Softwaretechnik (WS 2005/2006) - Übungsblatt 13

Jan Hendrik Dithmar - Matrikelnummer 2031259
Übungsgruppe Mi, 8-10 Uhr
Dirk Heine - Matrikelnummer 2030941
Übungsgruppe Di, 14-16 Uhr

Rückgabe: Übungsgruppe Di, 14-16 Uhr

Aufgabe 1

(a)

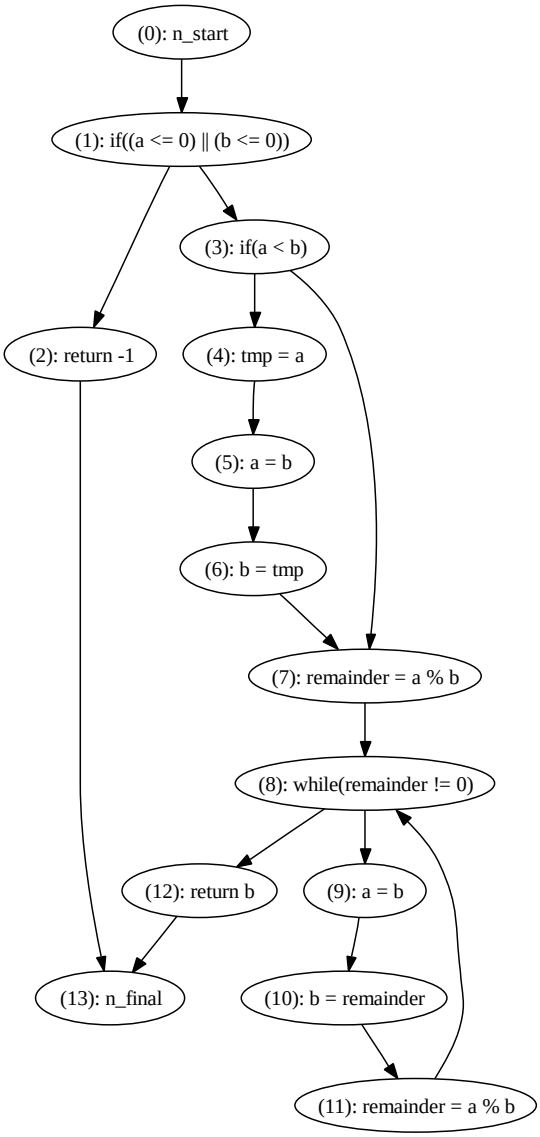

Abbildung 1: Aufgabe 1a

(b)

Anweisungsüberdeckung bedeutet, dass jeder Knoten im Kontrollflussgraphen einmal durchlaufen werden muss.

1. Testfall: `ggt(0,0)`
   Es wird die erste `if`-Schleife einmal durchlaufen. Es werden die Knoten (0), (1), (2), (13) erreicht.

2. Testfall: `ggt(7,8)`

   (a) Es wird die zweite `if`-Schleife durchlaufen.

   (b) Es wird einmal die `while`-Schleife durchlaufen bis 1 zurückgegeben wird.

   Es werden die Knoten (0), (1), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12), (13) abgedeckt.

Somit wurden alle Knoten mindestens einmal durchlaufen.

(c)

Um Zweigüberdeckung zu erreichen, muss jede Kante im Kontrollflussgraphen einmal ausgeführt werden. Die Zweigüberdeckung schließt die Anweisungsüberdeckung mit ein. Um im vorliegenden Beispiel Zweigüberdeckung zu erreichen, muss ein Testfall eingeführt werden, der die Transition von Knoten (3) nach Knoten (7) abdeckt. Dazu kann man z.B. einen dritten Testfall `ggt(5,3)` einführen.

(d)

Ein Problem bei der Pfadüberdeckung stellt sich dadurch, dass sie eigentlich keinen praktischen Nutzen hat. Die meisten Programme haben so viele mögliche Pfade, dass man unmöglich alle testen kann. Man beschränkt sich daher in der Regel auf den Boundary Interior-Pfadtest oder den strukturierten Pfadtest, die beide die Anzahl der zu testenden Pfade einschränken.

(e)

Die Idee besteht darin, die Anzahl der zu überprüfenden Pfade zu reduzieren, indem keine Pfade geprüft werden, bei denen Schleifen mehr als zweimal durchlaufen werden. Bei der Pfadüberdeckung muss sichergestellt sein, dass alle Pfade mindestens einmal durchlaufen werden. Für den Boundary Interior-Test fehlt noch der Pfad außerhalb aller Schleifen. Dazu kann man z.B. den Testfall `ggt(1,1)` einführen, in dem keine Schleife betreten wird.

(f)

```
$ gcov ggt
File 'ggt.c'
Lines executed: 93.75% of 16
ggt.c:creating 'ggt.c.gcov'

ggt.c.gcov:

        -:    0:Source:ggt.c
        -:    0:Graph:ggt.gcno
        -:    0:Data:ggt.gcda
        -:    0:Runs:1
        -:    0:Programs:1
        -:    1:#include <stdio.h>
        -:    2:
function ggt called 1 returned 100% blocks executed 91%
        1:    3:int ggt(int a, int b) {
        1:    4:    int tmp, remainder;
        1:    5:    if((a<=0) || (b<=0)) {
    #####:    6:        return -1;
        -:    7:    }
        1:    8:    if (a < b) {
        1:    9:        tmp = a;
        1:   10:        a = b;
        1:   11:        b = tmp;
        -:   12:    }
        1:   13:    remainder = a % b;
        7:   14:    while (remainder != 0) {
        6:   15:        a = b;
        6:   16:        b = remainder;
        6:   17:        remainder = a % b;
        -:   18:    }
        1:   19:    return b;
        -:   20:}
        -:   21:
function main called 1 returned 100% blocks executed 100%
        1:   22:int main() {
        1:   23:    printf("ggt(11324,23443)=%d\n",ggt(11324,23443));
        -:   24:}
        -:   25:
```

Aufgabe 2

(a)

Angenommen, eine Methode nimmt als Argumente Parameter die aktuelle Außentemperatur.

```
bool OutOfRangeTemp(float a) {
  if((a < -10) || (a > 40)) return true;
}
```

Da die Temperatur aber entweder < -10 Grad oder > 40 Grad sein kann, gibt es keinen Fall, in dem beide atomaren Bedingungen wahr werden können. Somit ist eine Mehrfach-Bedingungsüberdeckung nicht zu realisieren.

(b)

A: `!isAlreadyLoggedIn(user)`

B: `user.hasNeededCash()`

C: `user.isSuperUser()`

Testfall	1	2	3	4	5	6	7
A	T	F	F	T	T	F	T
B	F	T	F	T	F	T	T
A $\wedge$ B	F	F	F	T	F	F	T
C	F	F	T	F	T	T	T
A $\wedge$ B $\vee$ C	F	F	T	T	T	T	T

Aufgabe 3

(a)

Beim Mutationstesten werden absichtlich Fehler in ein Programm eingebaut, um die Güte und Vollständigkeit der Testfälle zu überprüfen.

(b)

Es werden lediglich 50 % der Fehler gefunden. Dies ist zu wenig. Die Testsuite muss insofern erweitert werden, dass entweder mehr Tests eingefügt werden oder Tests korrigiert werden, die bisher „falsch“ geprüft haben. Es sollte eine Mutanten-Kill-Quote von mindestens 90 % erreicht werden.

(c)

Die Restfehleranzahl E kann man wie folgt schätzen:

$$E = (M - X) \cdot \frac{N}{X}$$

Dabei sind:

- N die Gesamtanzahl der Mutationen
- M die Anzahl der gefundenen Fehler (incl. Mutanten)
- X die Anzahl der gefundenen Mutanten

Also folgt für E:

$$\begin{aligned} E &= (M - X) \cdot \frac{N}{X} \\ &= (1958 - 1950) \cdot \frac{2000}{1950} \\ &= 8,\overline{205128} \\ &\approx 8 \end{aligned}$$

Die Anzahl der Restfehler liegt bei ungefähr 8 (bzw. nicht mathematisch, sondern sinnvoll gerundet bei 9) Fehlern.