Universität Bremen

Korrekte Software: Grundlagen und Methoden SoSe 2016

Serge Autexier
Christoph Lüth

7. Übungsblatt

Ausgabe: 6.06.16
Abgabe: 16.06.16

7.1 *Verification Condition Generation Revisited* *15 Punkte*

In diesem Übungsblatt werden diewir Funktionen zur Berechnung der Verifikationsbedingungen des C-Analysewerkzeug `cat` aus Übung 5.1 erweitern, um mit Funktionen, Prozeduren, Strukturen, Feldern und Lokationen (*Lexp*) umgehen zu können.

Hierzu sind mehrere Schritte notwendig:

(1) die korrekte Behandlung von Zuweisungen (*Lexp* auf der linken Seiten statt nur eine Variable, Ersetzung des ganzen Terms *l* statt der einzelnen Variablen);

(2) die korrekte Behandlung von Funktionsdefinition mit **return** und Spezifikation mit **\result** und **\old**;

(3) die korrekte Behandlung von Funktionsaufrufen.

Für Schritt (1) und (2) haben wir das Rahmenwerk erweitert, so dass eine Methode

```
class Term {
  def replace(lhs: Term, rhs: Term) : Term
```

bereitsteht, die in dem Term die linke Seite lhs durch die rechte Seite rhs ersetzt. Die Methode ersetzt *nicht* unterhalb von **\old**.

Die Klasse Env wird so erweitert, dass die Nachbedingung der momentan analysierten Funktion im Environment gehalten werden kann, so dass bei **return** darauf zugegriffen werden kann.

Für Schritt (3) muss das Environment so erweitert werden, dass man einem Funktionsbezeicher die Spezifikation (Vor- und Nachbedingung) zuordnen kann.

Folgende Dateien (in `src/test/examples`) können als Testdateien für Schritt (1) und (2) genutzt werden (die Ausgabe der Referenzimplementierung ist im Repository zu finden):

`test-vc-structs.c`, `test-vc-fun.c`, `test-vc.fun-with-arrays.c`, `test-vc-arrays.c`, `test-vc-fac.c`

7.2 *Proving It* *5 Punkte*

Die meisten Beispiele aus dem vorherigen Aufgabenteil erzeugten triviale Beweisverpflichtungen, lediglich diejenigen für die Funktionen zero und max (aus `test-vc-fun-with-arrays.c`) sind nicht komplett trivial. Deshalb wollen wir diese jetzt in Isabelle beweisen.

1. Übersetzen Sie die Beweisverpflichtungen in eine Isabelle-Theorie mit neun Beweiszielen.

   (*2 Punkte*)

2. Beweisen Sie folgendes Hilfslemma in Isabelle:

   ```
   lemma step: "(!j. 0 <= j & j< (n::int)-1 --> P(j) ) & P(n-1)
                --> (!j. 0 <= j & j< n --> P(j))"
   ```

   (*1 Punkte*)

3. Beweisen Sie die Beweisverpflichtungen. Einige sind einfach, für andere werden Sie das Lemma step benötigen.

   (*2 Punkte*)

15:56:34 6. Juni 2016