Formale Grundlagen der Informatik III: Übung 4

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 2014/2015
Wird in der Übungsgruppe vom 17. Dezember bis 16 Januar behandelt

Aufgabe 1 CBMC: Model Checking

Auf Foliensatz 6 wurde die grobe Vorgehensweise zur Verifikation eines Programms ab Folie 20 vorgestellt. Versuchen Sie händisch die Schritte 1-5 des Ablaufs auf den folgenden Code anzuwenden.

```
int foo(int a, int b){
  int i;
  int result = a++;
  int steps = 3;
  for(i = 0; i < steps; i++){
    result += b;
    if(result > 30)
      break;
  }
  return result;
}
```

Aufgabe 2 CBMC: GCD Example (continued)

In der letzten Übung hatten Sie abschließend eine Funktion für die Berechnung des gcd aus zwei Zahlen vom Typ Integer in C implementiert. Bitte nutzen Sie Ihre Implementierung für die folgenden Aufgaben.
Sollten Sie keine funktionierende Implementierung haben, so können Sie mit folgendem Code arbeiten:

```
int gcd (int a, int b){
  int c;
  while (a != 0){
     c = a;
     a = b % a;
     b = c;
  }
  return b;
}
```

a) Versuchen Sie die korrekte Funktion Ihrer gcd Funktion durch CBMC verifizieren zu lassen. Formulieren Sie hierfür bei Bedarf Assertions. Starten Sie die Verifikation anschließend mit dem Befehl: *cbmc gcd.c --function gcd* (Je nach Implementierung müssen Sie Datei und Funktionsnamen anpassen!). Was stellen Sie fest?

b) In der vorherigen Aufgabe haben Sie betrachtet, wie Loop Unwinding funktioniert. Kann uns dieses Wissen bei dem vorherigen Problem weiterhelfen? Versuchen Sie die Zahl der Unwinding Operationen in CBMC zu beschränken. Was können Sie dabei beobachten?

c) Mit dem Befehl *cbmc gcd.c --function gcd --unwind 2* können Sie die Zahl der Unwindings beschränken. Was passiert, wenn Sie die Verifikation mit dem Parameter *--unwind 2* ausführen? Können Sie dieses Verhalten erklären? Führen Sie hierfür die Unwinding Operation händisch durch.

Aufgabe 3 CBMC: Verifizierter Taschenrechner

Wir wollen in dieser Aufgabe Funktionen für einen verifizierten Taschenrechner programmieren. Der Taschenrechner soll die vier grundlegende Rechenoperationen (+,-,*,/) unterstützen und eine erweiterte Operation (kgV) unterstützen.

a) Implementieren Sie die Funktionen *add* und *sub* zum Addieren und Subtrahieren von zwei Zahlen vom Typ **unsigned int**. Die Verwendung der Operatoren + und - ist hierfür nicht erlaubt.

b) Schreiben Sie anschließend Prüffunktionen, welche das Ergebnis Ihrer eigenen Funktionen gegen das Ergebnis von den Standardoperatoren **+** und **-** verifizieren.

c) Erweitern Sie nun Ihren Taschenrechner um die Funktionen *mul* und *div*. Sie dürfen hierbei die Operatoren * und / nicht verwenden. Eine Verwendung Ihrer vorherigen Funktionen ist zulässig.

d) Verifizieren Sie die Funktion von den zwei neuen Funktionen.

e) Schreiben Sie nun eine Funktion zur Berechnung des kgV und verifizieren Sie deren korrekte Funktionsweise.

Aufgabe 4 CBMC: Verifikation / Bugsuche

Das folgende Programm soll eine zugeteilte Liste traversieren. Es bestehen leider einige Fehler innerhalb des Codes.

```
#include <stdlib.h>
struct node {
  struct node* succ;
};
void traverse(struct node* n) {
  traverse(n->succ);
}
int main(int argc, char **argv) {
  struct node *list = (void*)0;
  struct node *new_node;
  int i;
  int size = 10;
  for (i = 0; i < size; i++) {
    new_node = malloc(sizeof(struct node));
    new_node->succ = list;
    list = new_node;
  }
  traverse(list);
  return 0;
}
```

a) Finden Sie diese Fehler mit der Hilfe von CBMC und korrigieren Sie die betroffenen Stellen.

b) Zeigen Sie, dass die Zahl der Aufrufe der Funktion *traverse* in Zusammenhang mit der Länge der Liste steht, indem Sie eine oder mehrere Assertions formulieren, die von CBMC gerpüft werden.

Hinweis: Denken Sie an die Funktionsweise von Loop Unwinding. Die Nutzung von Assertions kann erforderlich sein.