Softwarekonstruktion – Übung 6

6 Whitebox-Testen II

6.1 Grenze-Inneres-Überdeckung

Das gegebene Programmsegment soll mit einem Kontrollfluss-Testverfahren getestet werden:

```
1 while ( (a<b) OR (c<d) ) {
2     a:= a + 1;
3     c:= c + 1;
4 }
```

1. Geben Sie einen minimalen Satz von Testdaten an, der für die Bedingung der while-Schleife die minimal bestimmende Mehrfachbedingungsüberdeckung erfüllt.
2. Geben Sie einen minimalen Satz von Testdaten an, der die Grenze-Inneres-Überdeckung erfüllt.

Gegeben sei nun ein Programmsegment und sein aus einem Refactoring entstandenes Gegenstück.

```
1 factorial= counter;
2 counter--;
3 while ( counter > 0 ) {
4     factorial= factorial * counter;
5     counter--;
6 }
7 System.out.println(factorial);
```

```
1 /* Refactoring */}
2 factorial=1;
3 do  {
4     factorial= factorial * counter;
5     counter--;
6 } while ( counter > 0 )
7 System.out.println(factorial);
```

3. Geben Sie jeweils Testfälle für eine Grenze-Inneres-Überdeckung an. Worin unterscheiden sich die beiden Testfälle? Wodurch kommt der Unterschied zu Stande?

Auch das folgende Programmsegment soll mit einem Kontrollfluss-Testverfahren getestet werden:

```
1  x=a; y=b;
2
3  /* ggT (klassisch): */
4   while (a!= b) {
5       if (a>b) { a= a-b; }
6       else {b= b-a; }
7   }
8
9  /* ggT (modern): */
10  tmp = 0;
11  while (y != 0) {
12      tmp = x % y;
13      x = y;
14      y = tmp;
15  }
```

4. Geben Sie für den Testdatensatz $\{(8,2),(5,5),(6,3)\}$ den Grenze-Inneres-Überdeckungsgrad an.
5. Geben Sie einen minimalen Satz von Testdaten an, der die Grenze-Inneres-Überdeckung erfüllt.

Hinweis: Konvention für die Angabe von Testdatensätzen ist: ist nichts angegeben, wird von alphabetischer Sortierung ausgegangen.
Alternativ Variablen explizit zuweisen: $\{(a=2,d=1,c=2,b=1),(...),...\}$

1. z.B. $\{(2,1,2,1),(1,1,1,2),(1,2,1,1)\}$
   - Bei dieser Aufgabe ist nach einem Testdatensatz gefragt, der für die **Bedingung der while-Schleife** das Testkriterium erfüllt. Bei der Bestimmung der Testdaten sollen daher nicht Variablenmanipulationen durch den Programmablauf betrachtet werden.
2. Schleife muss 0x, 1x, > 1x durchlaufen werden. z.B. $\{(2,1,2,1),(1,1,1,2),(1,5,1,5)\}$
   - Hier sind 3 Kriterien unabhängig von einander zu erfüllen, somit sind 3 Testdaten nötig.
3. $T_1 = \{1,2,3\}$ und $T_R = \{1,2\}$. Der Unterschied besteht in den 0 Schleifendurchläufen
4. Die erste Schleife wird überdeckt, die zweite nicht, deswegen liegt ein Grenze-Inneres-Überdeckungsgrad von $\frac{1}{2}$ vor.
5. Man benötigt einen Testdatensatz, der $(0,0)$ enthält, sonst kann die zweite Schleife nicht keinmal durchlaufen werden. Außerdem müssen noch einfache und mehrfache Schleifendurchläufe abgedeckt werden, z.B. mit: $(6,3)$ und $(35,14)$ oder $(3,9)$. Daraus ergibt sich z.B. der Testdatensatz $\{(0,0),(6,3),(35,14)\}$.

6.2 Kontrollflussbezogenes Testen – Testfälle II

Das gegebene Programmsegment soll mit einem Kontrollfluss-Testverfahren getestet werden:

```
1 read(x,y,z)
2 if ( (x>0) AND (y>0) AND (z>0) ) { x:=x*y*z }
3 else { x:=y }
4 if ( x+y+z>0 ) { z:=-x }
```

Bei den Testdaten ist zu beachten, dass die zweite if-Abfrage ebenfalls überdeckt und ggf. überhaupt erreicht wird.

6.2.1 Geben Sie eine minimale Testmenge für eine Anweisungsüberdeckung an.

{(1,1,1),(-1,-1,-1)}

6.2.2 Geben Sie eine minimale Testmenge für eine einfache Bedingungsabdeckung an.

{(1,1,-1),(-1, -1, 1)}

6.2.3 Geben Sie eine minimale Testmenge für eine Mehrfachbedingungsabdeckung an.

{(1,1,1),(1, 1, -1),(1, -1, 1),(1, -1, -1),(-1, 1, 1),(-1, 1, -1),(-1, -1, 1),(-1, -1, -1)}

6.2.4 Geben Sie eine minimale Testmenge für eine minimal bestimmende Mehrfachbedingungsabdeckung an.

{(1,1,1),(1, 1, -1),(1, -3, 1),(-1, 1, 1)}

6.2.5 Geben Sie eine minimale Testmenge für eine Pfadabdeckung an.

Es gibt keine Pfadabdeckung, die die Bedingungen "`(x>0) AND (y>0) AND (z>0)`" und "`(x+y+z>0)`" voneinander abhängig sind und deswegen $x + y + z > 0$ immer gilt, wenn $((x > 0) AND (y > 0) AND (z > 0))$ gilt.

6.3 Kontrollflussbezogenes – Refactoring

Bei einem Refactoring wurde das Programmsegment aus Aufgabe 6.2 wie folgt geändert:

```
1  read(x,y,z)}
2  if ( x>0 ) {
3     if ( y>0 ) {
4        if ( z>0  ) { x:=x*y*z }
5        else  { x:=y }
6     }
7     else  { x:=y }
8  }
9  else  { x:=y }
10 if ( x+y+z>0 ) { z:=-x }
```

Das Programmsegment soll nun erneut getestet werden:

6.3.1 *Geben Sie eine minimale Testmenge für eine Anweisungsüberdeckung an. Vergleichen Sie Ihr Ergebnis mit dem Ergebnis von Aufgabe 6.2.1 und begründen Sie gegebenenfalls den Unterschied.*

{(1,1,1), (1,1,0), (1,0,0), (0,0,0)}
Durch die Änderung der Und-Bedingung in eine Verschachtelung und die mehrfach eingefügte Anweisung müssen nun auch mehre Anweisungen überdeckt und somit mehr Fälle berücksichtigt werden.

6.3.2 *Geben Sie eine minimale Testmenge für eine einfache Bedingungsabdeckung an. Vergleichen Sie Ihr Ergebnis mit dem Ergebnis von Aufgabe 6.2.2 und begründen Sie gegebenenfalls den Unterschied.*

{(1,1,1), (1,1,0), (1,0,0), (0,0,0)}
Durch die Änderung der Und-Bedingung in eine Verschachtelung muss hier sichergestellt werden, dass die entsprechenden Bedingungen auch erreicht werden.

6.3.3 *Geben Sie eine minimale Testmenge für eine Mehrfachbedingungsabdeckung an. Vergleichen Sie Ihr Ergebnis mit dem Ergebnis von Aufgabe 6.2.3 und begründen Sie gegebenenfalls den Unterschied.*

{(1,1,1),(1,1,0),(1,0,0),(0,0,0)}; Durch die Aufteilung der AND-Bedingung ändert sich der Test, da nun keine Kombination von atomaren Prädikaten mehr betrachtet werden muss. Dafür ist sicherzustellen, dass auch die jeweiligen Belegungen wirklich mindestens einmal ausgewertet werden. Die Überdeckung fällt also mit der C_2-Überdeckung zusammen.

6.3.4 Geben Sie eine minimale Testmenge für eine minimal bestimmende Mehrfachbedingungsabdeckung an. Vergleichen Sie Ihr Ergebnis mit dem Ergebnis von Aufgabe 6.2.4 und begründen Sie gegebenenfalls den Unterschied.

{(1,1,1),(1,1,0),(1,0,0),(0,0,0)}; fällt auch mit der C_2-Überdeckung zusammen.