Korrekte Software: Grundlagen und Methoden SoSe 19 Serge Autexier
Christoph Lüth

5. Musterlösung

Ausgabe: 16.05.19
Abgabe: 16.05.19

5.1 *Produkte*

Sie haben folgendes C0-Program gefunden:

```
/* This program computes the product */
x= 0;
c= 0;
while (c <= n) {
  x= x+ m;
  c= c+1;
  }
```

Angeblich soll es das Produkt berechnen.

(i) Welches Produkt wird berechnet? Geben Sie eine formale Spezifikation mit Vor- und Nachbedingung an.

(ii) Wie könnte eine Schleifeninvariante aussehen?

(iii) Aber leider ist das Program auch noch fehlerhaft! Korrigieren Sie den Fehler, und beweisen Sie jetzt die Korrektheit des korrigierten, nun (hoffentlich) korrekten Programmes mit dem Floyd-Hoare-Kalkül.

```
/* This program computes the product */
//{true}
// {0 = m × 0 ∧ 0 ≤ 0}
x= 0;
// {x = m × 0 ∧ 0 ≤ n}
c= 0;
// {x = m × c ∧ c ≤ n}
while (c < n) {
// {x = m × c ∧ c ≤ n ∧ c < n}
// {x = m × c ∧ c + 1 ≤ n}
// {x + m = m × (c + 1) ∧ c + 1 ≤ n}
  x= x+ m;
// {x = m × (c + 1) ∧ c + 1 ≤ n}
  c= c+1;
// {x = m × c ∧ c ≤ n}
}
// {x = m × c ∧ c ≤ n ∧ ¬(c < n)}
// {x = m × n}
```

5.2 *Noch mehr Produkte.*

Ihr Kollege sagt, folgendes Program sei noch wesentlich effizienter:

```
/* Binary multiplication of n and m */
x= 0;
while (n != 0) {
  if (n % 2 == 1) {
    x= x+ m;
    }
```

```
m= 2* m;
n= n/ 2;
}
```

Als ad-hoc-Erweiterung unserer Sprache benutzt das Programm den Modulo-Operator (%) in C, der den Rest bei der Division berechnet. Die mathemtische Notation dafür ist $x \bmod y$.

(i) Geben Sie Vor- und Nachbedigungen an.

(ii) Finden Sie eine Invariante, und beweisen Sie die Korrektheit des Programmes.

Hinweise:

- Beachten Sie, dass / die *ganzzahlige Division* berechnet (geschrieben $\div$). Es ist im Allgemeinen *nicht* das Inverse zur Multiplikation (d.h. es gilt *nicht*, dass $x = y \cdot (x \div y)$). Aber es gilt:

$$x = y \cdot (x \div y) + x \bmod y$$
$$x - x \bmod y = y \cdot (x \div y) \tag{1}$$