6. Übungsblatt zu Software Qualität

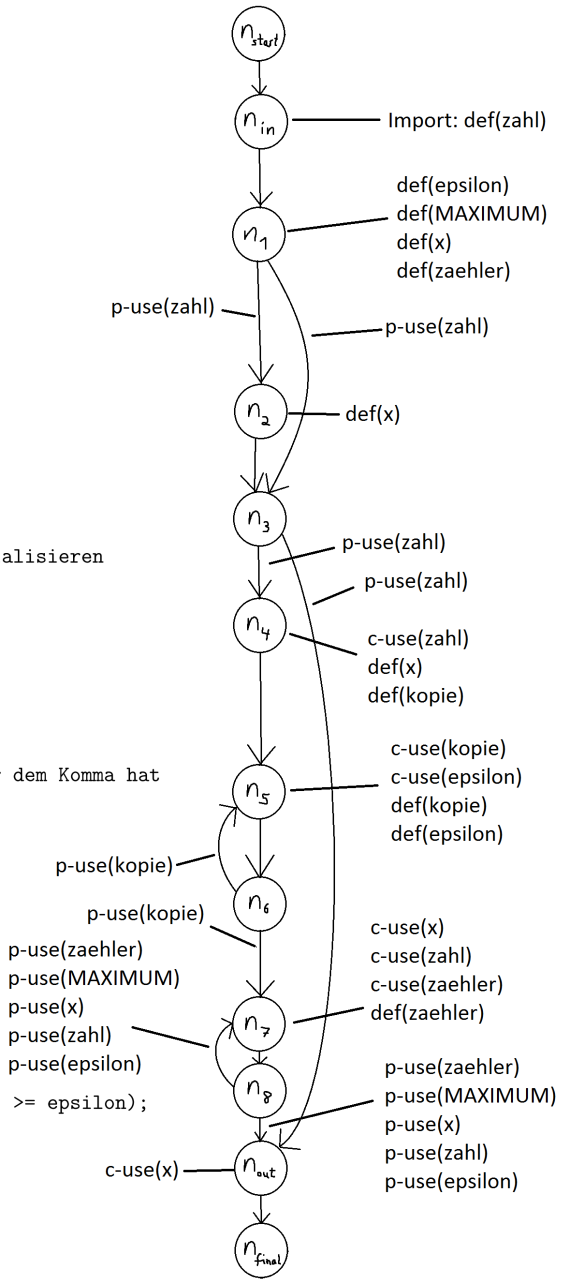
Michel Meyer, Manuel Schwarz

5. Dezember 2012

Aufgabe 7.1

(a)

```
08 public static double sqrtHeron(double zahl) {
09     // Schranke, vordefiniert fuer einstellige Zahl
10     double epsilon = 1e-15;
11     // max. Anzahl Folgenglieder
12     final int MAXIMUM = 100000;
13     // aktuelles Folgenglied
14     double x = 0;
15     // Folgengliednummer
16     int zaehler = 1;
17
18     // liefere -1 fuer Eingaben < 0
19     if (zahl < 0) {
20         x = -1;
21     }
22
23     // zahl <= 0 laesst sich nicht berechnen
24     if (zahl > 0) {
25         // aktuelles Folgenglied mit uebergebenen Zahl initialisieren
26         x = zahl;
27
28         // Kopie der uebergebenen Zahl erstellen
29         double kopie = zahl;
30
31         // Berechnung der Schranke
32         do {
33             // solange 'kopie' noch mehr als eine Stelle vor dem Komma hat
34             // ziehe eine Stelle ab...
35             kopie = kopie / 10;
36             // ... und multipliziere Schranke mit 10
37             epsilon = epsilon * 10;
38         } while (kopie > 1);
39
40         // Berechnung der Quadratwurzel
41         do {
42             // naechstes Folgenglied berechnen
43             x = (x + zahl / x) / 2.0;
44             zaehler++;
45         } while (zaehler < MAXIMUM && Math.abs(x * x - zahl) >= epsilon);
46     }
47     return x;
48 }
```



dcu* und *dpu

Variable	**Knoten** n_i	***dcu***(x, n_i)	***dpu***(x, n_i)
`zahl`	n_{in}	$\{n_4, n_7\}$	$\{(n_1, n_2), (n_1, n_3), (n_3, n_4), (n_3, n_{out}), (n_8, n_7), (n_8, n_{out})\}$
`epsilon`	n_1	$\{n_5\}$	$\{\}$
`epsilon`	n_5	$\{n_5\}$	$\{(n_8, n_7), (n_8, n_{out})\}$
`MAXIMUM`	n_1	$\{\}$	$\{(n_8, n_7), (n_8, n_{out})\}$
`x`	n_1	$\{n_{out}\}$	$\{\}$
`x`	n_2	$\{n_{out}\}$	$\{\}$
`x`	n_4	$\{n_7, n_{out}\}$	$\{(n_8, n_7), (n_8, n_{out})\}$
`zaehler`	n_1	$\{n_7\}$	$\{\}$
`zaehler`	n_7	$\{n_7\}$	$\{(n_8, n_7), (n_8, n_{out})\}$
`kopie`	n_4	$\{n_5\}$	$\{\}$
`kopie`	n_5	$\{n_5\}$	$\{(n_6, n_5), (n_6, n_7)\}$

(b) *All-defs*

Die Menge

$$\{(n_{start}, n_{in}, n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_{out}, n_{final})\}$$

testet alle definierten Variablen mindestens einmal mit *c-use* oder *p-use*.
Der eine Pfad reicht aus, denn:

1. Mindestens ein Knoten aus jeder *dcu*-Menge kommt in dem Pfad vor, womit die Definition aller Variablen außer `MAXMIMUM` durch ein *c-use* getestet wurde.

2. Für `MAXIMUM` ist meindestens einer der Kanten aus der *dpu*-Menge im Pfad vorhanden.

(c) *All-p-uses*

Die Menge

$$\begin{aligned}&\{(n_{start}, n_{in}, n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_{out}, n_{final}),\\ &\ (n_{start}, n_{in}, n_1, n_3, n_{out}, n_{final}),\\ &\ (n_{start}, n_{in}, n_1, n_3, n_4, n_5, n_6, n_5, n_6, n_7, n_8, n_7, n_8, n_{out}, n_{final})\}\end{aligned}$$

testet zu allen definierten Variablen alle *p-use*s.