Karlsruher Institut für Technologie

Lehrstuhl für Programmierparadigmen
Sebastian Ullrich sebastian.ullrich@kit.edu
Max Wagner maximilian.wagner@kit.edu

Semantik von Programmiersprachen – SS 2019

http://pp.ipd.kit.edu/lehre/SS2019/semantik

Blatt 13: Axiomatische Semantik

Besprechung: 22.07.2019

1. Welche der folgenden Aussagen sind richtig, welche falsch? (H)

(a) $\{\,\texttt{x == 0}\,\}\,\texttt{x := x + 1}\,\{\,\texttt{0 <= x}\,\}$ ist eine Zusicherung.

(b) In $\{\,\texttt{x} = n\,\}\,\texttt{y := x * n}\,\{\,\texttt{y} = n^2\,\}$ ist n eine logische Variable.

(c) Wenn $\vdash \{\,P\,\}\,\texttt{skip}\,\{\,Q\,\}$, dann $P = Q$.

(d) Es gibt ein c mit $\vdash \{\,\mathbf{tt}\,\}\,c\,\{\,\mathbf{ff}\,\}$.

(e) Statt der Regel Ass_P wäre auch die Regel $\vdash \{\,P\,\}\,x\ \texttt{:=}\ a\,\{\,P[x \mapsto \mathcal{A}\,[\![a]\!]]\,\}$ sinnvoll.

2. Quadratwurzelberechnung (H)

Zeigen Sie, dass folgende Zusicherung in der axiomatischen Semantik ableitbar ist.

$\{\,\texttt{a} \geq 0\,\}$
```
n := 0; m := 1; k := 0;
while (k <= a - 1) do (n := n + 1; k := k + m; m := m + 2)
```
$\{\,\texttt{n} - 1 < \sqrt{\texttt{a}} \leq \texttt{n}\,\}$

Hinweis: Die Invariante der Schleife ist etwas trickreich. Berechnen Sie ein paar einfache Beispiele und versuchen Sie herauszufinden, in welcher Beziehung `m`, `n` und `k` zueinander stehen.

3. Schnelle Division (Ü)

Arithmetische Ausdrücke in Aexp enthalten keinen Divisionsoperator. In dieser Aufgabe sollen Sie ein While-Programm schreiben, das zwei ganze Zahlen mit logarithmischer Laufzeit (in der Größe der Zahlen) dividiert, und dieses korrekt beweisen.

(a) Ergänzen Sie folgendes Programmskelett, sodass das vollständige Programm c folgende Spezifikation in Form einer Zusicherung erfüllt, die angegebenen Invarianten verwendet und die Eingabevariablen `x` und `y` unverändert lässt:

$$\{\,\texttt{x} \geq 0 \land \texttt{y} > 0\,\}\,c\,\{\,\texttt{x} = \texttt{q} \cdot \texttt{y} + \texttt{r} \land 0 \leq \texttt{r} < \texttt{y}\,\}$$

```
... ; // Invariante I_0 sicherstellen
while (z <= x) do ( ... // Invariante I_0 erhalten und z verdoppeln);
... ; // Invariante I_1 sicherstellen
while (not (n == 0)) do ( ... // Invariante I_1 erhalten und n verkleinern)
```

wobei

$$I_0 = (\texttt{z} = \texttt{y} \cdot 2^{\texttt{n}}) \land \texttt{n} \geq 0 \land \texttt{x} \geq 0 \land \texttt{y} > 0$$
$$I_1 = (\texttt{x} = \texttt{q} \cdot \texttt{z} + \texttt{r}) \land 0 \leq \texttt{r} < \texttt{z} \land (\texttt{z} = \texttt{y} \cdot 2^{\texttt{n}}) \land \texttt{n} \geq 0$$

Für diese Aufgabe können Sie neben den bisherigen arithmetischen Ausdrücken auch den neuen Verschiebe-Ausdruck $a \gg 1$ mit der Semantik

$$\mathcal{A}[\![a \gg 1]\!]\,\sigma = \left\lfloor \frac{\mathcal{A}[\![a]\!]\,\sigma}{2} \right\rfloor$$

verwenden, wobei $\lfloor x \rfloor$ die rationale Zahl x auf die nächstkleinere ganze Zahl abrundet.

(b) Weisen Sie nach, dass obige Zusicherung für Ihr Programm c ableitbar ist. Verwenden Sie dabei die angegebenen Invarianten für die Schleifen.

(c) Terminiert Ihr Programm immer?

(d) Eliminieren Sie jetzt den Operator $a \gg 1$ in Ihrem Programm, verwenden Sie bei Bedarf Schleifen. Wie müsste man den Korrektheitsbeweis anpassen? In welcher Komplexitätsklasse liegt Ihr Programm jetzt?