Schwächste Vorbedingung
Stärkste Nachbedingung

5.13 Definition

Sei A eine (S, Σ)-Algebra, α ein Programm und φ eine Formel über (S, Σ) und V die Menge der in α und φ auftretenden Variablen.

Die **schwächste Vorbedingung** $wlp_A(\alpha, \varphi)$ von α und φ über der Algebra A ist folgende Menge von Zuständen über A, V:

$$wlp_A(\alpha, \varphi) := \{z : \forall z'(z[[\alpha]]_A z' \Rightarrow A \models_{z'} \varphi)\}$$

Die **stärkste Nachbedingung** $spc_A(\varphi, \alpha)$ von φ und α über der Algebra A ist folgende Menge von Zuständen über A, V:

$$spc_A(\varphi, \alpha) := \{z : \exists z'(A \models_{z'} \varphi \wedge z'[[\alpha]]_A z)\}$$

Sei $Z \subseteq \mathcal{Z}(A, V)$ eine Menge von Zuständen über einer Algebra A und Variablenmenge V. Z heißt in A **definierbar**, falls es eine Formel φ mit freien Variablen in V und $Z = \{z : A \models_z \varphi\}$ gibt.

Literatur: $wlp_A(\alpha, \varphi)$ „Schwächste liberale Vorbedingung“

Eigenschaften von wlp_A und spc_A

5.14 Lemma

A sei eine Algebra, α ein Programm, φ, ψ Formeln und z, z' Zustände über der sich ergebenden Variablenmenge. Es gilt

- aus $z \in wlp_A(\alpha, \psi)$ und $z[[\alpha]]_A z'$ folgt $A \models_{z'} \psi$,
- aus $A \models \{\varphi\}\alpha\{\psi\}$ und $A \models_z \varphi$ folgt $z \in wlp_A(\alpha, \psi)$,
- aus $A \models_z \varphi$ und $z[[\alpha]]_A z'$ folgt $z' \in spc_A(\varphi, \alpha)$,
- aus $A \models \{\varphi\}\alpha\{\psi\}$ und $z' \in spc_A(\varphi, \alpha)$ folgt $A \models_{z'} \psi$,
- $wlp_A(\alpha, \psi)$ sei in A durch eine Formel $W_{\alpha,\psi}$ definierbar. Dann gilt
  $A \models \{W_{\alpha,\psi}\}\alpha\{\psi\}$ und für alle ξ folgt aus $A \models \{\xi\}\alpha\{\psi\}$ die Aussage $A \models (\xi \to W_{\alpha,\psi})$
  (d. h. sie wird von jeder anderen Vorbedingung impliziert).
- $spc_A(\varphi, \alpha)$ sei in A durch eine Formel $S_{\varphi,\alpha}$ definierbar. Dann gilt
  $A \models \{\varphi\}\alpha\{S_{\varphi,\alpha}\}$ und für alle ξ folgt aus $A \models \{\varphi\}\alpha\{\xi\}$ die Aussage $A \models (S_{\varphi,\alpha} \to \xi)$.
  (d. h. sie impliziert jede andere Nachbedingung).
- $wlp_A(X := t, \psi)$ definierbar durch $W_{X:=t,\psi} = [\psi]\{X/t\}$.
- $S_{\varphi,X:=t} = \exists Y([\varphi]\{X/Y\} \wedge X = t\{X/Y\})$.
  $(Y \notin \text{Var}(X, t, \varphi))$ (Definiert $spc_A(\varphi, X := t)$)

Beispiele

5.15 Beispiel Algebra Nat mit $- : nat \times nat \to nat$.

a) Sei $\varphi :: X > Y \qquad X := t; \quad t :: X - Y$
$[\varphi]\{X/t\} :: X - Y > Y$ offenbar

$$Nat \models \{X - Y > Y\} X := X - Y;\ \{X > Y\}$$

b) $\exists Z([\varphi]\{X/Z\} \wedge X = t\{X/Z\})$

$\exists Z(Z > Y \wedge X = Z - Y)\ \ (\underset{Nat}{\leftrightarrow} X > 0)$
Also ist
$Nat \models \{X > Y\} X := X - Y;\ \{X > 0\}$

c) $\models \{\mathsf{true}\} X := X - Y; \quad \{\exists Z(\mathsf{true} \wedge X = Z - Y)\}$
$\updownarrow$
$\{\exists Z X = Z - Y\}$

Beispiele (Forts.)

d) $\models \{X = Y\}X := X + Y;\ Y := X + Y;\ \{3X = 2Y\}$

semantisch klar!

Unter Anwendung von wlp

$\models \{3X = 2(X + Y)\}Y := X + Y;\ \{3X = 2Y\}$

$\models \{3(X + Y) = 2(X + Y + Y)\}X := X + Y;$
$\{3X = 2(X + Y)\}$

Dann ist

$\models \{3(X + Y) = 2(X + Y + Y)\}X := X + Y;$
$Y := X + Y;\ \{3X = 2Y\}.$

Es gilt:

$Nat \models X = Y \rightarrow 3(X + Y) = 2(X + Y + Y).$

Also gilt auch

$Nat \models \{X = Y\}X := X + Y;\ Y := X + Y;$
$\{3X = 2Y\}.$

Unter Anwendung von spc

$\models \{X = Y\}X := X+Y;\ \{\exists Z([X = Y]\{X/Z\} \wedge X = (X + Y)\{X/Z\})\}$ d.h.

$\models \{X = Y\}X := X+Y;\ \{\exists Z(Z = Y \wedge X = Z+Y)\}$

$\models \{X = Y\}X := X + Y;\ \{X = Y + Y)\}$

$\models \{X = Y + Y\}Y := X + Y;\ \{\exists V([X = Y + Y]\{Y/V\} \wedge Y = (X + Y)\{Y/V\}\}$ d.h. $\models \{X = Y + Y)\}Y := X + Y;\ \{\exists V(X = V + V \wedge Y = X + V)\}$

$\models_{Nat} \exists V(X = V + V \wedge Y = X + V) \rightarrow 3X = 2Y$

Beispiele (Forts.)

e) Sei N über $(nat, 0 \to nat, succ : nat \to nat)$.
$\mathbb{N}$ als Grundbereich, 0_N, $succ_N$ die üblichen.
Welche Teilmengen von $\mathbb{N}$ lassen sich durch Formeln über der Signatur von N darstellen? (Formeln mit einer freien Variablen X).
$X = succ^n(0)$ stellt $\{n\}$ dar.
$X = succ^n(X)$ stellt $\varnothing$ oder $\mathbb{N}$ $(n = 0)$ dar.
$succ^n(0) = succ^m(X)$ stellt $\underset{(m>n)}{\varnothing}$ oder $\{n - m\}$ dar.
Endliche und co-endliche (Komplement endlich)Teilmengen von $\mathbb{N}$.
$\exists Y : X = succ^n(Y)$:

f) Sei A eine Algebra. z Zustand über A und V. Wann ist z definierbar?
Hinreichende Bedingung: $V = \{X_1, \ldots, X_n : s\}$

$$z(X_1) = a_1 \ldots \quad z(X_n) = a_n \quad a_1, \ldots, a_n \in s_A$$

Angenommen es gibt Grundterme (Terme ohne Variablen) über Signatur von A mit $val_{A,\cdot}(t_i) = a_i$.
Dann definiert $(X_1 = t_1 \wedge \cdots \wedge X_n = t_n)$ den Zustand z.
Offenbar lässt sich dann jede endliche Menge von Zuständen definieren und falls die Grundbereiche der Algebra endlich sind, lässt sich jede Zustandsmenge durch eine Formel φ definieren.

Ausdrucksstarke Algebren

5.16 Definition

Sei (S, Σ) eine Signatur. Eine (S, Σ)-Algebra A heißt **ausdrucksstark** (expressiv), falls für jedes Programm α und jede Formel φ über (S, Σ) die Zustandsmenge $wlp_A(\alpha, \varphi)$ in A definierbar ist.

Bemerkung: Man kann diesen Begriff äquivalent über die Menge $spc(\varphi, \alpha)$ definieren.(Beweis: Übung macht den Meister).

5.17 Satz

Über einer ausdrucksstarken Algebra A ist $HC(A)$ vollständig.

Beweis: Es gelte $A \models \{\varphi\}\alpha\{\psi\}$, d. h. für z, z' Zustände folgt aus $A \models_z \varphi$ und $z[[\alpha]]_A z'$ stets $A \models_{z'} \psi$.

Zeige $\underset{HC(A)}{\vdash} \{\varphi\}\alpha\{\psi\}$.

Induktion über Aufbau von α: nur while Fall.

Sei $A \models \{\varphi\}$**while** B **do** β **end**; $\{\psi\}$.

Sei ξ Formel, die $wlp_A($**while** B **do** β **end**; $, \psi)$ in A definiert. Dann gilt

- $A \models (\varphi \rightarrow \xi)$ aus Definition der schwächsten Vorbedingung und $A \models \{\varphi\}$**while** B **do** β **end**; $\{\psi\}$.
  Also gilt $\underset{HC(A)}{\vdash} (\varphi \rightarrow \xi)$.

Ausdrucksstarke Algebren (Fort.)

- $A \models \{\xi \wedge B\}\beta\{\xi\}$ da:
  Sei $A \models_z (\xi \wedge B)$ und $z[[\beta]]_A z'$ für Zustände z, z'.
  Es ist $A \models_{z'} \xi$ zu zeigen.
  Zeige dazu $z' \in wlp_A(\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \beta\ \underline{\textbf{end}};\ , \psi)$.
  Sei also $z'[[\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \beta\ \underline{\textbf{end}};\ ]]_A z''$.
  Wegen $A \models_z B$ und $z[[\beta]]_A z'$, gilt auch $z[[\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \beta\ \underline{\textbf{end}};\ ]]_A z''$.
  Wegen $A \models_z \xi$ gilt auch $z \in wlp_A(\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \beta\ \underline{\textbf{end}};\ , \psi)$
  also $A \models_{z''} \psi$. Also $A \models_{z'} \xi$.
- $A \models ((\xi \wedge \neg B) \rightarrow \psi)$ ergibt sich wie folgt:
  Sei $A \models_z (\xi \wedge \neg B)$. Da die Schleife sofort abgebrochen wird, gilt $z[[\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \beta\ \underline{\textbf{end}};\ ]]_A z$. Wegen $A \models_z \xi$ kann man $z \in wlp_A(\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \beta\ \underline{\textbf{end}};\ , \psi)$ folgern.
  Mit der Definition von wlp folgt $A \models_z \psi$.

Mit der Induktionsvoraussetzung und der Schleifenregel erhält man die Ableitbarkeit in $HC(A)$.

Ohne Beweis.

- Die Algebra Nat ist ausdrucksstark.
- Die Algebra N ist nicht ausdrucksstark.
- Jede Algebra A mit endlichen Grundbereichen ist ausdrucksstark.

Beweis für Interessierte: Seite 36-39 Sp/Ha oder Loeckx/Sieber.

Erweiterungen der Programmiersprache

5.18 Bemerkung Die meisten Programmiersprachen bieten eine Vielzahl weiterer Programmkonstrukte an. Will man solche Erweiterungen der Sprache der While-Programme zulassen, so muss eine geeignete Syntax für diese Konstrukte gewählt werden und entsprechende Syntax-Regeln angegeben werden. Die denotationale Semantik bzw. die Interpretersemantik muss für diese Konstrukte erweitert werden und schließlich müssen die Regeln im Hoare-Kalkül angegeben werden um die Verifikationsaufgabe zu systematisieren.
Programmiersprachen bieten Möglichkeiten für Makros,(rekursive) Prozeduren, rekursive Programme, Sprünge usw.. Sie ermöglichen und unterstützen damit die strukturierte Programmentwicklung. Es stellt sich die Frage, ob diese Konstrukte die Berechnungsmächtigkeit erweitern oder ob die Klasse der While-berechenbaren Funktionen sich dadurch nicht verändert.

Konstrukte wie etwa „**repeat** α **until** B **end;**“ oder eine zählergesteuerte Schleife lassen sich leicht mit Hilfe der while-Schleife simulieren, d.h. sie sind eigentlich nur „syntaktischer Zucker“. Makros und nichtrekursive Prozeduren fallen auch in diese Kategorie.

An dieser Stelle sollen nur noch die rekursiven Prozeduren anhand von Beispielen erläutert werden. Für eine genauere Untersuchung siehe Sperschneider/Hammer. Prozeduren müssen vor ihrem Aufruf mit **call** P(...) deklariert werden und ihre Wirkung spezifiziert werden.
D.h man benötigt **Sonderzeichen: proc call in out**

Beispiele für Prozeduren

N mit Signatur $0 :\longrightarrow nat, succ : nat \longrightarrow nat$

β:: $Z := X;$
$Z' := 0;$
while $\neg Y = Z'$ **do**
$Z := succ(Z);$
$Z' := succ(Z');$
end;

Kopf: **proc** $ADD_1($**in** $X, Y : nat,$ **out** $Z : nat)$
Komment: $\{\text{true}\}$ **call** $ADD_1(X, Y, Z);\ \{Z = X +_{nat} Y\}$
begin
Rumpf: β
end.

γ:: $Z := 0;\ Z' := 0;$
while $\neg Y = Z'$ **do**
call $ADD_1(Z, X, X');$
$Z := X';$
$Z' := succ(Z');$
end;

Syntax des Prozeduraufrufs
call $P(t_1, \ldots, t_n, U_1, \ldots, U_m);$ wobei Typ(t_i) = Typ(X_i)
$U_1, \ldots, U_m$ kommen nicht in $t_1, \ldots, t_n$ vor: Nebenwirkungsfrei.

Beispiele für Prozeduren

proc G (**in** $X, Y : nat$, **out** $Z : nat$)
  {true} **call** $G(X, Y, Z)$; $\{X = succ(Y) \wedge Z = Y\}$
  begin
    if $\neg X = succ(Y)$ **then call** $G(X, succ(Y), Z)$;
                else $Z := Y$;
    end;
  end.

proc $PRED$ (**in** $X : nat$, **out** $Z : nat$)
  {true} **call** $PRED(X, Z)$; $\{(X = 0 \wedge Z = 0) \vee$
                $succ(Z) = X\}$
  begin
    if $X = 0$ **then** $Z := 0$; **else call** $G(X, 0, Z)$;
    end;
  end.

proc $PRED'$(**in** $X : nat$, **out** $Z : nat$)
  {true} **call** $PRED'(X, Z)$; $\{(X = 0 \wedge Z = 0) \vee$
                $succ(Z) = X\}$
  begin $Y := 0$; $Z := 0$;
    while $\neg X = Y$ **do** $Z := Y$; $Y := succ(Y)$; **end**;
  end.

Prozedurdeklarationen

Eine Prozedurumgebung über einer Signatur (S, Σ) besteht aus einer endlichen Menge Ω von Prozedurdeklarationen (Prozeduren) der Form

Kopf: **proc** P (**in** $\vec{X}$, **out** $\vec{Y}$)
Komment: $\{\varphi(\vec{X})\}$ **call** $P(\vec{X}, \vec{Y})$; $\{\psi(\vec{X}, \vec{Y})\}$
begin
Rumpf: β
end.

Mit folgenden Eigenschaften:

- Die verwandten Prozedurnamen sind paarweise verschieden.
- **proc** $P($**in** $\vec{X}$, **out** $\vec{Y})$ ist Prozedurkopf über (S, Σ).
- β ist ein rekursives Programm über (S, Σ) und zu jedem Prozeduraufruf **call** $Q(\vec{t}, \vec{U})$; in β gibt es eine Prozedur in Ω mit Namen Q und passendem Prozedurkopf.
- β verändert keine **in**-Parameter d. h. in β keine Anweisungen der Form $X_i := t$: oder **call** $Q(\vec{t} \ldots X_i \ldots)$, d. h. $X_i \notin \{\vec{U}\}$.
- $var(\varphi(\vec{X})) \subseteq \{\vec{X}\}$, $var(\psi(\vec{X}, \vec{Y})) \subseteq \{\vec{X}\} \cup \{\vec{Y}\}$.
- Keine „globalen“ Variablen in Prozeduren. Kommmentare werden zur Spezifikation und Verifikation des Programmverhaltens verwendet, sie beeinflussen nicht die Abarbeitung. Sie sind auch Hilfsmittel für die Wiederverwendung (suchen von Programmen mit bestimmten Eigenschaften).
- Beweisverpflichtung: $A \models \{\varphi(\vec{X})\}\beta\{\psi(\vec{X}, \vec{Y})\}$

Prozeduraufrufregel-Beispiel

Der hoarsche Kalkül $HC(\Omega)$ über Ω ergibt sich durch Hinzunahme der folgenden neuen Regel

$$\frac{\pi \rightarrow [\varphi]\{\vec{X}/\vec{t}\}, ([\psi]\{\vec{X}/\vec{t}, \vec{Y}/\vec{U}\} \wedge \exists \vec{U} \pi) \rightarrow \varrho}{\{\pi\} \ \underline{\textbf{call}} \ P(\vec{t}, \vec{U}); \ \{\varrho\}}$$

Zu jeder Prozedur P, die in Ω deklariert ist.

$HC(\Omega, A)$ sei $HC(\Omega)$ erweitert, um die in der Algebra A gültigen PL-Formeln. Eine Prozedurumgebung Ω heißt über einer Algebra A korrekt kommentiert, sofern für jede Prozedurdeklaration in Ω die partielle Korrektheitsaussage $\{\varphi(\vec{X})\}\beta\{\psi(\vec{X}, \vec{Y})\}$ in $HC(\Omega, A)$ ableitbar ist.

Beispiel 91-Funktion von McCarthy über (Nat, -)

proc $P91($ **in** $X : nat,$ **out** $Y : nat)$
  {true} **call** $P91(X, Y); \ \{\psi(X, Y)\}$
  begin
    if $X > 100$
      then
        $Y := X - 10;$
      else **call** $P91(X + 11, U);$ **call** $P91(U, Y);$
    end;
  end.

Beispiel 91-Funktion

Mit Spezifikation $\psi(X, Y)$ ::
$((X > 100 \rightarrow Y = (X - 10)) \wedge (X \leq 100 \rightarrow Y = 91))$
Behauptung:

$\underset{HC(\Omega,A)}{\vdash} \{\text{true}\}\beta\{\psi(X, Y)\}$, d. h. korrekt kommentierte Prozedur.

Syntaktisch korrekte Kommentierung von β:

$\{\text{true}\}$
if $X > 100$
  then
    $\{X > 100\}Y := X - 10;\ \{\psi(X, Y)\}$
  else $\{\neg X > 100\}$
    call $P91(X + 11, U);$
    $\{\neg X > 100 \wedge [\psi]\{X/X + 11, Y/U\}\}$
    call $P91(U, Y);$
    $\{\psi(X, Y)\}$
end;
$\{\psi(X, Y)\}$

Beweisverpflichtungen

- $X > 100 \rightarrow ((X > 100 \rightarrow (X - 10)$
  $= (X - 10)) \wedge (X \leq 100 \rightarrow (X - 10) = 91))$ ok.
- $\neg X > 100 \rightarrow$ true ok.
- $([\psi]\{X/X + 11, Y/U\} \wedge \exists U \neg X > 100)$
  $\rightarrow (\neg X > 100 \wedge [\psi]\{X/X + 11, Y/U\})$ ok.
- $(\neg X > 100 \wedge [\psi]\{X/X + 11, Y/U\}) \rightarrow$ true ok.
- $[\psi]\{X/U, Y/Y\} \wedge \exists U(\neg X > 100$
  $\wedge [\psi]\{X/X + 11, Y/U\}) \rightarrow \psi[X, Y]$

$((U > 100 \rightarrow Y = (U - 10)) \wedge (U \leq 100 \rightarrow Y = 91))$
$\wedge(\neg X > 100 \wedge (X + 11 > 100 \rightarrow U = X + 1) \wedge$
$(X + 11 \leq 100 \rightarrow U = 91))$
$\rightarrow (X > 100 \rightarrow Y = (X - 10)) \wedge (X \leq 100 \rightarrow Y = 91)$

Fall: $z(X) =$

$\cdot 100 \qquad \cdot z(U) = 101 \qquad z(Y) = z(U - 10) = 91$

$\cdot 90 \leq \cdot < 100 \qquad \cdot z(U) = z(X + 1) \leq 100 \qquad z(Y) = 91$

$\cdot \leq 89 \qquad z(U) = 91 \qquad z(Y) = 91$

$\rightsquigarrow$ Behauptung