Korrekte Software: Grundlagen und Methoden SoSe 17

Serge Autexier
Christoph Lüth

9. Übungsblatt

Ausgabe: 22.06.17
Abgabe: 29.06.17

9.1 *Stärkste Nachbedingungen* *20 Punkte*

In diesem Übungsblatt wollen wir den Funktionsumfang unseres C-Analysewerkzeugs `cat` noch um die Vorwärtsberechnung von Verifikationsbedingungen mit Hilfe der stärksten Nachbedingung (*strongest postcondition*) erweitern.

Die Eingabe ist dabei wie immer ein C-Programm, bei dem Funktionen mit Vor- und Nachbedingungen (erste optional) annotiert sind, und zwingend jede Schleife mit einer Invariante.

Analog zu der Berechung von Verifikationsbedingungen mit Hilfe der stärksten Nachbedingung (*awp*) nimmt unser Analysewerkzeug ein annotiertes Programm, und berechnet daraus die Verifikationsbedingungen mit Hilfe der approximierten stärksten Nachbedingung (*asp*). Wir schreiben dazu zwei Funktionen:

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

die exakt den in der Vorlesung vorgestellten entsprechen.

Die VCG ist in der Klasse StrongestPostcondition zu implementieren. Sie finden den entsprechenden Rahmen im git-Repository der Veranstaltung. Lösen Sie die Aufgabe so dass die entsprechenden Tests ein sinnvolles Ergebnis liefern (d.h. `sbt "testOnly cat.VCGTest"`).

12 Teilpunkte

Direkt wie in der Vorlesung vorgestellt wird so aus einer Sequenz von n Zuweisungen eine Nachbedingung mit n Existenzquantoren. Um das zu vermeiden, wollen wir eine spezialisierte Regel für den Fall entwerfen, dass die Vorbedingung schon von der Form $\exists S.P(S) \wedge \sigma = f(S)$ ist. Formulieren Sie diese Regel erst als Hoare-Regel, oder als neue Klausel für die Definition von *asp*, und implementieren Sie sie dann.

3 Teilpunkte

Die so generierten Verifikationsbedingungen sind nicht vereinfacht (insbesondere sind *Read*-Operationen nicht vereinfacht), und damit sehr länglich. Implementieren Sie deshalb eine Funktion

```
simpState(t: Term): Term
```

welche einen Term mit Hilfe der folgenden Regeln vereinfacht:

$$\begin{aligned} Read(Upd(s,l,v),l) &= v \\ Read(Upd(s,l,v),m) &= Read(s,m) \quad \text{wenn } l \neq m \\ Upd(Upd(s,l,v),l,w) &= Upd(s,l,w) \end{aligned}$$

Für die Ungleichheit $l \neq m$ können Sie nur davon ausgehen, dass benannte Variablen (`Var(Identifier(i), t, k)`) ungleich sind, wenn der Bezeichner `i` ungleich ist.

5 Teilpunkte