Technische Universität Berlin

Software and Embedded Systems Engineering Group
Prof. Dr. Sabine Glesner
www.sese.tu-berlin.de Sekr. TEL 12-4 Ernst-Reuter-Platz 7 10587 Berlin

Softwaretechnik und Programmierparadigmen WiSe 2022/2023

Prof. Dr. Sabine Glesner
Milko Monecke
Simon Schwan

Übungsblatt 15

Beispiellösung

Aufgabe 1: Terminierung

Beweist die *totale* Korrektheit folgender partiell korrekter Programme. Gibt es einen Algorithmus, der für jedes mögliche Programm eine passende Terminierungsfunktion findet?

a) `max(int a,int b):`

$\{true\}$
```
if a > b then
    m := a
else
    m := b
fi
```
$\{m \geq a \wedge m \geq b \wedge (m = a \vee m = b)\}$

b) `trinumber(int n):`

$\{n \geq 0\}$
```
s := 0;i := 0;
while i < n do
```
$\{i < n \wedge s = \sum_{j=0}^{i} j \wedge i \leq n\}$ Regel (5) $\{B \wedge I\}$
```
    i := i + 1;
    s := s + i
```
$\{s = \sum_{j=0}^{i} j \wedge i \leq n\}$ Regel (5) $\{I\}$
```
od
```
$\{s = \sum_{j=0}^{n} j\}$

c) `rest(int x, int y):`

$\{x \geq 0\}$
```
q := 0;
r := x;
while r >= y do
```
$\{r \geq y \land x = q * y + r \land r \geq 0\}$ Regel (5) $\{B \land I\}$
```
    r := r - y;
    q := q + 1
```
$\{x = q * y + r \land r \geq 0\}$ Regel (5) $\{I\}$
```
od;
```
$\{x = q * y + r \land r \geq 0 \land r < y\}$

Referenz: Hoare Kalkül

(1) Skip-Axiom: $\{P\}$ `skip` $\{P\}$

(2) Zuweisungsaxiom: $\{P[x \leftarrow E]\}$ `x := E` $\{P\}$

(3) Sequenzregel:

$$\frac{\{P\}\ S_1\ \{R\} \quad \{R\}\ S_2\ \{Q\}}{\{P\}\ S_1; S_2\ \{Q\}}$$

(4) if-then-else-Regel:

$$\frac{\{B \land P\}\ S_1\ \{Q\} \quad \{\neg B \land P\}\ S_2\ \{Q\}}{\{P\}\ \texttt{if}\ B\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \texttt{fi}\ \{Q\}}$$

(5) while-Regel:

$$\frac{\{B \land I\}\ S\ \{I\}}{\{I\}\ \texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}\ \{\neg B \land I\}}$$

(6) Konsequenzregel:

$$\frac{\{P \Rightarrow P'\} \quad \{P'\}\ S\ \{Q'\} \quad \{Q' \Rightarrow Q\}}{\{P\}\ S\ \{Q\}}$$

(7) Terminierung:

$$\frac{\{B \land I \land (t = m)\}\ S\ \{I \land (t < m)\},\ B \land I \Rightarrow t \geq 0}{\{I\}\ \texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}\ \{\neg B \land I\}}$$

Vorgehen: finde Terminierungsfunktion $t \mapsto \mathbb{N}$, sodass

1. $B \land I \Rightarrow t \geq 0$ und
2. $\{B \land I \land (t = m)\}\ S\ \{I \land (t < m)\}$ gilt.

Lösung:

a) Terminiert natürlich, weil keine Schleife.

b) Terminierung: $t = n - i$

1. $\{i < n \wedge s = \sum_{j=0}^{i} j \wedge i \leq n\} \Rightarrow n - i \geq 0$ Regel (7) $B \wedge I \Rightarrow n - i \geq 0$

2. `while i < n do`

$\{... \wedge (n - i = m)\}$	Regel (7) $\{B \wedge I \wedge (t = m)\}$
$\Rightarrow \{\ldots \wedge (n - i - 1 < m)\}$	Regel (6)
$\Rightarrow \{\ldots \wedge (n - (i + 1) < m)\}$	Regel (6) (2) $\{P[i \leftarrow i + 1]\}$
`i := i + 1;`	
$\{\ldots \wedge (n - i < m)\}$	Regel (3) (2) $\{P[s \leftarrow s + i]\}$
`s := s + i;`	
$\{... \wedge (n - i < m)\}$	Regel (7) $\{I \wedge (t < m)\}$

`od`

c) Terminierung: $t = r - y$

1. $\{r \geq y \wedge x = q * y + r \wedge r \geq 0\} \Rightarrow r - y \geq 0$ Regel (7) $B \wedge I \Rightarrow r - y \geq 0$

2. `while r >= y do`

$\{\ldots \wedge (r - y = m)\}$	Regel (7) $\{B \wedge I \wedge (t = m)\}$
$\not\Rightarrow \{\ldots \wedge (r - 2y < m)\}$	Regel (6), terminiert nur wenn $y > 0$
$\Rightarrow \{\ldots \wedge (r - y - y < m)\}$	Regel (6) (2) $\{P[r \leftarrow r - y]\}$
`r := r - y;`	
$\{\ldots \wedge (r - y < m)\}$	Regel (3) (2) $\{P[q \leftarrow q + 1]\}$
`q := q + 1`	
$\{\ldots \wedge (r - y < m)\}$	Regel (7) $\{I \wedge (t < m)\}$

`od;`

Wenn es einen generellen Algorithmus gäbe, könnte dieser generell die Terminierung von Programmen entscheiden, was das Halteproblem lösen würde. Daher kann es keinen solchen Algorithmus geben. Die Semi-Entscheidbarkeit des Halteproblems zeigt sich auch hier: wenn *eine* Terminierungsfunktion gefunden wird, dann terminiert das Programm sicher. Andernfalls müsste *für alle* möglichen Kandidaten gezeigt werden, dass diese nicht die Terminierung zeigen können. Da es unendlich viele mögliche Kandidaten gibt, kann dieser Teil nicht entschieden werden (der Algorithmus terminiert nicht).