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);
```

$\mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;z=z-(x-y-z);"}, x = y + z)$

$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;z=z-(x-y-z);"}, x = y + z)$

z eingesetzen

$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, x = y + (z - (x - y - z)))$

Innere Klammer auflösen

$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, x = y + (-x + y - 2z))$

Klammer auflösen

$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, x = -x + 2y + 2z)$

$-x$ auf beiden Seiten

$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, 0 = -2x + 2y + 2z)$

$\div 2$ auf beiden Seiten

$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, 0 = -x + y + z)$

y einsetzen

$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);"}, 0 = -x + (2y + 2x + z) + z)$

Term vereinfachen

$$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);"}, 0 = x + 2y + 2z)$$

x einsetzen

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

wp weglassen

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

ausmultiplizieren

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

Klammer auflösen, vereinfachen

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

$\div 2$ auf beiden Seiten

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

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

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

y ganz nach links schreiben

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

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

(b) Verzweigung:

```
if (x < y) {
  x = y + z;
} else if (y > 0) {
  z = y - 1;
} else {
  x -= y -= z;
}
```

$Q :\equiv x > z$

(c) Mehrfachauswahl:

```
switch (z) {
  case "x":
    y = "x";
  case "y":
    y = --z;
    break;
  default:
    y = 0x39 + "?";
}
```

$Q :\equiv$ `'x'` $= y$

Hinweis zu den ASCII-Codes

- `'x'` $= 120_{(10)}$
- `'y'` $= 121_{(10)}$
- `0x39` $= 57_{(10)}$
- `'?'` $= 63_{(10)}$

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

```
if (z == "x") {
  y = "x";
  y = z - 1;
} else if (z == "y") {
  y = z - 1;
} else {
  y = 0x39 + "?";
}
```

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

```
if (z == 120) {
  y = 120;
  y = 120 - 1;
} else if (z == 121) {
  y = 121 - 1;
} else {
  y = 57 + 63;
}
```

Vereinfachung / Zusammenfassung:

```
if (z == 120) {
  y = 120;
  y = 119;
} else if (z == 121) {
  y = 120;
} else {
  y = 120;
}
```