Grundlagen der Programmierung

Vorlesung 8 vom 02.12.2004

Sebastian Iwanowski
FH Wedel

Verifikation von Verzweigungen

S

{Vorbedingung}	φ
if Ausdruck	β
then	
{then-Vorbedingung}	φ_1
then-Anweisung	**T**
{then-Nachbedingung}	ψ_1
else	
{else-Vorbedingung}	φ_2
else-Anweisung	**E**
{else-Nachbedingung}	ψ_2
{Nachbedingung}	ψ

Berechnung der stärksten Nachbedingung: Gegeben φ, berechne ψ

1) Setze $\varphi_1 = \varphi \wedge \beta$ und berechne das stärkste ψ_1 $\quad (\psi_1 \rightarrow \psi)$

2) Setze $\varphi_2 = \varphi \wedge \neg\beta$ und berechne das stärkste ψ_2 $\quad (\psi_2 \rightarrow \psi)$

3) Lösung: $(\psi_1 \rightarrow \psi) \wedge (\psi_2 \rightarrow \psi) \Leftrightarrow (\neg\psi_1 \vee \psi) \wedge (\neg\psi_2 \vee \psi)$

$\Leftrightarrow (\neg\psi_1 \wedge \neg\psi_2) \vee \psi \Leftrightarrow \neg(\psi_1 \vee \psi_2) \vee \psi$

$\Leftrightarrow \mathbf{(\psi_1 \vee \psi_2) \rightarrow \psi}$

Also ist $\psi_1 \vee \psi_2$ *die stärkste Nachbedingung*

Verifikation von Verzweigungen

S

{Vorbedingung}	φ
if Ausdruck	β
then	
{then-Vorbedingung}	φ_1
then-Anweisung	T
{then-Nachbedingung}	ψ_1
else	
{else-Vorbedingung}	φ_2
else-Anweisung	E
{else-Nachbedingung}	ψ_2
{Nachbedingung}	ψ

Zusammenhang der Anweisungen S, T und E:

$$\{\varphi\}\ S\ \{\psi\} \Leftrightarrow (\{\varphi \wedge \beta\}\ T\ \{\psi\}) \wedge (\{\varphi \wedge \neg\beta\}\ E\ \{\psi\})$$

Es wird hier keine Aussage über den Zusammenhang von ψ_1, ψ_2 und ψ gemacht !

Grundlagen der Programmierung

1. Einführung
   - Grundlegende Eigenschaften von Algorithmen und Programmen
2. Logik
   - Aussagenlogik
   - Prädikatenlogik
3. Programmentwicklung und –verifikation
   - Grundlagen der Programmverifikation
   - Zuweisungen und Verbundanweisungen
   - Verzweigungen
   - ➔ Schleifen
   - Modularisierung
   - Rekursion
4. Entwurf und Analyse von Algorithmen
   - Klassifikation von Algorithmen
   - Programmierung von Algorithmen
   - Bewertung von Algorithmen

Verifikation und Konstruktion von Schleifen

Definition einer Schleife:

```
while Schleifenbedingung do
Rumpfanweisung
```

`Schleifenbedingung` muss eine logische Funktion sein, die nur von Variablen abhängen darf, die mit Werten belegt sind.

Funktionsweise:

1) Zunächst wird **`Schleifenbedingung`** ausgewertet.

2) Wenn **`Schleifenbedingung`** falsch ist, wird die Schleife sofort beendet.
   Wenn **`Schleifenbedingung`** wahr ist, wird **`Rumpfanweisung`** ausgeführt.
   Dann wird bei Schritt 1) fortgefahren.

Verifikation von Schleifen

Verifikationstechnik:

	{Vorbedingung}	φ
W	**while Schleifenbedingung do**	β
	{Eintrittsbedingung}	φ_i
	Rumpfanweisung	
	{Austrittsbedingung}	ψ_i
	{Nachbedingung}	ψ

Definition:

Es sei φ_i die Eintrittsbedingung vor der i-ten Ausführung der Rumpfanweisung und ψ_i die Austrittsbedingung nach der i-ten Ausführung der Rumpfanweisung
Die Schleife werde nach k Ausführungen beendet.

Dann gilt:

1) $\varphi_1 \Leftrightarrow \varphi \wedge \beta$ **{φ_1} Rumpfanweisung {ψ_1}**

2) $\varphi_2 \Leftrightarrow \psi_1 \wedge \beta$ **{φ_2} Rumpfanweisung {ψ_2}**

.

i) $\varphi_i \Leftrightarrow \psi_{i-1} \wedge \beta$ **{φ_i} Rumpfanweisung {ψ_i}**

.

k) $\varphi_k \Leftrightarrow \psi_{k-1} \wedge \beta$ **{φ_k} Rumpfanweisung {ψ_k}**

k+1) $\psi \Leftrightarrow \psi_k \wedge \neg\beta$ **{φ_k} Rumpfanweisung {ψ_k}**

Verifikation von Schleifen

Verifikationstechnik:

W

```
{Vorbedingung}                    φ
while Schleifenbedingung do       β
      {Eintrittsbedingung}        φ_i
      Rumpfanweisung
      {Austrittsbedingung}        ψ_i
{Nachbedingung}                   ψ
```

Gegeben ψ, berechne φ: Wie findet man die schwächste Vorbedingung P für φ ?

Beobachtung:

0) Sei P_0 die schwächste Vorbedingung, falls die Schleife gar nicht durchlaufen wird.

1) Sei P_1 die schwächste Vorbedingung, falls die Schleife genau einmal durchlaufen wird.

i) Sei P_i die schwächste Vorbedingung, falls die Schleife genau i-mal durchlaufen wird.

Lösung: Dann gilt: $P = P_0 \vee P_1 \vee \ldots \vee P_i \vee \ldots$

Problem: Im allgemeinen Fall könnten sich die P_i's alle unterscheiden !

Damit kann keine allgemeine Lösungstechnik angegeben werden !

Verifikation von Schleifen

Verifikationstechnik:

W

```
{Vorbedingung}                    φ
while Schleifenbedingung do       β
      {Eintrittsbedingung}        φ_i
      Rumpfanweisung
      {Austrittsbedingung}        ψ_i
{Nachbedingung}                   ψ
```

Einfachere Aufgabe:

Gegeben φ und ψ:

Beweise, dass gilt: **{φ} W {ψ} !**

Verifikation von Schleifen

Beispiel 1:

```
{ Vorbedingung }    φ

f := 1;
z := 0;
while (z < n) do
  begin
     z := z + 1;
     f := z • f
  end

{ Nachbedingung }   ψ
```

Was berechnet dieses Programmstück ?

Gibt es Einschränkungen für die Bedingungen φ oder ψ ?

Verifikation von Schleifen

Beispiel 2:

```
Gegeben seien n Zahlen a[1] .. a[n];

{ Vorbedingung }    φ

z := 0;
while (z < n) do
  begin
     z := z + 1;
     if m < a[z]
       then
          m := a[z]
  end

{ Nachbedingung }   ψ
```

Was berechnet dieses Programm ?

Gibt es Einschränkungen für die Bedingungen φ oder ψ ?

Verifikation von Schleifen

Zur Erinnerung:

$\varphi_i \Leftrightarrow \psi_{i-1} \wedge \beta \; \{\psi_{i-1} \wedge \beta\}$ **Rumpfanweisung** $\{\psi_i\}$

Neue Idee: Invariantensuche

Suche einen Anteil I, der in jedem ψ_i enthalten ist (I ist ***Invariante***).
Dann gilt: $\psi_i \Leftrightarrow I \wedge V_i$ wobei V_i der variante Anteil ist

Bei mindestens einem Schleifendurchlauf gilt:

1)	$\exists k\,(V_k \Rightarrow \neg\beta)$	*Der variante Anteil muss irgendwann einmal die Schleifenbedingung negieren, anderenfalls handelt es sich um eine Endlosschleife*
2)	$\psi \Rightarrow I \wedge \neg\beta$	*In der Nachbedingung gilt immer die Invariante und niemals die Schleifenbedingung.*
3)	$\varphi \Rightarrow I \wedge \beta$	*Die Vorbedingung erfüllt immer die Invariante und die Schleifenbedingung.*

Fazit: *Die Invariante gilt **immer** (vor, während und nach der Schleife)*

Verifikation von Schleifen

Gegeben φ und ψ, beweise, dass gilt: **{φ} W {ψ} !**

W

```
{Vorbedingung}                          φ
while Schleifenbedingung do             β
      {Eintrittsbedingung}              φ_i
      Rumpfanweisung
      {Austrittsbedingung}              ψ_i
{Nachbedingung}                         ψ
```

Lösungsverfahren:

1) Finde eine Invariante I und eine Variante V_i!
2) Zeige, dass I vor der Schleife und nach jedem Schleifendurchlauf gilt!
3) Zeige, dass V_i sich in jedem Schleifendurchlauf derart ändert, dass irgendwann die Bedingung β verletzt sein muss!
4) Zeige, dass aus den nach Beendigung der Schleife geltenden Prädikaten I und ¬β die gewünschte Nachbedingung ψ folgt!

Konstruktion von Schleifen

Lösungsverfahren:

1) Finde zu gegebenem ψ eine Zerlegung in I und ¬β: $I \wedge \neg\beta \Rightarrow \psi$
2) Sorge dafür, dass I schon vor der Schleife gilt: Finde S_0 mit: $\{\varphi\}\ S_0\ \{I\}$
3) Finde eine Rumpfanweisung S derart, dass:

a) $\{I \wedge \beta\}$	S	$\{I\}$	(Invariantenbedingung)
b) $\{I \wedge \beta \wedge (z=z_0\}$	S	$\{z<z_0\}$	(Fortschrittsbedingung)
c) $I \wedge (z \leq 0)$	$\Rightarrow$	¬β	(Terminierungsbedingung)

$\{\varphi\}$
S_0
$\{I\}$
```
while β do
begin
        {I ∧ β ∧ (z=z0) }
        S
        {I ∧ (z<z0) }
end
```
$\{I \wedge \neg\beta\}$
$\{\psi\}$

Konstruktion von Schleifen

Beispiel: Division von ganzen Zahlen mit Rest:

x div y = q mod r

{(x ≥ 0) ∧ (y > 0)}

S_0

{I}
while β do
begin
{I ∧ β ∧ ($z=z_0$} }

S

{I ∧ ($z<z_0$) }
end
{I ∧ ¬β}
{(x = q•y+r) ∧ (0 ≤ r < y)}

Beim nächsten Mal:

Programmentwicklung und –verifikation:
Modularisierung