II.1.4 Verifikation

Dienstag, 10. November 2020 17:30

Bsp: Fakultät von n

(Fakultät von negativen Zahlen und 0 wird als 1 definiert.)

$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$

Bsp: $n=3$

$i=3$
$res=1$

$\downarrow$

$i=2$
$res=3$

$\downarrow$

$i=1$
$res=3\cdot 2$

Ergebnis: 6

Hoare-Kalkül

Tony Hoare

$\langle \underbrace{\varphi}_{\text{phi}} \rangle \quad P \quad \langle \underbrace{\psi}_{\text{psi}} \rangle$

- Rahmen/Anleitung zur Verifikation
- Verifikation leicht überprüfbar
- Verifikation teilweise automatisierbar

Schreibweise der Regeln:

$$\frac{\ldots}{\ldots}$$

← wenn das wahr ist,

← dann ist auch das wahr

Zuweisungsregel

- Oberhalb des Strichs steht nichts
  $\Rightarrow$ Aussage unterhalb des Strichs ist immer wahr.
- Damit nach Ausführung von

  $x = t;$

  die Nachbedingung φ gilt,
  muss vorher $\varphi[x/t]$ gegolten haben.

$$\langle \underbrace{5 \geq y}_{\varphi[x/5]} \rangle \quad x = \underbrace{5}_{t}; \quad \langle \underbrace{x \geq y}_{\varphi} \rangle$$

$$\langle 5 > 4 \rangle \quad x = 5; \quad \langle x > 4 \rangle$$

Schreibweise:
Ergänze Programm um Zusicherung (Assertions), die wahre Aussagen über die Prog.-Variablen an dieser Stelle enthalten.

Konsequenzregel 1

: Man kann die Vorbedingung verschärfen.

$$\langle \overbrace{5 \geq y}^{\varphi} \rangle \quad x = 5; \quad \langle \overbrace{x \geq y}^{\psi} \rangle$$

$$\underbrace{3 \geq y}_{\alpha} \Rightarrow \underbrace{5 \geq y}_{\varphi}$$

$$\langle \overbrace{3 \geq y}^{\alpha} \rangle \quad x = 5; \quad \langle \overbrace{x \geq y}^{\psi} \rangle$$

Schreibweise:

- Wenn 2 Zusicherungen direkt untereinander stehen, dann muss die untere aus der oberen folgen.
- Wenn zwischen 2 Zusicherungen eine Anweisung steht,

- Wenn zwischen 2 Zusicherungen eine Anweisung steht, dann muss es einer Anwendung einer Regel des H-Kalküls entsprechen.

Konsequenzregel 2: Man kann die Nachbedingung abschwächen.

$$\overbrace{\langle 3 \geq y \rangle}^{\varphi} \quad x = 5; \quad \overbrace{\langle x \geq y \rangle}^{\psi}$$

$$\underbrace{x \geq y}_{\psi} \quad \Rightarrow \quad \underbrace{x+1 \geq y}_{\beta}$$

$$\overbrace{\langle 3 \geq y \rangle}^{\varphi} \quad x = 5; \quad \overbrace{\langle x+1 \geq y \rangle}^{\beta}$$

Logische Verknüpfungen:

$\wedge$	"und"	(&&)
$\vee$	"oder"	(||)
$\neg$	"nicht"	(!)

Sequenzregel

Setzt Teilbeweise für Prog.-Teile zusammen.

$\langle true \rangle$

$\langle 5 = 5 \rangle$

x = 5;

$\langle x = 5 \rangle$

$\langle x * x + 6 = 31 \rangle$

res = x * x + 6;

$\langle res = 31 \rangle$

Bedingungsregel 1

Jetzt mit "if"-Anweisungen ohne "else"

Bedingungsregel 1

behandelt "if"-Anweisungen ohne "else"

$\langle true \rangle$

$\langle y = y \rangle$

res = y;

$\langle res = y \rangle$ — φ

if (x > y) {

$\langle res = y \wedge x > y \rangle$ — $\varphi \wedge B$

$\langle x = max(x,y) \rangle$

res = x;

$\langle res = max(x,y) \rangle$ — ψ

}

$\langle res = max(x,y) \rangle$ — ψ

und $\underbrace{res = y}_{\varphi} \wedge \underbrace{\neg\, x > y}_{\neg B} \Rightarrow \underbrace{res = max(x,y)}_{\psi}$

Bedingungsregel 2

$\langle true \rangle$

if (x < 0) {

$\langle true \wedge x < 0 \rangle$

$\langle -x = |x| \rangle$

res = -x;

$\langle res = |x| \rangle$

}

else {

$\langle true \wedge \neg\, x < 0 \rangle$

$\langle x = |x| \rangle$

res = x;

$\langle res = |x| \rangle$

}

$\langle res = |x| \rangle$

Schleifenregel

Idee: Finde eine Schleifeninvariante φ:

Wenn φ vor Schleifenrumpf P gilt

und der Schleifenrumpf wird ausgeführt (weil Bedingung B gilt)

dann gilt φ nach dem Schleifenrumpf immer noch.

Man muss eine Schleifeninvariante φ finden, die 3 Eigenschaften erfüllt:

ⓐ φ muss Schleifeninvariante sein

$\langle \varphi \wedge i > 1 \rangle$

res = res * i; i = i - 1;

$\langle \varphi \rangle$

ⓑ φ muss aus der Vorbedingung folgen.

$i = n \wedge res = 1 \Rightarrow \varphi$

ⓒ Nachbedingung muss aus Schleifeninvariante φ und negierter Schleifenbedingung folgen.

$\varphi \wedge \neg i > 1 \Rightarrow res = n!$

Tipp: Starte mit Nachbedingung und passe sie so an, dass daraus eine Schleifeninvariante wird. Führe dazu die Schleife mit typischen Variablenwerten aus.

i	res	n
4	1	4
3	4	4
2	4·3	4
1	4·3·2	4

Schleifeninvariante:

$i! \cdot res = n!$

Zusicherungen können mit "assert" ins Java-Programm geschrieben.
Bei Ausführung mit

java -ea ...

werden Zusicherungen mit überprüft.

Terminierung

Finde Variante V, die ≥ 0 ist, wenn die Schleife ausgeführt wird und die in jedem Schleifendurchlauf kleiner wird.

Bsp: Variante ist i

- $i > 1 \Rightarrow i \geq 0$
- $\langle i = m \wedge i > 1 \rangle$
  $\langle i-1 < m \rangle$
  `res = res * i;`
  $\langle i-1 < m \rangle$
  `i = i-1;`
  $\langle i < m \rangle$

Programmierer sollte sich zu jeder Schleife Invariante und Variante überlegen.

Additions-Bsp

Terminierung: Variante ist x

- $x > 0 \Rightarrow x \geq 0$
- $\langle x = m \wedge x > 0 \rangle$
  $\langle x-1 < m \rangle$
  `x = x - 1;`
  $\langle x < m \rangle$
  `res = res + 1;`

$\langle x < m \rangle$

$res = res + 1;$

$\langle x < m \rangle$

Subtraktions-Bsp

Terminierung: Variante ist $x - z$

- $x > z \Rightarrow x - z \geq 0$
-

$\langle x - z = m \wedge x > z \rangle$
$\langle x - (z+1) < m \rangle$

$z = z + 1;$

$\langle x - z < m \rangle$

$res = res + 1;$

$\langle x - z < m \rangle$