Überdeckungskriterien beim Glass-Box-Testing

FOLIEN: HTTPS://WWW.STEFAN-WINTER.NET/APPLICATION-MATERIALS/

Software-Tests und Testendekriterien

```
public int sum(int a, int b)
```

Die Funktion gibt die Summe der beiden Parameter a und b zurück.

Eingaberaum: $2^{64} \approx 18{,}4 \times 10^{18}$

Bei 1000 Tests pro Sekunde: >584 Millionen Jahre Testlaufzeit

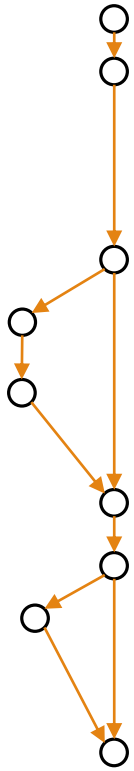
Alternative: **Testendekriterien**

Glassbox-Testing:
Überdeckungsbasierte Testendekriterien

- Kontrollflussbasierte Überdeckung
- Datenflussbasierte Überdeckung

Kumulative Normalverteilung in „Blackscholes“

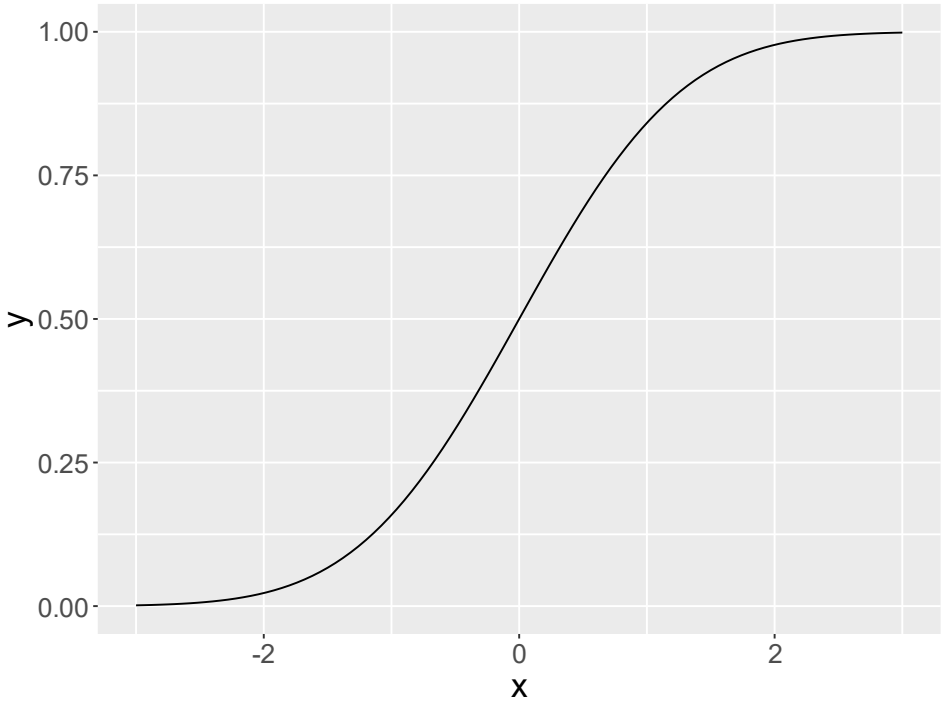
„Kontroll-flussgraph“ CFG



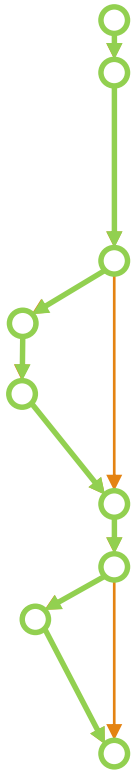
```
fptype CNDF ( fptype InputX )
{
    int sign = 0;
    fptype OutputX;

    // Check for negative value of InputX
    if (InputX < 0.0) {
        InputX = -InputX;
        sign = 1;
    }
    OutputX = computeNPrimeX(InputX);
    if (sign) {
        OutputX = 1.0 - OutputX;
    }
    return OutputX;
}
```

Kontrollflussgraph $G = (V, E)$:

- Menge von Knoten V
- Menge von Kanten $E \subseteq V \times V$



Anweisungsüberdeckung



```
fptype CNDF ( fptype InputX )
{
    int sign = 0;
    fptype OutputX;

    // Check for negative value of InputX
    if (InputX < 0.0) {
        InputX = -InputX;
        sign = 1;
    }
    OutputX = computeNPrimeX(InputX);
    if (sign) {
        OutputX = 1.0 - OutputX;
    }
    return OutputX;
}
```

Anzahl CFG-Knoten: $|V| = 9$
Anzahl durch Tests besuchte CFG-Knoten: $|V_t|$

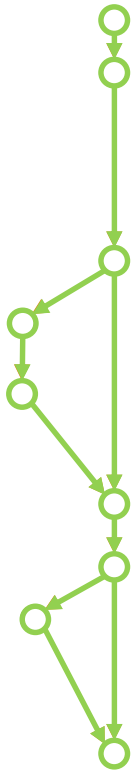
Anweisungsüberdeckung: $c_0 = \frac{|V_t|}{|V|}$

Testendekriterium: $c_0 = 1$

100% Anweisungsüberdeckung mit einer Testeingabe:

- InputX = -0.5
- $|V_t| = 9$
- $c_0 = 1$

Zweigüberdeckung



```
fptype CNDF ( fptype InputX )
{
    int sign = 0;
    fptype OutputX;

    // Check for negative value of InputX
    if (InputX < 0.0) {
        InputX = -InputX;
        sign = 1;
    }
    OutputX = computeNPrimeX(InputX);
    if (sign) {
        OutputX = 1.0 - OutputX;
    }
    return OutputX;
}
```

Anzahl CFG-Kanten („Zweige“):
$|E| = 10$
Anzahl durch Tests verfolgte CFG-Kanten: $|E_t|$

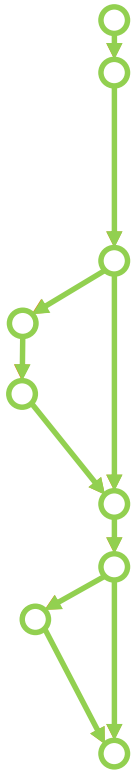
Zweigüberdeckung: $c_1 = \frac{|E_t|}{|E|}$

Testendekriterium: $c_1 = 1$

100% Zweigüberdeckung mit zwei Testeingaben:

- InputX = -0.5
- $|E_t| = 8$
- InputX = 0.5
- $|E_t| \mathrel{+}= 2$
- $c_1 = 1$

Pfadüberdeckung



```
fptype CNDF ( fptype InputX )
{
    int sign = 0;
    fptype OutputX;

    // Check for negative value of InputX
    if (InputX < 0.0) {
        InputX = -InputX;
        sign = 1;
    }
    OutputX = computeNPrimeX(InputX);
    if (sign) {
        OutputX = 1.0 - OutputX;
    }
    return OutputX;
}
```

Anzahl CFG-Pfade vom Start- zum Endknoten:

$$|P| = \left| \begin{array}{c} \{p = (v_1, \dots, v_9). \\ \forall (v_i, v_{i+1}) \subset p \\ \Rightarrow (v_i, v_{i+1}) \in E\} \end{array} \right| = 4$$

Anzahl durch Tests verfolgte CFG-Pfade: $|P_t|$

Pfadüberdeckung: $c_2 = \frac{|P_t|}{|P|}$

Testendekriterium: $c_2 = ?$

Max. 50% Pfadüberdeckung für

- InputX = -0.5
- InputX = 0.5
- $|P_t| = 2$
- $c_2 = 0.5$

Zusammenfassung und Ausblick

Im Vortrag behandelt:

- Welches Problem sollen Überdeckungskriterien beim Glassbox-Testing lösen?
- Was ist ein Kontrollflussgraph?
- Was bedeutet Anweisungs-/Zweig-/Pfadüberdeckung?

Weiter zu vertiefende Themen:

- Kontrollflussbasierte Überdeckungskriterien für sicherheitskritische Systeme
- Datenflussbasierte Überdeckungskriterien
- Mutation Testing als Alternative zu strukturellen Testendekriterien

Übung:

- Überdeckungsanalyse für GNU coreutils mit gcov und lcov

Literatur:

- Spillner & Linz: Basiswissen Softwaretest, dpunkt-Verlag
- Liggesmeyer: Software-Qualität