ECE454/544: Fault-Tolerant Computing & Reliability Engineering

Lecture #9–

Fault Tree Analysis

Instructor: Dr. Liudong Xing

Fall 2022

Administrative Issues (Oct. 11, Tuesday)

- Homework#4 assigned and due **Oct. 12, Wednesday**
- Project Meeting
  - Due **Oct. 28, Friday**

Review of Lecture#7&8

- Review of random variables and related concepts
- Quantitative measures
  - Time to failure (T): a random variable describing the time elapsing from when a component is put into operation until it fails for the first time
  - Failure function F(t): the cumulative distribution function (c.d.f) of the r.v. T
  - Reliability/survivor function R(t)=1-F(t)
  - Failure rate (hazard rate/function) z(t)
  - Relationship between F(t), f(t), z(t), and R(t)
  - The bathtub curve for the failure rate
    - Burn-in/infant mortality period
    - Useful-life period
    - Wear-out period
  - Mean time to failure (MTTF)
  - Mean residual life (MRL)
- Exponential time to failure distribution has constant failure rate and memory-less property

Topics

- Introduction to fault tree analysis
- Fault tree construction
- Fault tree analysis using cut-sets

Reference & acknowledgement:

J. B. Dugan and S. A. Doyle. "New Results in Fault-Tree Analysis" *Tutorial notes presented at Annual Reliability and Maintainability Symposium*, January 1997 (Section 1)

Fault Tree Analysis

- Fault trees were first developed in 1962 at Bell Telephone lab to facilitate analysis of the Minuteman missile launching system

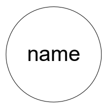
- **What is a fault tree?**
  - Not a tree (in the graph-theoretic sense)
  - a graphical representation of a logical function
  - shows logical relationship between an event (failure) and its causes

Why use fault tree analysis?

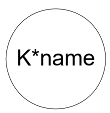
- A fault tree model precisely documents which failure scenarios have been considered and which have not.

- Fault tree analysis provides a logical framework for understanding the ways in which a system can fail (i.e., combinations of component failures that can lead to system failure), which is often as important as understanding how a system can succeed

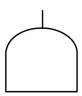
Fault Tree Construction (I)



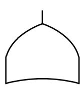
- **Basic Event**: Corresponds to a basic failure event (usually a component failure) in the system
  - Characterized by failure rate or failure probability



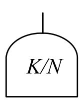
- **Replicated Basic Event**: Represents K statistically and functionally identical copies of a component
  - Characterized by failure rate or failure probability of one copy



- **AND gate**: output event occurs only if ALL input events occur



- **OR gate**: output event occurs if one or more input events occur

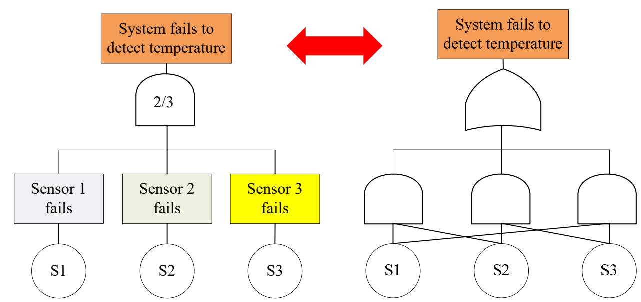


- **K/N gate**: output event occurs if K or more of N input events occur

K/N Gate

- A *K/N* vote gate can be expanded into OR combinations of C_N^K AND gates

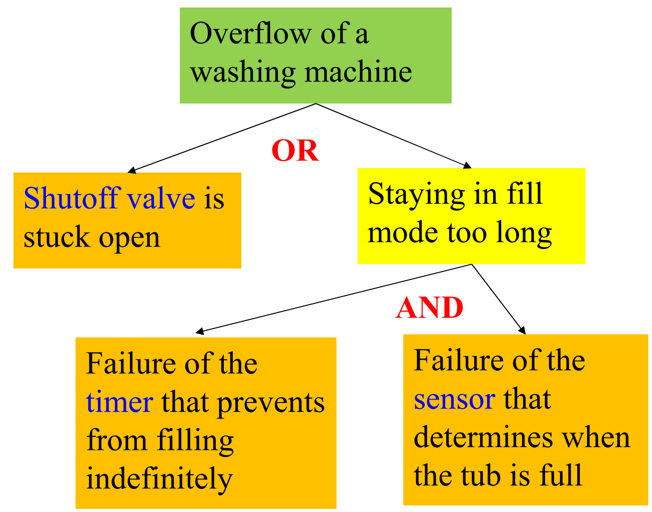
- Example: a control system has 3 sensors; 2 out of 3 required to detect temperature



Fault Tree Construction (II)

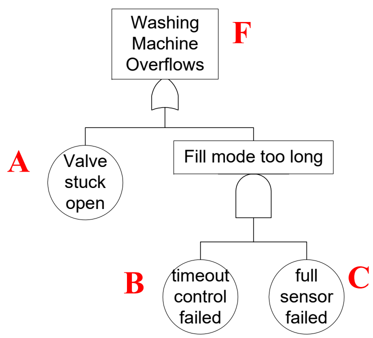
- Top-down approach
  - Begin with the failure scenario being considered
  - Decompose the failure symptom into its possible causes
  - Each possible cause is then investigated and further refined until the basic causes of the failure are understood
- The construction of fault trees provides a systematic method for analyzing and documenting the potential causes of system failure

An Example



Fault Tree Construction (III)

- Simple example fault tree (FT1)



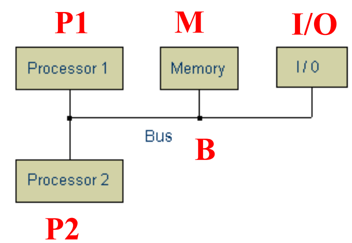
Structure Function:

Fail = valve_fail OR (timer_fail AND sensor_fail)

$F = A + BC$

Hands-on Problem (1)

- System is operational *iff* 1 processor, memory, I/O and bus are functioning



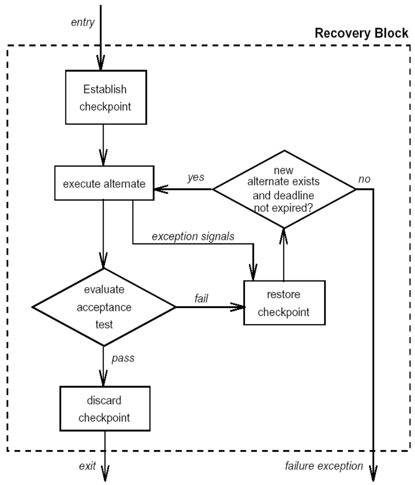
  – **Find fault tree model of the system**

Fault Tree Models for Fault Tolerant Software Systems

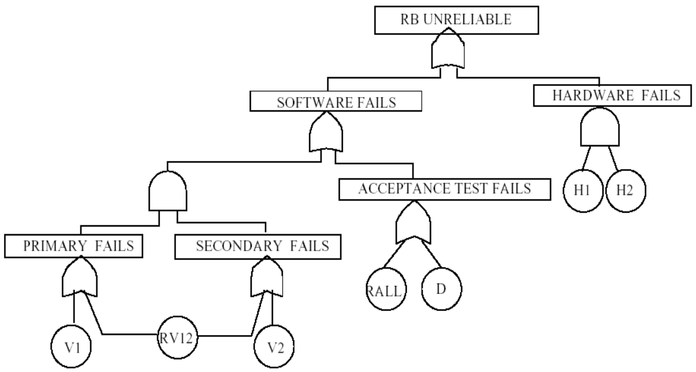
- Recovery blocks (RB)
- N-version programming (NVP)
- N-self-checking programming (NSCP)

Recovery Block (L#6, revisit)

- Three software elements
  - A primary routine (PR) executing a critical function
  - An acceptance test checking the results of the PR after each execution
  - One or more secondary/alternate routines
    - Performing the same function as the PR
- Implementation [Lyu96]



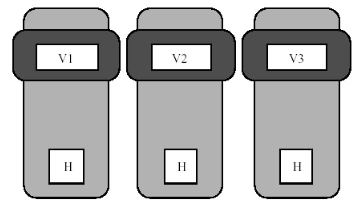
Fault Tree Model of DRB



- **RV12**: the input for a single computation activates a related faults between two software versions
- **RALL**: a related fault affects all versions and the acceptance test, caused by imperfect specifications
- **V#**: the input for a single computation activates an unrelated fault
- **H#**: a hardware fault affects the task computation
- **D:** an independent fault in the decider (acceptance test, majority voter, comparator)

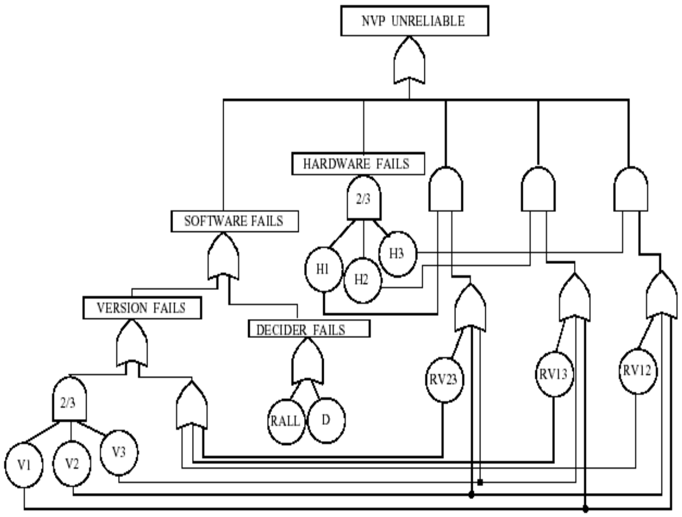
An Example of NVP (L#6, revisit)

- Three identical hardware components, each running a distinct software version

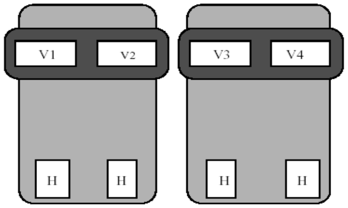


- Failure scenarios
  - Coincident unrelated faults
    - Software faults
    - Hardware faults
  - Related software faults
  - Combinations of hardware and software faults
    - A hardware host fails and one of the software component on another host also fails due to an unrelated or related fault

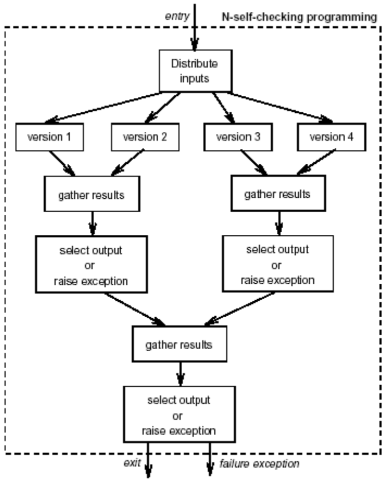
The Fault Tree of NVP



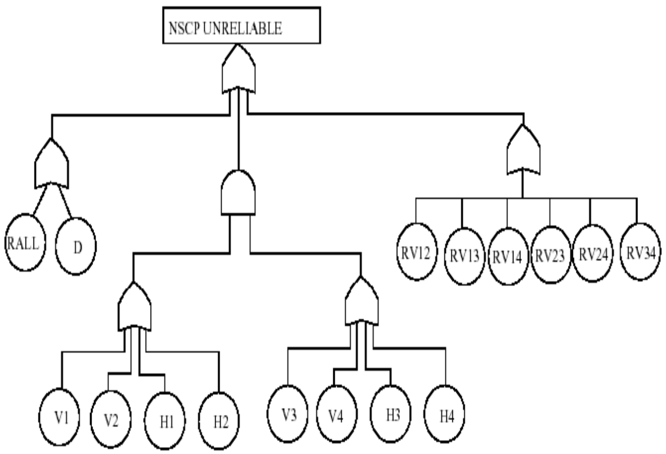
N-Self-Checking Programming (L#6, revisit)



- 4 software versions and 4 hardware components, grouped into two pairs



Fault Tree Model of NSCP



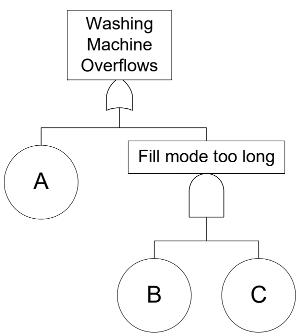
The NSCP system is vulnerable to related faults, whether they involve versions in the same error confinement area or not!

Agenda

✓ Introduction to fault tree analysis
✓ Fault tree construction
- **Fault tree analysis using cut-sets**

Fault Tree Analysis using Cutsets

- Many fault tree analysis techniques begin with the generation of the set of **cutsets**.
- A cutset is a set of basic events. If all the basic events in the set occur then the top event (system failure) occurs.
- A **mincut (minimal cutset)** is one that contains no redundant elements. If an element is removed from a mincut it ceases to be a cutset.



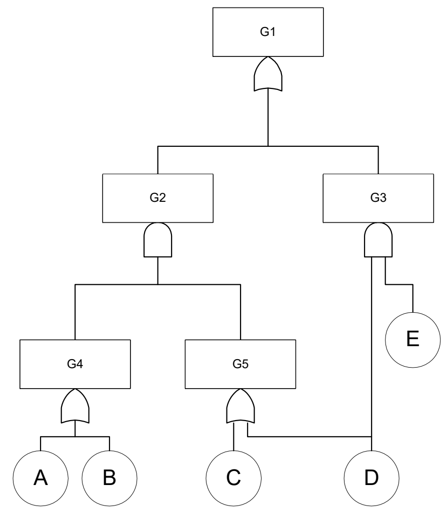
Cutsets: {{A}, {A,B}, {A,C}, {A,B,C}, {B,C}}

Mincuts: {{A}, {B,C}}

Note:

We often simply refer to cutsets when we mean mincuts.

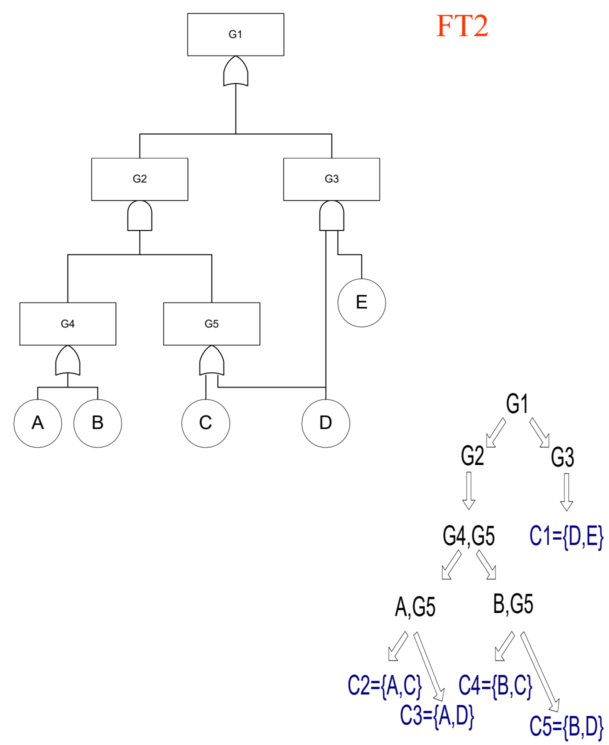
Cutsets for this larger example?



Cutset Generation

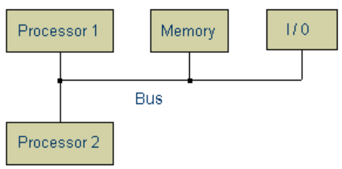
- Top-down algorithm:
  - Starts at the top event of the fault tree and constructs the set of cutsets by considering the gates at each lower level
  - A set of cutsets is expanded at each lower level of the tree until the set of basic events is reached
  - If the gate being considered is an AND gate then all the inputs must occur to enable the gate so a gate is replaced at the lower level by a listing of all its inputs
  - If the gate being considered is an OR gate then the cutset being built is split into several cutsets one containing each input to the OR gate
  - If a gate being expanded is a K-out-of-N gate then its expansion is a combination of the OR and AND expansions. The K-out-of-N gate is expanded into the C_N^K combinations of input events that can cause the gate to occur

Cutset Generation For Large FT



Hands-on Problem (1, Cont'd)

- System is operational *iff* 1 processor, memory, I/O and bus are functioning



  - Find fault tree model of the system
  - **Find the minimal cut sets**

Fault Tree Analysis using Cutsets (Cont'd)

- Qualitative analysis
- Quantitative analysis

Qualitative Analysis

- Qualitative analysis of the fault tree usually consists of studying the minimal cutsets
  - Determine the existence of the single-point failures
    - A single-point of failure is any component whose failure by itself can cause system failure
    - Identified by cutsets with a single element
    - Revisit examples: FT1, FT2
  - Determine the system vulnerability resulting from a particular component failure
    - FT2: once D fails, the system is vulnerable to a failure of either A, B, or E

Quantitative Analysis

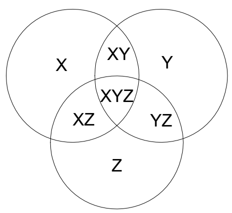
- Quantitative analysis of fault trees is used to determine the occurrence probability of the top event (system failure), given the probability of occurrence for the basic events

$$Pr\{System \quad Failure\} = Pr\{\bigcup_i C_i\}$$

  - Inclusion-exclusion (I/E)
  - Sum of disjoint products (SDP)

Example: Pr(A∪B)?

Quantitative Analysis using Inclusion-Exclusion (I/E)



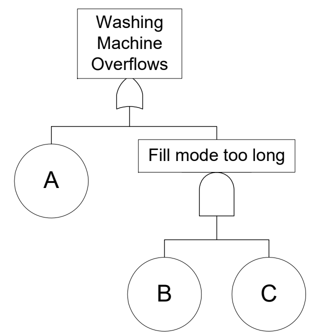
Pr{X + Y + Z}
= Pr{X} + Pr{Y} + Pr{Z}
- Pr{XY} - Pr{XZ} - Pr{YZ}
+ Pr{XYZ}

In general:
+ each single item
- each 2-way pair
+ each 3-way combination
…
+- combination of all items

$$\begin{aligned} Pr\{\bigcup_{i=1}^{n} C_i\} &= \sum_{i=1}^{n} Pr\{C_i\} \\ &- \sum_{i<j} Pr\{C_i \bigcap C_j\} \\ &+ \sum_{i<j<k} Pr\{C_i \bigcap C_j \bigcap C_k\} \\ &\mp \cdots \\ &\pm Pr\{\bigcap_{i=1}^{n} C_i\} \end{aligned}$$

- The bounds on the system failure probability can be determined by using only a portion of the terms in above equation

I/E Method: Example



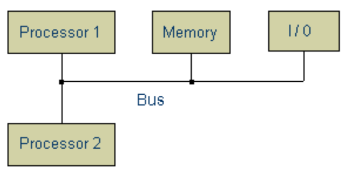
- F = A + BC
- Suppose
  Pr{A} = 0.01
  Pr{B} = 0.05
  Pr{C} = 0.075

and that all failures are s-independent

Find the system unreliability, i.e., the probability that the washing machine overflows.

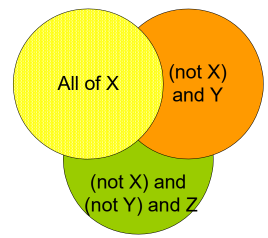
Hands-on Problem (1, Cont'd)

- System is operational *iff* 1 processor, memory, I/O and bus are functioning



  - Find fault tree model of the system
  - Find the minimal cut sets
  - **Find the system unreliability formula using the I/E method**, assuming the reliability of each component as follows
    - Two processors have the same reliability: *p*
    - Memory: *m*
    - I/O: *d*
    - Bus: *b*

Quantitative Analysis using Sum-of-Disjoint-Products (SDP)



Pr{X + Y + Z}
= Pr{X}
+ Pr{(not X) AND Y}
+ Pr{(not X) and (not Y) AND Z}

- Need to make each term disjoint from each previous term. In general:

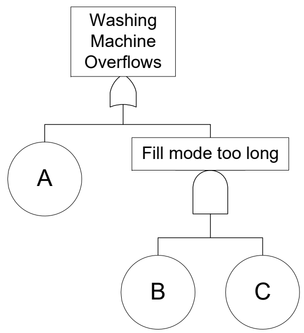
$$\bigcup_{i=1}^{n} C_i = C_1 \cup (\overline{C_1}C_2) \cup (\overline{C_1}\,\overline{C_2}C_3) \cup \ldots \cup (\overline{C_1}\,\overline{C_2}\,\overline{C_3}\ldots\overline{C_{n-1}}C_n)$$

- No subtraction, more efficient than I/E

$$Unreliablity = \Pr\{\bigcup_{i=1}^{n} C_i\}$$

$$= P(C_1) + P(\overline{C_1}C_2) + P(\overline{C_1}\,\overline{C_2}C_3) + \ldots + P(\overline{C_1}\,\overline{C_2}\,\overline{C_3}\ldots\overline{C_{n-1}}C_n)$$

SDP Method: Example



- F = A + BC
- Suppose

  Pr{A} = 0.01

  Pr{B} = 0.05

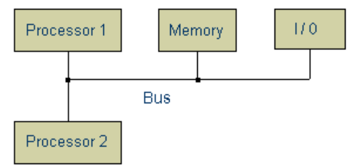
  Pr{C} = 0.075

and that all failures are s-independent

Find the system unreliability, i.e., the probability that the washing machine overflows.

Hands-on Problem (1, Cont'd)

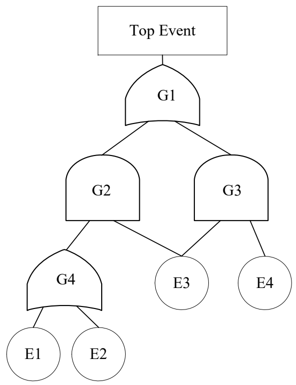
- System is operational *iff* 1 processor, memory, I/O and bus are functioning



- Find fault tree model of the system
- Find the minimal cut sets
- **Find the system unreliability formula using the SDP method,** assuming the reliability of each component as follows
  - Two processors have the same reliability: *p*
  - Memory: *m*
  - I/O: *d*
  - Bus: *b*

Review Question

For the fault tree below, calculate the probability of occurrence for the top event in the fault tree.



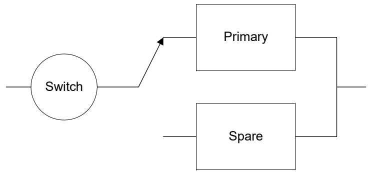
- Assume occurrence probabilities of basic events are:

  Pr(E1)=0.1, Pr(E2)=0.05,

  Pr(E3)=0.01, Pr(E4)=0.02

Future Topic:
Dynamic Fault Trees

- Traditional (static) fault trees cannot model sequence dependent failures, in which the *order* that events occur is important.
- Sequence dependencies do exist in practical systems



  – Failure criteria depends on the *order* in which the failure occur.

Summary of Lecture #9

- Fault tree is not a tree in the graph-theoretic sense; it provides a logical framework for expressing combinations of component failures that can lead to system failure
- Top-down construction of fault trees provides a systematic method for analyzing and documenting the potential causes of system failure
- Qualitative analysis of fault trees based on cutsets can identify the single-point failures and system vulnerability
- Quantitative analysis of fault trees using cutsets
  - Inclusion/Exclusion (I/E)
  - Sum of Disjoint Products (SDP)
  - SDP is more efficient than I/E

Things to Do

- Homework
- ECE544 Project Meeting
  - Due **October 28, Friday**