

### Aufgabe 3: „Formale Verifikation“

Sei  $wp(A, Q)$  die schwächste Vorbedingung (weakest precondition) eines Programmfragments  $A$  bei gegebener Nachbedingung  $Q$  so, dass  $A$  alle Eingaben, die  $wp(A, Q)$  erfüllen, auf gültige Ausgaben abbildet, die  $Q$  erfüllen.

Bestimmen Sie schrittweise und formal (mittels Floyd-Hoare-Kalkül) jeweils  $wp(A, Q)$  für folgende Code-Fragmente  $A$  und Nachbedingungen  $Q$  und vereinfachen Sie dabei den jeweils ermittelten Ausdruck so weit wie möglich.

Die Variablen  $x, y$  und  $z$  in folgenden Pseudo-Codes seien ganzzahlig (vom Typ `int`). Zur Vereinfachung nehmen Sie bitte im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

(a) Sequenz:

```
1  x = -2 * (x + 2 * y);
2  y += 2 * x + y + z;
3  z -= x - y - z;
```

$Q \equiv x = y + z$

Code umformulieren:

```
1  x = -2 * (x + 2 * y);
2  y = y + 2 * x + y + z;
3  z = z - (x - y - z);
```

$wp("x = -2 * (x + 2 * y); y = 2 * y + 2 * x + z; z = z - (x - y - z);", x = y + z)$

$z$  eingesetzt

$\equiv wp("x = -2 * (x + 2 * y); y = 2 * y + 2 * x + z; ", x = y + (z - (x - y - z)))$

Innere Klammer auflösen

$\equiv wp("x = -2 * (x + 2 * y); y = 2 * y + 2 * x + z; ", x = y + (-x + y - 2z))$

Klammer auflösen

$\equiv wp("x = -2 * (x + 2 * y); y = 2 * y + 2 * x + z; ", x = -x + 2y + 2z)$

$-x$  auf beiden Seiten

$\equiv wp("x = -2 * (x + 2 * y); y = 2 * y + 2 * x + z; ", 0 = -2x + 2y + 2z)$

$\div 2$  auf beiden Seiten

$\equiv wp("x = -2 * (x + 2 * y); y = 2 * y + 2 * x + z; ", 0 = -x + y + z)$

$y$  einsetzen

$\equiv wp("x = -2 * (x + 2 * y); ", 0 = -x + (2y + 2x + z) + z)$

Term vereinfachen

$$\equiv \text{wp}("x = -2 \cdot (x + 2 \cdot y);", 0 = x + 2y + 2z)$$

$x$  einsetzen

$$\equiv \text{wp}("", 0 = (-2(x + 2y)) + 2y + 2z)$$

wp weglassen

$$\equiv 0 = (-2(x + 2y)) + 2y + 2z$$

ausmultiplizieren

$$\equiv 0 = (-2x - 4y) + 2y + 2z$$

Klammer auflösen, vereinfachen

$$\equiv 0 = -2x - 2y + 2z$$

$\div 2$  auf beiden Seiten

$$\equiv 0 = -x - y + z$$

$x$  nach links holen mit  $+x$  auf beiden Seiten

$$\equiv x = -y + z$$

$y$  ganz nach links schreiben

$$\equiv x = z - y$$

$$x = -2 \cdot (x + 2 \cdot y)$$

(b) Verzweigung:

```

1  if (x < y) {
2    x = y + z;
3  } else if (y > 0) {
4    z = y - 1;
5  } else {
6    x -= y -= z;
7  }

```

$$Q \equiv x > z$$

**1. Fall:**  $x < y$

**2. Fall:**  $x \geq y \wedge y > 0$

**3. Fall:**  $x \geq y \wedge y \leq 0$

Code umformulieren:

```

1  if (x < y) {
2    x = y + z;
3  } else if (x >= y && y > 0) {
4    z = y - 1;
5  } else {

```

$$\begin{array}{l}
6 \quad y = y - z; \\
7 \quad x = x - y; \\
8 \quad \} \\
\\
\text{wp}(\text{"if}(x < y)\{x=y+z;\}\text{else if}(x \geq y \& \& y > 0)\{z=y-1;\}\text{else}\{y=y-z; x=x-y;\}\text{"}, x > z) \\
\equiv \quad \quad \quad (\text{In mehrere kleinere wp-Kalküle aufsplitten}) \\
\\
\left( (x < y) \wedge \text{wp}(\text{"x=y+z;"}, x > z) \right) \vee \\
\left( (x \geq y \wedge y > 0) \wedge \text{wp}(\text{"z=y-1;"}, x > z) \right) \vee \\
\left( (x \geq y \wedge y \leq 0) \wedge \text{wp}(\text{"y=y-z; x=x-y;"}, x > z) \right) \\
\\
\equiv \quad \quad \quad (\text{Code einsetzen}) \\
\\
\left( (x < y) \wedge \text{wp}(\text{"", } y + z > z) \right) \vee \\
\left( (x \geq y \wedge y > 0) \wedge \text{wp}(\text{"", } x > y - 1) \right) \vee \\
\left( (x \geq y \wedge y \leq 0) \wedge \text{wp}(\text{"y=y-z;"}, x - y > z) \right) \\
\\
\equiv \quad \quad \quad (\text{wp-Kalkül-Schreibweise weg lassen, Code weiter einsetzen}) \\
\\
\left( (x < y) \wedge y + z > z \right) \vee \\
\left( (x \geq y \wedge y > 0) \wedge x > y - 1 \right) \vee \\
\left( (x \geq y \wedge y \leq 0) \wedge \text{wp}(\text{"", } x - (y - z) > z) \right) \\
\\
\equiv \quad \quad \quad (\text{Terme vereinfachen, wp-Kalkül-Schreibweise weg lassen}) \\
\\
\left( x < y \wedge y > 0 \right) \vee \\
\left( x \geq y \wedge y > 0 \right) \vee \\
\left( (x \geq y \wedge y \leq 0) \wedge x - (y - z) > z \right) \\
\\
\equiv \quad \quad \quad (\text{letzten Term vereinfachen})
\end{array}$$

$$\begin{aligned} & (x < y \wedge y > 0) \vee \\ & (x \geq y \wedge y > 0) \vee \\ & ((x \geq y \wedge y \leq 0) \wedge x - y > 0) \end{aligned}$$

$\equiv$

(ein  $\wedge$  eliminieren)

$$\begin{aligned} & (x < y \wedge y > 0) \vee \\ & (x \geq y \wedge y > 0) \vee \\ & (y \leq 0 \wedge x > y) \end{aligned}$$

---

<sup>a</sup>  $x > y - 1 \wedge x \geq y$  ergibt  $x \geq y$   
<sup>a</sup>  $x > y - 1 \wedge x \geq y$  ergibt  $x \geq y$

(c) Mehrfachauswahl:

```

1  switch (z) {
2      case "x":
3          y = "x";
4      case "y":
5          y = --z;
6          break;
7      default:
8          y = 0x39 + "?";
9  }
```

$Q \equiv 'x' = y$

Hinweis zu den ASCII-Codes

- 'x' = 120<sub>(10)</sub>
- 'y' = 121<sub>(10)</sub>
- 0x39 = 57<sub>(10)</sub>
- '?' = 63<sub>(10)</sub>

Mehrfachauswahl in Bedingte Anweisungen umschreiben. Dabei beachten, dass bei fehlendem **break** die Anweisungen im folgenden Fall bzw. ggf. in den folgenden Fällen ausgeführt werden:

```

1  if (z == "x") {
2      y = "x";
3      y = z - 1;
4  } else if (z == "y") {
5      y = z - 1;
```

```

6 } else {
7   y = 0x39 + "?";
8 }

```

Da kein `break` im Fall `z == "x"`. `--z` bedeutet, dass die Variable erst um eins verringert und dann zugewiesen wird.

```

1 if (z == 120) {
2   y = 120;
3   y = 120 - 1;
4 } else if (z == 121) {
5   y = 121 - 1;
6 } else {
7   y = 57 + 63;
8 }

```

Vereinfachung / Zusammenfassung:

```

1 if (z == 120) {
2   y = 120;
3   y = 119;
4 } else if (z == 121) {
5   y = 120;
6 } else {
7   y = 120;
8 }

```