Korrekte Software: Grundlagen und Methoden SoSe 17

Serge Autexier
Christoph Lüth

7. Übungsblatt

Ausgabe: 29.05.17
Abgabe: 08.06.17

7.1 *Verification Condition Generation* *20 Punkte*

In diesem Übungsblatt wollen wir das C-Analysewerkzeug `cat` um eine Funktion zur Berechnung der Verifikationsbedingungen erweitern, in Fachkreisen auch als *verification condition generation (VCG)* bekannt.

Wir konzentrieren uns erst einmal auf die Verifikationsbedingungen aus der schwächsten Vorbedingung. Die Eingabe ist dabei immer ein C-Programm, bei dem Funktionen mit Vor- und Nachbedingungen (erste optional) annotiert sind, und zwingend jede Schleife mit einer Invariante. Hier ist das notorische Beispiel:

```
int fac(int n)
/** pre 1 ≤ n;
    post p == faculty(n);
  */
{
 int p;
 int c;

 p= 1;
 c= 1;
 while (c≤ n)
    /** inv p == faculty(c− 1) && c ≤ n+1;  */  {
    p= p∗c;
    c= c+1;
    }

}
```

Unser Analysewerkzeug nimmt ein annotiertes Programm, und berechnet daraus die Verifikationsbedingungen mit Hilfe der approximierten schwächsten Vorbedingung (*awp*). Wir könnten dazu zwei Funktionen schreiben

```
def wvc(ctxt: Env)(s: Stmt, post: Term, ret: Term): List[Term]
def awp(ctxt: Env)(s: Stmt, post: Term, ret: Term): Term
```

die exakt den in der Vorlesung vorgestellten entsprechen (*7 Punkte*); aus Effizienzgründen, damit das Programm nicht zweimal traversiert werden muss, bietet es sich an, beide in *einer* Funktion zusammenzufassen:

```
def awpvc(ctxt: Env)(s: Stmt, post: Term, ret: Term): (Term, List[Term])
```

Die VCG ist in der Klasse WeakestPrecondition zu implementieren. Sie finden den entsprechenden Rahmen im git-Repository der Veranstaltung. Lösen Sie die Aufgabe, so dass die entsprechenden Tests (`sbt test`) durchlaufen.

Sie können zuerst die VCG für den nicht-erweiterten Hoare-Kalkül (also ohne **return** und Funktionsaufruf) implementieren (*13 Punkte*), und ihre Implementierung dann erweitern.