Max Wisniewski , Alexander Steen

Tutor: Ansgar Schneider

Aufgabe 1

Beweisen Sie die Formel

$$\{true\}\ x\ :=\ 7;\ y\ :=x+3;\ \{y=10\}$$

im Hoare- Kalkül.
Lösung:

$$\dfrac{\dfrac{\{true\} \Rightarrow \{7=7\}}{\{7=7\}\ x\ :=\ 7;\ \{x=7\}}\ (1),(2) \qquad \dfrac{\dfrac{\{x=7\} \Rightarrow \{x+3=10\} \wedge \{true\}}{\{x=7\} \Rightarrow \{y=10_{y\leftarrow x+3}\} \wedge \{y=10_{y\leftarrow x+3}\ y\ :=x+3;\ \{y=10\}}\ (2)}{\{x=7\}y:=x+3\{y=10\}}\ (1)}{\{true\}\ x\ :=\ 7;\ y\ :=x+3;\ \{y=10\}}\ (3)$$

Legende:

(1) Zuweisung

(2) Konsequenz

(3) Komposition

Aufgabe 2

Beweisen Sie die Gültigkeit des Axioms (A.4), d.h. zeigen Sie die Gültigkeit der Formel:

$$\{Q[output.T/output]\}\ output\ T\ \{Q\}$$

im Bezug auf die denotationelle Semantik von WHILE.

Beweis:
Um die Gültigkeit des Axioms bezglich der denotationellen Semantik zu zeigen, muss bewiesen werden, dass $\forall z \in ZUSTAND\ :\ z \models Q[output.T/output] \Rightarrow C[\![output\,T]\!] \models Q$ gilt.
Wir beginnen also mit einer Skolemisierung des Quantors. Sei also $z \in ZUSTAND$ ein beliebiger aber fester Zustand mit $z = (s, input, output)$ dann ist nach denotationeller Semantik $z' = C[\![output\,T]\!] = (s, input, output')$, wobei $output' = output.T[\![T]\!]$

Wenn wir das Predikat unter dem Zustand nun auswerten gilt

$$Q[(\forall x \in s\ :\ x < -s(x)), output/\underline{output}, input/\underline{input}].$$

Nun betrachten wir also Doppelsubstitutionen

$$(1)\ \ Q[output.T/output][(\forall\, x \in s\ :\ x/s(x), output/\underline{output}, input/\underline{input}]$$

und

$$(2)\ \ Q[(\forall\, x \in S\ :\ x/s(x), output/\underline{output'}, input/\underline{input}].$$

Wir betrachten nun alle vorkommen vom ehemaligen $\{\underline{output}\}$ im Predikat Q. Kam in (1) $\{\underline{output}\}$ vor, so wurde der Term erst subsituiert und dann eingesetzt
$\Rightarrow \{\underline{output}\}[output.T/output][ifT \in SthenT/s(T), output/\underline{output}]$
$= \{output.T\}[ifT \in SthenT/s(T), output/\underline{output}]]$. Wenn T keine Variable ist, so wird beim substituieren vollständig ersetzt und wir setzen an dieser Stelle vorraus, dass die Auswertung von Termen in der axiomatischen Semantik äquivalent zur denotationellen Semantik ist.

Für (2) gilt nun $\{output\}[output/\underline{output'}] = \{output\}[output/output.T[\![T]\!]$, da nun T in beiden Fällen den selben Wert ergibt (vorrausgesetzt Termauswertung funktioniert korrekt), sehen wir, dass in jedem *output* nun der selbe Wert steht.

Da in simultanen Substitutionen kein Wert doppelt geschrieben darf (definition), kann der Wert nicht an anderer Stelle wieder verändert werden.

Aufgabe 3

Führen Sei einen Korrektheitsbeweis unter Verwendung der axiomatischen Semantik zu folgendem Programm:

```
sum := 0; read x;
while not (x=0) do
    sum ;= sum + x;
    read x;
output sum
```

Beweis:

Wäre wir sehr diabolisch und äußerst faul, könnten wir sagen, dass diese Aufgabenstellung unvollständig ist, da zu einem Korrektheitsbeweis von einem Programm immer eine Aussage gehört, die anhand des Programmes zu beweisen ist. Wir könnten uns also überlegen, dass wir zeigen wollen, dass nach der Ausfürhung $\{true\}$ gilt, d.h. das Programm terminiert korrekt und das einzige, was wir erhalten ist, dass die eingabe am Anfang nicht leer ist und mindestens eine 0 enthält.

Wir nehmen aber zum lernen hier an dieser Stelle einfach an, dass wir zeigen sollen, dass das folgende Hoaretripel gülitg ist: $\{\underline{output} = a', \underline{output} = e_1, e_2, ...e_t.0.y, \forall i \leq t\ :\ e_i \in ZAHL \setminus \{0\}\}$
C
$\{\underline{output} = a'.sum, \underline{input} = y, sum = \sum_{k=1}^{t} e_k\}$.
Wir verändern das Programm geringfügig, da es uns unmögich ist, eine Invariante zu finden, die alle Bedingungen erfüllt, ohne wirklich etwas über den Zustand der Grenze auszusagen:

```
i := t; sum := 0; read x;
while not (x=0) do
    i := i - 1;
    sum ;= sum + x;
    read x;
output sum
```

Die Invariante für die While-Schleife die wir wählen ist $I \equiv \{sum = \sum_{k=1}^{t-i} e_k, x = e_{t-i+1}, \underline{input} = e_{t-i+2}, .., e_t.0.y, t - i \geq 0, \forall r \leq i\ :\ e_r \in ZAHL \setminus \{0\}\}$

$\{\underline{output} = a', \underline{output} = e_1, ..., e_t.0.y, \forall i \leq t \ : \ e_i \in ZAHL \setminus \{0\}\}$
$\Rightarrow$
$\{0 = \sum_{k=1}^{0} e_k, \underline{input} = e_1, ..., e_t.0.y, 0 \geq 0, \forall r \leq t \ : \ e_r \in ZAHL \setminus \{0\}\}$

```
i := t;
```

$\{0 = \sum_{k=1}^{t-i} e_k, \underline{input} = e_{t-i+1}, ..., e_t.0.y, t-i \geq 0, \forall r \leq i \ : \ e_r \in ZAHL \setminus \{0\}\}$

```
sum := 0;
```

$\{sum = \sum_{k=1}^{t-i} e_k, \underline{input} = e_{t-i+1}, .., e_t.0.y, t-i \geq 0, \forall r \leq i \ : \ e_r \in ZAHL \setminus \{0\}\}$

```
read x;
```

$I = \{sum = \sum_{k=1}^{t-i} e_k, x = e_{t-i+1}, \underline{input} = e_{t-i+2}, .., e_t.0.y, t-i \geq 0, \forall r \leq i \ : \ e_r \in ZAHL \setminus \{0\}\}$

```
while not (x=0) do
    Da x ≠ 0kann, musst − i > 0gelten, daher gilt die Nachstehende Bedingung.
```

$\{sum = \sum_{k=1}^{t-i} e_k, x = e_{t-i+1}, \underline{input} = e_{t-i+2}, ..., e_t.0.y, t-i > 0, \forall r \leq i \ : \ ZAHL \setminus \{0\}\}$

```
    i := i - 1;
```

$\{sum = \sum_{k=1}^{t-i-1} e_k, x = e_{t-i}, \underline{input} = e_{t-i+1}, ..., e_t.0.y, t-i \geq 0, \forall r \leq i \ : \ ZAHL \setminus \{0\}\}$

```
    sum := sum + x;
```

$\{sum = \sum_{k=1}^{t-i} e_k, x = e_{t-i}, \underline{input} = e_{t-i+1}, ..., e_t.0.y, t-i \geq 0, \forall r \leq i-1 \ : \ ZAHL \setminus \{0\}\}$

```
    read x;
```

$I = \{sum = \sum_{k=1}^{t-i} e_k, x = e_{t-i+1}, \underline{input} = e_{t-i+2}, .., e_t.0.y, t-i \geq 0, \forall r \leq i \ : \ e_r \in ZAHL \setminus \{0\}\}$

```
od
```

$I \wedge \neg b = \{sum = \sum_{k=1}^{t-i} e_k, x = e_{t-i+1}, \underline{input} = e_{t-i+2}, .., e_t.0.y, i = 0, \forall r \leq i \ : \ e_r \in ZAHL \setminus \{0\}, x = 0\}$
$\overset{e_i=0}{\Rightarrow} sum = \sum_{k=1}^{t} e_k, x = e_{i-1} = 0, \underline{input} = y \forall r \leq 0 \ : \ e_r \in ZAHL \setminus \{0\}, i = 0\}$

$\underline{output} = a', \underline{input} = y, sum = \sum_{k=1}^{t} e_k\}$

```
output sum
```

$\{\underline{output} = a'.sum, \underline{input} = y, sum = \sum_{k=1}^{t} e_k\}$

Wir haben hier etwas getrickst um eine sinnvolle Invarinte zu erhalten, aber so lief der Beweis anstandslos durch. Es fehlt nur noch zu zeigen, dass die Rankingfunktion existiert um die Terminierung des Programmes zu zeigen.
Sei $r(i) = i$. Wie wir gesehen haben, ist $i > 0$ zu beginn jedes Schleifen durchlaufes (Argument im Hoarebeweis) und die Folge fällt streng monoton ($i := i - 1$). Daraus können wir schlussfolgern, dass

unser Programm immer terminiert.

Unser Hoaretripel, das wir aufgestellt haben, ist also gültig.