Institut für Echtzeitsysteme
und Softwaretechnik
Prof. Dr. Derk Rembold

Hochschule
Albstadt-Sigmaringen
Albstadt-Sigmaringen University

2019

Praktikum Betriebssicherheit

Aufgabenblatt 1

Fehlerbaumanalyse

Anforderungen:

- Die Aufgabe wird in Python programmiert.
- Die Aufgabe wird von jedem Teilnehmer einzeln erstellt!
- Der Teilnehmer kommt zur Abnahme auf den Dozenten zu. Die Abnahme erfolgt für jeden Teilnehmer einzeln. Die Kenntnis des Quellcodes wird erwartet.
- Programmcode wird auf Ilias hochgeladen. Die Lokation wird im Praktikum bekanntgegeben. Das File hat folgendes Format:
  - <Name>_<Vorname>_<Matrikelnummer>_Aufgabe_1_Programmcode.py

- **Es gibt eine Frist für die Abnahme, Abgabe des Berichts und Hochladen der Files. Diese wird im Praktikum bekanntgegeben.**

Einleitung

Der Fehlerbaum ist ein Werkzeug zur logischen Verknüpfung von Komponenten und Teilsystemen. Unteranderem will man die Kombination der Ausfälle ermitteln, die zu einem unerwünschten Ergebnis führen. Hier ist insbesondere die Bestimmung der Minimal Cut Sets interessant und Teil dieser Praktikumsaufgabe. Der Fehlerbaum besteht aus einer Menge von Verknüpfungselementen. Bei dieser Aufgabe sollen aber nur Standardeingang, Nicht-Verknüpfung, und Oder-Verknüpfung behandelt werden.

Aufgabe

Es soll mit Hilfe von den Verknüpfungselementen Standardeingang, Oder-Verknüpfung und Und-Verknüpfung ein beliebiger Fehlerbaum aufgebaut werden. Dafür sollen die drei Elemente als Klassen definiert werden. Die Klassen sollen Listen (nodes, siehe Bild 2) als Attribute enthalten, die Instantzen von weiteren Verknüpfungselementen aufnehmen können.

Der folgende Fehlerbaum (Bild 1) soll mit einem Programm innerhalb des Praktikums modelliert werden:

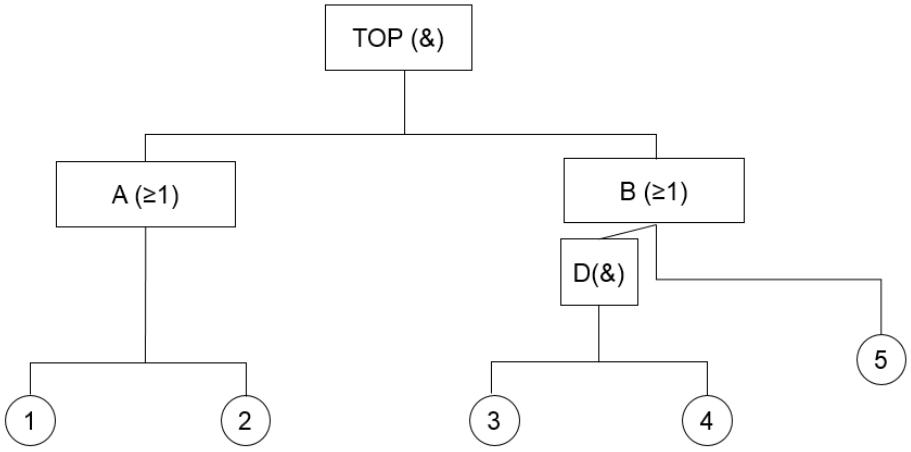

Bild 1: Fehlerbaum

TOP und D sind dabei Und-Verknüpfungselemente, A und B sind Oder-Verknüpfungselemente. Standardeingänge sind als 1, 2, 3, 4, 5 bezeichnet.

Die Verknüpfungselemente sollen als Klassen definiert werden und sollen die folgende Struktur erhalten, siehe Bild 2, Bild 3, Bild 4:

```
class ANDNODE:
    def __init__(self,name):
        self.name = name
        self.nodes=[]
    def add(self,node):
        self.nodes.append(node)
        return
    #............
    def topdown(self, mat):
        #..............
        return mat
```

Bild 2: Und-Verknüpfungselement

```
class ORNODE:
    def __init__(self, name):
        self.name = name
        self.nodes=[]
    def add(self,node):
        self.nodes.append(node)
        return
    #............
    def topdown(self, mat):
        #.............
        return  mat
```

Bild 3: Oder-Verknüpfungselement

```
class EVENT:
    name=''
    def __init__(self,name):
        self.name = name
    #...............
    def topdown(self, mat):
        return mat
```

Bild 4: Standardeingang

a.) Bauen Sie aus den instanziierten Verknüpfungselementen den Fehlerbaum aus Bild 1 auf. Verwenden Sie dabei die Methoden add(…) um darunterliegende Objekte an die Verknüpfungselemente anzuhängen. Beispiel:

TOP = ANDNODE(,TOP‘)
A = ORNODE(,A‘)
E1 = EVENT(,1‘)

TOP.add(A)
TOP.add(E1)

Kontrollfrage: Wie würde der Fehlerbaum nach diesen Befehlsfolgen aussehen?

b.) Es soll aus dem Fehlerbaum die Minimal Cut Sets ermittelt werden. Dafür sollen die Methoden topdown() implementiert werden. topdown() nimmt als Parameter eine Matrix für die Topdown-Analyse entgegen. Diese ist ein zwei-dimensionales Array. Die Methode topdown() liefert dann eine veränderte Matrix gemäß des Topdown-Algorithmus. Beispiel (bezieht sich nicht auf den Fehlerbaum oben):

mat=[[TOP]]
mat = TOP.topdown(mat)

Output:
[[1][2][3]]
[[1][2][4]]

c.) Programmieren Sie den Graphen mit Graphviz/Matplotlib und drucken Sie diesen, sodass dieser dem Graphen aus Bild 1 gleicht.

d.) Verändern Sie bei der Abnahme den Fehlerbaum und berechnen Sie in Anwesenheit des Dozenten die neuen Minimal Cut Sets. Der Fehlerbaum wird vom Dozenten vorgegeben. Der vom Programmcode erzeugte Graph soll dabei automatisch angepasst werden.