Korrekte Software: Grundlagen und Methoden SoSe 17

Serge Autexier
Christoph Lüth

3. Übungsblatt

Ausgabe: 27.04.17
Abgabe: 04.05.17

In diesem Übungsblatt wollen wir die theoretischen Konzepte zur Semantik von Programmen praktisch umsetzen. Hierzu werden wir für unsere bereits eingeführte Sprache C0 die Funktionen zur operationalen und denotationalen Semantik implementieren. Dazu laden Sie sich das Rahmenwerk für das zur Übung 3 gehörende C0-Analyse-Werkzeug (*C0 Analysing Tool* oder `cat`) aus dem von der Webseite verlinkten Git-Repository herunter.

3.1 *Operationale Semantik* *10 Punkte*

Implementieren Sie gemäß den der Vorlesung angegebenen Regel die operationale Semantik für C0. Dazu vervollständigen Sie in der Datei `OperationalSemantics.scala` (im Verzeichnis `src/main/scala`) die Funktionen evaluateExpr und evaluateStmt um die Fälle für alle in C0 betrachteten arithmetischen und boolschen Ausdrücke (Expr), bzw. alle fehlenden Programmkonstrukte (Stmt).

Sie können Ihre Funktionen testen, indem Sie das `cat`-Werkzeug aus der Kommandozeile aufrufen. Die Argument hierzu sind `--opsemeval=<fun> --state=<state>` um den Rumpf der Funktion `<fun>` im Zustand `<state>` auszuführen. Darüber hinaus sind in SemanticsTest.scala (im Testverzeichnis src/test/scala) einige Tests vorbereitet.

3.2 *Denotationale Semantik* *10 Punkte*

Implementieren Sie gemäß der Definition aus der Vorlesung eine denotationale Semantik für C0. Dazu vervollständigen Sie in der Datei `DenotationalSemantics.scala` die Funktionen `dsExpr` und `dsStmt` um die Fälle für alle in C0 betrachteten arithmetischen und boolschen Ausdrücke (Expr) bzw. Programmkonstrukte (Stmt). Das Denotat eines Ausdrucks bzw. eines Programms ist dann ein Wert vom Typ

```
type DenSem[Int] = Function[basicState, Option[Int]] // Ausdrücke
type DenSem[BasicState] = Function[BasicState, Option[BasicState]] // Statements
```

oder mit anderen Worten, partielle Funktion von Zuständen (BasicState) auf ganze Zahlen bzw. Zustände.

Für While-Schleifen verwenden Sie die vorgegebenen Funktion gamma , die der in der Vorlesung gegebenen Fixpunktapproximationsfunktional entspricht, außer dass sie zusätzliche als erstes Argument das Denotat der Schleifenbedingung und als zweites Argument das Denotat des Schleifenrumpfes erwartet. Das dritte Argument ist das φ bzw. ψ aus der Vorlesung, und repräsentiert die bis dahin aufgefalteten Schleifenrümpfe.[1]

Zum Auruf der denotationalen Semantik aus der Kommandozeile rufen Sie `cat` mit den Argumenten `--densemeval=<fun> --state=<state>` auf.

[1] Der Fixpunkt wird hier von Scala automatisch durch die rekursive Definition von gamma gebildet.