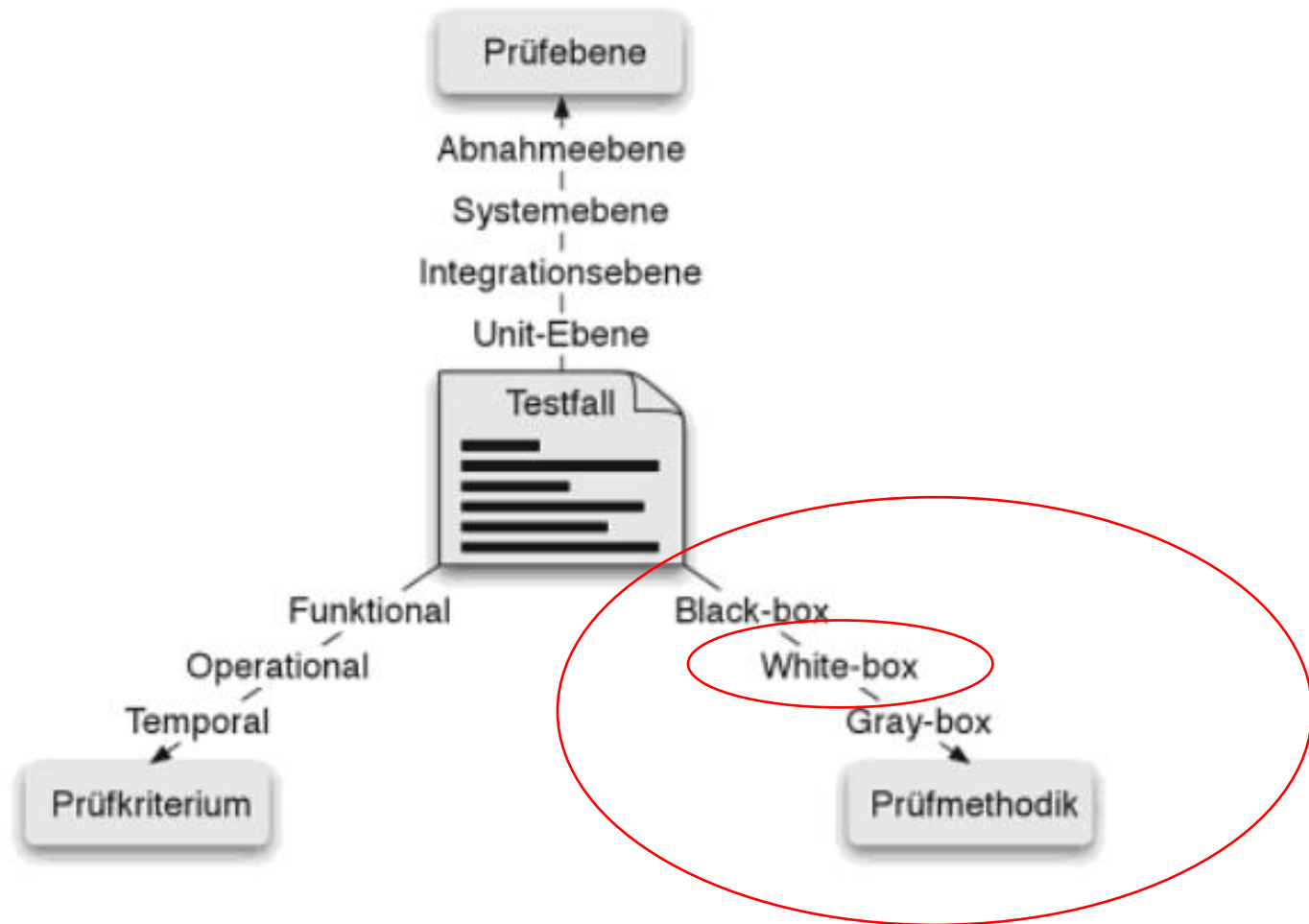


- Motivation
- Testklassifikation
- Black Box Testtechniken
- ➡ ■ White Box Testtechniken
- Testmetriken
- Grenzen des Software Tests
- Testautomatisierung

Testklassifikation



Black Box Test:

Input basiert ausschließlich auf der funktionalen Beschreibung.

Nun: White Box Test:

Basiert auf der Analyse der inneren Programmstrukturen.

White Box Test = Strukturtest

- **Kontrollflussorientierte Tests**

Konstruktion der Testfälle ausschließlich auf Basis der internen Berechnungspfade eines Programms. Beschaffenheit der Testdaten spielt keine Rolle.

- **Datenflussorientierte Tests**

Heranziehen zusätzlicher Kriterien aus der Beschaffenheit der manipulierten Daten.

1. Strukturanalyse

Aus dem Programmcode wird der Kontrollflussgraph extrahiert.

2. Testkonstruktion

Ableitung der Testfälle entsprechend eines vorher definierten Überdeckungskriterium.

3. Testdurchführung

Kontrollflussmodellierung - Bsp

```
int manhattan(int a, int b){
```

```
    if(a < 0){
```

```
        a = -a;
```

```
    }
```

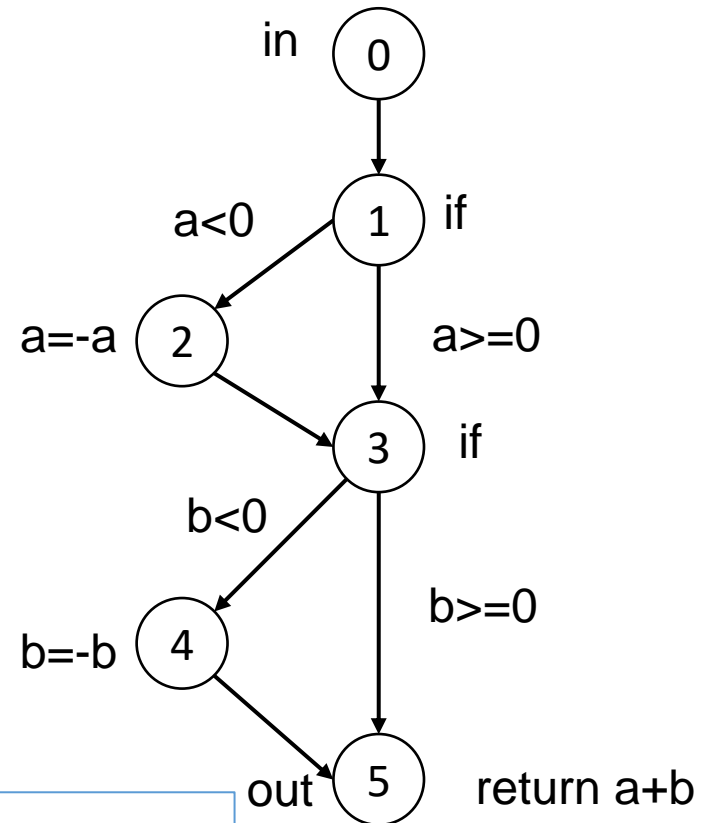
```
    if(b < 0){
```

```
        b = -b;
```

```
    }
```

```
    return a + b;
```

```
}
```



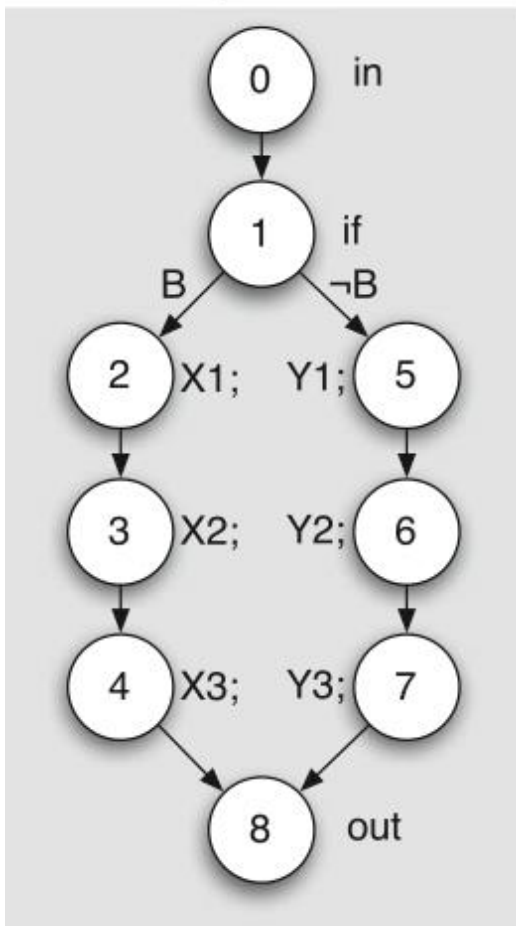
**Immer:
Ein einziger Einstiegspunkt,
Ein einziger Ausstiegspunkt!**

- **Expandierte Kontrollflussgraphen**
Jeder Befehl ein separater Knoten.
- **Teilkollabierte Kontrollflussgraphen**
zwei oder mehrere sequenzielle Befehle in einem einzigen Knoten.
- **Kollabierte Kontrollflussgraphen (am häufigsten verwendet)**
Verzweigungsfreie Befehlsblöcke in einem einzigen Knoten.

Klassifikationsmerkmale - Bsp

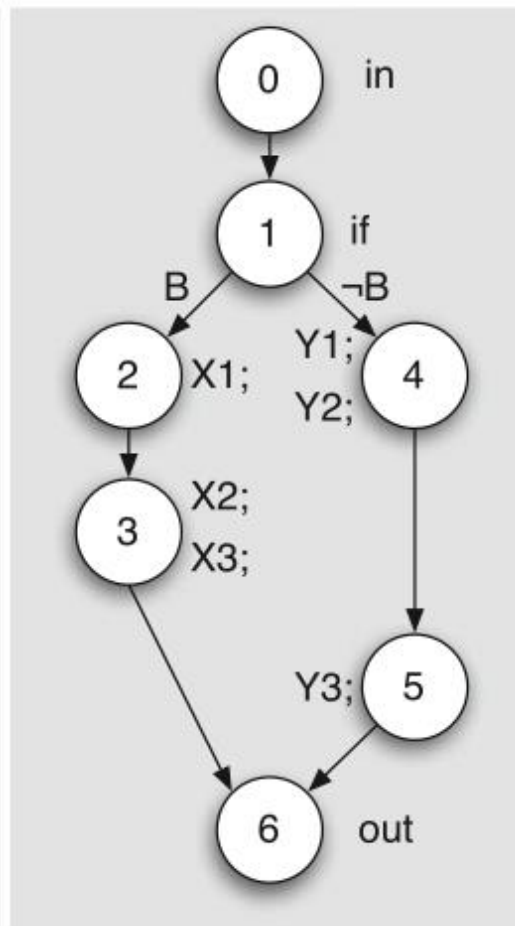
expandiert

Expandiert



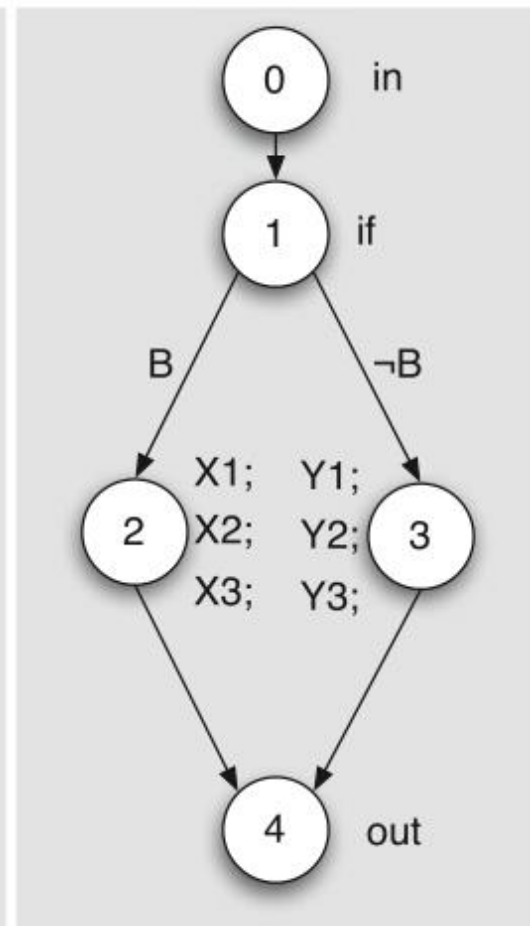
teilkollabiert

Teilkollabiert

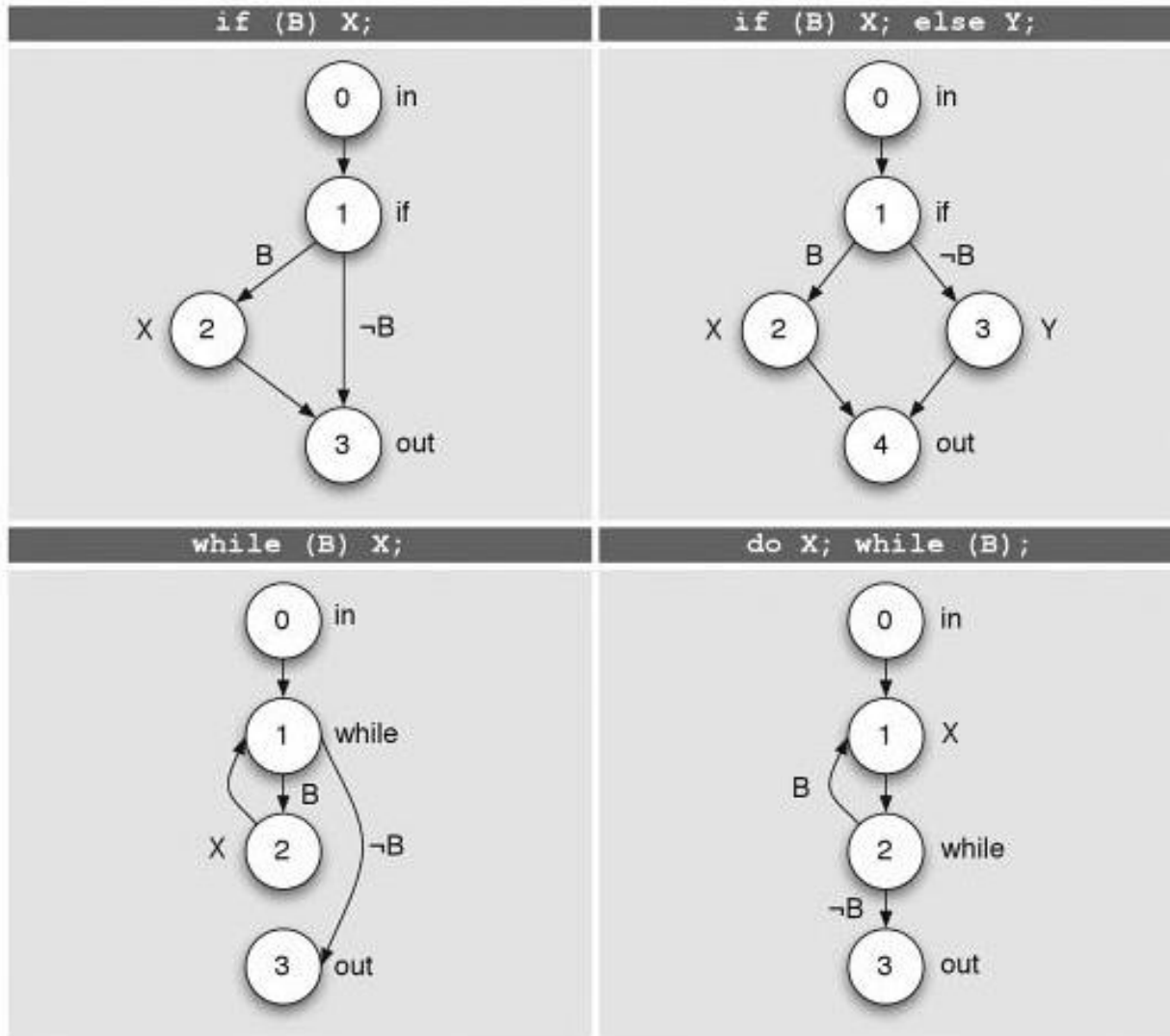



kollabiert

Kollabiert



Kontrollflussgraphen elementarer Konstrukte

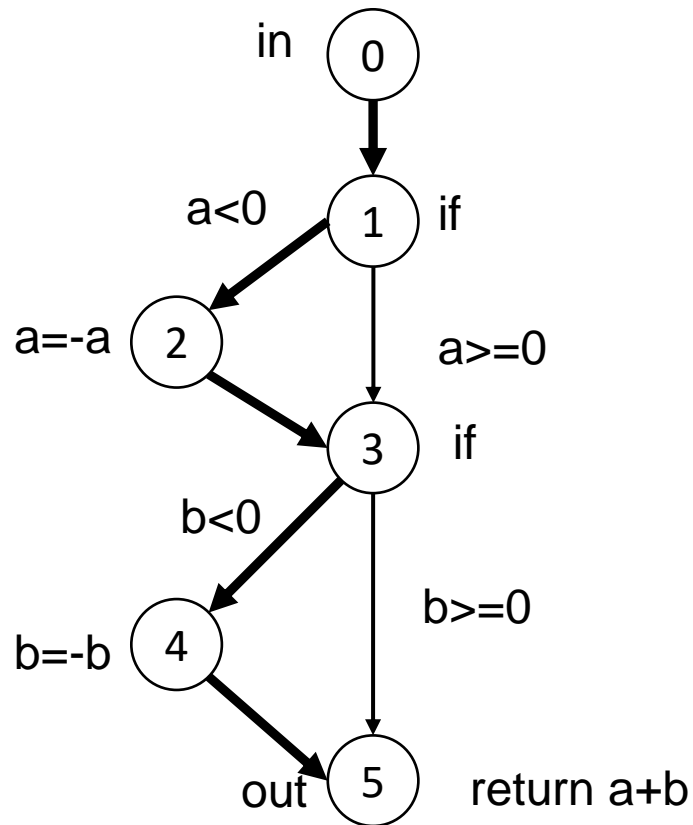


- 
- Anweisungsüberdeckung
 - Zweigüberdeckung
 - Pfadüberdeckung
 - Bedingungsüberdeckung
 - McCabe Überdeckung
 - Defs Uses Überdeckung
 - Required k-Tupel Überdeckung

Anweisungsüberdeckung

- Testmenge wird so gewählt, dass alle Knoten des Kontrollflussgraphen mindestens einmal durchlaufen werden.
- Auch als C_0 Test bezeichnet.
- Schwächstes der hier vorgestellten Kriterien.

Anweisungsüberdeckungstest - Bsp



Bsp manhattan:

Testfälle:

- manhattan(-1,-1)

Überdeckung: $\{0,1,2,3,4,5\}$

Diskussion der Anweisungsüberdeckung

- Mit wenigen Testfällen zu erfüllen im Vergleich zu den anderen Überdeckungskriterien.
- Wichtige Fälle bleiben unberücksichtigt.
- Dennoch häufig schwer zu erreichen.
- Wird die Anweisungsüberdeckung nicht zu 100% erfüllt, ist das Risiko nicht kalkulierbar.
- Standard **RTCA DO-178B** für Software Anwendungen in der Luftfahrt verlangt Anweisungsüberdeckung für alle Komponenten, deren Ausfall zu einer *bedeutenden aber nicht kritischen* Fehlfunktion führen kann.

Bsp für schwer zu realisierende Anweisungsüberdeckung

```
uint8_t *hardToTest(...)
{
    /*Get the file properties*/
    if(stat(filename,&fileProperties) != 0){
        return NULL;
    }

    /*Open file*/
    if(!(file = fopen(filename,"r"))){
        return NULL;
    }

    /*Allocate memory*/
    if(!(data = (uint8_t *) malloc(fileProperties.st_size))){
        return NULL;
    }

    return data;
}
```

Schwierigkeiten in dem Bsp

- Wie erhalten Sie die File Properties, können das File aber nicht öffnen?
- Wie erzeugen Sie einen Fehler für malloc?
- ➔ malloc durch eine eigene Funktion ersetzen, damit die vorliegende Funktion getestet werden kann.
- ➔ Derartige Situationen können dazu führen, dass die C_0 -Überdeckung nur aufwändig zu erreichen ist.

- Anweisungsüberdeckung
- ➔ ▪ **Zweigüberdeckung**
- Pfadüberdeckung
- Bedingungsüberdeckung
- McCabe Überdeckung
- Defs Uses Überdeckung
- Required k-Tupel Überdeckung

- Auch als C_1 -Überdeckung bezeichnet.
- Jede Kante des Kontrollflussgraphen muss durch mindestens einen Testfall durchlaufen werden.
- Wird oftmals als Minimalkriterium definiert.
- Sollte immer angestrebt werden.

Zweigüberdeckung

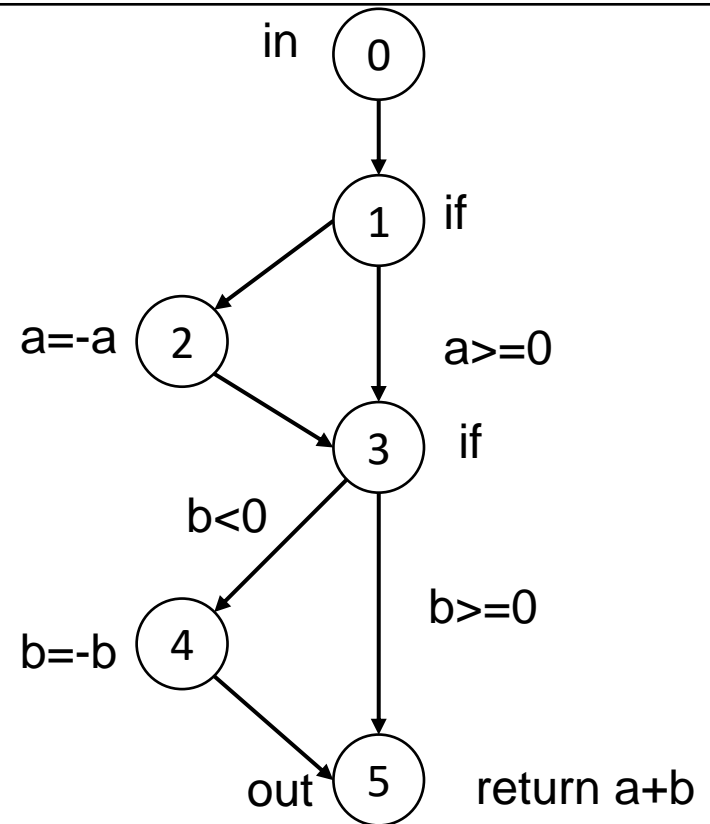
```
int manhattan(int a, int b){
```

```
    if(a < 0){  
        a = -a;  
    }
```

```
    if(b < 0){  
        b = -b;  
    }
```

```
    return a + b;
```

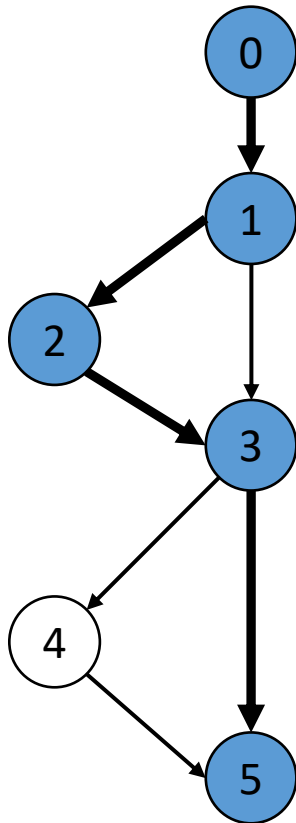
```
}
```



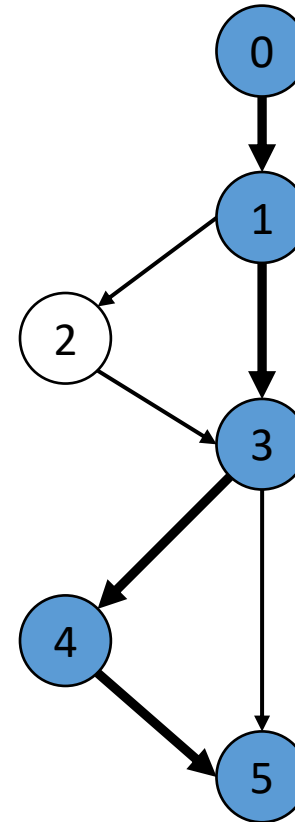
Aufgabe: Definieren Sie Testfälle, so dass eine Zweigüberdeckung von 100 % erfüllt ist!

Zweigüberdeckung – Bsp manhattan

manhattan(-1,1)




manhattan(1,-1)



Zweigüberdeckung

Standard RTCA DO-178B für Software Anwendungen in der Luftfahrt verlangt Zweigüberdeckung für alle Komponenten, deren Ausfall zu einer schweren, aber noch nicht katastrophalen Fehlfunktion führen kann.

- Anweisungsüberdeckung
- Zweigüberdeckung
-  ▪ Pfadüberdeckung
- Bedingungsüberdeckung
- McCabe Überdeckung
- Defs Uses Überdeckung
- Required k-Tupel Überdeckung

Pfadüberdeckung

- Dann erfüllt, wenn für jeden möglichen Pfad von Eingangsknoten zu Ausgangsknoten ein separater Testfall existiert.
- Mächtigste White Box Prüftechnik
- Anzahl der Pfade explodiert schnell, insbsd. wenn Schleifen vorhanden sind.

Pfadüberdeckung

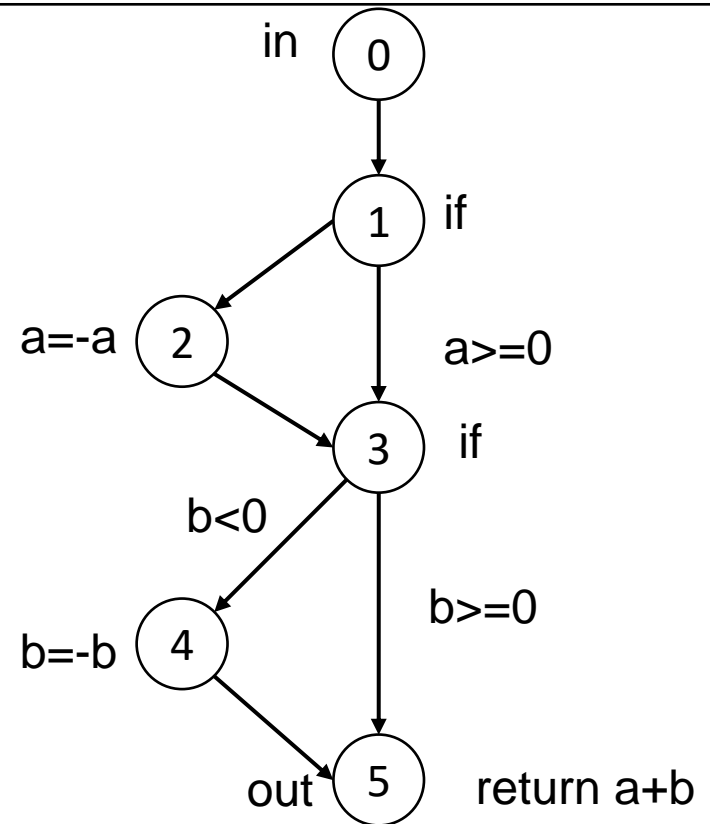
```
int manhattan(int a, int b){
```

```
    if(a < 0){  
        a = -a;  
    }
```

```
    if(b < 0){  
        b = -b;  
    }
```

```
    return a + b;
```

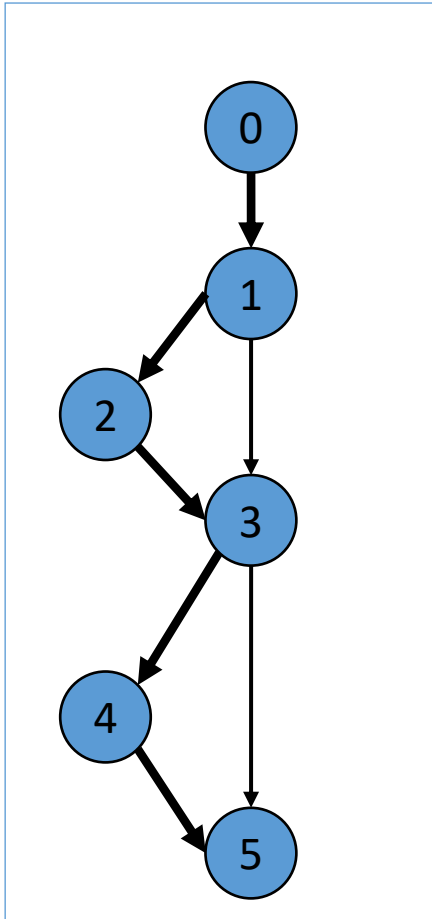
```
}
```



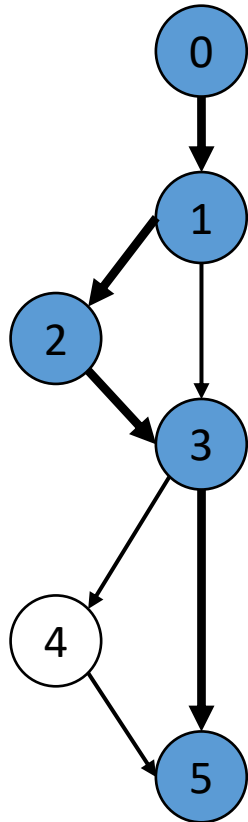
Aufgabe: Definieren Sie Testfälle, so dass eine Pfadüberdeckung von 100 % erfüllt ist!

Pfadüberdeckung- Bsp manhattan

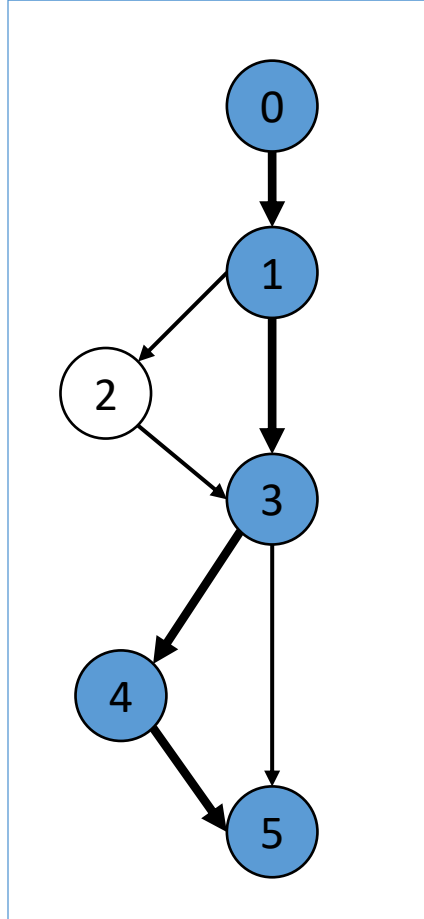
manhattan(-1,-1)



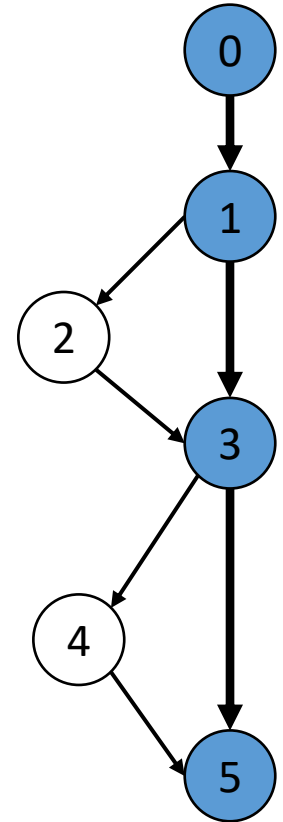
manhattan(-1,1)



manhattan(1,-1)



manhattan(1,1)



Probleme der Pfadüberdeckung - Bsp

0 `void foobar(int *a)`

1 `{`

2 `for(int i = 0; i < 512 ; i++) { for ...`

3 `if(a[i]) {`

4 `foo();`

5 `}`

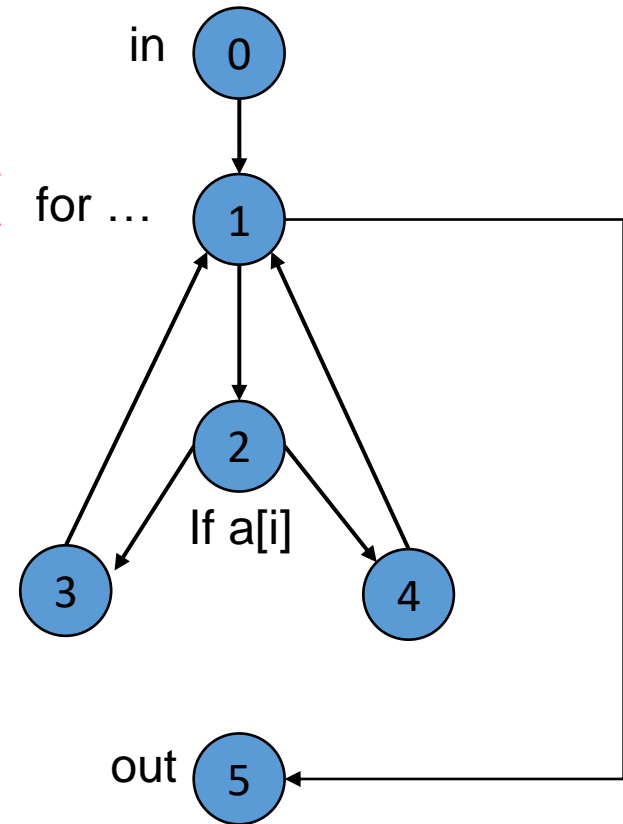
6 `else{`

7 `bar();`

8 `}`

9 `}`

10 `}`



➔ 2^{512} verschiedene Pfade

Varianten der Pfadüberdeckung

- Pfadüberdeckung in der Praxis normalerweise nicht realisierbar, nur theoretisch interessant.
- ➔ Varianten, um die Anzahl der Testfälle zu reduzieren.

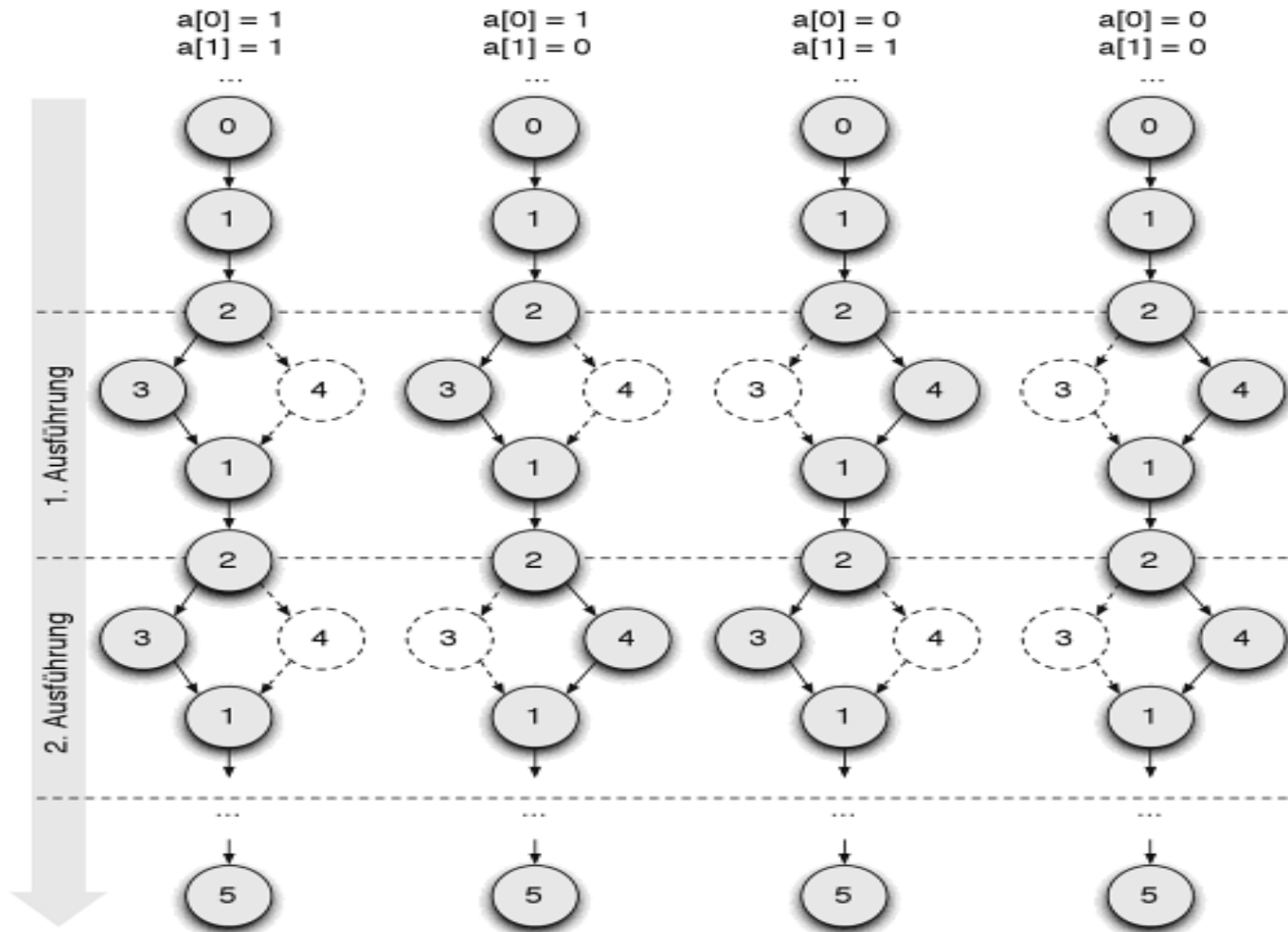
Idee: es ist nicht nötig, Schleifen öfter als ein paar Iterationen zu durchlaufen um fast alle Fehler zu finden.

Boundary Interior Pfadüberdeckung:

Für jede Schleife drei Gruppen von Testfällen:

- **Äußere Pfade:**
Schleifen werden nicht betreten.
- **Grenzpfade**
Genau eine Iteration.
- **Innere Pfade**
Mindestens eine weitere Iteration. Testfälle werden so gewählt, dass innerhalb der ersten beiden Iterationen alle möglichen Pfade abgearbeitet werden.

Bsp: Interior Pfade der Funktion foobar



Strukturierte Pfadüberdeckung:

Verallgemeinerung des Boundary-Interior Tests:

Es werden alle möglichen Ausführungspfade bis zur k -ten Schleifeniteration durchlaufen (statt nur die ersten beiden).

Immer noch sehr viele Pfade, insbds. bei verschachtelten Schleifen)

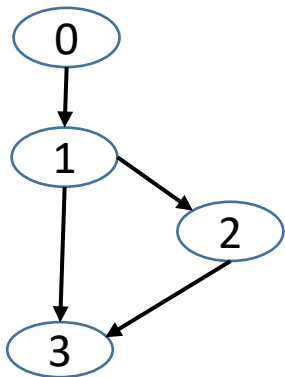
➔ **Modifizierung:** k Iterationen nur für Schleifen, die keine inneren Schleifen mehr enthalten.

- Anweisungsüberdeckung
- Zweigüberdeckung
- Pfadüberdeckung
- ➔ • Bedingungsüberdeckung
- McCabe Überdeckung
- Defs Uses Überdeckung
- Required k-Tupel Überdeckung

Achtung bei der Zweigüberdeckung

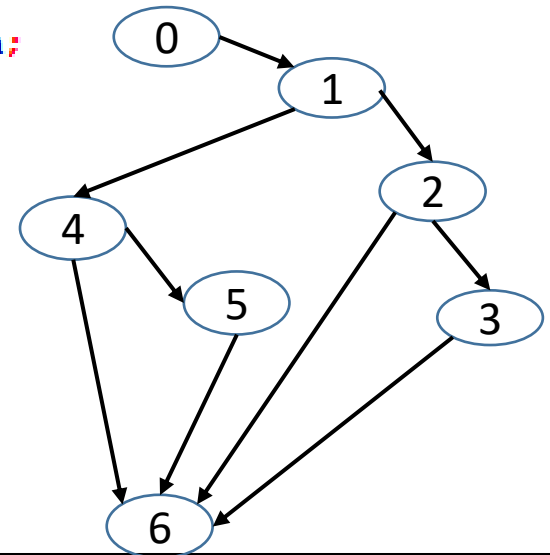
```

0 void first(){
1     if( (A && !B) || (!A && B){
2         do_something();
3     }
4     return;
5 }
  
```



```

0 void second(){
1     if(A){
2         if( !B ){
3             do_something();
4         }
5         }else{
6             if(B){
7                 do_something();
8             }
9         }
10    }
11    return;
12 }
  
```



Identische Funktionalität,
Unterschiedliche
Implementierung
➔ Unterschiedliche Anzahl
von Testfällen

Zusätzlich zu Kontrollflussgraph: Einbeziehung der logischen Struktur der *Bedingungen*

Verschiedene Kriterien:

- Einfache Bedingungsüberdeckung
- Minimale Mehrfachbedingungsüberdeckung
- Mehrfachbedingungsüberdeckung

Beispiel: $\text{if } (A \ \&\& \ !B) \ || \ (!A \ \&\& \ B)$

Einfache Bedingungsüberdeckung:

Alle atomaren Prädikate müssen mindestens einmal beide Wahrheitswerte annehmen.

→ $A = 0, B = 1$ $A = 1, B = 0$

Minimale Mehrfachbedingungsüberdeckung

Alle atomaren und zusammengesetzten Prädikate müssen mindestens einmal beide Wahrheitswerte annehmen.

→ $A = 0, B = 0$ $A = 0, B = 1$ $A = 1, B = 0$

Mehrfachbedingungsüberdeckung:

Alle Wahrheitskombinationen müssen getestet werden

→ $A = 0, B = 0$ $A = 0, B = 1$ $A = 1, B = 0$ $A = 1, B = 1$

- Anweisungsüberdeckung
- Zweigüberdeckung
- Pfadüberdeckung
- Bedingungsüberdeckung
- McCabe Überdeckung
- Defs Uses Überdeckung
- Required k-Tupel Überdeckung

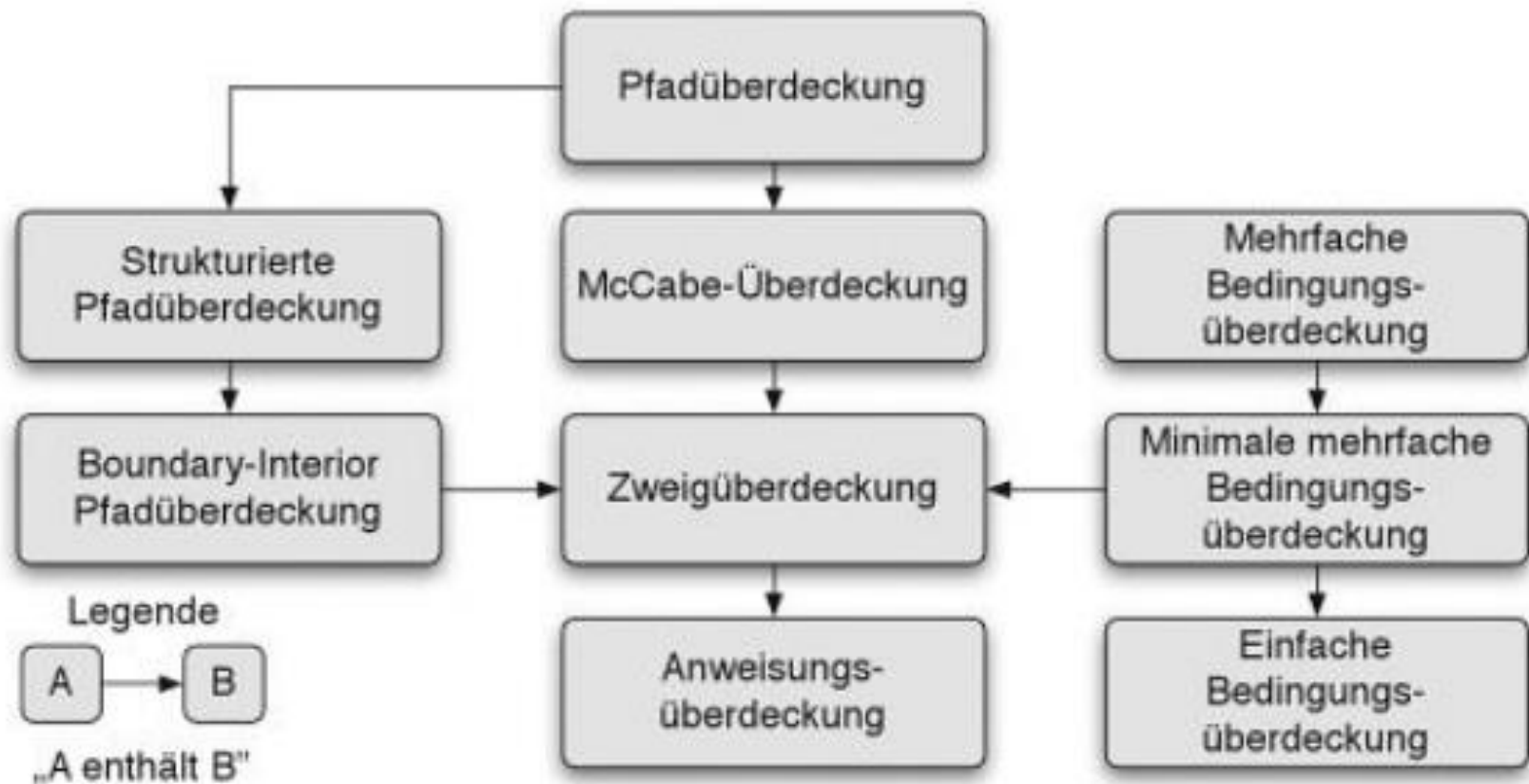
- **McCabe Überdeckung**
kontrollflussorientiertes Verfahren.
- **Defs Uses Überdeckung**
Ableitung der Testfälle aus dem Datenfluss.
- **Required k-Tupel Überdeckung**
Ebenfalls Ableitung der Testfälle aus dem Datenfluss.

Hier nicht weiter betrachtet

Nachzulesen unter Dirk W. Hoffmann: Software-Qualität, 2 Auflage, Springer Vieweg

Verfahren sind zu komplex um in der Praxis eine große Rolle zu spielen.

Kontrollflussorientierte Strukturtests



Datenflussorientierte Strukturtests

