Grundlagen der Theoretischen Informatik

Sebastian Iwanowski
FH Wedel

Kap. 3: Verifikationstechniken
Teil 1: Verifikation mit Hoare-Tripeln

Programmentwicklung

Konstruktionsproblem:

Gegeben eine Spezifikation:

Funktion, die einem Argument einen Funktionswert zuordnet

Entwirf ein Programm, das jedes Argument des Definitionsbereichs als Eingabe akzeptiert und den zugehörigen Funktionswert als Ausgabe produziert.

Verifikationsproblem:

Gegeben eine Spezifikation und ein Programm:

Beweise, dass das Programm für jedes Argument der Spezifikation als Eingabe den zugehörigen Funktionswert der Ausgabe berechnet.

Zusatzaufgaben:

1. Was wird bei anderen Eingaben als den zulässigen Argumenten berechnet ?
2. Welche Bedingungen muss die Eingabe erfüllen, um bestimmte Ausgaben auszuschließen ?

Programmverifikation

Etwas allgemeiner:

Gegeben eine Spezifikation:
Funktion, die Startzuständen eindeutige Endzuständen zuordnet.

Gegeben ein Programm:

Beweise, dass das Programm für jeden Startzustand den von der Spezifikation geforderten Endzustand erreicht und danach stoppt.

Zusatzaufgaben:

1. Welche Zustände werden bei anderen als den geforderten Startzuständen erreicht ?
2. Welche Bedingungen müssen die Startzustände erfüllen, um bestimmte Endzustände auszuschließen ?

Anmerkung: Die gleichen Aufgaben können natürlich auch für beliebige Zwischenzustände untersucht werden.

Programmverifikation

Formalismus zum Lösen der Verifikationsaufgaben:
<u>Hoare-Tripel</u>

$$\{\varphi\} \quad P \quad \{\psi\}$$

Vorbedingung Programm Nachbedingung

Da ein Programm eine Sequenz von Anweisungen ist, kann man dieses Vorgehen auf die einzelnen Anweisungen reduzieren:

$$\{\varphi\} \quad S \quad \{\psi\}$$

Vorbedingung Anweisung Nachbedingung

Fragestellungen:

1. Gegeben φ, finde stärkste Nachbedingung ψ
2. Gegeben ψ, finde schwächste Vorbedingung φ

Programmverifikation

Beispiel für das Arbeiten mit Hoare-Tripeln:

`{ V }`	φ
`z := x • y ;`	S
`{ z ≥ 0 }`	ψ
`w := sqrt (z)`	

$\varphi_1 \Leftrightarrow (x > 0) \wedge (y > 0)$ ist eine Vorbedingung für ψ: $\{\varphi_1\}$ S $\{\psi\}$

$\varphi_2 \Leftrightarrow (x < 0) \wedge (y < 0)$ ist auch eine Vorbedingung für ψ: $\{\varphi_2\}$ S $\{\psi\}$

Welche ist die schwächste Vorbedingung V ?

$V \Leftrightarrow ((x>0)\wedge(y>0)) \vee ((x<0)\wedge(y<0)) \vee (x=0) \vee (y=0)$
ist die schwächste Vorbedingung für ψ:

$\{V\}$ S $\{\psi\}$ Außerdem gilt: $\{\varphi\}$ S $\{\psi\} \Rightarrow (\varphi \rightarrow V)$

Programmverifikation

Semantik von Hoare-Tripeln:

$\{\, \varphi \,\}$	S	$\{\, \psi \,\}$	bedeutet:
Vorbedingung	**Anweisung**	**Nachbedingung**	

Bei Vorliegen von φ gilt nach Beendigung der Anweisung S die Bedingung ψ.

<u>Merke:</u> φ_1 ist schwächer als φ_2 bedeutet: $\varphi_2 \rightarrow \varphi_1$ (φ_2 ist stärker als φ_1)

Also gilt: $\top$ (w) ist die schwächste aller Bedingungen und $\perp$ (f) die stärkste.

Folgerung: Die schwächste Vorbedingung für $\top$ ist die Bedingung, die garantiert, dass S zu einem Ende kommt.

Forderung (Axiom): Die schwächste Vorbedingung für $\perp$ ist $\perp$.

(nach Dijkstra: Gesetz des ausgeschlossenen Wunders)

Programmverifikation

Logischer Zusammenhang von Vorbedingungen:

$(\{\varphi_1\}\ S\ \{\psi\}) \wedge (\varphi_2 \rightarrow \varphi_1) \qquad \Rightarrow \qquad \{\varphi_2\}\ S\ \{\psi\}$

Die Vertauschung gilt nicht:

$\cancel{(\{\varphi_1\}\ S\ \{\psi\}) \wedge (\varphi_1 \rightarrow \varphi_2) \qquad \Rightarrow \qquad \{\varphi_2\}\ S\ \{\psi\}}$

Logischer Zusammenhang von Nachbedingungen:

$(\{\varphi\}\ S\ \{\psi_1\}) \wedge (\psi_1 \rightarrow \psi_2) \qquad \Rightarrow \qquad \{\varphi\}\ S\ \{\psi_2\}$

Die Vertauschung gilt nicht:

$\cancel{(\{\varphi\}\ S\ \{\psi_1\}) \wedge (\psi_2 \rightarrow \psi_1) \qquad \Rightarrow \qquad \{\varphi\}\ S\ \{\psi_2\}}$

Verifikation von Zuweisungen

Definition einer Zuweisung:

```
x := Ausdruck
```

Hierbei ist **x** ein beliebiger Variablenname und **Ausdruck** eine beliebige Funktion, die unter Umständen von Variablen abhängt. Die Variablen in **Ausdruck** müssen zum Zeitpunkt der Anweisung Werte haben.

Funktionsweise einer Zuweisung:

{ Prädikate für die Werte von x_1, ..., x_k und eventuell weitere Prädikate }

x := Ausdruck (x_1, ..., x_k)

{ Neues Prädikat für den Wert von x und eventuell weitere Prädikate }

Zunächst wird **Ausdruck (x_1, ..., x_k)** ausgewertet.
Der sich ergebende Funktionswert wird danach in die Variable x geschrieben.

Verifikation von Zuweisungen

Beispiel für die Verifikation einer Zuweisung:

`{ (x > 0) ∧ (y > 0)}`	φ
`z := x - sqrt (y) ;`	S
`{ N }`	ψ

Welche ist die stärkste Nachbedingung N ?

Welche ist die schwächste Vorbedingung für $\psi \Leftrightarrow$ `(z > 0)` ?

Verifikation von Zuweisungen

Die Zuweisungsvariable darf auch im zu berechnenden Ausdruck vorkommen:

`{ (x > 0) }`	φ
`x := x - sqrt (x) ;`	**S**
`{ N }`	ψ

Welche ist die stärkste Nachbedingung N ?

Welche ist die schwächste Vorbedingung für $\psi \Leftrightarrow$ `(x > 0)` ?

Verifikation von Zuweisungen

Der allgemeine Fall einer Zuweisung:

`{ V (x) }`	φ
`x := f(x) ;`	**S**
`{ N (x) }`	ψ

Verfahren zur Berechnung der schwächsten Vorbedingung zu gegebener Nachbedingung:

- Ersetze in N (x) jedes x durch f (x) ➔ Das Ergebnis ist die schwächste Vorbedingung.

Verfahren zur Berechnung der stärksten Nachbedingung zu gegebener Vorbedingung:

- Ersetze in V (x) jedes x durch f (x) ➔ Das Ergebnis ist eine Nachbedingung,
  aber nicht unbedingt die stärkste !

Verifikation von Zuweisungen

Zuweisung ohne Verwendung der Zuweisungsvariable im Ausdruck:

$\{V(x_1,...,x_k)\}$	φ
`z := A(`$x_1,...,x_k$`);`	S
$\{N(x_1,...,x_k,z)\}$	ψ

Finde die stärkste Nachbedingung ψ zu gegebenem φ:

$$N(x_1,...,x_k,z) \Leftrightarrow V(x_1,...,x_k) \wedge (z = A(x_1,...,x_k))$$

Beispiel:

$\{ (x > 0) \wedge (y > 0)\}$	φ
`z := x - sqrt (y) ;`	S
$\{ N \}$	ψ

Stärkste Nachbedingung:

$$N(x,y,z) \Leftrightarrow (x > 0) \wedge (y > 0) \wedge (z = x - \mathrm{sqrt}\ (y))$$

Verifikation von Zuweisungen

Zuweisung mit Verwendung der Zuweisungsvariable im Ausdruck:

$$\{V_1(x_1,...,x_k) \wedge V_2(x)\} \qquad \varphi$$

$$x := A(x_1,...,x_k,x); \qquad S$$

$$\{N(x_1,...,x_k,x)\}\} \qquad \psi$$

Finde die stärkste Nachbedingung ψ zu gegebenem φ:

$N(x_1,...,x_k,x) \Leftrightarrow$
$\quad V_1(x_1,...,x_k) \wedge$
$\quad (x \in$ Wertebereich von $A(x_1,...,x_k,z)$
$\quad\quad$ wenn gilt: $(V_1(x_1,...,x_k) \wedge V_2(z))$

Beispiel: $\{ (x > 0) \}$ φ
$x := x - \mathrm{sqrt}\ (x)$; S
$\{ N \}$ ψ

Stärkste Nachbedingung:

$N(x) \Leftrightarrow x \in$ Wertebereich von $f(z)= z - \mathrm{sqrt}\ (z)$ für $(z > 0)$

Verifikation von Zuweisungen

$\{V(x_1,...,x_k,z)\}$	φ
$z := A(x_1,...,x_k,z);$	S
$\{N_1(x_1,...,x_k) \wedge N_2(z)\}\}$	ψ

Finde die schwächste Vorbedingung φ zu gegebenem ψ:

$$V(x_1,...,x_k,z) \Leftrightarrow N_1(x_1,...,x_k) \wedge N_2(A(x_1,...,x_k,z)))$$

Beispiel:

`{V}`	φ
`x := x - sqrt (x) ;`	S
`{ x > 0 }`	ψ

Schwächste Vorbedingung:

$V(x) \Leftrightarrow (x - sqrt\ (x) > 0)$

Verifikation von Verbundanweisungen

Definition einer Verbundanweisung:

```
begin
     Anweisung 1;
     Anweisung 2;
     ...
     Anweisung n
end
```

Hierbei dürfen die Anweisungen 1 bis n beliebige Anweisungen sein: von einfachen Zuweisungen bis hin zu ineinandergeschachtelten Kontrollstrukturen.

Funktionsweise einer Verbundanweisung:

Die Anweisungen werden der Reihe nach **hintereinander** ausgeführt.
Eine **parallele** Ausführung findet **nicht** statt.

Verifikation von Verbundanweisungen

Verifikationstechnik:

```
{Vorbedingung}
begin
     Anweisung 1;
     {Zwischenbedingung 1}
     Anweisung 2;
     {Zwischenbedingung 2}
     ...
     {Zwischenbedingung n-1}
     Anweisung n
end
{Nachbedingung}
```

Achtung:

Geübte Verifizierer schreiben nicht jede Zwischenbedingung auf. Sie müssen aber alle Zwischenbedingungen im Kopf durcharbeiten, da es keine parallele Abarbeitung gibt !

Verifikation von Verbundanweisungen

Beispiel für die Verifikation einer Verbundanweisung:

Spezifikation: 2 mit Werten belegte Variablen sollen ihre Werte tauschen.

Programm:

```
{ (x = Wert1) ∧ (y = Wert2)}   φ
begin
     x := y ;
     { (x = Wert2) ∧ (y = Wert2)}
     y := x ;
     { (x = Wert2) ∧ (y = Wert2)}
end
{ (x = Wert2) ∧ (y = Wert1)}   ψ
```

Unter welchen Bedingungen ist das Programm korrekt ?

Antwort: nur wenn `Wert1 = Wert2`

Verifikation von Verbundanweisungen

Wie tauscht man verschiedene Werte ?

Lösung:

```
{ (x = Wert1) ∧ (y = Wert2)}  φ
begin
     z := x ;
    { (x = Wert1) ∧ (y = Wert2) ∧ (z = Wert1)}
     x := y ;
    { (x = Wert2) ∧ (y = Wert2) ∧ (z = Wert1)}
     y := z ;
    { (x = Wert2) ∧ (y = Wert1) ∧ (z = Wert1)}
end
{ (x = Wert2) ∧ (y = Wert1)}  ψ
```

Keine Einschränkung der Vor- oder Nachbedingung !

Damit ist bewiesen, dass das Programm die Spezifikation erfüllt.