1. Übungsblatt

Ausgabe: 04.12.1998

Aufgabe 1: Semantik von Programmiersprachen

Die Variation **IMP'** der Programmiersprache **IMP** hat folgende Syntax für Befehle:

$C ::= X := e \mid c_0; c_1; \mid$ `if` b `then` c_0 `else` c_1 `fi` $\mid$ `repeat` c `until` b `done`

wobei Boolsche und arithmetische Ausdrücke gegenüber **IMP** unverändert bleiben. Die Semantik des `repeat`-Konstruktes soll sein, daß das Kommando c solange ausgeführt wird, wie die Boolsche Bedingung b wahr ist, aber *mindestens einmal*.

1. Gebt eine operationale Semantik für **IMP'** in der bekannten Weise als Regeln der Form $\langle c, \sigma \rangle \to \sigma$, die die Auswertung des Kommandos c im Zustand σ beschreiben.
2. Gebt eine denotationale Semantik für **IMP'** in der bekannten Form in der jedes Kommando c durch eine partielle Funktion $\mathcal{D}[\![c]\!]$ von Zuständen nach Zuständen denotiert wird.
3. Skizziert den Beweis, daß beide Semantiken äquivalent sind.

Wo die Semantiken und der Beweis gegenüber **IMP** unverändert bleiben, reicht eine Skizze der Semantik oder des Beweises.

(Dieses ist Aufgabe 5.9, Seite 68, in Winskels Buch.)

Aufgabe 2: Programmverifikation

Die ganzzahlige Wurzel einer Zahl $n \in \mathbb{N}$ ist definiert als die größte ganze Zahl, die kleiner oder gleich der Wurzeln von n ist:

$$sqrt(n) = k \Longleftrightarrow k^2 \leq n \wedge (k+1)^2 > n$$

Folgendes **IMP**-Programm berechnet in der Variablen M die ganzzahlige Wurzel der natürlichen Zahl in der Variablen N:

$SQRT \equiv$
```
M := 0;
while (M+1)*(M+1)<= N do M:= M+1 done
```

1. Verifiziert $SQRT$ formal im Hoare-Kalkül. Dazu wird zuerst eine Spezifikation $\{P\}SQRT\{Q\}$ aufgestellt, die dann mit den Regeln des Hoare-Kalküls bewiesen wird. Der entscheidende Schritt ist wie immer, die Invariante für die Schleife zu finden.

Aufgabe 3: Entwicklung durch Transformation

1. Gebt eine operationale Spezifikation für die Funktion

   ```
   length :: 'a list => nat
   ```

   welche die Länge einer Liste berechnen soll.

   (Vgl. die operationale Spezifikation der Listenumkehrfunktion `rev :: 'a list => 'a list` aus der Vorlesung, siehe Folien.)

2. Transformiert diese Spezifikation in eine endrekursive (*tail*-rekursive) Form, ähnlich der Transformationen für die Fakultät aus der Vorlesung.