Korrekte Software: Grundlagen und Methoden SoSe 17

Serge Autexier
Christoph Lüth

1. Übungsblatt

Ausgabe: 10.04.17
Abgabe: 20.04.17

In diesem Übungsblatt machen wir unsere ersten praktischen Erfahrungen mit der Sprache C0 und der Implementierung der zugehörigen Werkzeuge. Dazu laden Sie sich das Rahmenwerk für das C0-Analyse-Werkzeug (*C0 Analysing Tool* oder `cat`[1]) von der Webseite herunter.

1.1 *Variable Initialisation Analysis* — *20 Punkte*

Wir wollen `cat` um eine Analyse erweitern, die prüft, ob alle Variablen auch vor der Benutzung initialisiert wurden. (Ein klassisches Problem der statischen Programmanalyse.)

Ihr Verfahren soll den Trait Analysis implementieren, und eine Methode

```
def run(env:Context, body: Stmt): List[(Identifier, Position)]
```

implementieren, welche eine Liste der Variablen (Identifier) und Positionen zurückgibt, an der diese Variable ohne Initialisierung benutzt wird.

Hinweise:

1. Implementieren Sie zwei Funktionen, welche eine einzelne Anweisung bzw. eine Liste von Anweisungen analysieren:

   ```
   type PosId= (Identifier, Position)
   def checkStmt(s: Stmt, varmap: HashSet[Identifier]):
                 (HashSet[Identifier], List[PosId]) = ???
   def checkStmts(stmts: List[Stmt], varmap: HashSet[Identifier]):
                 (HashSet[Identifier], List[PosId]) = ???
   ```

   Der Rückgabewert dieser Funktionen ist eine Menge der nach dieser Anweisung initialisierten Bezeichner, sowie das eigentliche Ergebnis (Liste der nichtinitialisert benutzten Bezeichner mit Position). Die Menge der benutzten Bezeichner wird mit jeder Zuweisung angereichert. Für Fallunterscheidungen muss eine Variable in beiden Zweigen initialisiert werden, damit sie nach der Falluntercheidung initialisiert ist.

2. Eine nützliche Hilfsfunktion ist

   ```
   def identifiersOf(e: Expr): HashSet[Identifier] = ???
   ```

   welche die Menge der in einem Ausdruck auftretenden Bezeichner berechnet.

3. Sie können davon ausgehen, dass die zu prüfenden Funktionen keine Parameter haben (ansonsten müssten wir Parameter gesondert behandeln, da sie immer initialisiert sind).

4. Sie brauchen keine Datentypen außer den elementaren zu behandeln, und können insbesondere die Operationen auf Feldern und Strukturen (**struct**) vernachlässigen.

5. Sie brauchen nur Zuweisungen, Falluntercheidungen, Schleifen und Sequenzen von Anweisungen zu behandeln.

6. Der Test VarInitSpec beinhaltet einige elementare Tests der zu implementierenden Funktion, ist aber nicht vollständig.

[1] Unsere Marketingabteilung behauptete, „Katzen gehen im Internet immer“, deshalb der Name.