdw. die Variablenbelegung (= Zustand) σ Formel P erfüllt –, die boolesche Ausdrücke und Konnektoren entsprechend $\mathcal{B}\,[\![\_]\!]$ interpretiert, sowie
2. eine Substitutionsfunktion $\_[x \mapsto a]$, die mit der Interpretationsfunktion kommutiert:

$$\mathcal{I}\,[\![P[x \mapsto a]]\!]\,\sigma = \mathcal{I}\,[\![P]\!]\,(\sigma[x \mapsto \mathcal{A}\,[\![a]\!]\,\sigma])$$

Eine Formel P gilt, wenn sie für alle Belegungen σ wahr ist, d.h., $\mathcal{I}\,[\![P]\!]\,\sigma = \mathbf{tt}$ für alle σ.

In den Beispielen verwenden wir Bexp als Sprache und $\mathcal{B}[\![\_]\!]$ als Interpretation. Die Substitutionsfunktionen $a[x \mapsto a']$ bzw. $b[x \mapsto a']$ ersetzen alle Vorkommen von x im arithmetischen bzw. booleschen Ausdruck a bzw. b durch a'. Für sie gilt auch folgendes Substitutionslemma:

Lemma 55 (Substitutionslemma für arithmetische und boolesche Ausdrücke).

$$\mathcal{A}[\![a[x \mapsto a']]\!]\,\sigma = \mathcal{A}[\![a]\!]\,(\sigma[x \mapsto \mathcal{A}[\![a']\!]\,\sigma]) \qquad \text{und} \qquad \mathcal{B}[\![b[x \mapsto a']]\!]\,\sigma = \mathcal{B}[\![b]\!]\,(\sigma[x \mapsto \mathcal{A}[\![a']\!]\,\sigma])$$

Beweis. Induktion über a bzw. b und ausrechnen. □

Definition 49 (Annotiertes Programm). In einem annotierten Programm ist jede Schleife mit einer Schleifeninvariante I annotiert.

$$\mathsf{ACom} \quad c ::= \texttt{skip} \mid x \texttt{ := } a \mid c_1\texttt{; } c_2 \mid \texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2 \mid \texttt{while (}b\texttt{) \{}I\texttt{\} do } c$$

Die Funktion strip :: ACom ⇒ Com entfernt alle Annotationen eines Programms:

$$\begin{array}{ll}
\text{strip}(\texttt{skip}) & = \texttt{skip} \\
\text{strip}(x \texttt{ := } a) & = x \texttt{ := } a \\
\text{strip}(c_1\texttt{; } c_2) & = \text{strip}(c_1)\texttt{; strip}(c_2) \\
\text{strip}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2) & = \texttt{if (}b\texttt{) then strip}(c_1)\texttt{ else strip}(c_2) \\
\text{strip}(\texttt{while (}b\texttt{) \{}I\texttt{\} do } c) & = \texttt{while (}b\texttt{) do strip}(c)
\end{array}$$

Die berechnete Vorbedingung $\text{pre}(c, Q)$ für das annotierte Programm c und die Nachbedingung Q ist im Wesentlichen die schwächste freie Vorbedingung – bis auf die annotierten Invarianten:

$$\begin{array}{ll}
\text{pre}(\texttt{skip}, Q) & = Q \\
\text{pre}(x \texttt{ := } a, Q) & = Q[x \mapsto a] \\
\text{pre}(c_1\texttt{; } c_2, Q) & = \text{pre}(c_1, \text{pre}(c_2, Q)) \\
\text{pre}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2, Q) & = \texttt{(}b \texttt{ --> pre}(c_1, Q)\texttt{) \&\& (not } b \texttt{ --> pre}(c_2, Q)\texttt{)} \\
\text{pre}(\texttt{while (}b\texttt{) \{}I\texttt{\} do } c, Q) & = I
\end{array}$$

wobei b_1 `-->` b_2 syntaktischer Zucker für `not` b_1 `||` b_2 ist.

Der Verifikationsbedingungsgenerator generiert als syntaktische Formeln die Implikationen, die bei einer Ableitung von $\vdash \{\, \text{pre}(c, Q)\,\}\, \text{strip}(c)\, \{\, Q\,\}$ in den Schritten mit der Regel $\textsc{Cons}_\text{P}$ auftreten.

$$\begin{array}{ll}
\text{vc}(\texttt{skip}, Q) & = \texttt{true} \\
\text{vc}(x \texttt{ := } a, Q) & = \texttt{true} \\
\text{vc}(c_1\texttt{; } c_2, Q) & = \text{vc}(c_1, \text{pre}(c_2, Q))\texttt{ \&\& vc}(c_2, Q) \\
\text{vc}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2,) & = \text{vc}(c_1, Q)\texttt{ \&\& vc}(c_2, Q) \\
\text{vc}(\texttt{while (}b\texttt{) \{}I\texttt{\} do } c,) & = \texttt{(}\underbrace{b \texttt{ \&\& } I \texttt{ --> pre}(c, I)}_{\text{Schleifeninvariante}}\texttt{) \&\& (}\underbrace{\texttt{not } b \texttt{ \&\& } I \texttt{ --> } Q}_{\text{Schleifenende}}\texttt{) \&\& vc}(c, I)
\end{array}$$

Beispiel 43. Sei

$$w = \texttt{while (}\underbrace{\texttt{not (i == n)}}_{=b}\texttt{) \{}\underbrace{\texttt{2 * x == i * (i + 1)}}_{=I}\texttt{\} do (}\underbrace{\texttt{x := x + i; i := i + 1}}_{=c}\texttt{)}$$

$$Q = \texttt{2 * x == n * (n + 1)}$$

Dann gilt:

$$\begin{aligned}
\mathrm{pre}(c, I) &= I[\texttt{i} \mapsto \texttt{i + 1}, \texttt{x} \mapsto \texttt{x + i}] = \texttt{2 * (x + i) == (i + 1) * ((i + 1) + 1)} \\
\mathrm{pre}(w, Q) &= I \\
\mathrm{vc}(c, I) &= \texttt{true \&\& true} \\
\mathrm{vc}(w, Q) &= \texttt{(not (i == n) \&\& } I \texttt{ --> } \mathrm{pre}(c, I)\texttt{) \&\& (not not (i == n) \&\& } I \texttt{ --> } Q\texttt{) \&\& } \mathrm{vc}(c, I)
\end{aligned}$$

Theorem 56 (Korrektheit des Verifikationsbedingungsgenerators).
Wenn $\mathrm{vc}(c, Q)$ gilt, dann $\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(c, Q)]\!] \,\}\, \mathrm{strip}(c)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$.

Beweis. Induktion über c (Q beliebig):

- Fall `skip`: Wegen $\mathrm{pre}(\texttt{skip}, Q) = Q$ und $\mathrm{strip}(\texttt{skip}) = \texttt{skip}$ gilt nach Regel $\textsc{Skip}_\mathrm{P}$:

$$\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(\texttt{skip}, Q)]\!] \,\}\, \mathrm{strip}(\texttt{skip})\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$$

- Fall x `:=` a: Wegen $\mathrm{pre}(x \texttt{ := } a, Q) = Q[x \mapsto a]$ und $\mathcal{I}\,[\![Q[x \mapsto a]]\!] = \mathcal{I}\,[\![Q]\!]\,[x \mapsto \mathcal{A}\,[\![a]\!]]$ und $\mathrm{strip}(x \texttt{ := } a) = x \texttt{ := } a$ folgt die Behauptung $\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(x \texttt{ := } a, Q)]\!] \,\}\, \mathrm{strip}(x \texttt{ := } a)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$ nach Regel $\textsc{Ass}_\mathrm{P}$.

- Fall c_1`;` c_2: Induktionsannahmen (für beliebige Q):
  Wenn $\mathrm{vc}(c_1, Q)$ gilt, dann $\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(c_1, Q)]\!] \,\}\, \mathrm{strip}(c_1)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$.
  Wenn $\mathrm{vc}(c_2, Q)$ gilt, dann $\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(c_2, Q)]\!] \,\}\, \mathrm{strip}(c_2)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$.

  Sei Q beliebig, so dass $\mathrm{vc}(c_1 \texttt{;}\ c_2, Q)$ gelte. Zu zeigen: $\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(c_1 \texttt{;}\ c_2, Q)]\!] \,\}\, \mathrm{strip}(c_1 \texttt{;}\ c_2)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$.

  Wegen $\mathrm{vc}(c_1 \texttt{;}\ c_2, Q) = \mathrm{vc}(c_1, \mathrm{pre}(c_2, Q))$ `&&` $\mathrm{vc}(c_2, Q)$ gelten auch $\mathrm{vc}(c_1, \mathrm{pre}(c_2, Q))$ und $\mathrm{vc}(c_2, Q)$ und damit nach Induktionsannahme

  $$\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(c_1, \mathrm{pre}(c_2, Q))]\!] \,\}\, \mathrm{strip}(c_1)\, \{\, \mathcal{I}\,[\![\mathrm{pre}(c_2, Q)]\!] \,\} \quad \text{und} \quad \vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(c_2, Q)]\!] \,\}\, \mathrm{strip}(c_2)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$$

  Daraus folgt nach Regel $\textsc{Seq}_\mathrm{P}$ die Behauptung, da $\mathrm{pre}(c_1 \texttt{;}\ c_2, Q) = \mathrm{pre}(c_1, \mathrm{pre}(c_2, Q))$.

- Fall `if (`b`) then` c_1 `else` c_2: Induktionsannahmen (für beliebige Q):
  Wenn $\mathrm{vc}(c_1, Q)$ gilt, dann $\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(c_1, Q)]\!] \,\}\, \mathrm{strip}(c_1)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$.
  Wenn $\mathrm{vc}(c_2, Q)$ gilt, dann $\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(c_2, Q)]\!] \,\}\, \mathrm{strip}(c_2)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$.

  Sei Q beliebig, so dass $\mathrm{vc}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2, Q)$ gelte.
  Zu zeigen: $\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2, Q)]\!] \,\}\, \mathrm{strip}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$.
  Nach Regel $\textsc{If}_\mathrm{P}$ und Definition von strip genügt es, folgende zwei Ableitungen zu zeigen:

  $$\begin{aligned}
  &\{\, \lambda\sigma.\ \mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![\mathrm{pre}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2, Q)]\!]\,\sigma \,\}\, \mathrm{strip}(c_1)\, \{\, \mathcal{I}\,[\![Q]\!] \,\} \\
  &\{\, \lambda\sigma.\ \neg\, \mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![\mathrm{pre}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2, Q)]\!]\,\sigma \,\}\, \mathrm{strip}(c_2)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}
  \end{aligned}$$

  Wegen $\mathrm{vc}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2, Q) = \mathrm{vc}(c_1, Q)$ `&&` $\mathrm{vc}(c_2, Q)$ gelten auch $\mathrm{vc}(c_1, Q)$ und $\mathrm{vc}(c_2, Q)$, womit die Induktionsannahmen (für Q) anwendbar sind. Außerdem gilt für die Vorbedingungen:

  $$\begin{aligned}
  &\mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![\mathrm{pre}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2, Q)]\!]\,\sigma \\
  &\quad = \mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![\texttt{(}b \texttt{ --> } \mathrm{pre}(c_1, Q)\texttt{) \&\& (not } b \texttt{ --> } \mathrm{pre}(c_2, Q)\texttt{)}]\!]\,\sigma \\
  &\quad = \mathcal{B}\,[\![b]\!]\,\sigma \wedge (\mathcal{B}\,[\![b]\!]\,\sigma \implies \mathcal{I}\,[\![\mathrm{pre}(c_1, Q)]\!]\,\sigma) \wedge (\neg\, \mathcal{B}\,[\![b]\!]\,\sigma \implies \mathcal{I}\,[\![\mathrm{pre}(c_2, Q)]\!]\,\sigma) \\
  &\quad = \mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![\mathrm{pre}(c_1, Q)]\!]\,\sigma \implies \mathcal{I}\,[\![\mathrm{pre}(c_1, Q)]\!]\,\sigma
  \end{aligned}$$

  und entsprechend

  $$\neg\, \mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![\mathrm{pre}(\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2, Q)]\!] = \neg\, \mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![\mathrm{pre}(c_2, Q)]\!]\,\sigma \implies \mathcal{I}\,[\![\mathrm{pre}(c_2, Q)]\!]\,\sigma$$

  Damit folgen die zu zeigenden Zusicherungen aus den Induktionsannahmen jeweils mit der Regel $\textsc{Cons}_\mathrm{P}$.

- Fall `while (`b`) {`I`} do` c: Induktionsannahme (für beliebiges Q):
  Wenn $\mathrm{vc}(c, Q)$, dann $\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(c, Q)]\!] \,\}\, \mathrm{strip}(c)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$.

  Sei Q beliebig, so dass $\mathrm{vc}(\texttt{while (}b\texttt{) \{}I\texttt{\} do }c, Q)$ gelte.
  Zu zeigen: $\vdash \{\, \mathcal{I}\,[\![\mathrm{pre}(\texttt{while (}b\texttt{) \{}I\texttt{\} do }c, Q)]\!] \,\}\, \mathrm{strip}(\texttt{while (}b\texttt{) \{}I\texttt{\} do }c)\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$.

  Da $\mathrm{vc}(\texttt{while (}b\texttt{) \{}I\texttt{\} do }c, Q)$ gilt, gelten auch $\mathrm{vc}(c, I)$ sowie $\mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![I]\!]\,\sigma \implies \mathcal{I}\,[\![\mathrm{pre}(c, I)]\!]\,\sigma$ und $\neg\, \mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![I]\!]\,\sigma \implies \mathcal{I}\,[\![Q]\!]\,\sigma$ für alle σ.

  Dies lässt wie folgt zusammenfügen:

$$
\begin{array}{ll}
\mathcal{I}\,[\![\mathrm{pre}(\texttt{while (}b\texttt{) \{}I\texttt{\} do }c, Q)]\!] = \mathcal{I}\,[\![I]\!] & \\
\qquad \texttt{while (b) do} & \textsc{While}_{\mathrm{P}} \\
\lambda\sigma.\ \mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![I]\!]\,\sigma & \\
\qquad\qquad \implies & \textsc{Cons}_{\mathrm{P}} \\
\mathcal{I}\,[\![\mathrm{pre}(c, I)]\!] & \\
\qquad\qquad \mathrm{strip}(c) & \text{Induktionsannahme} \\
\mathcal{I}\,[\![I]\!] & \\
\lambda\sigma.\ \neg\, \mathcal{B}\,[\![b]\!]\,\sigma \wedge \mathcal{I}\,[\![I]\!]\,\sigma & \\
\qquad\qquad \implies & \textsc{Cons}_{\mathrm{P}} \\
\mathcal{I}\,[\![Q]\!] &
\end{array}
$$

□

Korollar 57. Wenn $\mathrm{vc}(c, Q)$ und P `-->` $\mathrm{pre}(c, Q)$ gelten, dann gilt $\vdash \{\, P \,\}\, \mathrm{strip}(c)\, \{\, Q \,\}$.

Beispiel 44 (Fortsetzung von Bsp. 43). Sei $P =$ `x == 0 && i == 0`. Dann gelten P `-->` I und $\mathrm{vc}(w, Q)$:

$$
\begin{aligned}
\mathcal{B}\,[\![P \texttt{ --> } I]\!]\,\sigma &= (\mathtt{x} = 0 \wedge \mathtt{i} = 0 \implies 2 \cdot \mathtt{x} = \mathtt{i} \cdot (\mathtt{i} + 1)) = \mathbf{tt} \\
\mathcal{B}\,[\![\mathrm{vc}(w, Q)]\!]\,\sigma &= (\mathtt{i} \neq \mathtt{n} \wedge 2 \cdot \mathtt{x} = \mathtt{i} \cdot (\mathtt{i} + 1) \implies 2 \cdot (\mathtt{x} + \mathtt{i} + 1) = (\mathtt{i} + 1) \cdot (\mathtt{i} + 1 + 1)) \wedge \\
&\qquad (\mathtt{i} = \mathtt{n} \wedge 2 \cdot \mathtt{x} = \mathtt{i} \cdot (\mathtt{i} + 1) \implies 2 \cdot \mathtt{x} = \mathtt{n} \cdot (\mathtt{n} + 1)) \wedge \mathbf{tt} \wedge \mathbf{tt} \\
&= \mathbf{tt}
\end{aligned}
$$

Somit gilt auch $\vdash \{\, P \,\}\, w\, \{\, Q \,\}$.

Korrektheit eines Verifikationsbedingungsgenerators (Thm. 56) sagt noch nichts darüber aus, ob dieser auch verwendbar ist. Die Verifikationsbedingung `false` wäre auch für alle Programme korrekt, ebenso wie die berechnete Vorbedingung `false`, aber diese beiden wären für die Verifikation von Programmen unbrauchbar. Wichtig ist deswegen die Umkehrung: Vollständigkeit.

Lemma 58 (Monotonie von $\mathrm{pre}(c, \_)$ **und** $\mathrm{vc}(c, \_)$**).**
Wenn $\mathcal{I}\,[\![P]\!] \implies \mathcal{I}\,[\![Q]\!]$, dann auch $\mathcal{I}\,[\![\mathrm{pre}(c, P)]\!] \implies \mathcal{I}\,[\![\mathrm{pre}(c, Q)]\!]$ und $\mathcal{I}\,[\![\mathrm{vc}(c, P)]\!] \implies \mathcal{I}\,[\![\mathrm{vc}(c, Q)]\!]$.

Beweis. Induktion über c (P und Q beliebig) und Ausrechnen. Wesentlich ist, dass P in $\mathrm{pre}(c, P)$ und $\mathrm{vc}(c, P)$ nur rechts der Implikationen auftritt. □

Theorem 59. Wenn $\vdash \{\, \mathcal{I}\,[\![P]\!] \,\}\, c\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$, so dass alle Prädikate im Ableitungsbaum als Formeln darstellbar sind, dann gibt es ein annotiertes Programm c' mit $c = \mathrm{strip}(c')$, sodass $\mathrm{vc}(c', Q)$ und P `-->` $\mathrm{pre}(c', Q)$ gelten.

Beweis. Induktion über $\vdash \{\, \mathcal{I}\,[\![P]\!] \,\}\, c\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$:

- Fälle $\textsc{Skip}_{\mathrm{P}}$, $\textsc{Ass}_{\mathrm{P}}$: Trivial bei Wahl $c' = c$.
- Fall $\textsc{Seq}_{\mathrm{P}}$: Induktionsannahmen: $\vdash \{\, \mathcal{I}\,[\![P]\!] \,\}\, c_1\, \{\, \mathcal{I}\,[\![Q]\!] \,\}$ und $\vdash \{\, \mathcal{I}\,[\![Q]\!] \,\}\, c_2\, \{\, \mathcal{I}\,[\![R]\!] \,\}$ und es gibt c_1' und c_2' mit $c_1 = \mathrm{strip}(c_1')$ und $c_2 = \mathrm{strip}(c_2')$, sodass $\mathrm{vc}(c_1', Q)$, P `-->` $\mathrm{pre}(c_1', Q)$, $\mathrm{vc}(c_2', R)$ und Q `-->` $\mathrm{pre}(c_2', R)$ gelten.

Wähle $c' = c'_1 \texttt{;}\ c'_2$. Da Q `-->` $\mathrm{pre}(c'_2, R)$ gilt, folgt aus den Induktionsannahmen mit der Monotonie (Lem. 58), dass auch $\mathrm{vc}(c', R) = \mathrm{vc}(c'_1, \mathrm{pre}(c'_2, R))$ und P `-->` $\mathrm{pre}(c', R) = P$ `-->` $\mathrm{pre}(c'_1, \mathrm{pre}(c'_2, R))$ gelten.

- Fall IfP: Induktionsannahmen: $\vdash \{\, \mathcal{I}\,[\![b \texttt{ \&\& } P]\!]\,\}\, c_1\, \{\, \mathcal{I}\,[\![Q]\!]\,\}$ und $\vdash \{\, \mathcal{I}\,[\![\texttt{not } b \texttt{ \&\& } P]\!]\,\}\, c_2\, \{\, \mathcal{I}\,[\![Q]\!]\,\}$ und es gibt c'_1 und c'_2 mit $c_1 = \mathrm{strip}(c'_1)$ und $c_2 = \mathrm{strip}(c'_2)$, sodass $\mathrm{vc}(c'_1, Q)$, b `&&` P `-->` $\mathrm{pre}(c'_1, Q)$, $\mathrm{vc}(c'_2, Q)$ und `not` b `&&` P `-->` $\mathrm{pre}(c'_2, Q)$ gelten.

  Wähle $c' =$ `if (`b`) then` c'_1 `else` c'_2. Dann gelten offensichtlich

  $$\mathrm{vc}(c', Q) = \mathrm{vc}(c'_1, Q) \texttt{ \&\& } \mathrm{vc}(c'_2, Q)$$

  und

  $$P \texttt{ --> } \mathrm{pre}(c', Q) = P \texttt{ --> ((}b \texttt{ --> } \mathrm{pre}(c'_1, Q)\texttt{) \&\& (not } b \texttt{ --> } \mathrm{pre}(c'_2, Q)\texttt{))}$$

- Fall WhileP: Induktionsannahmen: $\vdash \{\, \mathcal{I}\,[\![b \texttt{ \&\& } I]\!]\,\}\, c\, \{\, \mathcal{I}\,[\![I]\!]\,\}$ und es gibt ein c^* mit $c = \mathrm{strip}(c^*)$, sodass $\mathrm{vc}(c^*, I)$ und b `&&` I `-->` $\mathrm{pre}(c^*, I)$ gelten.

  Wähle $c' =$ `while (`b`) {`I`} do` c^*. Dann gelten offensichtlich

  $$\mathrm{vc}(c', \texttt{not } b \texttt{ \&\& } I) = \texttt{(}b \texttt{ \&\& } I \texttt{ --> } \mathrm{pre}(c^*, I)\texttt{) \&\& (not } b \texttt{ \&\& } I \texttt{ --> not } b \texttt{ \&\& } I\texttt{) \&\& } \mathrm{vc}(c^*, I)$$

  und

  $$I \texttt{ --> } \mathrm{pre}(c', \texttt{not } b \texttt{ \&\& } I) = I \texttt{ --> } I$$

- Fall ConsP: Induktionsannahme: $\vdash \{\, \mathcal{I}\,[\![P']\!]\,\}\, c\, \{\, \mathcal{I}\,[\![Q']\!]\,\}$ und es gibt ein c' mit $c = \mathrm{strip}(c')$, sodass $\mathrm{vc}(c', Q')$ und P' `-->` $\mathrm{pre}(c', Q')$ gelten. Außerdem $\mathcal{I}\,[\![P]\!] \implies \mathcal{I}\,[\![P']\!]$ und $\mathcal{I}\,[\![Q']\!] \implies \mathcal{I}\,[\![Q]\!]$.

  Wegen der Monotonie (Lem. 58) gelten auch $\mathrm{vc}(c', Q)$ und P' `-->` $\mathrm{pre}(c', Q)$. Da P `-->` P', gilt auch P `-->` $\mathrm{pre}(c', Q)$. □

In Thm. 59 ist ganz wesentlich, dass alle Prädikate des Ableitungsbaums in der Formelsprache ausdrückbar sind. Ansonsten gibt es Fälle (Bsp. 42), bei denen zwar Vor- und Nachbedingung als Formel ausgedrückt werden können, aber nicht die benötigten Zwischenbedingungen. Wären von Anfang an nur Formeln als Vor- und Nachbedingung zugelassen, gäbe es dieses Problem nicht. Bei praktischen Anwendungen von Verifikationsbedingungsgeneratoren definiert man üblicherweise das Hoare-Kalkül so, dass Vor- und Nachbedingungen syntaktische Formeln sind.

Korollar 60. Sei die Formelsprache expressiv. Wenn $\vdash \{\, \mathcal{I}\,[\![P]\!]\,\}\, c\, \{\, \mathcal{I}\,[\![Q]\!]\,\}$, dann gibt es ein c' mit $c = \mathrm{strip}(c')$, sodass $\mathrm{vc}(c', Q)$ und P `-->` $\mathrm{pre}(c', Q)$ gelten.

Beweis. Aus $\vdash \{\, \mathcal{I}\,[\![P]\!]\,\}\, c\, \{\, \mathcal{I}\,[\![Q]\!]\,\}$ gilt nach Thm. 49 (Korrektheit), dass $\vDash \{\, \mathcal{I}\,[\![P]\!]\,\}\, c\, \{\, \mathcal{I}\,[\![Q]\!]\,\}$. Nach dem Beweis der Vollständigkeit (Thm. 52) gibt es eine Ableitung $\vdash \{\, \mathcal{I}\,[\![P]\!]\,\}\, c\, \{\, \mathcal{I}\,[\![Q]\!]\,\}$, in der nur schwächste freie Vorbedingungen vorkommen, für die es nach Annahme Formeln gibt. Damit ist Thm. 59 anwendbar.

Der Umweg über Korrektheit und Vollständigkeit ist notwendig, weil der Ableitungsbaum von $\vdash \{\, \mathcal{I}\,[\![P]\!]\,\}\, c\, \{\, \mathcal{I}\,[\![Q]\!]\,\}$ Prädikate enthalten kann, die nicht als Formel darstellbar sind. Expressivität fordert schließlich nur, dass die schwächsten freien Vorbedingungen ausdrückbar sind. □