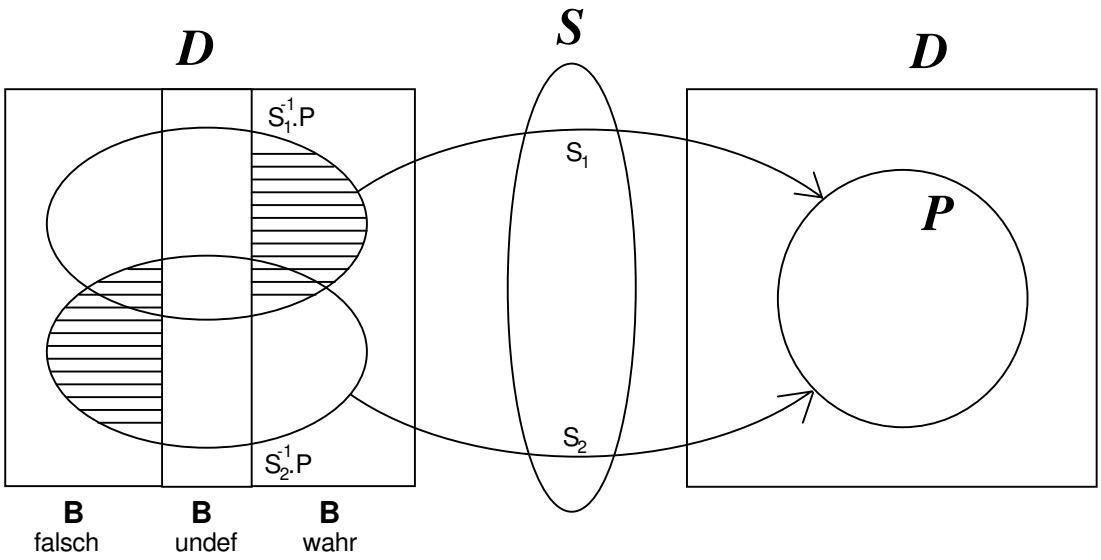


Beweisregel IF1 umf: Sei $S = \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ endif}$. Ist $\{V \text{ und } B\}S_1\{P\}$ umfassend und $\{V \text{ und nicht } B\}S_2\{P\}$ umfassend, so ist $\{V\}S\{P\}$ ebenfalls umfassend.

Beweis: $\{V \text{ und } B\}S_1\{P\}$ umf. und $\{V \text{ und nicht } B\}S_2\{P\}$ umf

$\Leftrightarrow \quad B \cap V \cap S_1^{-1}.D = S_1^{-1}.P$ und $\overline{B} \cap V \cap S_2^{-1}.D = S_2^{-1}.P$

$\Leftrightarrow \quad \left(B \cap V \cap S_1^{-1}.D\right) \cup \left(\overline{B} \cap V \cap S_2^{-1}.D\right) = S_1^{-1}.P \cup S_2^{-1}.P$

$\Leftrightarrow \quad V \cap \left(B \cap S_1^{-1}.D \cup \overline{B} \cap S_2^{-1}.D\right) = S_1^{-1}.P \cup S_2^{-1}.P$

$\Leftrightarrow \quad V \cap S^{-1}.D = S^{-1}.P$

$\Leftrightarrow \quad \{V\}S\{P\}$ umfassend.

Beweisregel IF2 umf: Sei $S = \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ endif}$. Falls $\{V_1\}S_1\{P\}$ umfassend und $\{V_2\}S_2\{P\}$ umfassend, dann ist $\{V_1 \text{ und } B \text{ oder } V_2 \text{ und nicht } B\}S\{P\}$ ebenfalls umfassend.

Beweis: $\{V_1\}S_1\{P\}$ umf. und $\{V_2\}S_2\{P\}$ umf.

$\Leftrightarrow \quad S_1^{-1}.P \subseteq V_1$ und $S_2^{-1}.P \subseteq V_2$

$\Leftrightarrow \quad B \cap S_1^{-1}.P \subseteq B \cap V_1$ und $\overline{B} \cap S_2^{-1}.P \subseteq \overline{B} \cap V_2$

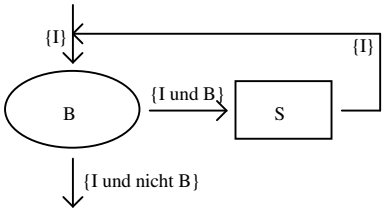
$\Leftrightarrow \quad B \cap S_1^{-1}.P \cup \overline{B} \cap S_2^{-1}.P \subseteq B \cap V_1 \cup \overline{B} \cap V_2$

$\Leftrightarrow \quad S^{-1}.P \subseteq V$

$\Leftrightarrow \quad \{V_1 \text{ und } B \text{ oder } V_2 \text{ und nicht } B\}$ umfassend.

Korrektheitsbeweise der while-Anweisung, Schleifeninvarianten

Beweisregel W1: Sei $W = \text{while } B \text{ do } S \text{ endwhile}$. Falls $\{I \text{ und } B\}S\{I\}$, dann gilt unabhängig von I $\{I\}W\{I \text{ und nicht } B\}$. Beweis anschaulich durch Grafik:



Solange I und B gilt, wird die Schleife fortgesetzt. Erst wenn B falsch wird, wird die Schleife verlassen. Ausserhalb der Schleife muß nun I und nicht B gelten.

Beweisregel W2: Falls $\{V\}init\{I\}$ und $\{I \text{ und } B\}S\{I\}$ und I und nicht $B \Rightarrow P$, so gilt
$\{V\}(init;W)\{P\}$.

Lemma WL2: Für alle $n \in N_0$ und $d \in D$ gilt, wie bereits auf Seite 5 gezeigt:
$B.(S^n.d) = falsch$ und $\bigwedge_{j=0}^{n-1} B.(S^j.d) = wahr \quad \Rightarrow \quad W.d = S^n.d$

Lemma WL3: Falls ① $n \in N_0$ und ② $\{I \text{ und } B\}S\{I\}$
und ③ $d \in I$ und ④ $d \in (S^n)^{-1}.D$
und ⑤ $\bigwedge_{j=0}^{n-1} B.(S^j.d) = wahr$

dann gilt $S^n.d \in I$.

Beweis durch Induktion über n:

Verankerung: Für $n = 0$ ist $S^0.d = d \overset{(3)}{\in} I$.

Schritt: Annahme: Lemma WL3 gilt für k, d.h.
$\{I \text{ und } B\}S\{I\}$ und $d \in I$ und $d \in (S^k)^{-1}.D$ und $\bigwedge_{j=0}^{k-1} B.(S^j.d) = wahr \quad \Rightarrow \quad S^k.d \in I$

Zu zeigen ist nun
$\{I \text{ und } B\}S\{I\}$ und $d \in I$ und $d \in (S^{k+1})^{-1}.D$ und $\bigwedge_{j=0}^{k} B.(S^j.d) = wahr \quad \Rightarrow \quad S^{k+1}.d \in I$

Es ist $S^{k+1}.d \in I$

$\Leftrightarrow \quad S.(S^k.d) \in I$

$\overset{Vorbed}{\Leftarrow} \quad S^k.d \in I \cap B \cap S^{-1}.D$ und $\{I \text{ und } B\}S\{I\}$

$\Leftrightarrow \quad S^k.d \in I$ und $S^k.d \in B$ und $S^k.d \in S^{-1}.D$ und $\{I \text{ und } B\}S\{I\}$

$\Leftrightarrow \quad S^k.d \in I$ und $B.(S^k.d) = wahr$ und $d \in (S^{k+1})^{-1}.D$ und $\{I \text{ und } B\}S\{I\}$

$\overset{Annahme}{\Leftarrow} \quad d \in I$ und $B.(S^k.d) = wahr$ und $\bigwedge_{j=0}^{k-1} B.(S^j.d) = wahr$
und $d \in (S^{k+1})^{-1}.D$ und $\{I \text{ und } B\}S\{I\}$

$\Leftrightarrow \quad d \in I$ und $\bigwedge_{j=0}^{k} B.(S^j.d) = wahr$ und
$d \in (S^{k+1})^{-1}.D$ und $\{I \text{ und } B\}S\{I\}$

Beweis von W1: Sei $d \in I \cap W^{-1}.D$. Zu zeigen ist $W.d \in I$ und $B.(W.d) = falsch$.
Nach Lemma WL2 gilt $W.d = S^n.d$ mit $B.(S^n.d) = falsch$ und $\bigwedge_{j=0}^{n-1} B.(S^j.d) = wahr$

Nach Lemma WL3 gilt $S^n.d \in I$.
Zusammen gilt $W.d = S^n.d \in I \cap \overline{B}$. Es folgt $W.d \in I \cap \overline{B} \quad \Leftrightarrow \quad \{I\}W\{I \text{ und nicht } B\}$

Beweis von W2: $\{V\}(init;W)\{P\}$

$\overset{(F1)}{\Leftarrow} \quad \{V\}init\{I\}$ und $\{I\}W\{P\}$

$\overset{(B1)}{\Leftarrow} \quad \{V\}init\{I\}$ und $\{I\}W\{I \text{ und nicht } B\}$ und $(I \text{ und nicht } B \Rightarrow P)$

$\overset{(W1)}{\Leftarrow} \quad \{V\}init\{I\}$ und $\{I \text{ und } B\}S\{I\}$ und $(I \text{ und nicht } B \Rightarrow P)$

Das vielzitierte *I* dieser Beweise ist die sogenannte *Schleifeninvariante*.

Beispiel für einen Korrektheitsbeweis

Wir betrachten folgendes Programmstück *S*:

```
k:=1
while k≤n und A(k)≠x do
   k:=k+1
endwhile
g := (k≤n)
```

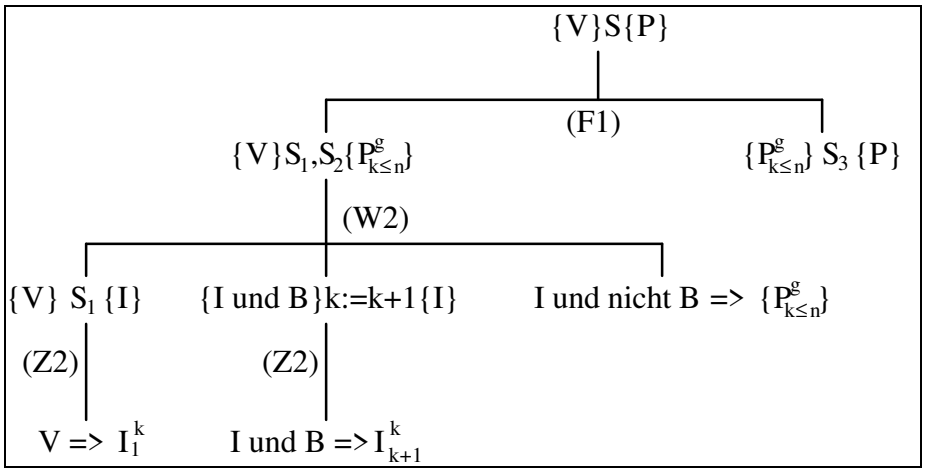
Dieses Programmstück führt eine lineare Suche aus. Im Feld A wird das erste x gesucht. Falls x gefunden wird, so terminiert die Schleife mit k≤n, und A(k)=x. Wir wollen beweisen: $\{V\}S\{P\}$ mit

$$V = \{n \in Z \wedge 0 \le n\} \text{ und}$$

$$P = \begin{Bmatrix} n,k \in Z \wedge 1 \le k \le n+1 \wedge \\ g \in B \wedge \left(\wedge_{i=1}^{k-1} A(i) \ne x\right) \end{Bmatrix} \wedge \begin{Bmatrix} k \le n \wedge A(k) = x \wedge g \\ \vee k = n+1 \wedge \overline{g} \end{Bmatrix}$$

Die Schleifenvariante ist als $I = \left\{n \in Z \wedge k \in Z \wedge 1 \le k \le n+1 \wedge \left(\wedge_{i=1}^{k-1} A(i) \ne x\right)\right\}$ gewählt.

Das Programmstück wird zunächst in drei Teilstücke S_1, S_2, S_3 zerlegt, wobei S_1 die Initialisierung, S_2 die Schleife und S_3 das Schlußstück ist. Unter Berücksichtigung der zur Verfügung stehenden Beweismethoden wird der Korrektheitsbeweis aufgesplittet:



Wir haben nun die "Blätter" zu zeigen, und beginnen von links nach rechts:

① $I_1^k \quad \Leftrightarrow \quad n \in Z \wedge 1 \in Z \wedge 1 \le 1 \le n+1 \wedge \left(\wedge_{i=1}^{0} A(i) \ne x\right)$

$\Leftrightarrow \quad n \in Z \wedge 1 \le n+1$

$\Leftrightarrow \quad n \in Z \wedge 0 \le n \quad \Leftrightarrow \quad V$

② $I \wedge B \quad \Leftrightarrow \quad n \in Z \wedge k \in Z \wedge 1 \le k \le n+1 \wedge \left(\wedge_{i=1}^{k-1} A(i) \ne x\right) \wedge k \le n \wedge A(k) \ne x$

$\Leftrightarrow \quad n \in Z \wedge k \in Z \wedge 1 \le k \le n \wedge \left(\wedge_{i=1}^{k} A(i) \ne x\right)$

$\Rightarrow \quad n \in Z \wedge k \in Z \wedge 0 \le k \le n \wedge \left(\wedge_{i=1}^{k} A(i) \ne x\right)$

$\Leftrightarrow \quad n \in Z \wedge k \in Z \wedge 1 \le k+1 \le n+1 \wedge \left(\wedge_{i=1}^{k} A(i) \ne x\right)$

$\Leftrightarrow \quad I_{k+1}^k$

③ $I \wedge \overline{B} \quad \Leftrightarrow \quad n \in Z \wedge k \in Z \wedge 1 \le k \le n+1 \wedge \left(\wedge_{i=1}^{k-1} A(i) \ne x\right) \wedge \left(k > n \vee A(k) = x\right)$

$\Leftrightarrow \quad n \in Z \wedge k \in Z \wedge 1 \le k \le n+1 \wedge \left(\wedge_{i=1}^{k-1} A(i) \ne x\right) \wedge \left(k > n \vee k \le n \wedge A(k) = x\right)$

$\Leftrightarrow \quad \left\{\begin{matrix} n \in Z \wedge k \in Z \wedge 1 \le k \le n+1 \\ \wedge_{i=1}^{k-1} A(i) \ne x \end{matrix}\right\} \wedge \left\{\begin{matrix} 1 \le k \le n \wedge A(k) = x \\ \vee k = n+1 \wedge k > n \end{matrix}\right\}$

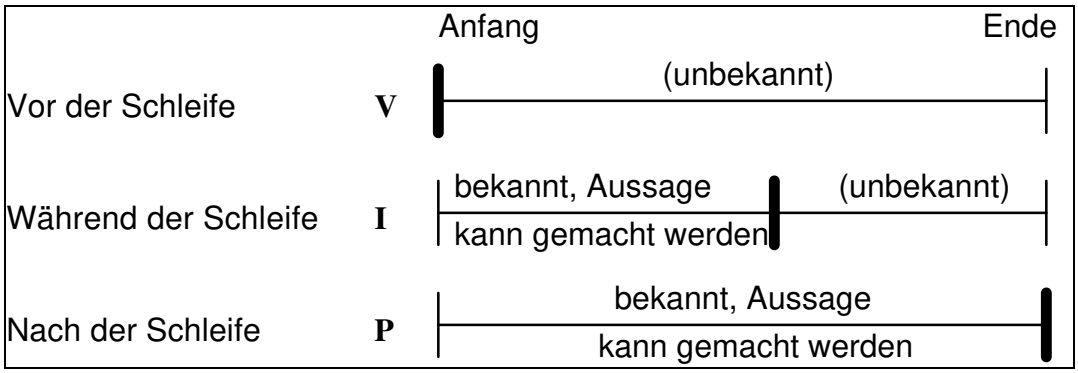
$\Leftrightarrow \quad \left\{\begin{matrix} n \in Z \wedge k \in Z \wedge 1 \le k \le n+1 \\ (k \le n) \in B \wedge \left(\wedge_{i=1}^{k-1} A(i) \ne x\right) \end{matrix}\right\} \wedge \left\{\begin{matrix} k \le n \wedge k \le n \wedge A(k) = x \\ \vee k = n+1 \wedge k > n \end{matrix}\right\}$

$\Leftrightarrow \quad P_{k \le n}^{g}$

④ $\left\{P_{k \le n}^{g}\right\} S_3 = \left(g := (k \le n)\right) \{P\}$ folgt automatisch mit (Z2).

Damit ist $\{V\} S \{P\}$ bewiesen.

Eine Schleifeninvariante sollte immer eine Aussage über die bereits betrachteten Werte einschließen. Vorbedingungen und Postbedingungen sind im Prinzip Spezialfälle der Schleifeninvarianten. Dies läßt sich an einer Grafik verdeutlichen:



Im obigen Beispiel ist z.B.:

$I\big|_{k=1} \quad \Leftrightarrow \quad \left\{n \in Z \wedge 1 \in Z \wedge 1 \le 1 \le n+1 \wedge \left(\wedge_{i=1}^{0} A(i) \ne x\right)\right\}$

$\Leftrightarrow \quad \{n \in Z \wedge 1 \le n+1\}$

$\Leftrightarrow \quad \{n \in Z \wedge 0 \le n\} \qquad \Leftrightarrow \qquad V$

Aus dieser Idee läßt sich auch in manchen Fällen aus Vorbedingung, Nachbedingung und Schleifeninvariante der Programmcode erschließen, gemäß den Fragestellungen: "Wie muß S beschaffen sein, damit $\{V\} S \{I\}$ gilt?" oder "Wie muß die Bedingung B beschaffen sein, damit $I \wedge \overline{B} \Rightarrow P$ gilt?

Was ist ein mathematischer Satz?

In allgemeiner Form ist ein Satz von der Art: "Falls *Hypothese*, dann gilt *These*". Eine Hypothese wird möglichst schwach formuliert, die These so stark wie möglich. Bei der Korrektheitsbeweisführung gibt es mehrere Arten von Sätzen, Sätze zur partiellen und totalen Korrektheit sowie Aussagen bezüglich Werten von Programmvariablen und Aussagen bezüglich der Struktur von Datenumgebungen.

Zu dem Beispiel des Suchalgorithmus von Seite 20 lassen sich folgende Sätze formulieren:

Satz 1: $\{V\}S\{P\}$

Satz 2:
$$\left\{\begin{array}{c} \text{Menge}(x) \neq \varnothing \wedge \left(\wedge_{i=1}^{n} \text{Menge}(A(i)) \neq \varnothing\right) \\ V \wedge \text{Menge}(k) \supseteq \{1 \ldots n+1\} \wedge \text{Menge}(g) \supseteq B \\ \left(\wedge_{i=1}^{n} \text{Menge}(x) = \text{Menge}(A(i))\right) \end{array}\right\}$$

ist eine strikte Vorbedingung von *P* bezüglich *S*.

Satz 3: $S.d \overset{strukt}{=} d \quad \forall d \in S^{-1}.D$

Satz 4: $S.d = d$ mit Ausnahme der Werte der Programmvariablen *k* und *g* $\forall d \in S^{-1}.D$

Konstruktion einer Schleife

Für die Konstruktion einer Schleife und den Beweis ihrer Korrektheit, falls Vorbedingung und Nachbedingung bereits gegeben sind, sind 5 Schritte notwendig:

① Schleifeninvariante *I* festlegen
② Initialisierung so wählen, daß $\{V\}init\{I\}$ gilt.
③ Programmstück *S* innerhalb der Schleife und die Bedingung *B* wählen, so daß $\{I \wedge B\}S\{I\}$ und
④ $I \wedge \overline{B} \Rightarrow P$ gelten
⑤ *S* muß die Berechnung in Richtung *P* vorantreiben, d.h. die Schleifenterminierung muß gesichert sein.

Durch Abarbeitung der Punkte 2, 3 und 4 ist mit dem Satz (W2) automatisch die Korrektheit der Schleife sichergestellt. Die Konstruktion der Schleife beginnt mit dem ersten Punkt, der Überlegung der geeigneten Schleifeninvariante (gemäß den auf Seite 17 angestellten Überlegungen). Als nächstes folgen die Initialisierung (festgelegt durch die Vorbedingung und die Invariante), und die Bedingung wird gemäß Punkt ④ so gewählt, daß bei Terminierung der Schleife die Postbedingung erreicht ist.

Als letzer Schritt wird nun gemäß den Punkten ③ und ⑤ das Programmstück gewählt.

Bestimmung der Schleifenbedingung

Punkt ④ des Schleifenbeweises läßt sich umformen:

$$\begin{array}{lll} I \wedge \overline{B} \Rightarrow P & \Leftrightarrow & \overline{I} \vee B \vee P \\ & \Leftrightarrow & I \Rightarrow \left(\overline{B} \Rightarrow P\right) \\ & \Leftrightarrow & I \Rightarrow \left(\overline{P} \Rightarrow B\right) \end{array}$$

Wir betrachten als Beispiel die Summierung aller Elemente eines Feldes (aus den Übungen)

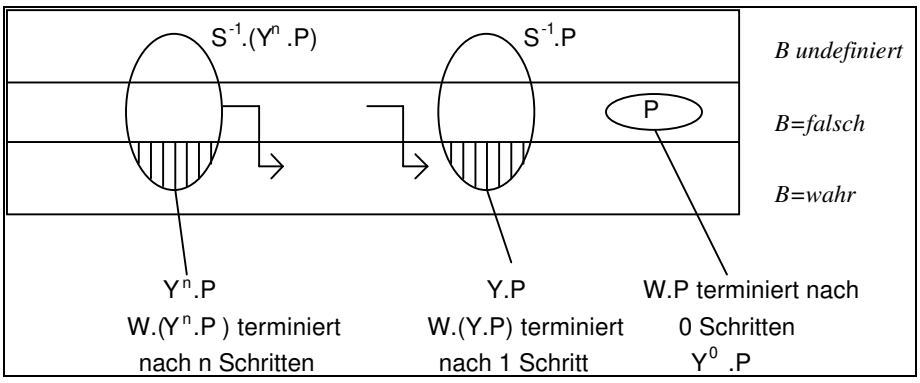
```
declare (sum, R, 0); declare (i,Z,0);
while i<n
  i   := i+1;
  sum := sum + x(i);
endwhile
release i
```

mit der aus den Übungen bekannten Vor- und Postbedingungen $V = \{n \in Z \wedge n \geq 0\}$ und $P = \left\{sum = \sum_{i=1}^{n} x(i)\right\}$ und der Schleifeninvariante $I = \left\{n, i \in Z \wedge 0 \leq i \leq n \wedge sum = \sum_{j=1}^{i} x(j)\right\}$. Die oben angegebene Umformung besagt, daß B aus $\overline{P}$ unter der Annahme folgt, daß I wahr ist; diese Annahme ist natürlich während des gesamten Schleifendurchlaufs gegeben. Es ist:

$$\begin{aligned}
\overline{P} \quad &\Leftrightarrow \quad sum \neq \sum_{j=1}^{n} x(j) && \text{(mit Annahme von } I \text{ wahr)} \\
&\Leftrightarrow \quad \sum_{j=1}^{i} x(j) \neq \sum_{j=1}^{n} x(j) \\
&\Leftrightarrow \quad i \neq n && \text{(mit Annahme von } I \text{ wahr)} \\
&\Leftrightarrow \quad i < n \quad \Leftrightarrow \quad B
\end{aligned}$$

Umfassende Vorbedingung einer while-Schleife

Die while-Schleife terminiert, sobald die Schleifenbedingung falsch wird. Es ist daher klar, daß $P \subseteq \overline{B}$ ist, d.h. $B.d = falsch \quad \forall d \in P$. Sei $W = \text{while } B \text{ do } S \text{ endwhile}$. Wir definieren nun eine Funktion Y mit $Y.M = B \cap S^{-1}.M$, d.h. Y liefert alle Datenumgebungen, in denen B gilt und auf die S definiert ist. Die Funktion Y stellt daher sicher, daß der Schleifenkern S in $W.Y$ mindestens einmal durchlaufen wird. Damit unterscheidet sich die Funktion von S^{-1}, denn in $S^{-1}.M$ darf $B = falsch$ oder auch undefiniert sein; deshalb ist im allgemeinen $W.\left(S^{-1}.M\right) \neq W.M$, es gilt jedoch $W.(Y.M) = W.M$. Grafisch läßt sich das wie folgt darstellen:



Satz: Falls $Q \in \left\{Y^n.P \middle| n \geq 0 \wedge P \subseteq \overline{B}\right\}$, dann gilt, daß $\{Q\}W\{P\}$ strikt und umfassend ist, mit anderen Worten: $Q = W^{-1}.P$.

Korollar: Sei $Q \in \left\{Z_n \middle| n \geq 0 \wedge P \subseteq \overline{B}\right\} \wedge Z_0 = P \wedge$
$\left(\forall i > 0: Z_i = B \wedge c_{i-1}\right) \wedge \left(\forall i > 0: \{c_{i-1}\} S \{Z_{i-1}\} \text{ umfassend}\right)$

Dann gilt: $\{Q\}W\{P\}$

Bemerkung: Q stellt die Terminierung der Schleife sicher.
Q ist eine mögliche Schleifeninvariante.

Das Korollar hier ist nichts anderes als der vorige Satz, allerdings mit eingesetzter formaler Definition der Funktion Y.

Systemprobleme

Betrachte noch einmal das Programm zur linearen Suche:

```
k:=1
while k≤n und A(k)≠x do
   k:=k+1
endwhile
```

Was passiert nun, wenn $n = 0$ ist? Beim Eintritt in die Schleife wird die Schleifenbedingung getestet. In der Schleifenbedingung befindet sich jetzt jedoch eine Referenz auf $A(n+1)$. Dieses Feldelement ist nicht notwendigerweise definiert.

Das Verhalten in diesem Fall ist nun Systemabhängig. Üblicherweise werden logische Ausdrücke von links nach rechts ausgewertet, bis das Ergebnis des Ausdrucks *feststeht*. Ein auf diese Weise arbeitendes System wird zunächst feststellen, daß der Ausdruck $n+1 \le n$ falsch ist, und den evtl. undefinierten zweiten Ausdruck nicht mehr auswerten, da das Ergebnis der und-Verknüpfung nicht mehr zu ändern ist.

Andere Systeme, die durchaus existieren können, werten beide Ausdrücke aus, wobei der erste falsch ist und der zweite undefiniert, weshalb diese Systeme den Gesamtausdruck zu undefiniert evaluieren.

Um dieses Problem theoretisch zu erfassen, setzen wir die Funktionen *fort*: Ist f eine Funktion $X \to Y$, so definieren wir $X' = X \oplus \{undef\}$ und $Y' = Y \oplus \{undef\}$ und setzen die Funktion f wie folgt fort:

$$f'.\chi = \begin{cases} f.\chi & \text{falls } x \in X \\ willkürlich & \text{sonst} \end{cases}$$

Beide Systeme setzen nun ihre eigenen Werte für den Wert *willkürlich* ein. System 1 setzt $\text{und}'(falsch, undef) = falsch$; System 2 setzt $\text{und}'(falsch, undef) = undef$. Es gibt drei gebräuchliche Fortsetzungen der Boole'schen Algebra. Die erste definiert das Ergebnis aller Operationen zu undefiniert, wenn ein Operand undefiniert ist. Die zweite Logik ist die MinMax-Logik, die *undef* als einen arithmetischen Wert zwischen *wahr* und *falsch* ansieht, und dann für und' die Minimumsfunktion und für oder' die Maximumsfunktion verwendet. Die dritte sinnvolle Logik ist die Kurzschlußlogik, die Parameter in einem Boole'schen Ausdruck nach der MinMax-Logik solange von links nach rechts auswertet, bis das Ergebnis feststeht. Zu den verschiedenen Logiken siehe auch Übungsblatt 9.

Zuweisungen

Wir betrachten als Beispiel die Summierung aller Elemente eines Feldes (aus den Übungen)

```
declare (sum, R, 0); declare (i,Z,0);
while i<n
  i   := i+1;
  sum := sum + x(i);
endwhile
release i
```

mit der Schleifeninvariante $I = \left\{ n, i \in Z \wedge 0 \le i \le n \wedge sum = \sum_{j=1}^{i} x(j) \right\}$ sowie der Bedingung $B = \{ i < n \}$.

Hier sollen nun zwei zentrale Themen des Schleifensatzes behandelt werden:

① Wie prüft man, ob $\{I \text{ und } B\} i := i+1; sum := sum + x(i) \{I\}$ ist?

② Was ist die umf. Vorbedingung für $\{V\} \text{declare}(sum, R, 0); \text{declare}(i, Z, 0) \{I\}$?

Nach Beweisregel (F1) gilt die Aussage der Fragestellung ①, wenn

①.1 $\{I \text{ und } B\} i := i+1 \left\{ I^{sum}_{sum+x(i)} \right\}$ und

①.2 $\left\{ I^{sum}_{sum+x(i)} \right\} sum := sum + x(i) \{I\}$

Dabei gilt ①.2 sofort nach Beweisregel (Z1). ①.1 bleibt noch zu zeigen. Diese Aussage läßt sich mit (Z2) weiter umschreiben. ①.1 gilt genau dann, wenn $I \text{ und } B \Rightarrow \left(I^{sum}_{sum+x(i)} \right)^{i}_{i+1}$. Es ist

$I^{sum}_{sum+x(i)} = n, i \in Z \wedge 0 \le i \le n \wedge sum + x(i) = \sum_{j=1}^{i} x(j)$ und daher

$$\begin{aligned}
\left(I^{sum}_{sum+x(i)} \right)^{i}_{i+1} &= n, i+1 \in Z \wedge 0 \le i+1 \le n \wedge sum + x(i+1) = \sum_{j=1}^{i+1} x(j) \\
&= n, i \in Z \wedge -1 \le i \le n-1 \wedge sum = \sum_{j=1}^{i} x(j) \\
&\Leftarrow n, i \in Z \wedge 0 \le i \le n-1 \wedge sum = \sum_{j=1}^{i} x(j) \\
&= n, i \in Z \wedge 0 \le i \le n \wedge i < n \wedge sum = \sum_{j=1}^{i} x(j) \\
&= I \text{ und } B
\end{aligned}$$

Wiederum gilt nach (F1), daß ② umfassend ist, wenn

②.1 $\{V\} \text{declare}(sum, R, 0) \left\{ I^{i}_{0} \right\}$ umfassend und

②.2 $\left\{ I^{i}_{0} \right\} \text{declare}(i, Z, 0) \{I\}$ umfassend

Gemäß (Z1) gilt sofort ②.2. Äquivalent gilt nach (Z1), daß $\left\{ \left(I^{i}_{0} \right)^{sum}_{0} \right\} \text{declare}(sum, R, 0) \left\{ I^{i}_{0} \right\}$ umfassend ist, und $\left(I^{i}_{0} \right)^{sum}_{0}$ daher eine geeignete Vorbedingung ist. Es ist

$$\begin{aligned}
\left(I^{i}_{0} \right)^{sum}_{0} &= n, 0 \in Z \wedge 0 \le 0 \le n \wedge 0 = \sum_{j=1}^{0} x(j) \\
&= n \in Z \text{ und } 0 \le n
\end{aligned}$$

Zusammenfassung

Bei der Verwendung von **declare** und **release** Anweisungen kann zum einen nach Werten von Variablen, zum anderen nach der Struktur von Datenumgebungen gefragt werden. Für erstens können für declare die Beweisregeln (Z1) und (Z2) wie oben verwendet werden. Für release gilt, daß $\{P\} \text{release } s \{P\}$, falls s in P nicht vorkommt. Für Aussagen über die Struktur der Datenumgebungen können die Definitionen der Anweisungen direkt verwendet werden.

Ein Aufruf eines Unterprogrammes mit formalen Parametern muß in einen äquivalenten Aufruf ohne formale Parameter umgeschrieben werden, bevor Beweisregeln aufgestellt werden können.