Dependable Systems

Dependability Modeling

Dr. Peter Tröger

Sources:

Eusgeld, Irene et al.: Dependability Metrics. 4909. Springer Publishing, 2008
Menasce, Daniel A.; Almeida, Virgilio A.: Capacity Planning for Web Services: Metrics, Models, and Methods. Prentice Hall, 2002. , 0-13-065903-7
Krishna B. Misra: Handbook of Performability Engineering. Springer. 2008
Hazard Analysis Techniques for System Safety, by Clifton A. Ericson, II
www.fault-tree.net
http://www.hq.nasa.gov/office/codeq/doctree/fthb.pdf

Dependability Modeling

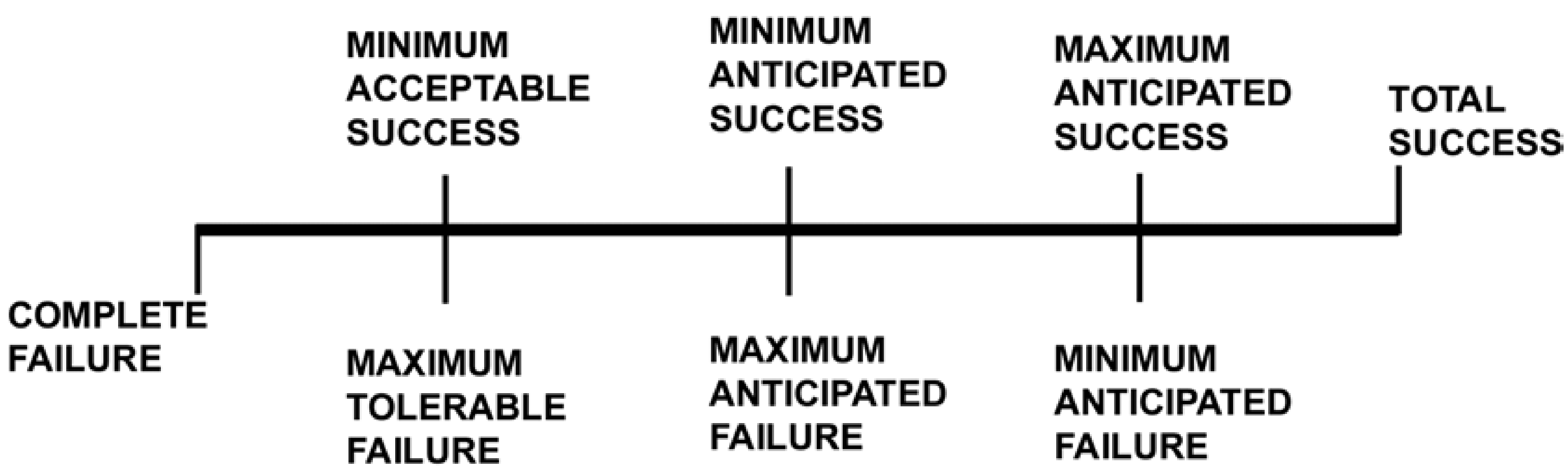
- Default approach: Utilize a formalism to model system dependability
  - Quantify the availability of components, calculate system availability based on this data and a set of assumptions (the availability model)
    - Most models expose the same expressiveness
    - Each formalism allows to focus on certain aspects
    - Structure-based models: Reliability block diagram, fault tree
    - State-based models: Markov chain, petri net
- System understanding evolved from hardware to software to IT infrastructures
  - Example: Organization management influence on business service reliability
    - Information Technology Infrastructure Library (ITIL)
    - CoBiT(Control Objectives for Information and related Technology)

History

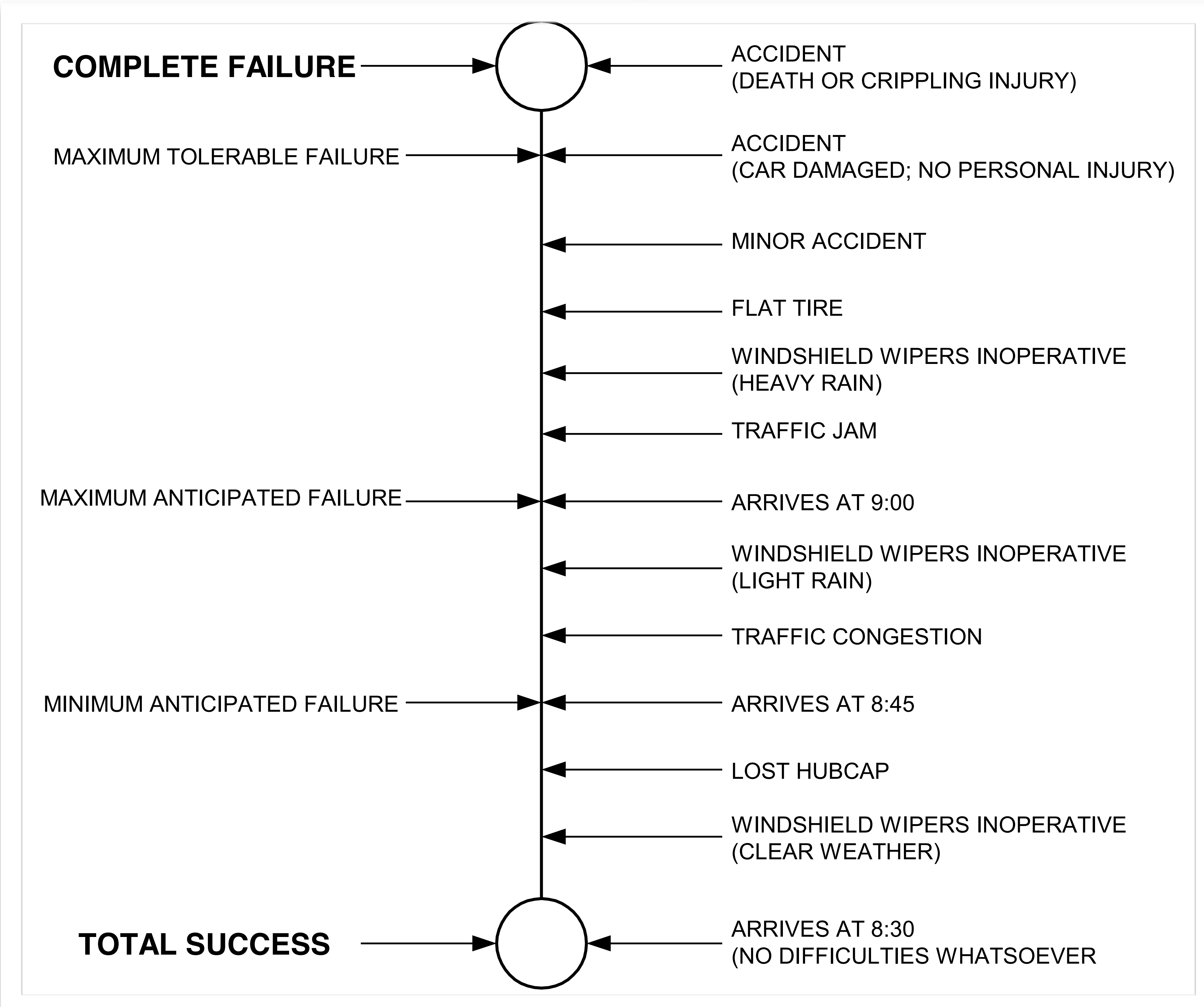
- Methods for risk and reliability assessment originate in the early 60‘s
  - US aerospace and missile programs
  - Importance for NASA grew after Challenger accident in 1986
- Importance for nuclear industry grew after Three Mile Island accident in 1979
  - Fault Tree Handbook, NUREG-0492, US Nuclear Regulatory Commission
- Meanwhile established methodologies and commercial / academic tools
  - SAVE, SHARPE, Fault Tree+, AvSim+, ReliabilityWorkbench, BlockSim, Figaro/KB3, Galileo/ASSAP, BQR Care, ...

Dependability Modeling

- The *Failure Space-Success Space* concept
  - Often easier to agree on what constitutes a system failure
  - Success tends to be associated with system efficiency, which makes it harder to formulate events in the model („The car drives fast.“, „The car stops driving.“)
  - In practice, there are more ways to success than to failure



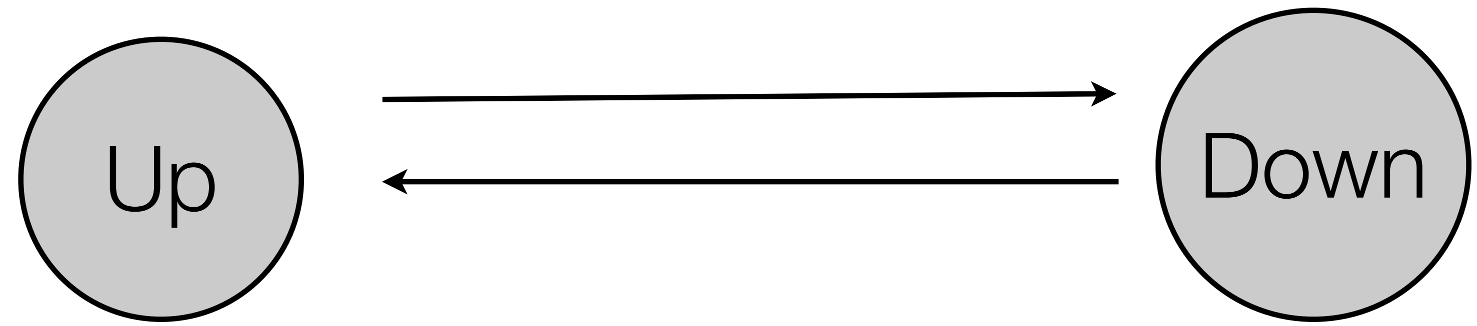
Example: Failure Space



Dependability Modeling

- System analysis approaches
  - **Inductive methods** - Reasoning from specific cases to a general conclusion
    - Postulate a particular fault or initiating event, find out system effect
    - Determine **what** system (failure) states are possible
    - Trivial approach: „parts count“ method
    - Examples: Failure Mode and Effect Analysis (FMEA), Preliminary Hazards Analysis (PHA), Event Tree Analysis, Reliability Block Diagrams (RBD), ...
  - **Deductive methods** - Postulate a system failure, find out what system modes or component behaviors contribute to this failure
    - Determine **how** a particular system state can occur
    - Examples: Fault Tree Analysis (FTA)

General Rules



- Components are either fully working or completely failed
- Two options for expressing the probability that the success / failure event occurs
  - Based on **(un)reliability data**
    - Model contains probability for a given point in time, or (un)reliability function
  - Based on **availability data**
    - Model contains numerical probability for (non-)failure at **any** point in time
    - Demands definition of probability distribution function and its parameters (typically exponential distribution)
- All failure and repair events are pair-wisely stochastically independent

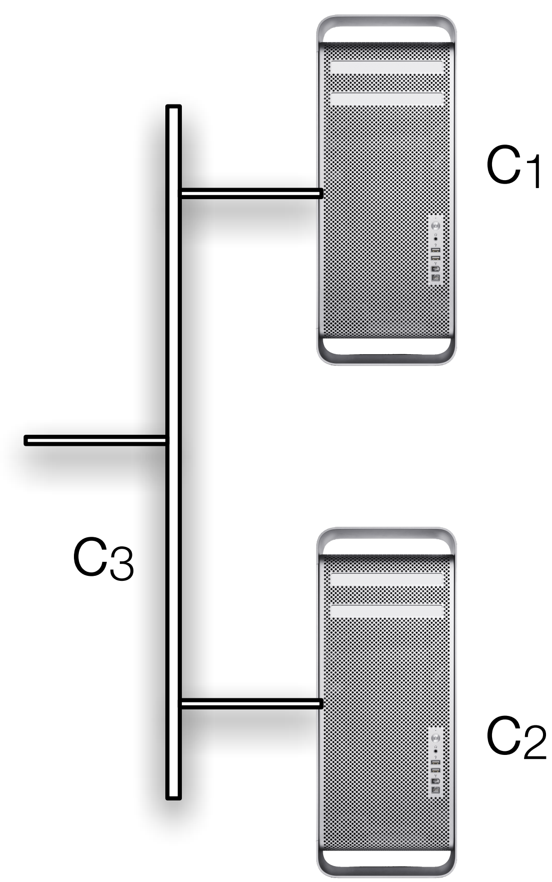
Inductive Modeling - Boolean Algebra Approach

- For stochastically independent events:

$$Pr(\phi_1 \wedge \phi_2) = Pr(\phi_1) \cdot Pr(\phi_2)$$
$$Pr(\phi_1 \vee \phi_2) = Pr(\phi_1) + Pr(\phi_2) - Pr(\phi_1 \wedge \phi 2)$$
$$Pr(\neg \phi) = 1 - Pr(\phi)$$

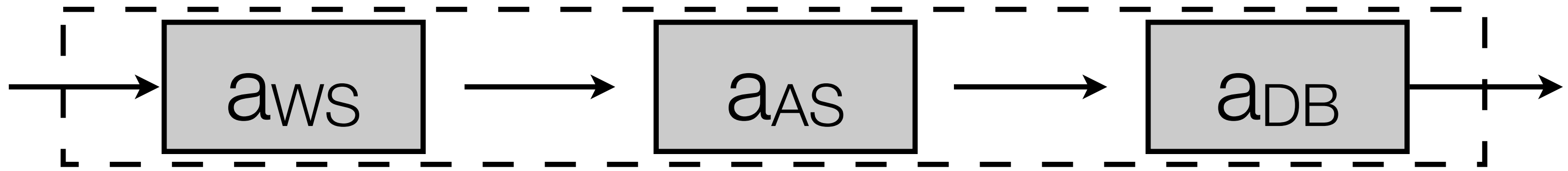
- **c_i : The binary event that component c_i is operational at any given point in model time**

- **$a_i = Pr(c_i)$: Probability that c_i occurs -> Availability !**

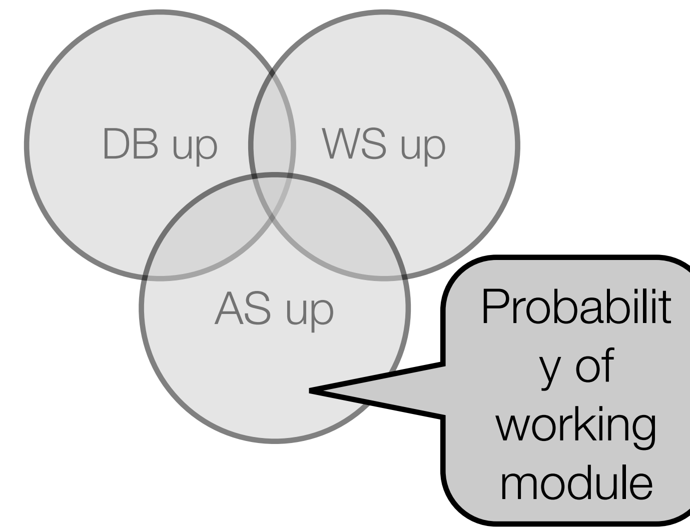


$$\phi = (c_1 \vee c_2) \wedge c_3$$
$$\begin{aligned} Pr(\phi) &= Pr((c_1 \vee c_2) \wedge c3) \\ &= (a_1 + a_2 - a_1 \cdot a_2) \cdot a_3 \\ &= a_1 a_3 + a_2 a_3 - a_1 a_2 a_3 \end{aligned}$$

Serial Case



- Help from probability theory: *The probability of an event expressed as the intersection of independent events is the product of the probabilities of the independent events.*



- Example: Chain of web server (a=0.9), application server (a=0.95) and database server (a=0.99)
    - Benefit of replacing the database with an expensive model (a=0.999) ?
    - Benefit of replacing the web server with a new model (a=0.95) ?

$$\phi_S = c_{WS} \wedge c_{AS} \wedge c_{DB}$$

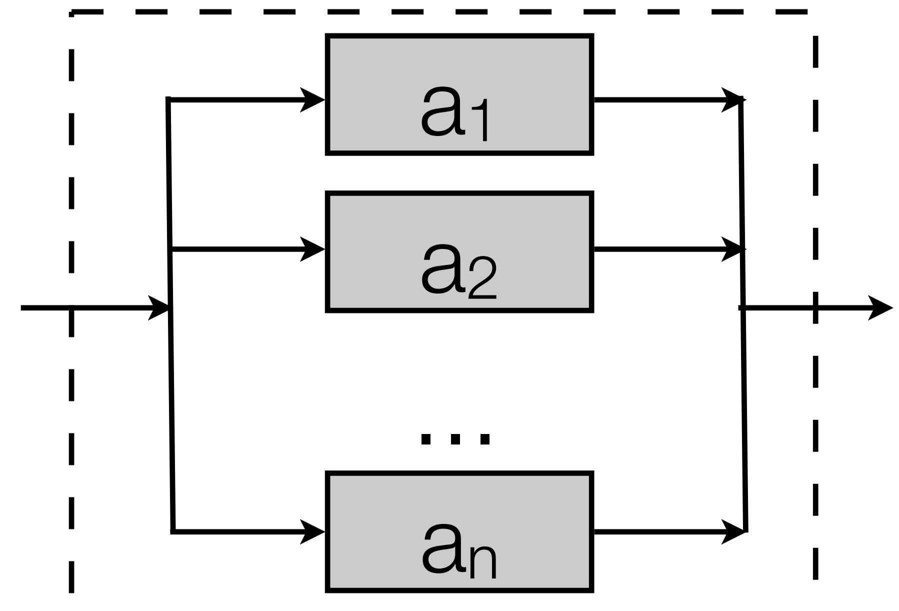
Redundancy structure

Component available

$$A_S = a_1 \times a_2 ... a_n = \prod_{i=1}^{n} a_i$$

Parallel Case

- Parallel case
  - Search engine, cluster node a=0.85 (around 2 months outage / year)
  - How many servers to reach 5 nines of site availability ?



$$\phi_S = a_1 \vee a_2 \vee ... \vee a_n$$

Redundancy structure

Component available

$$A_S = 1 - P_{alldown}$$

$$A_S = 1 - ((1 - a_1) \times (1 - a_2) \times ...(1 - a_n))$$

$$A_S = 1 - \prod_{i=1}^{n}(1 - a_i)$$

$$n_{min} = \lceil \frac{ln(1-A_S)}{ln(1-a)} \rceil$$

K-of-N Systems

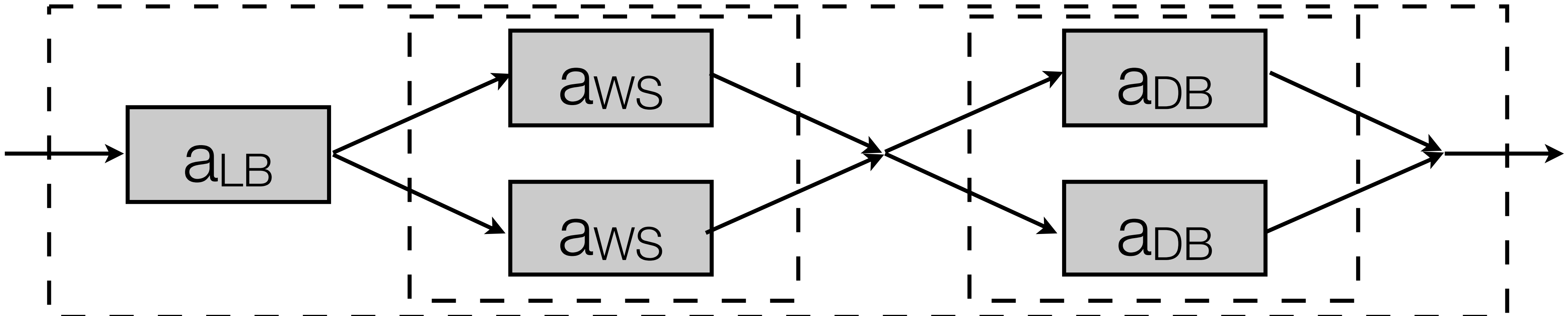
- *At least K out of N identical independent components need to work to have a functioning system*
- Algebraic investigation only feasible with exponential failure distribution
    - At the beginning, there are N operational units, so failure rate equals $N \cdot \lambda$
    - After first component failure, the failure rate goes down to $(N-1) \cdot \lambda$
    - This goes until the (K+1)th failure has occurred

$$MTTF = \sum_{K \leq j \leq N} \frac{1}{\lambda j}$$

    - K=1 is the same as the parallel case, K=N is the same as the serial case
- For identical components, survival probability can be computed as:

$$A_S(k, N, a) = \sum_{i=k}^{N} \binom{N}{i} a^i (1-a)^{n-i}$$

Examples

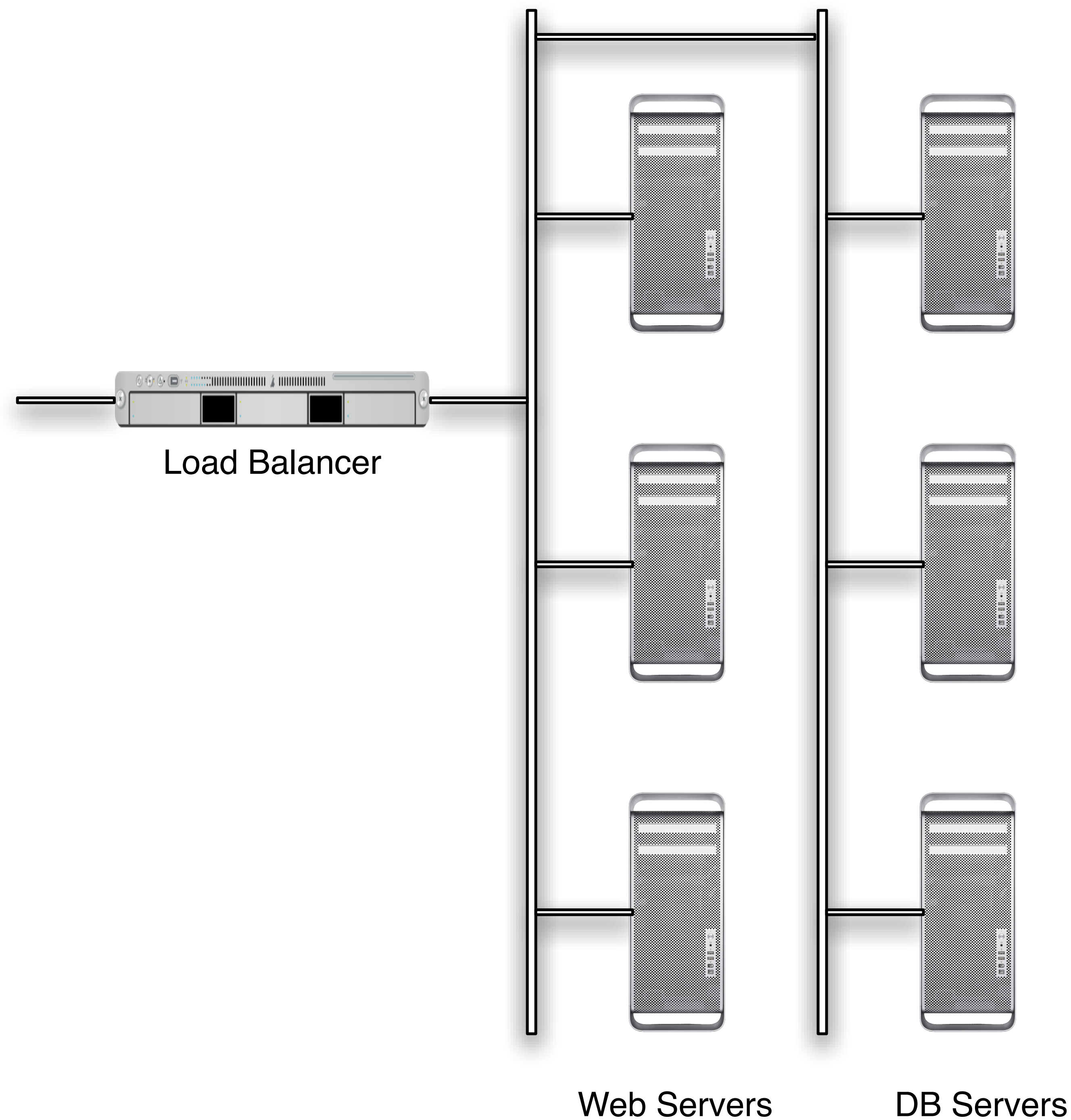


$$\phi_S = c_{LB} \wedge (c_{WS1} \vee c_{WS2}) \wedge (c_{DB1} \vee c_{DB2})$$

$$A_{site} = a_{LB} \times A_{WSset} \times A_{DBset}$$
$$= a_{LB} \times [1 - (1 - a_{WS})^{n_{WS}}] \times [1 - (1 - a_{DB})^{n_{DB}}]$$

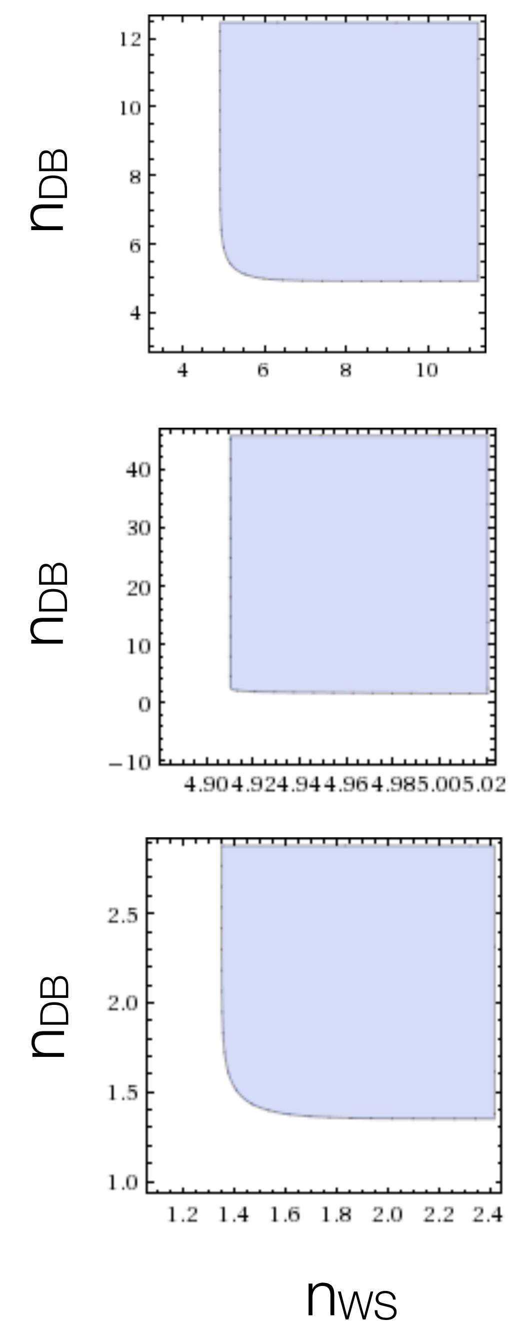
Examples

- Online brokerage site to be designed - choice of components needed
- Site availability aimed at 99.99%
- Setup: Load balancer, similar web server hardware, replicated database
- Question: What is the least expensive configuration that reaches 99.99% ?
  - Choice between low-end (a=0.85) and high-end (a=0.999) servers
  - Must also consider purchase and maintenance costs per setup



Examples

a	a	Minimum n	Minimum n	A
0,85	0,85	6	5	99,99 %
0,85	0,999	5	2	99,991 %
0,999	0,999	2	2	99,999 %



Example: How to reach 4 nine‘s ?

Examples

- Three identical hard drives in a parallel setup, two of them must operate

$$A_S = a_1 a_2 a_3 + (1-a_1)a_2 a_3 + a_1(1-a_2)a_3 + a_1 a_2 (1-a_3)$$

$$A_S = \binom{3}{3} a^3 (1-a)^0 + \binom{3}{2} a^2 (1-a)$$

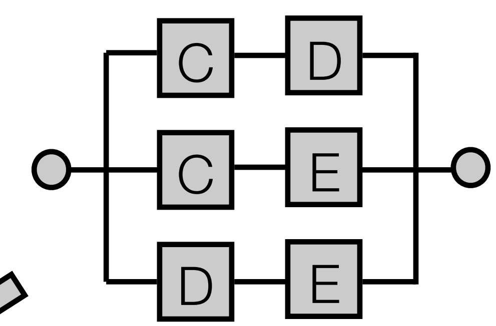
$$A_S = a^3 + 3a^2(1-a)$$

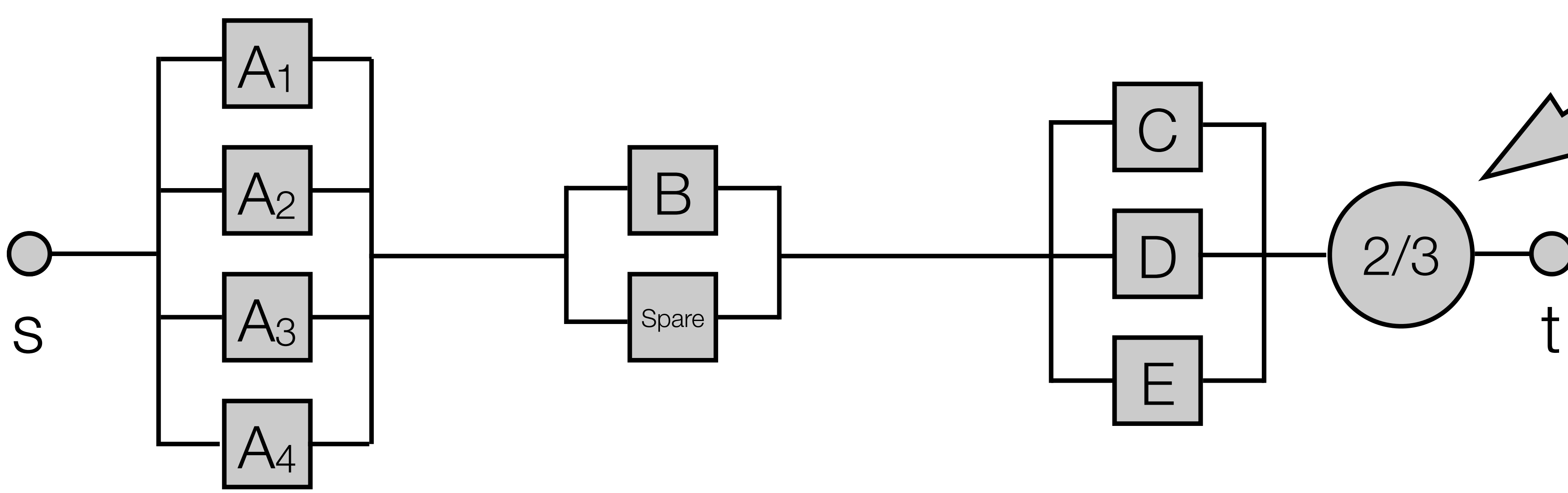
- Example: Disk RAID system with K=3, N=4, MTTF=1800h, MTTR=4.5h

$$MTTF = \tfrac{1}{4} MTTF_{Disk} + \tfrac{1}{3} MTTF_{Disk} = 1050h \qquad A_{Disk} = \tfrac{1800}{1800+4.5} = 0.9999628$$

Reliability Block Diagrams (RBD)

- Model logical interaction for success-oriented analysis of system reliability
- Building blocks: **series structure, parallel structure, k-out-of-n structure**
- System is available only if there is a path between **s** and **t**
- Granularity based on data and *lowest actionable item* concept
- Structure formula can be obtained from RBD by identifying the *subset of nodes that disconnects s from t* if removed

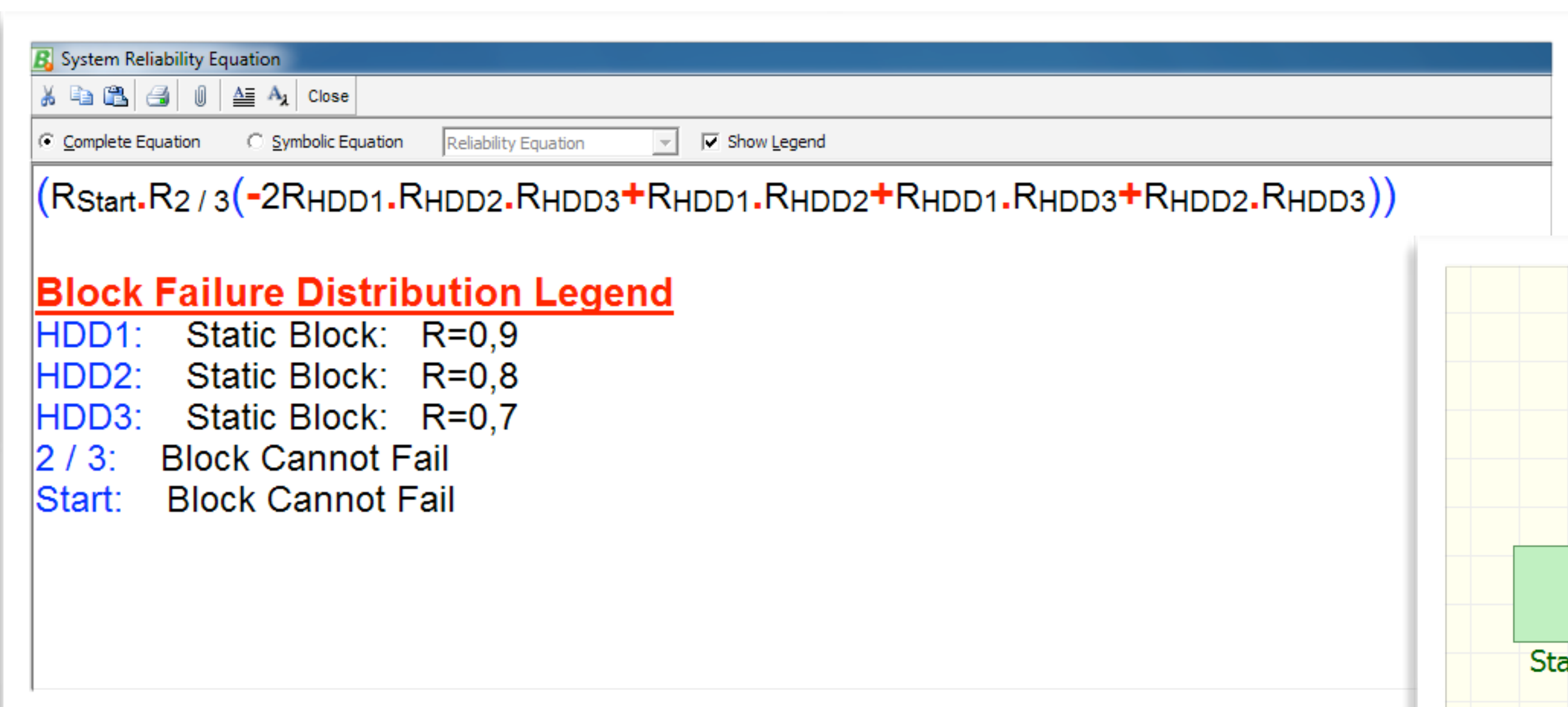


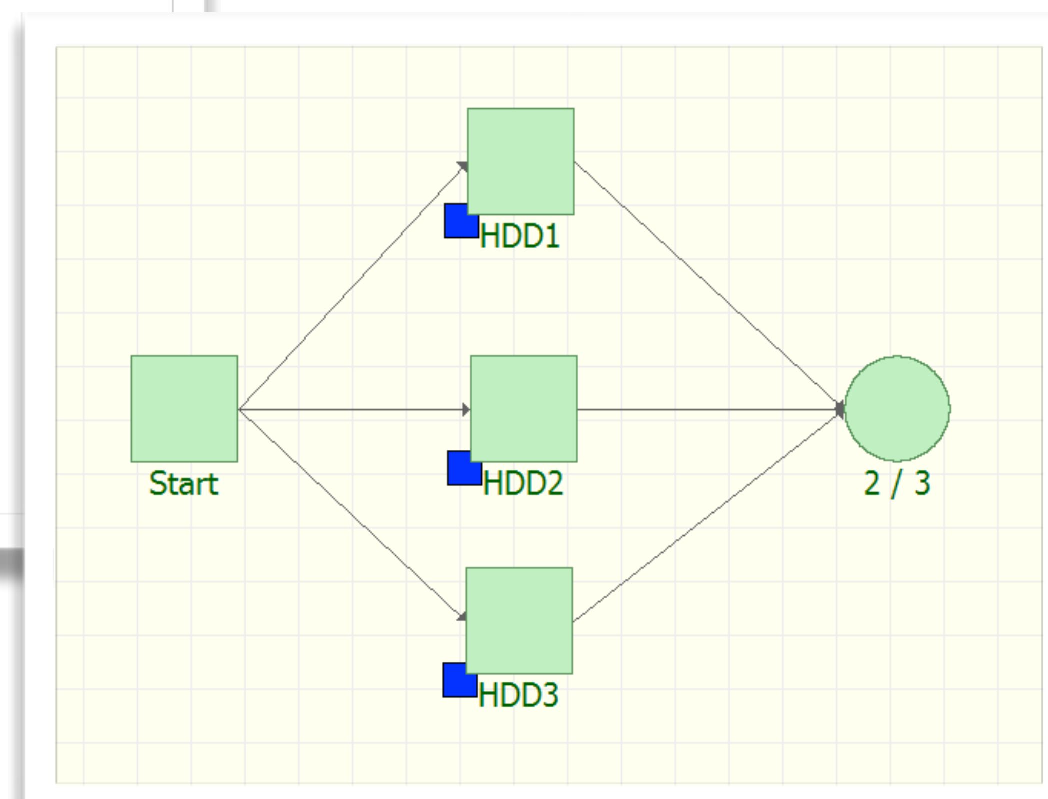


RBD: k-of-N for Nonidentical Components [ReliaSoft]

- Example: 2-out-of-3 different hard drives must remain functional
  - Different manufacturers with different device reliability

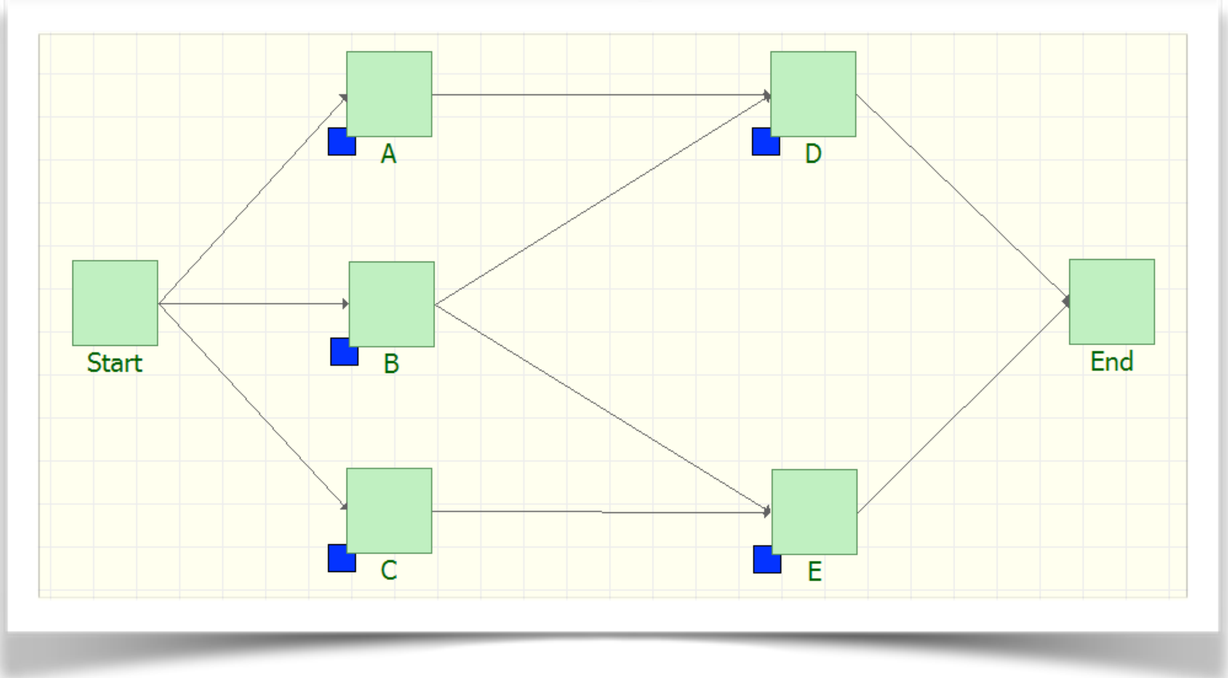
$$A_S = a_1 a_2 a_3 + (1 - a_1) a_2 a_3 + a_1 (1 - a_2) a_3 + a_1 a_2 (1 - a_3)$$





Complex RBDs

- Break down into serial and parallel sections not always obvious, for example:
  - A or B or C must work
  - If A works, D must work
  - If B works, than D or E must work
  - If C works, E must work



- **Decomposition method**:
  Identify key component, compute reliability with and without it, combine them
- **Event space method**:
  System reliability is the probability of the union of all mutually exclusive events that lead to system success
- **Path Tracing method**:
  Calculate probability of all possible paths through the RBD, combine for system survival probability

Complex RBDs

System Reliability Equation

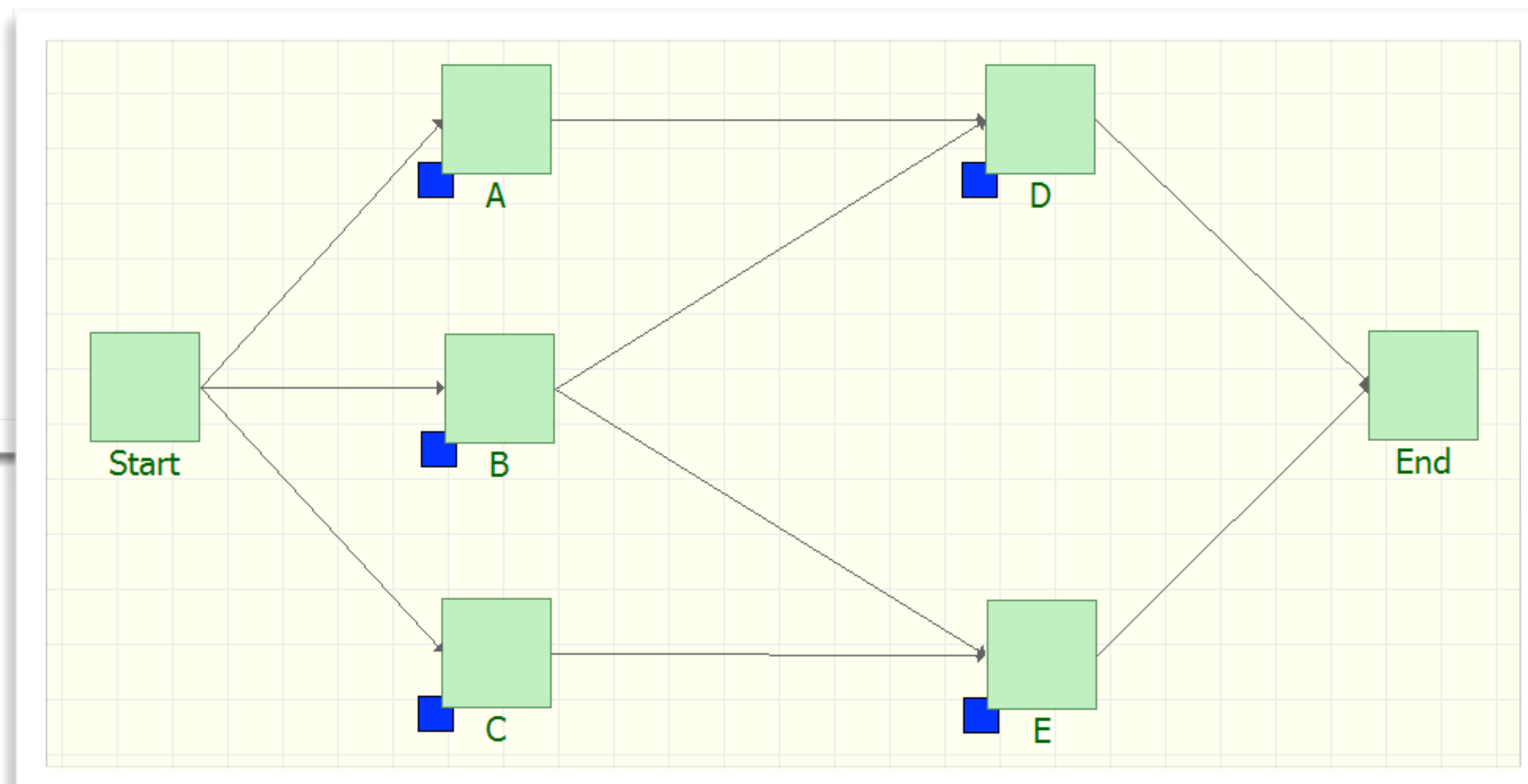
Close

Complete Equation | Symbolic Equation | Reliability Equation | Show Legend

$$(R_{Start}.R_{End}(R_D.R_E.R_A.R_C.R_B-R_D.R_E.R_A.R_C-R_D.R_E.R_B-R_D.R_A.R_B-R_E.R_C.R_B+R_D.R_A+R_D.R_B+R_E.R_C+R_E.R_B))$$

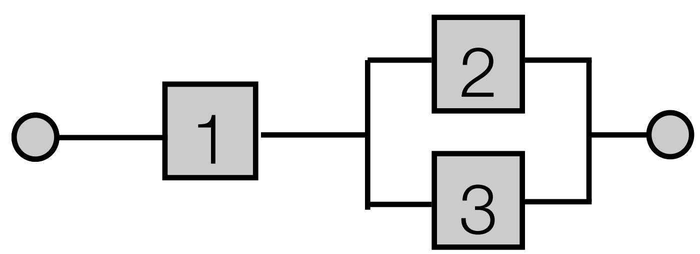
Block Failure Distribution Legend

B: Static Block: R=0,8
C: Static Block: R=0,7
A: Static Block: R=0,9
D: Static Block: R=0,9
E: Static Block: R=0,9
Start: Block Cannot Fail
End: Block Cannot Fail



More on Structure Functions [Rausand]

- State of each component described by a binary variable (1 -> functioning, 0 -> failed)
- **State vector** describes system state at specific point in time
- Binary **structure function** of the system based on current state vector



Structure function

$$\phi(X(t)) = X_1(t)(X_2(t) + X_3(t) - X_2(t)X_3(t))$$

State vector

State variable

$$R(t) = (Pr(\phi(X(t)) = 1)$$

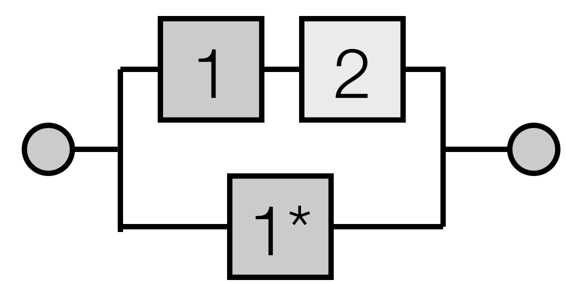
Coherent Structures [Rausand]

$$\phi(x)_{serial} = x_1 \cdot x_2 \cdots x_n = \prod_{i=1}^{n} x_i$$

$$\phi(x)_{parallel} = 1 - (1 - x_1)(1 - x_2) \cdots (1 - x_n) = 1 - \prod_{i=1}^{n} (1 - x_i) = \coprod_{i=1}^{n} x_i$$

$$\phi(x)_{k-out-of-N} = \left\{ \begin{array}{lll} 1 & if & \sum_{i=1}^{n} x_i \geq k \\ 0 & if & \sum_{i=1}^{n} x_i < k \end{array} \right\}$$

- In the description of a system structure, **relevant** components contribute to the *functioning ability* of the system
  - **Component irrelevance** with respect only to a specific system function



- **Coherent system structure**: All components are relevant
  - Any coherent system with n components is functioning at least as well as a corresponding system where all n components are in series, and at most as well as one with all components in parallel:

$$\prod_{i=1}^{n} x_i \leq \phi(x) \leq \coprod_{i=1}^{n} x_i$$

Coherent Structures [Rausand]

- Given two state vectors **x** and **y** for the same structure function (= system)

$$x = (x_1, x_2, \ldots, x_n)$$
$$y = (y_1, y_2, \ldots, y_n)$$

- Serial or parallel replication per component expressed by combined state vectors

$$x \cdot y = (x_1 y_1, x_2 y_2, \ldots, x_n y_n)$$
$$x \sqcup y = (x_1 \sqcup y_1, x_2 \sqcup y_2, \ldots, x_n \sqcup y_n)$$

- Theorem by Rausand et al. shows redundancy impact on coherent structure:

$$\phi(x \sqcup y) \geq \phi(x) \sqcup \phi(y)$$
$$\phi(x \cdot y) \leq \phi(x) \cdot \phi(y)$$

  - The „value“ of the structure function (=system) with component-level parallel redundancy is higher than the „value“ with system-level parallel redundancy

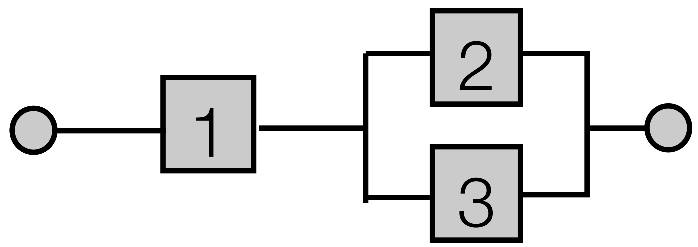
- If the system with component-level redundancy would fail, then the system-level redundancy design would also fail

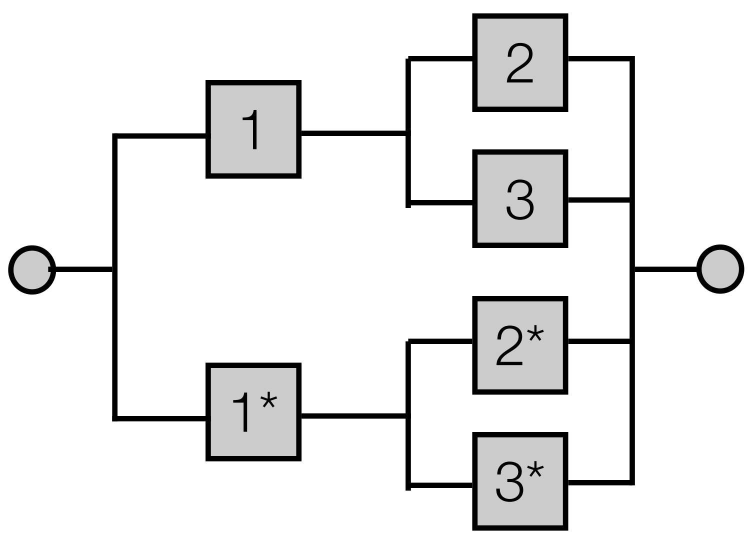
- There may (!) be cases where only the component-level redundancy design survives

  - In other words: Structure function is binary -> there are state vectors with

$$\phi(x \sqcup y) = 1$$
$$\phi(x) \sqcup \phi(y) = 0$$

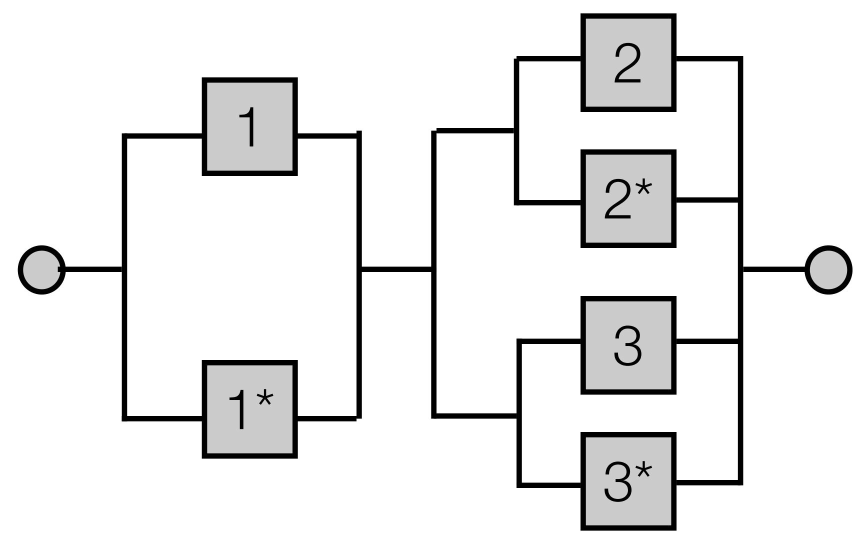
Coherent Structures [Rausand]





Redundancy at system level

$$\phi(x) \sqcup \phi(y)$$



Redundancy at component level

$$\phi(x \sqcup y)$$

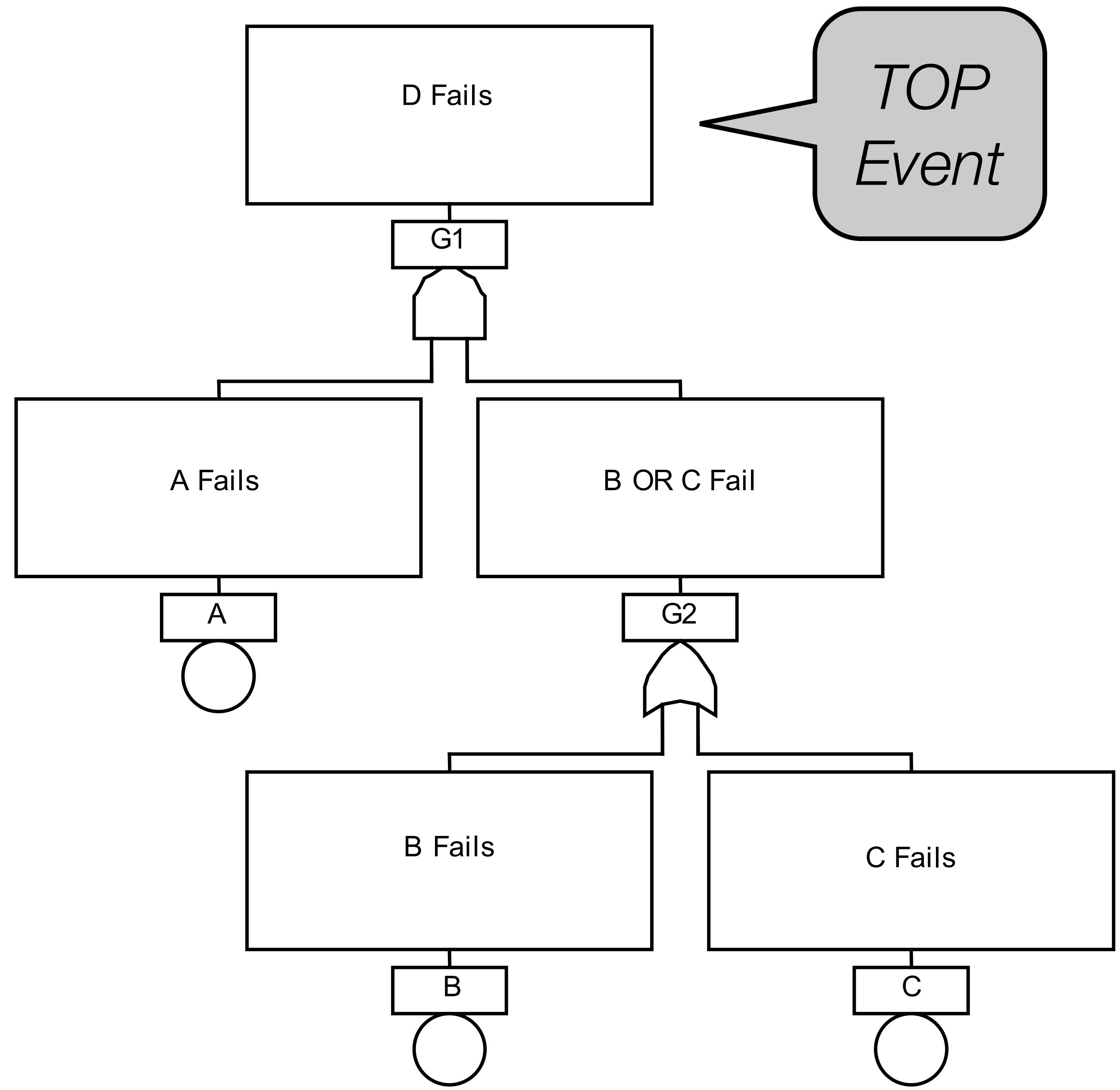
Example cases:
- 1 fails
- 1, 1* fail
- 1, 2*, 3* fail

Deductive Analysis - Fault Trees

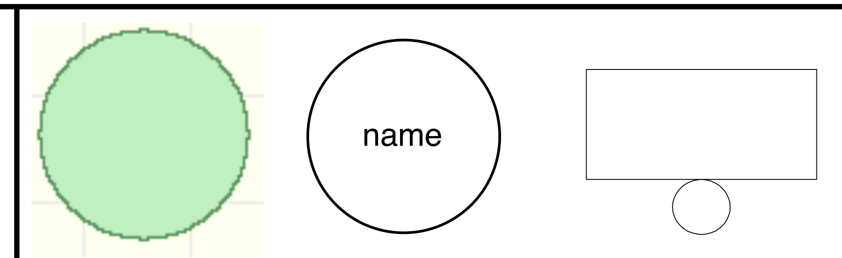
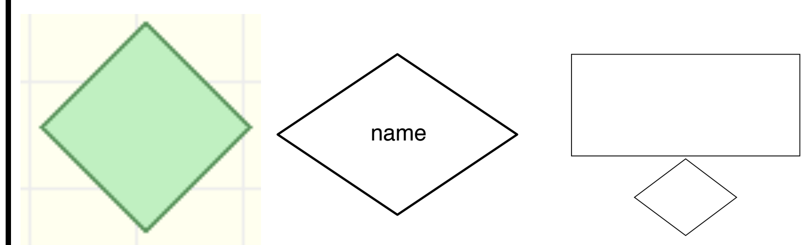
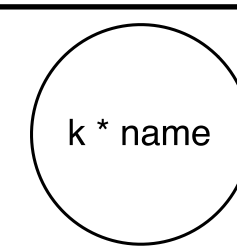
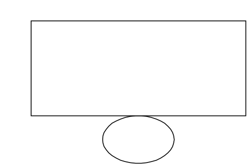
- Structure analysis effort grows exponentially with the number of components
- Fault Trees
  - Invented 1961 by H. Watson (Bell Telephone Laboratories)
    - Facilitate analysis of the launch control system of the intercontinental Minuteman missile
  - Used by Boeing since 1966, meanwhile adopted by different industries
  - Root cause analysis, risk assessment, safety assessment
- Basic idea
  - Technique for describing **the possible ways** in which an **undesired system state** can occur
  - Complex system failures are broken down into basic events

Fault Tree Analysis

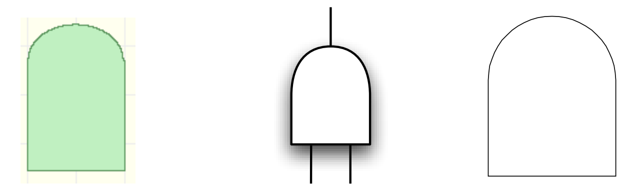
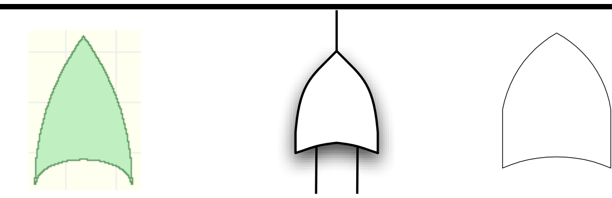
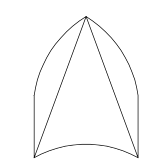
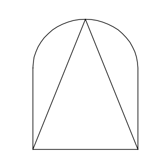
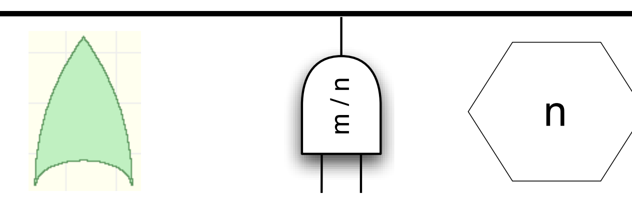
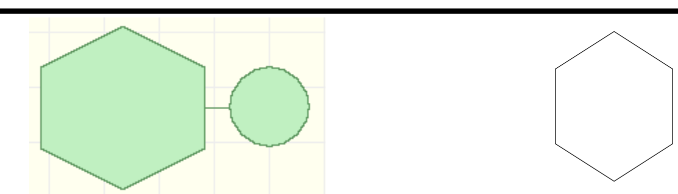
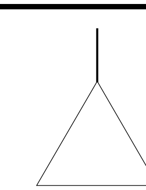
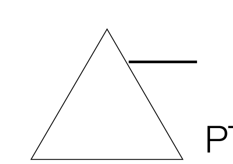
- Basic events (faults) can be associated with component hardware failures, human errors, software errors, or any other pertinent events
- Probability of a higher-level event can be calculated by lower level probabilities
- Graphical representation of structure formula, helps to identify fault classes
- Includes only faults that contribute to the top event
- In itself not a quantitative model, but can be evaluated as one
- Events and gates are not system components !



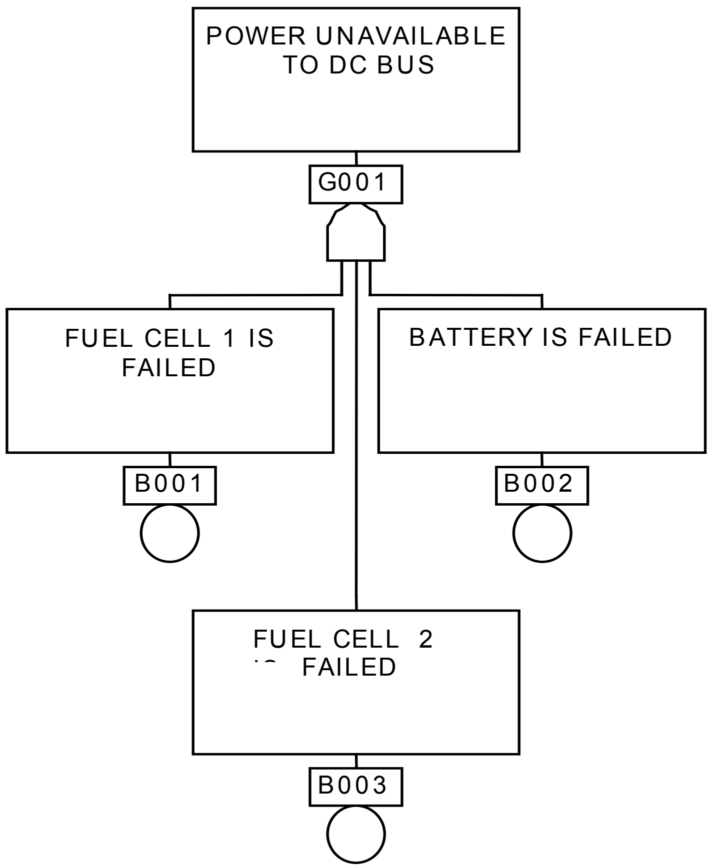
Static Fault Trees

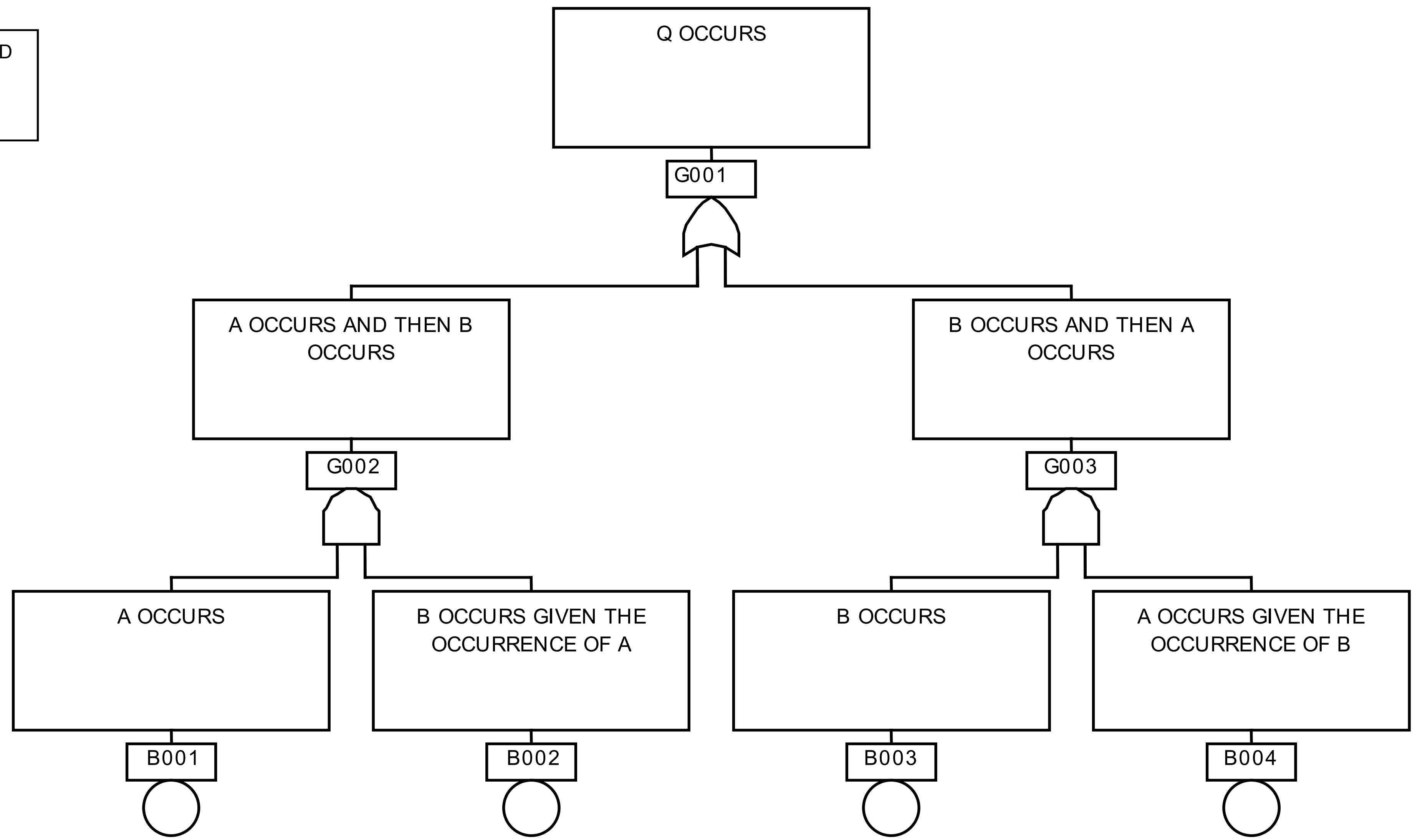
Basic event - Initiating fault, limit of resolution for the fault tree has been reached	
Undeveloped event - No information available or insignificant consequences	
House event - An event that is expected to occur and typically does not denote a failure (e.g. phase change)	
Replicated basic event - A given number of k statistically identical copies of a component	
Conditioning event - Restrictions that apply to the attached gate (e.g. INHIBIT / PRIORITY AND)	

Static Fault Trees

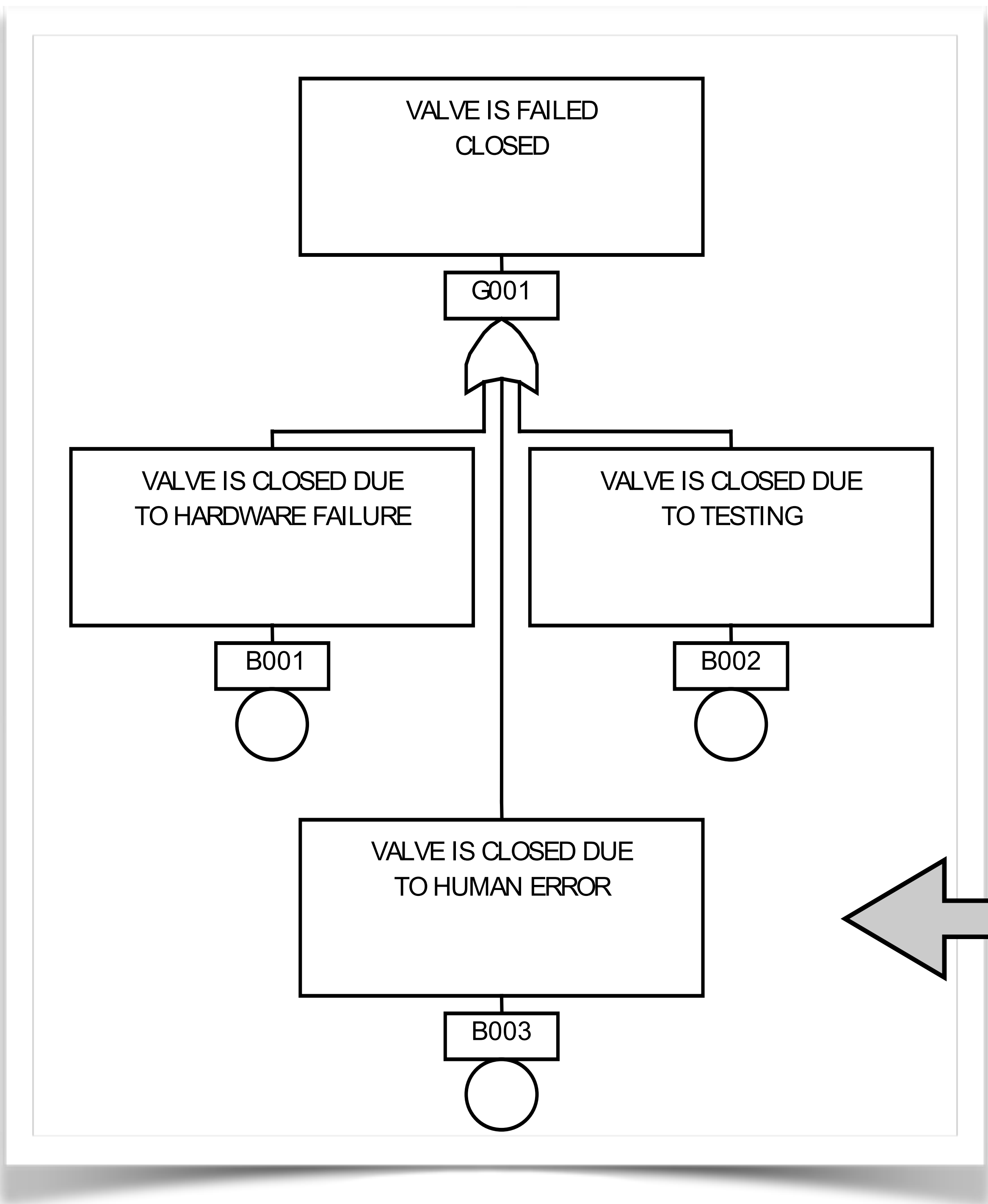
AND gate - Output event occurs if all input events occur	
OR gate - Output event occurs if one or more input events occur	
EXCLUSIVE OR gate - Output event occurs if exactly on of the input events occur	
PRIORITY AND gate - Output event occurs if all input events occur in the specific order	
COMBINATION / VOTING OR gate - Output event occurs if the given number of input events occur	
INHIBIT gate - Output event occurs if the single input event occurs and the enabling condition is given	
TRANSFER IN gate - Tree is further developed at the occurrence of the corresponding TRANSFER OUT gate	
TRANSFER OUT gate - This portion of the tree must be attached at the corresponding TRANSFER IN	

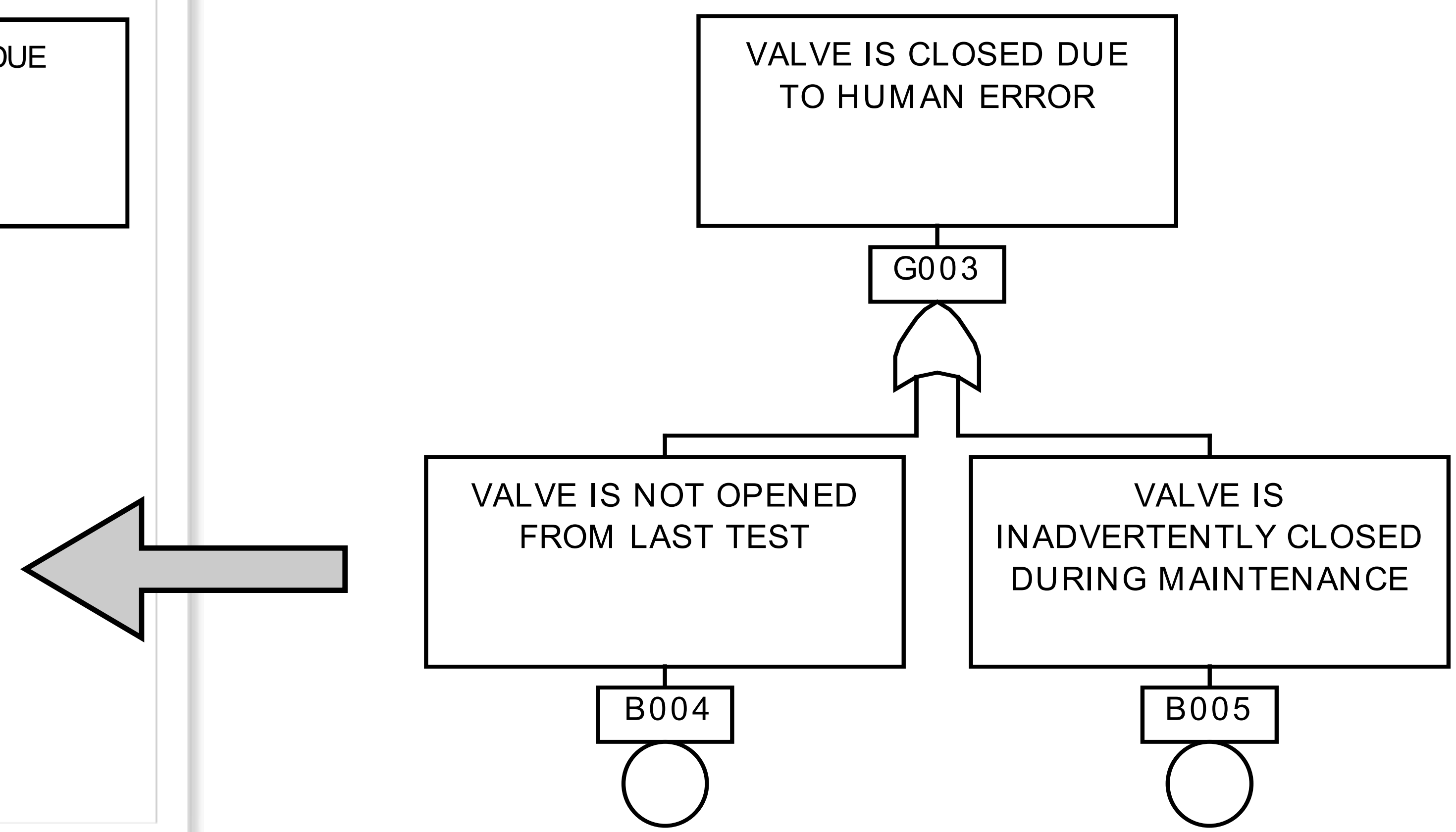
Examples: AND Gate



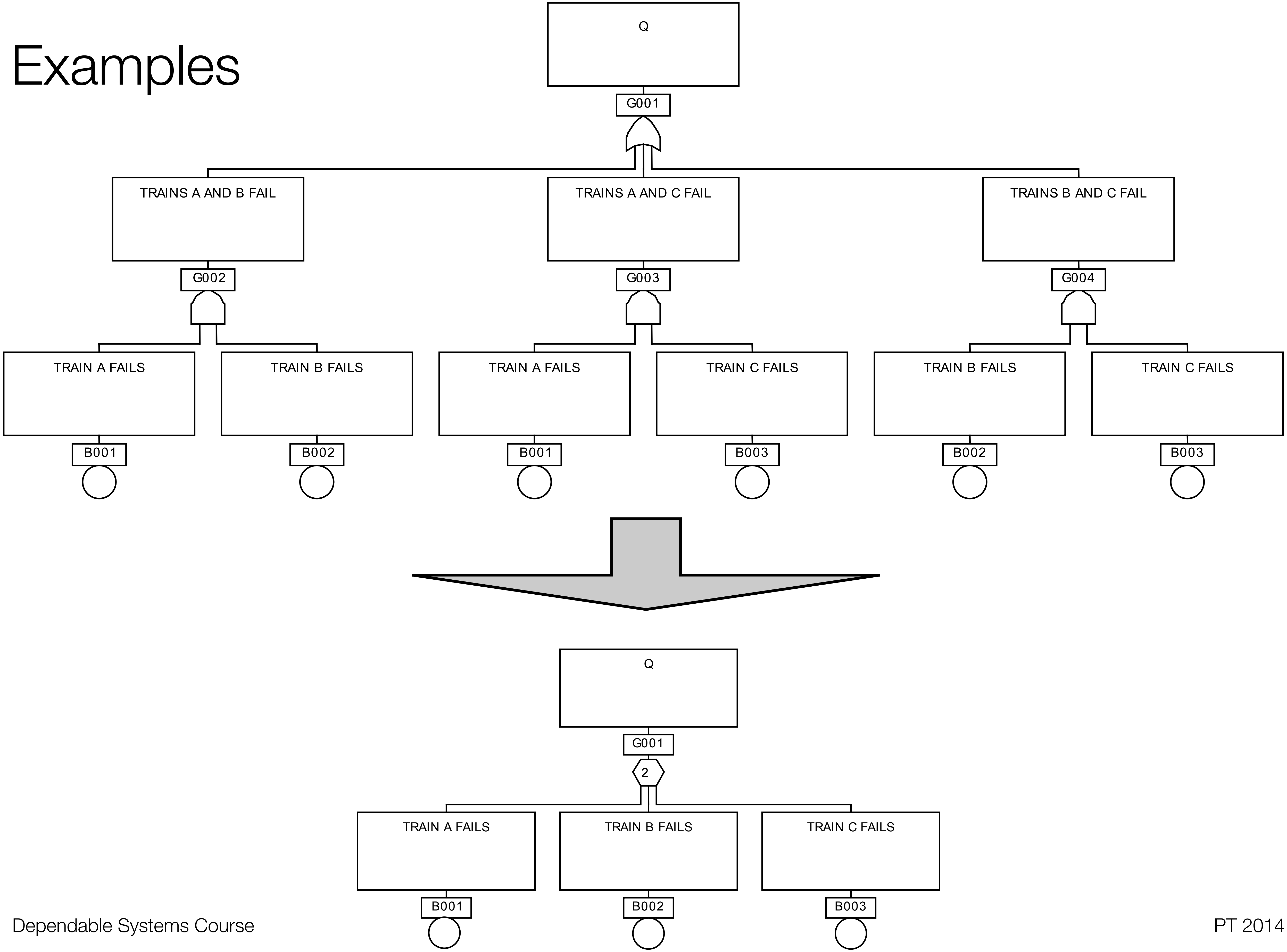


Examples: OR Gate

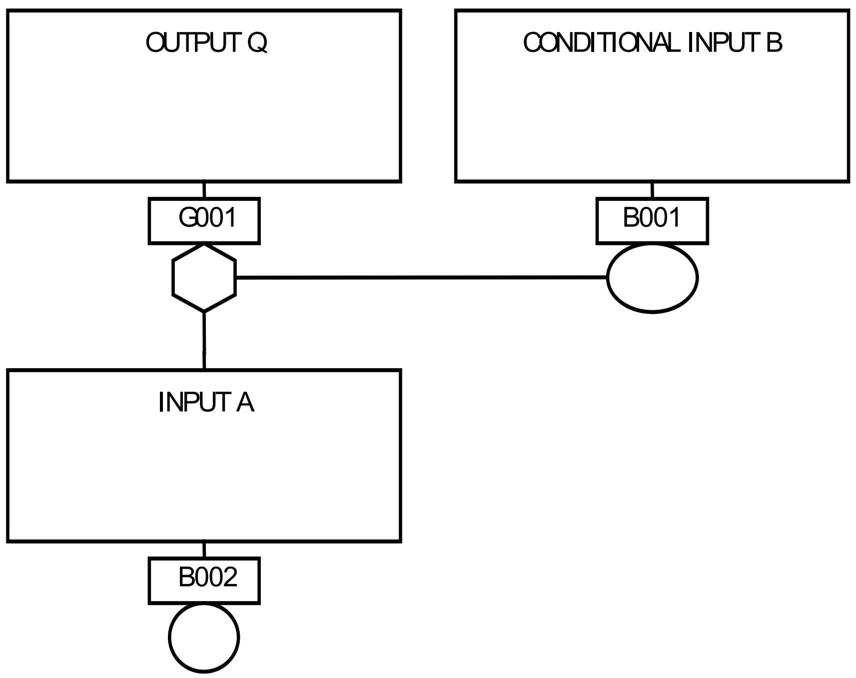


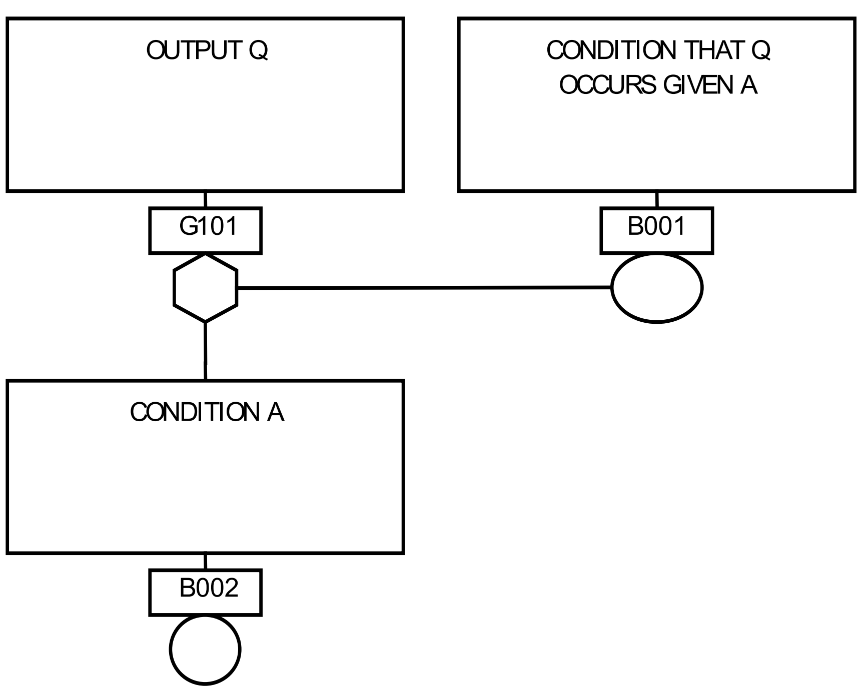


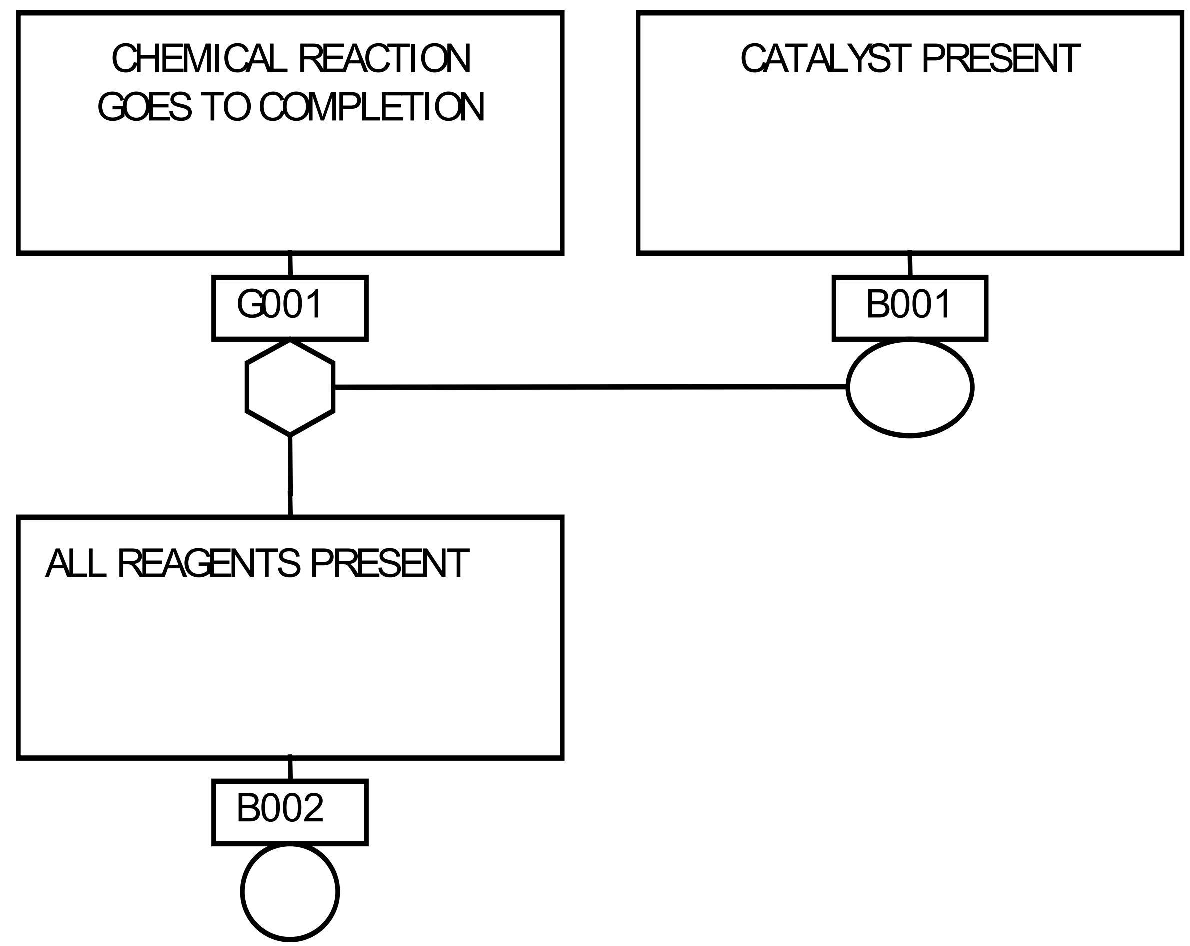
Examples


Q
G001
TRAINS A AND B FAIL
TRAINS A AND C FAIL
TRAINS B AND C FAIL
G002
G003
G004
TRAIN A FAILS
TRAIN B FAILS
TRAIN A FAILS
TRAIN C FAILS
TRAIN B FAILS
TRAIN C FAILS
B001
B002
B001
B003
B002
B003
Q
G001
2
TRAIN A FAILS
TRAIN B FAILS
TRAIN C FAILS
B001
B002
B003

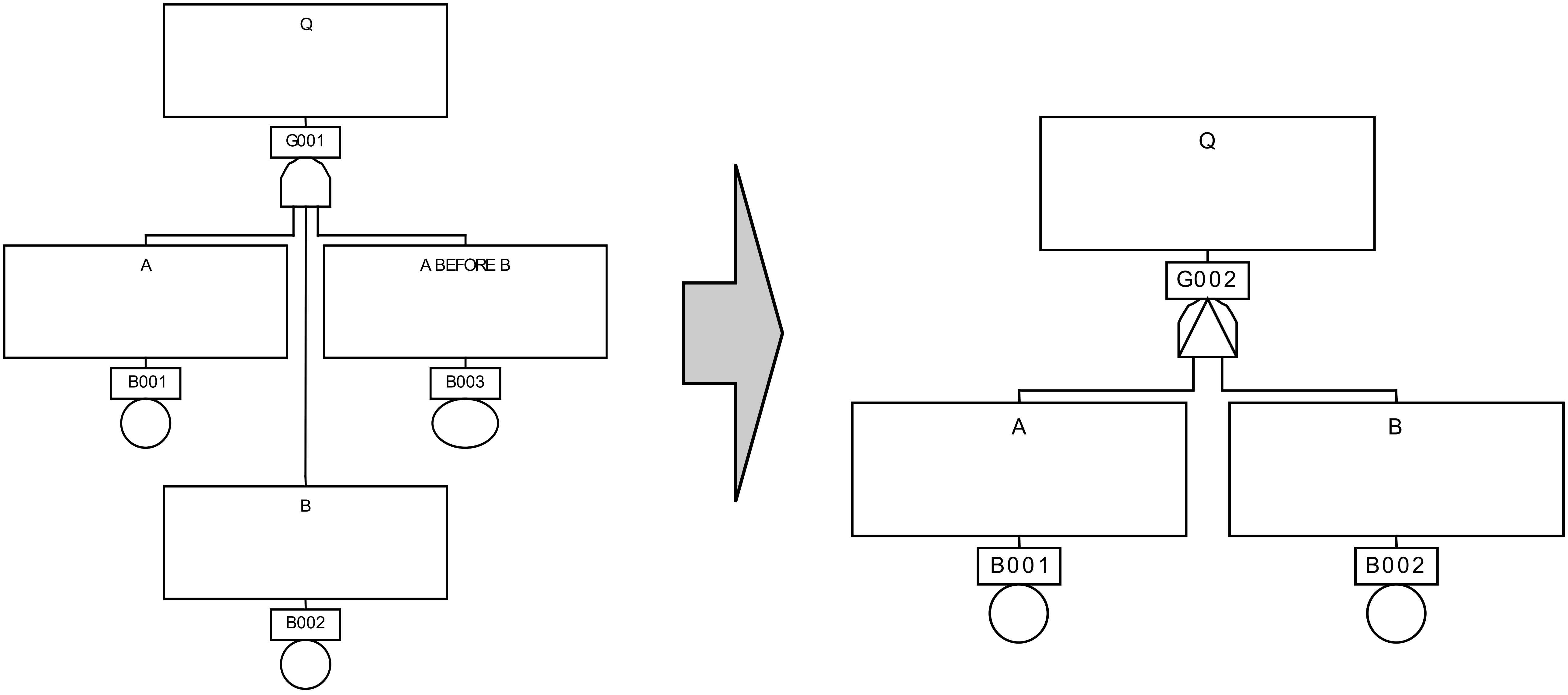
Examples: INHIBIT Gate / Conditioning Event





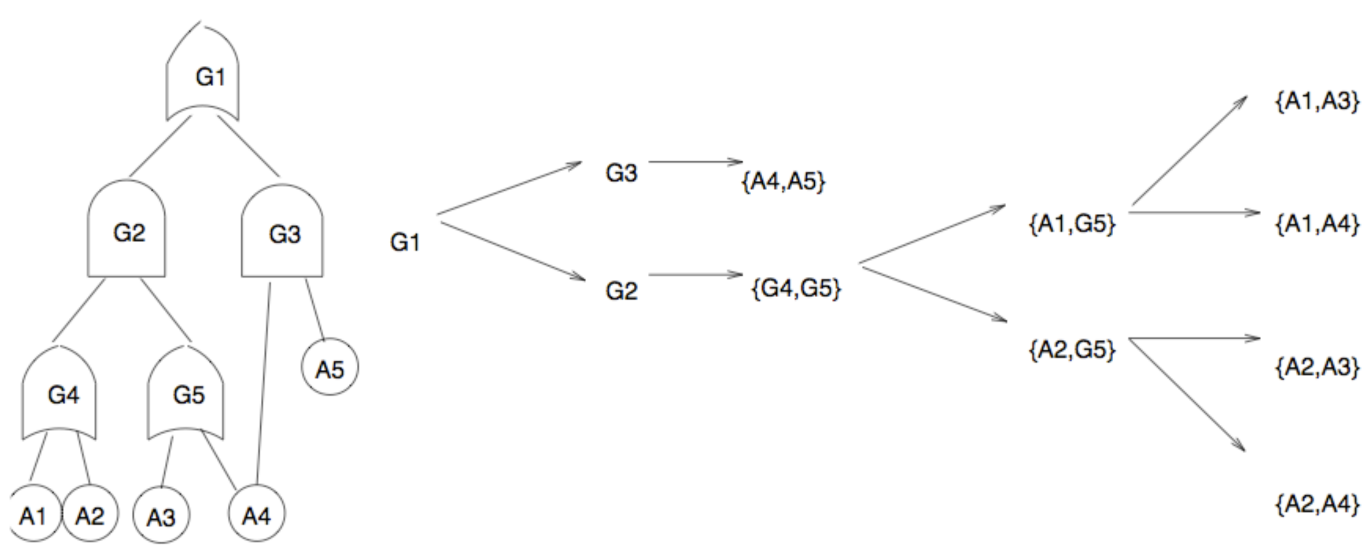


Examples: Priority AND Gate



Cut Sets

- **Cut set:** Any group of basic events which, if all occur at the same time, cause the TOP event
- **Minimal cut set (mincut)**: Minimal combination of basic events that induce TOP
  - ‚Minimal‘: All basic events are needed to let the TOP event occur
  - A long *mincut* shows low vulnerability, a short *mincut* shows high vulnerability
  - Analysis of events by lead to *Common Cause Susceptibilities* identification (e.g. temperature)
  - A **singleton cut set** shows a *single point of failure*
- **Path set:** Set of events whose nonoccurence ensures that TOP does not occur



(C) Joanne Bechta Dugan

FTA Cutsets

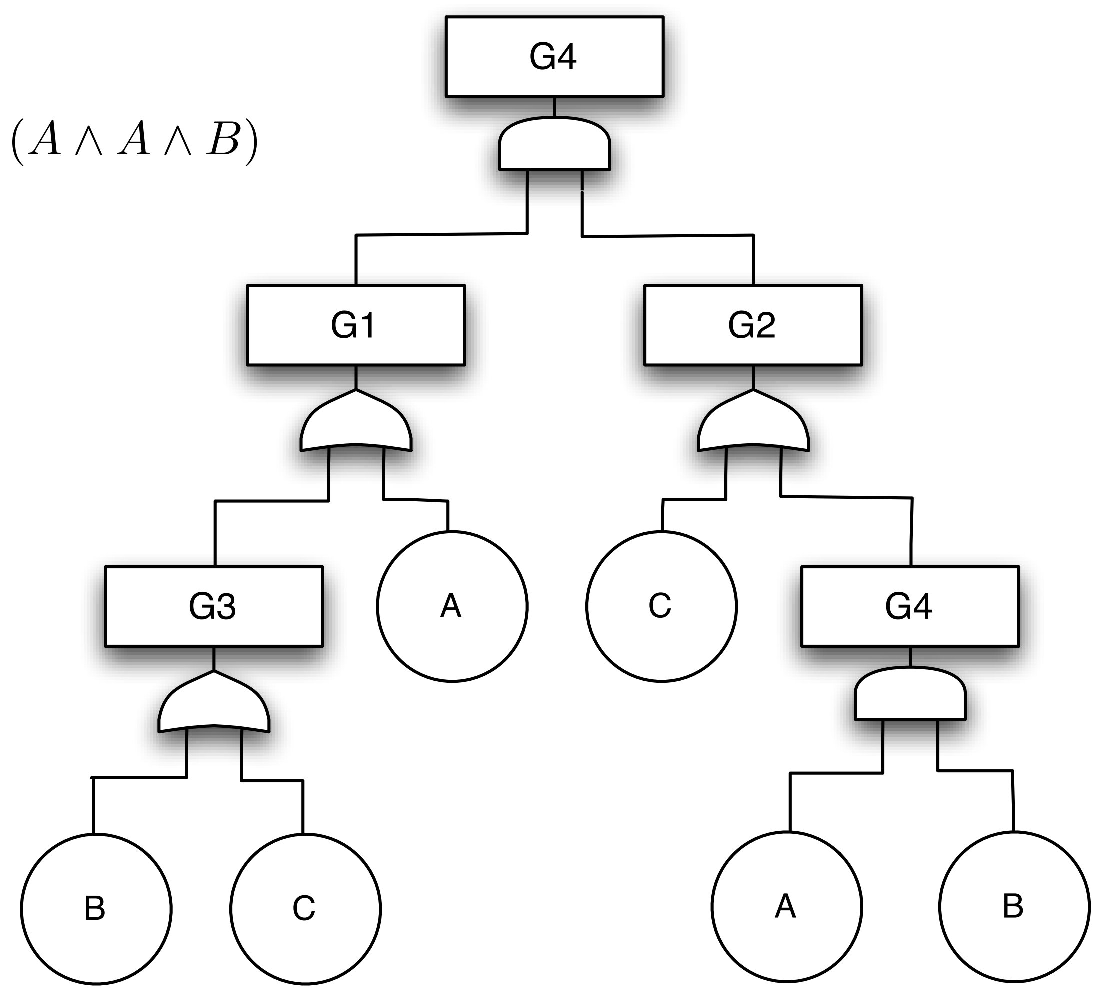
- Determine probabilities for cut sets to find **critical path**
  - Critical and weak links in a system design
- Analyze cut set for
  - Unexpected root cause combinations
  - Weak points in the design
  - Bypass of intended safety features
  - Common cause problems
- Methods for cut set finding
  - Boolean reduction, bottom-up reduction, top-down reduction, mapping to binary decision diagram, Shannon decomposition, genetic algorithms, ...

Boolean Reduction Example

$$(A \vee B) \wedge (C \vee D) = (A \wedge C) \vee (A \wedge D) \vee (B \wedge C) \vee (B \wedge D)$$

$$A \vee A = A \qquad A \wedge A = A \qquad A \vee (A \wedge B) = A$$

$$\begin{aligned} TOP &= (B \vee C \vee A) \wedge (C \vee A \wedge B) \\ &= (B \wedge C) \vee (B \wedge A \wedge B) \vee (C \wedge C) \vee (C \wedge A \wedge B) \vee (A \wedge C) \vee (A \wedge A \wedge B) \\ &= (B \wedge C) \vee (A \wedge B) \vee C \vee (C \wedge A \wedge B) \vee (A \wedge C) \vee (A \wedge B) \\ &= (B \wedge C) \vee (A \wedge B) \vee C \vee (C \wedge A \wedge B) \vee (A \wedge C) \\ &= A \wedge B \vee C \end{aligned}$$



-> 2 resulting minimal cut sets
(== all cut sets ?)

Example by Dr. John Andrews / Loughborough University

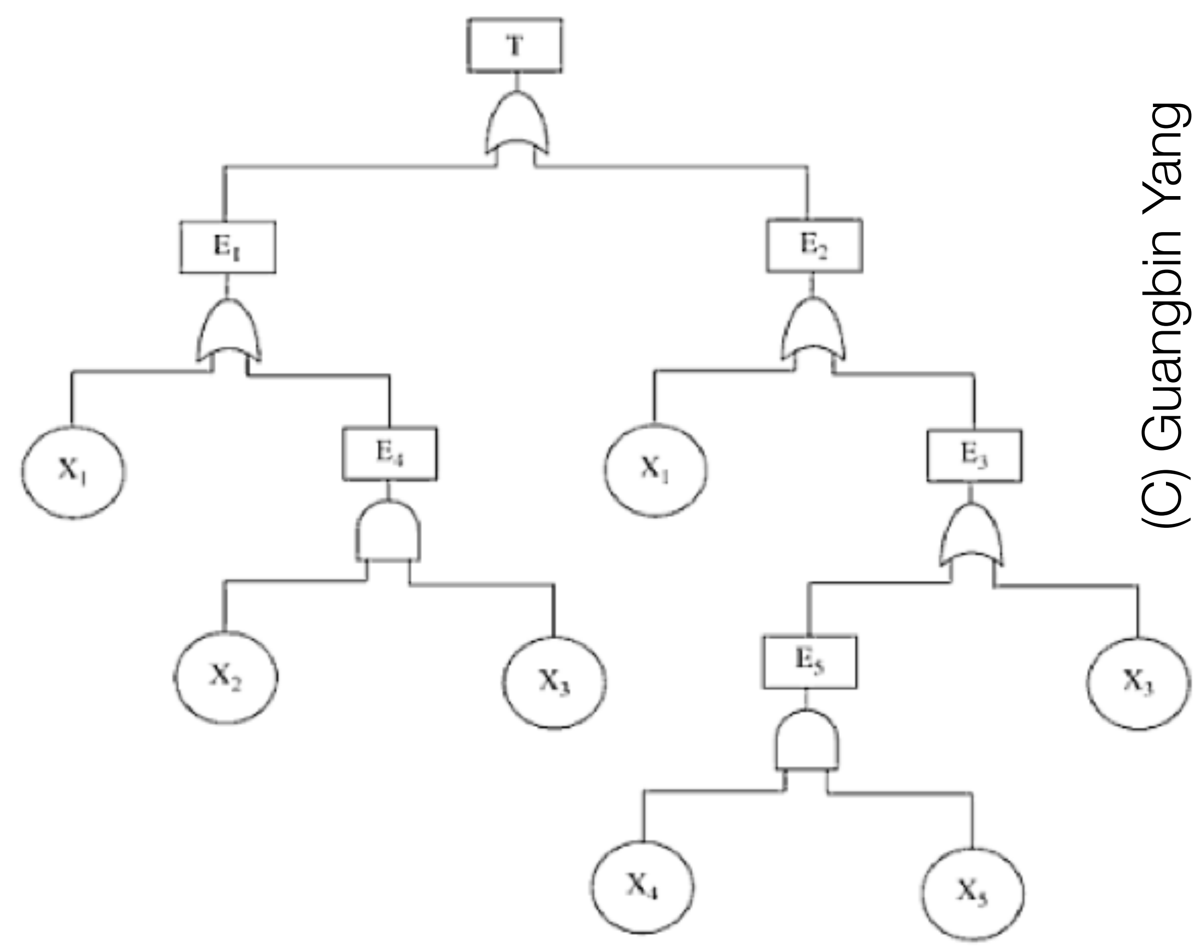
Quantitative Analysis of Fault Trees

$$TOP = X_1 \lor X_3 \lor X_4 \land X_5$$

$$\begin{aligned} P(A \cup B \cup C) = {} & P(A) + P(B) + P(C) \\ & - P(A \cap B) - P(A \cap C) \\ & - P(B \cap C) + P(A \cap B \cap C) \end{aligned}$$

$$Pr(TOP) = Pr(X_1) + Pr(X_3) + Pr(X_4 * X_5) - Pr(X_1 * X_3) - Pr(X_1 * X_4 * X_5) - Pr(X_3 * X_4 * X_5) + Pr(X_1 * X_3 * X_4 * X_5)$$

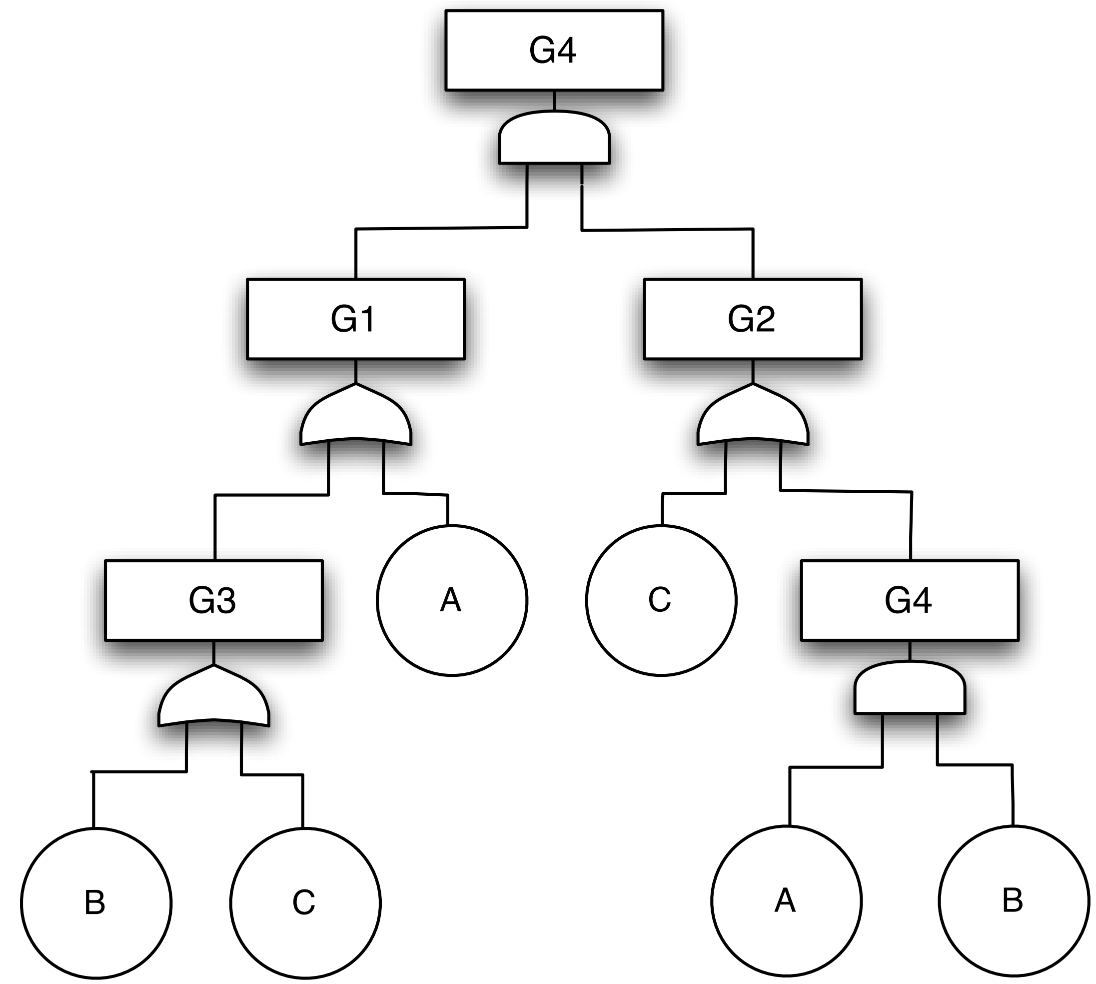
- Determine probability of TOP event by
  - Assuming independence of basic events
  - Utilize probability of independent basic events to compute probability of TOP event



(C) Guangbin Yang

Method for Obtaining Cut Sets (MOCUS) [Rausand]

- Start at the TOP event
  - OR gate: Each input to the gate is written in separate rows
  - AND gate: Each input to the gate is written in separate columns
  - Iteratively replace gates in rows and columns
- Each resulting row forms a cut set



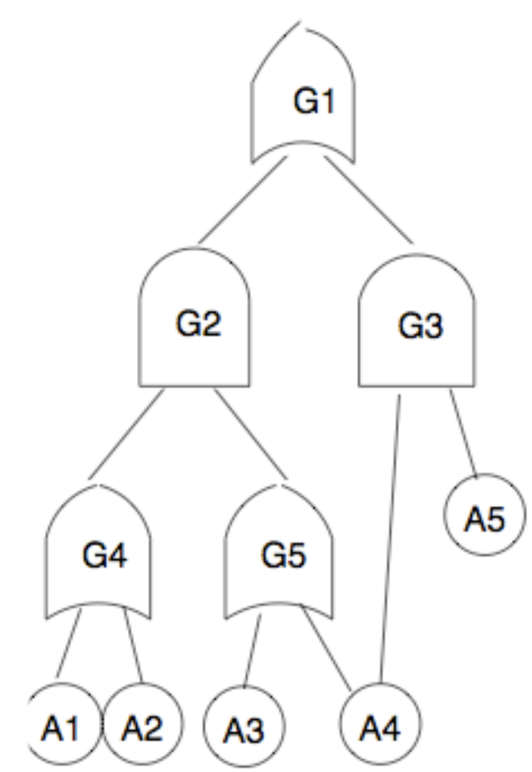
G1,G2	G3, G2	G3, C	B, C	B, C	B, C	B, C	B, C	B, C
	A, G2	G3, G4	C, C	C, C	C, C	C, C	C, C	**C**
		A, G2	B, G4	B, A, B	B, A, B	B, A, B	B, A, B	**A, B**
			C, G4	C, A, B	C, A, B	C, A, B	C, A, B	C, A, B
			A, G2	A, G2	A, G2	A, C	A, C	A, C
						A, G4	A, A, B	

Quantitative Analysis of Cut Sets

- Set of minimal cut sets -> „minimal failure set"
- **Set of minimal cut sets** can also be determined **for any intermediate event**
- Can help with quantitative analysis
  - Finding the **dominant minimal cut set**: Calculate the probability of each minimal cut set, sort by probability
  - Identification of **importance** of cut sets or single events
    - Importance $E_i(t)$ of minimal cut set i at time t
      - Determine cut set unavailability $Q_i(t)$ -> multiply probabilities of events
      - Determine system unavailability $Q_S(t)$ -> $\mathbf{E_i(t) = Q_i(t) / Q_S(t)}$
    - Importance $e_k(t)$ of component k at time t
      - Sum up all $Q_i(t)$ for cut sets that contain k -> $\mathbf{e_k(t) = Q_{k\_1}(t) + Q_{k\_2}(t) + \ldots / Q_S(t)}$

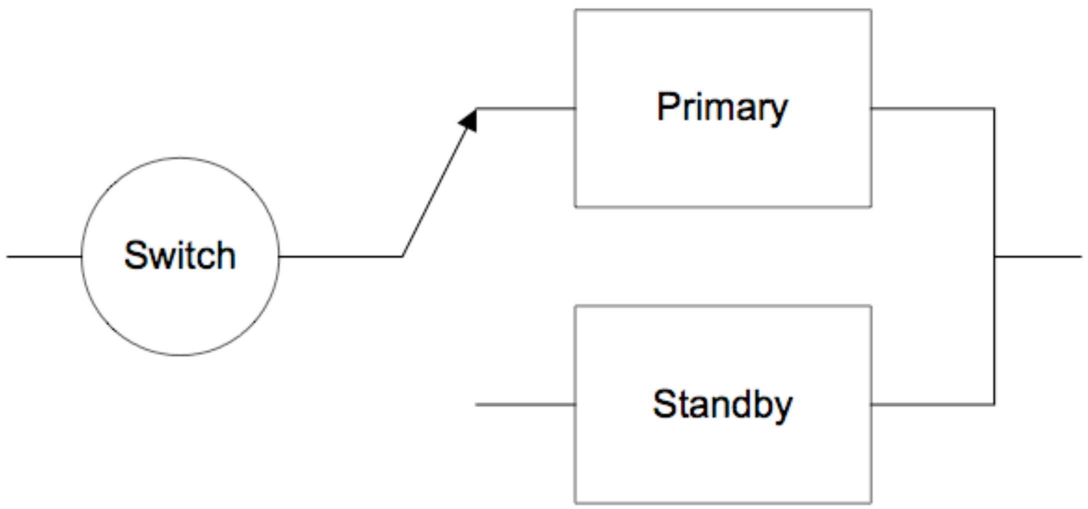
Fixing Cut Sets

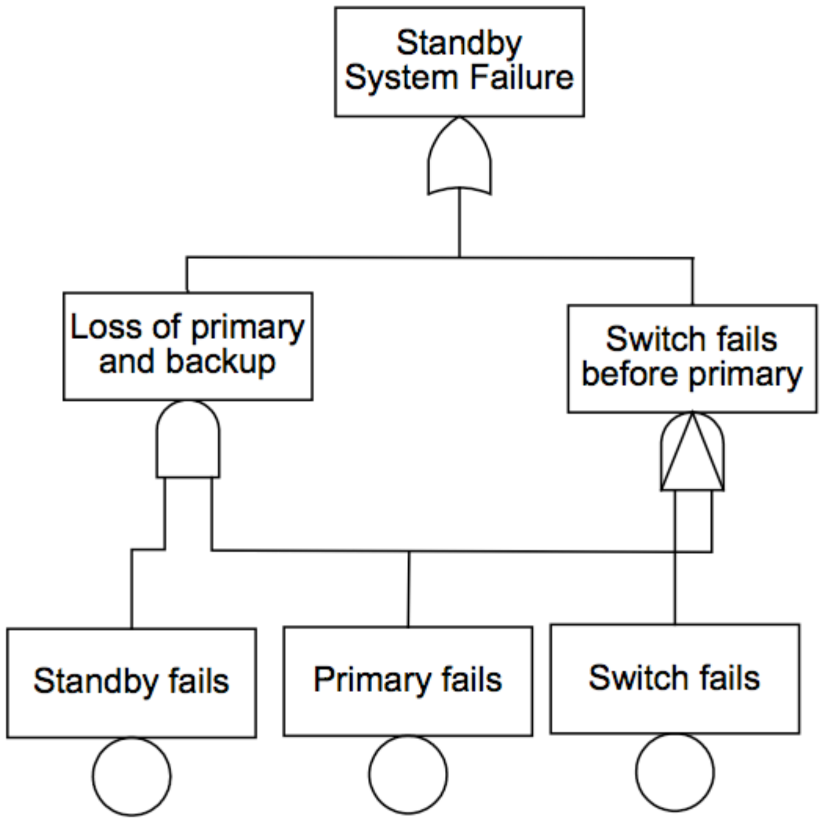
- AND gates can be protected by disallowing one of the inputs
  - Exhaustive testing or formal proof to show that the component cannot fail
  - Test for failure condition and recovery routine
- OR gates can be protected by disallowing all inputs or by providing error recovery
- Example
  - Protect G3 by preventing failure of A4
  - Protect G2 by
    - preventing failure of A3
    - preventing failure of both A1 and A2
    - providing fault tolerance for G4



Dynamic Fault Trees (DFT)

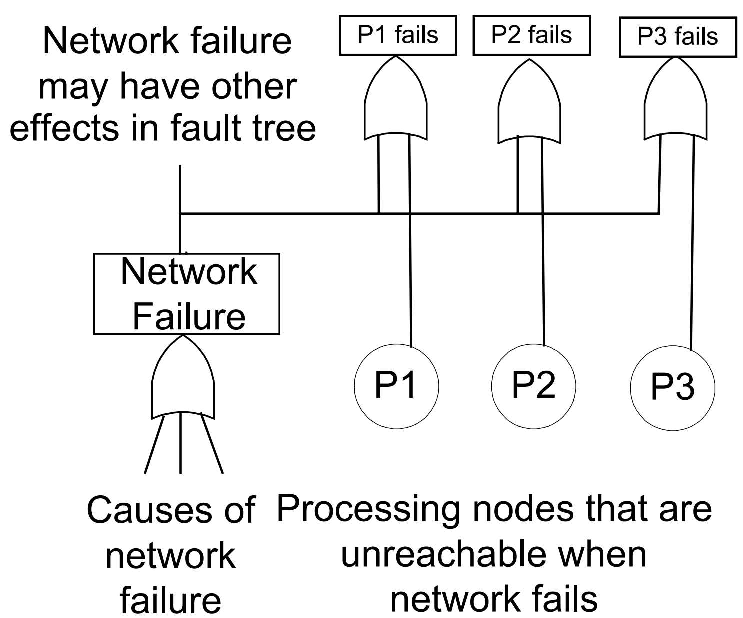
- Failure criteria of a system might depend not only on logical combination of basic events in the same time frame
  -> sequence-dependent failure
- Dynamic fault tree gates support sequences and dynamic probability changes
- Dynamic sub parts of the fault tree are typically analyzed by Markov model
- Example
  - Failure of switch only relevant if it happens before outage of primary
  - What is the probability of „switch fails before primary“ ?

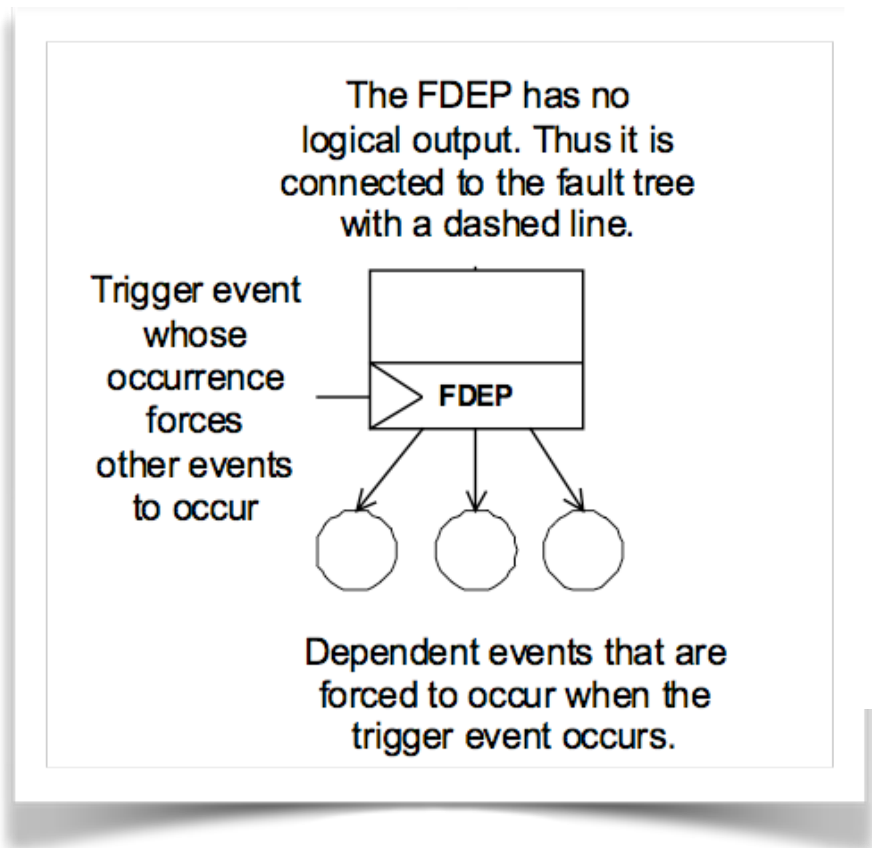


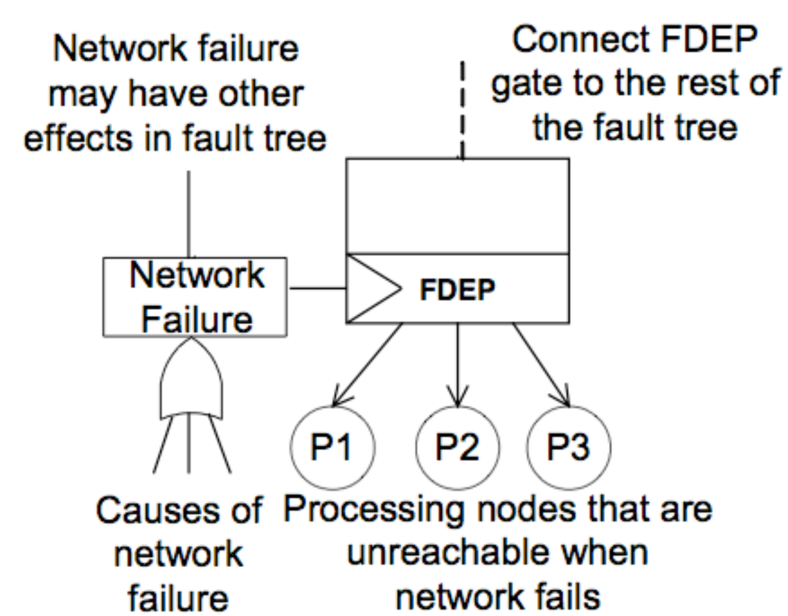


Dynamic Fault Trees

- **Functional dependency (FDEP) gate**
  - Single trigger input event, *forces* dependent events to occur on activation
  - No logical gate output - connected through a dashed line
  - Separate occurrence of the dependent events has no effect on trigger event

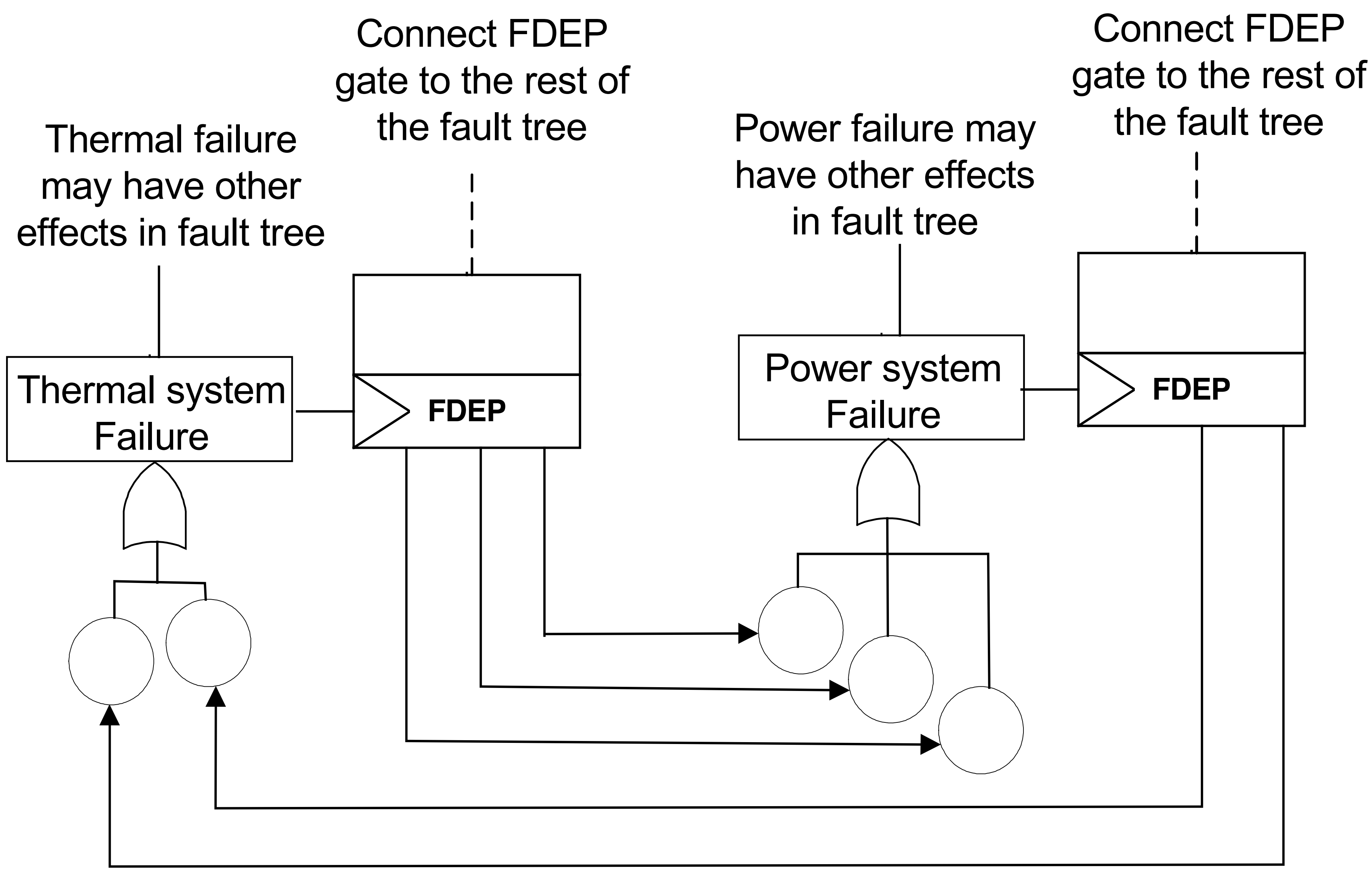






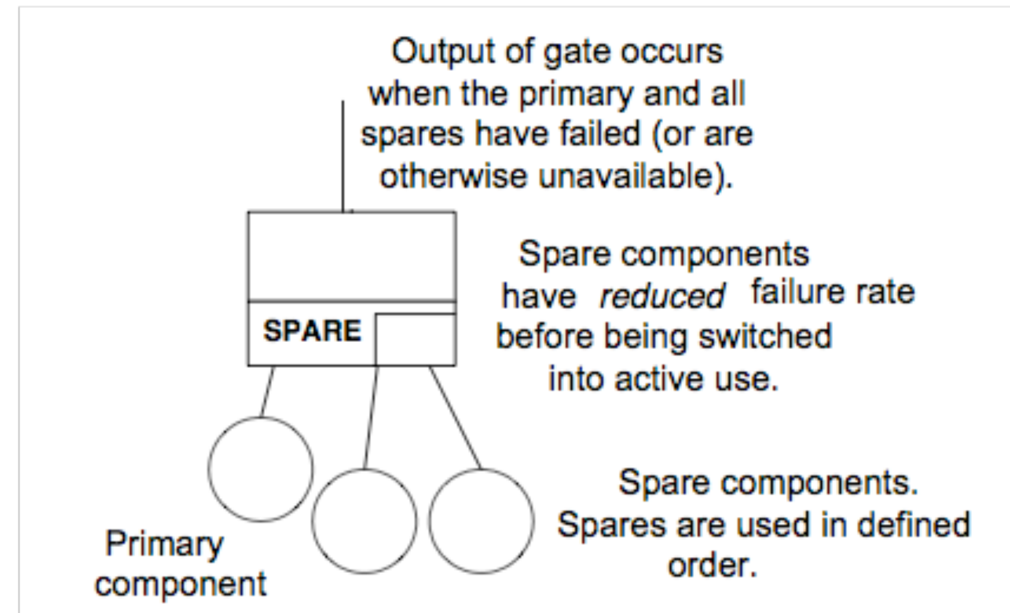
[Vesseley]

FDEP for Interdependency Modeling

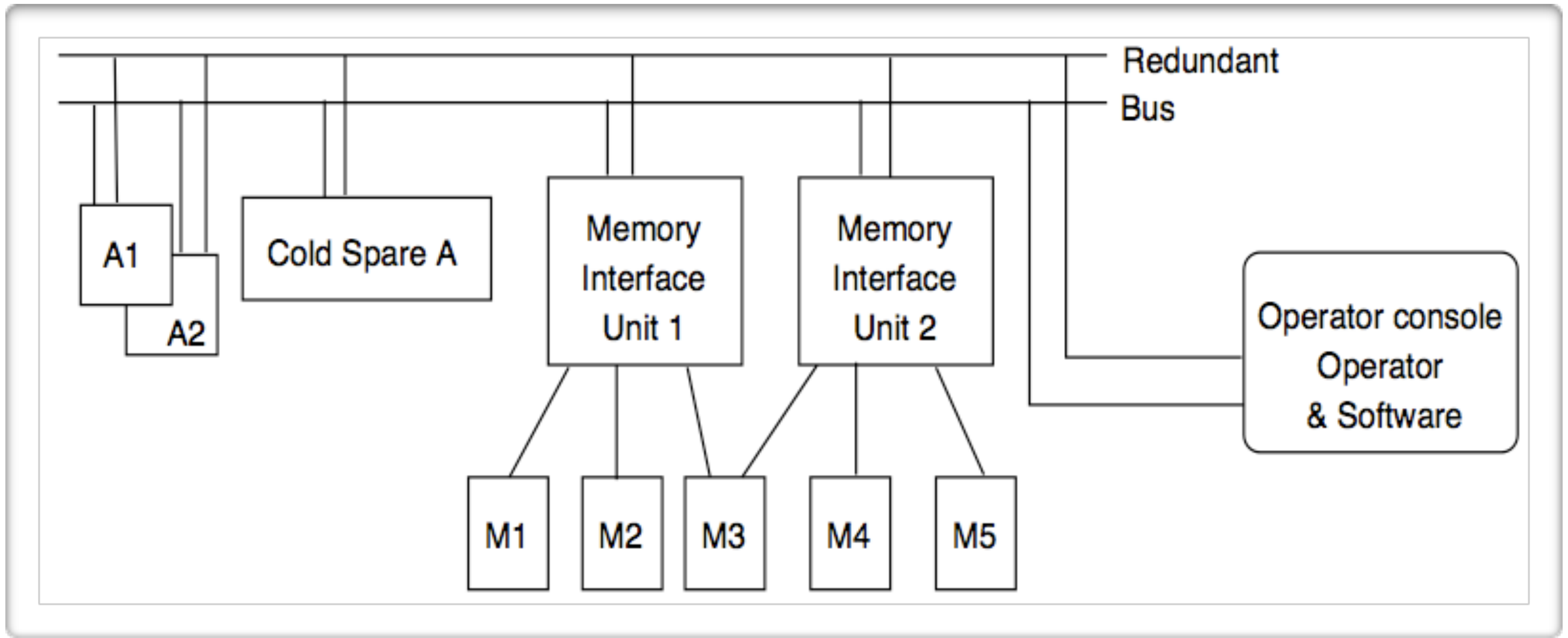


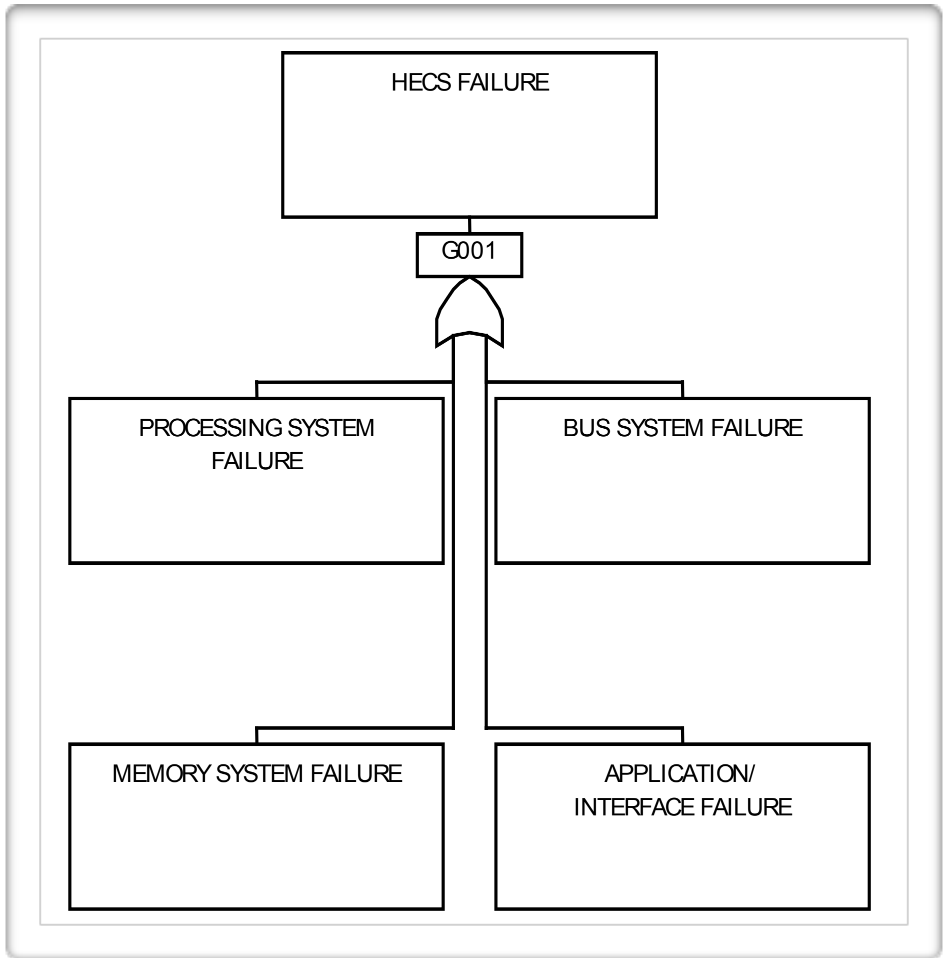
Dynamic Fault Trees

- **Cold Spare (CSP) Gate**
  - One primary basic input event,
    one or more *ordered* cold spare input events
  - Alternate inputs are initially unpowered,
    serve as replacement for primary
  - Output occurs if all the input events occurred
    - Primary and all spares fail
  - Support modeling of *cold spares* (zero failure rate when unpowered),
    warm spares (reduced failure rate when unpowered) or *hot spares*
  - *Dormancy factor* multiplies the failure rate when the unit is in spare
    - Defines decrease of failure probability without primary event



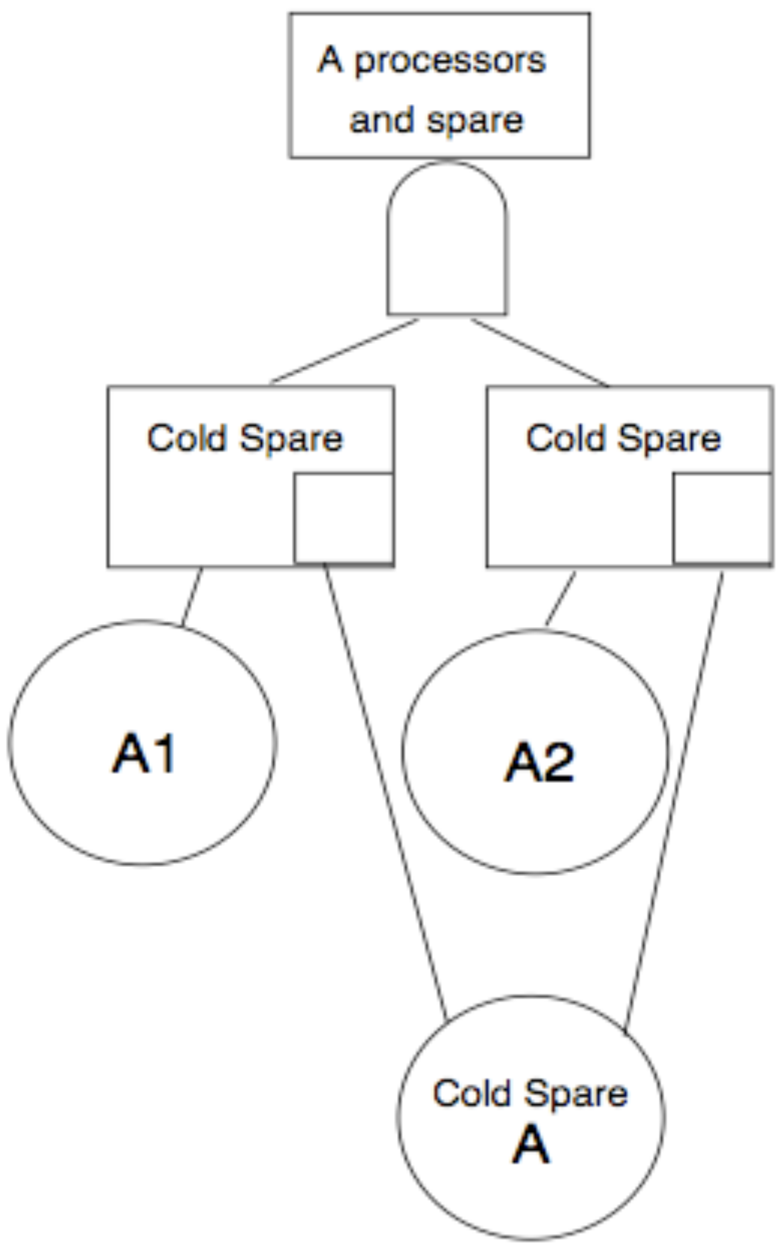
Hypothetic Example Computer System (HECS)

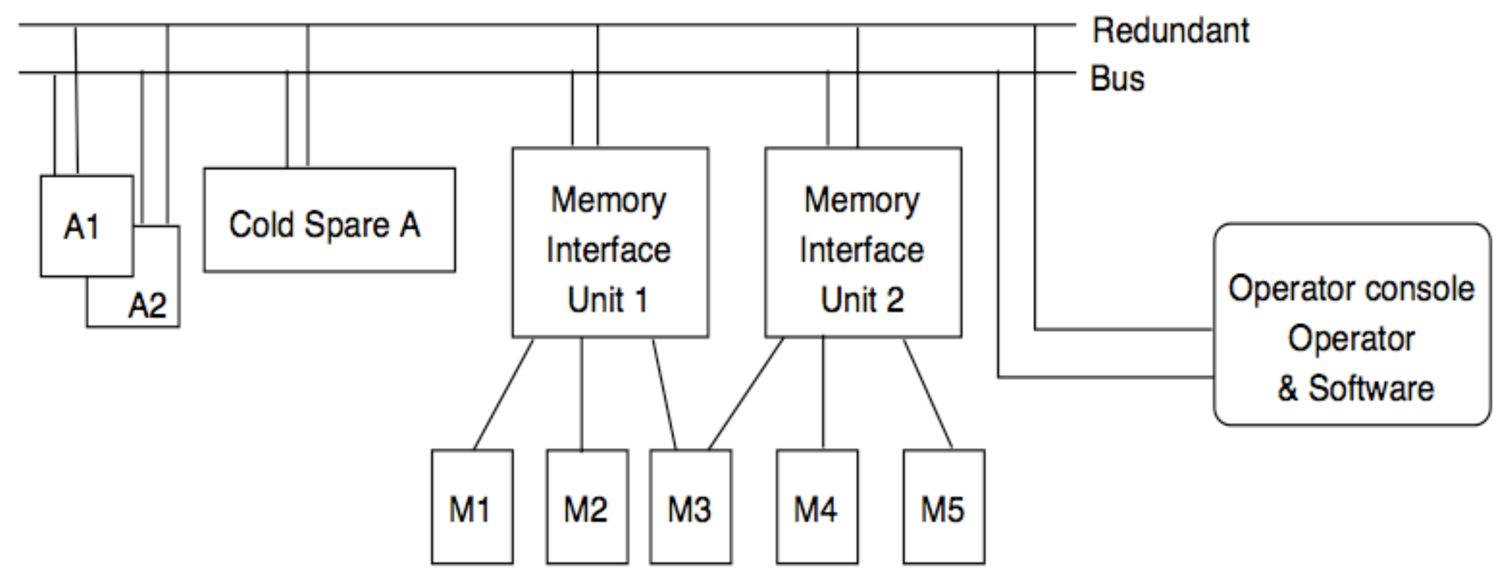




- Minimum demands for operation
  - One functional processor from redundant pair + cold spare
  - Three memory modules connected by at least one memory interface unit
  - One bus
  - Operator + console + software

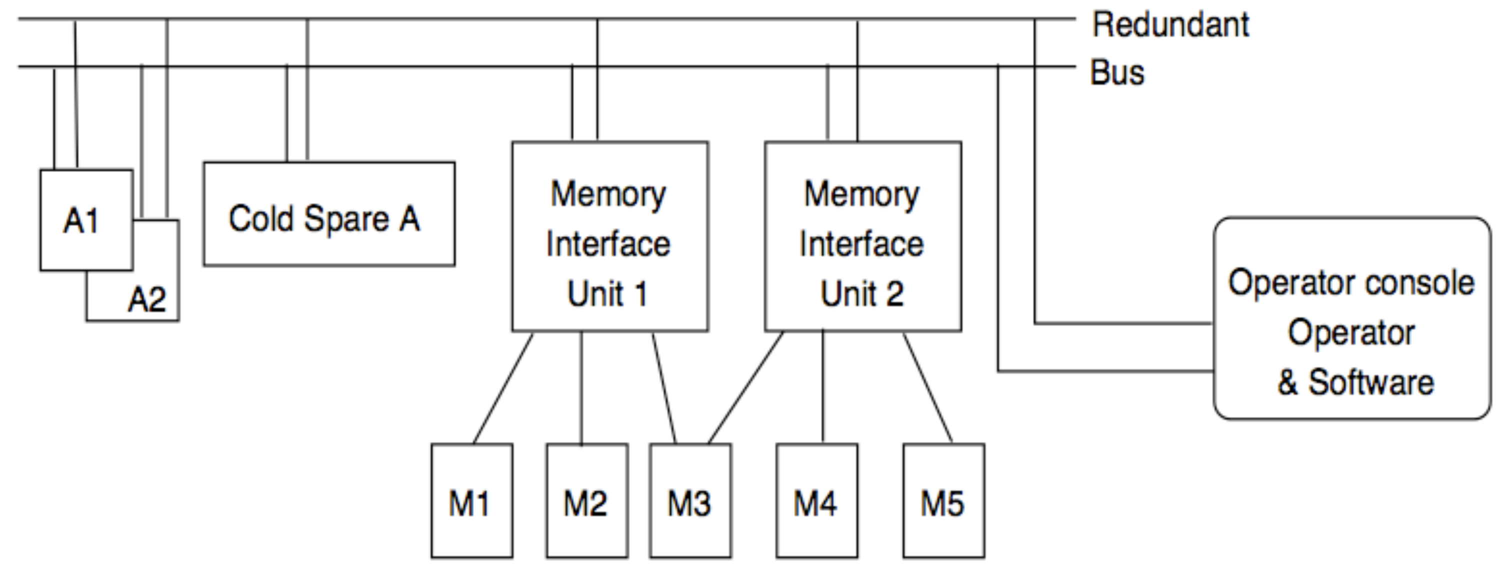
HECS Example

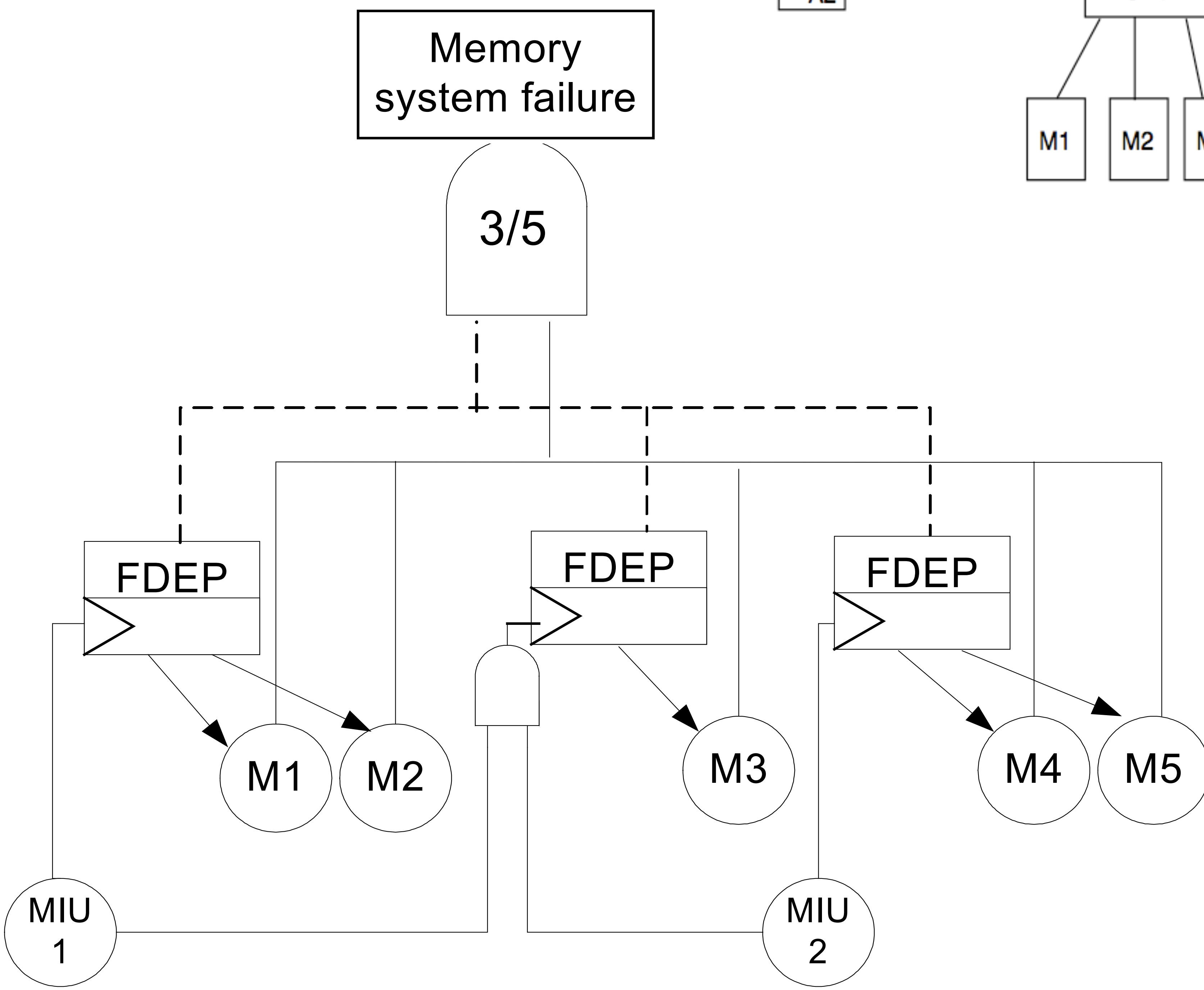




- Failure rate of active processor is different from cold spare failure rate when not activated
  - Cold spare - dormancy factor of 0

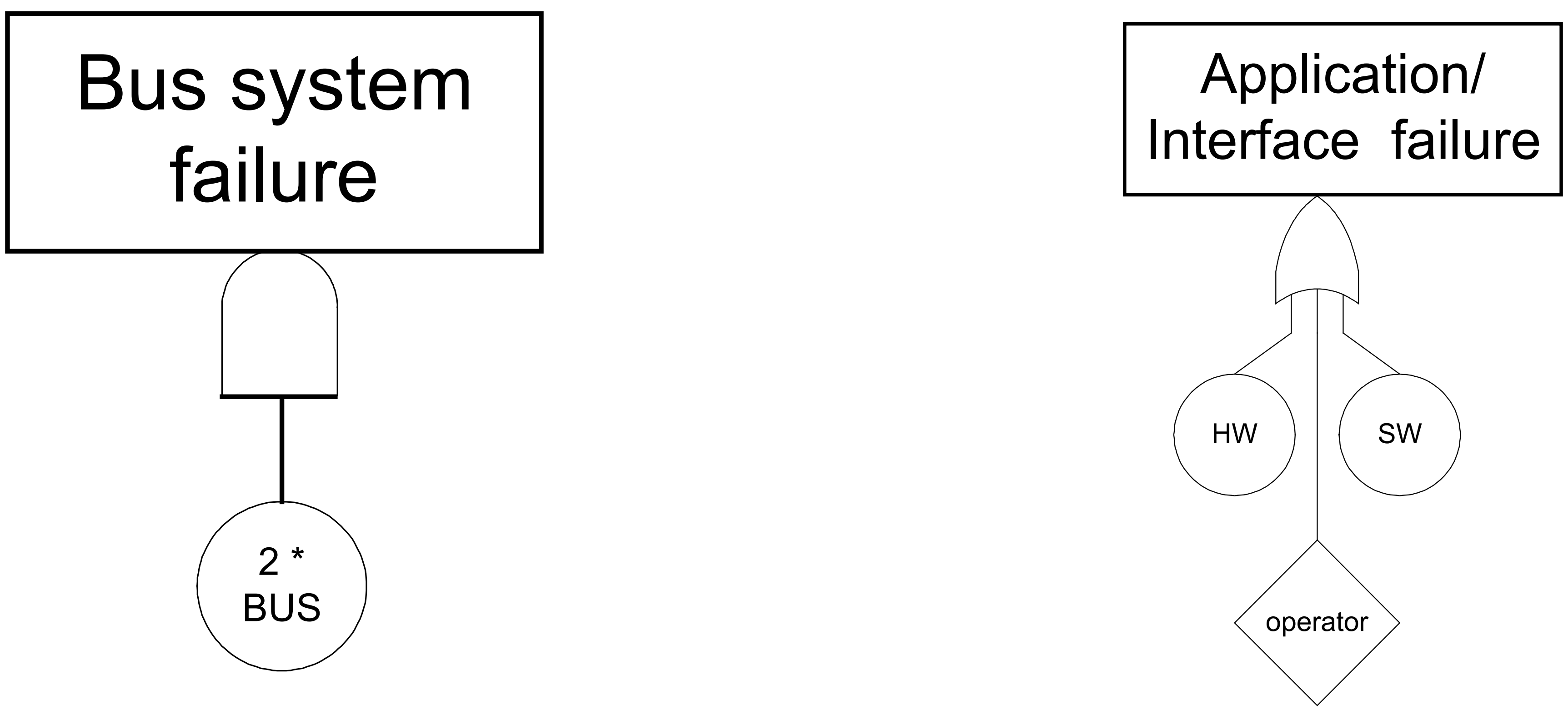
HECS Example



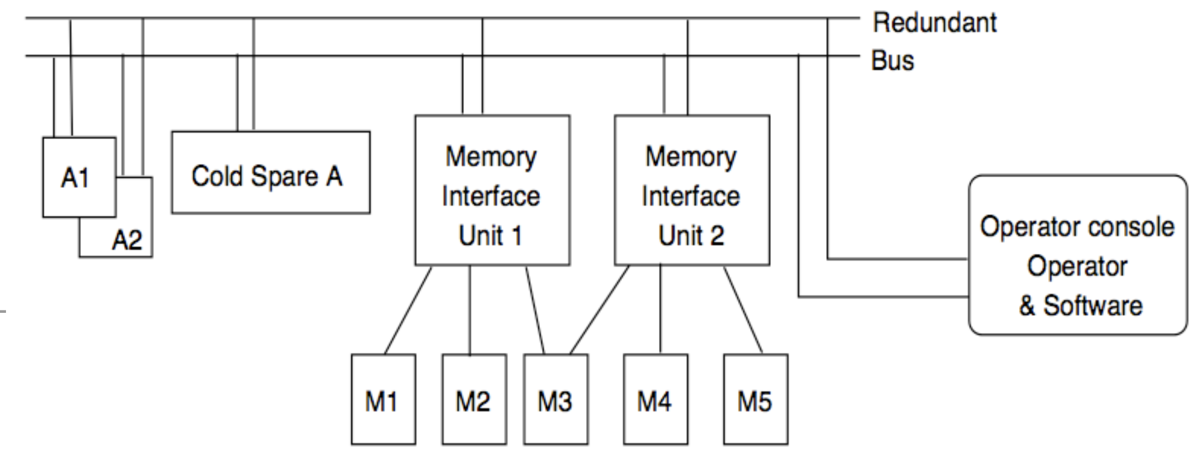


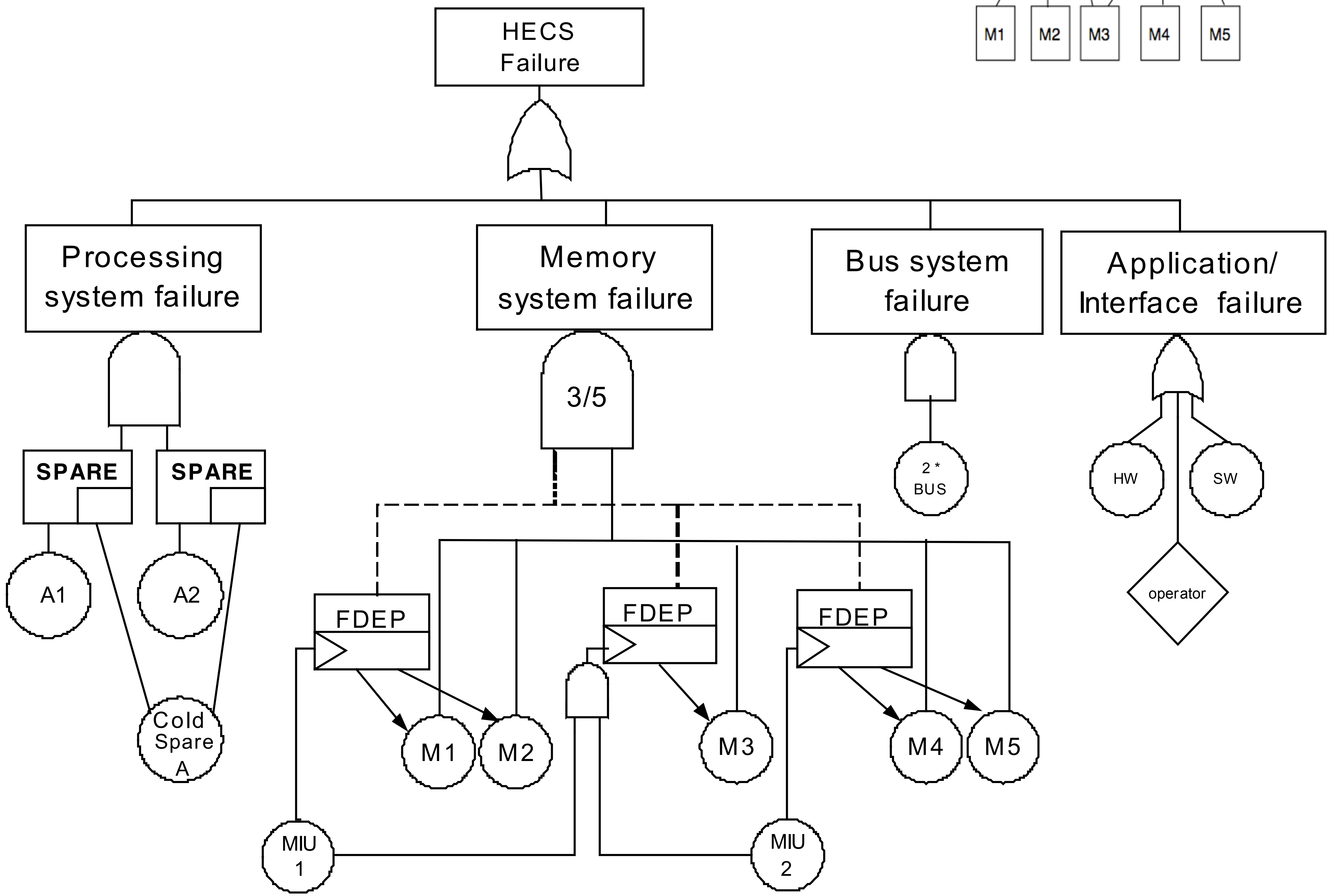
- Dashed line does not count for k/N gate

HECS Example

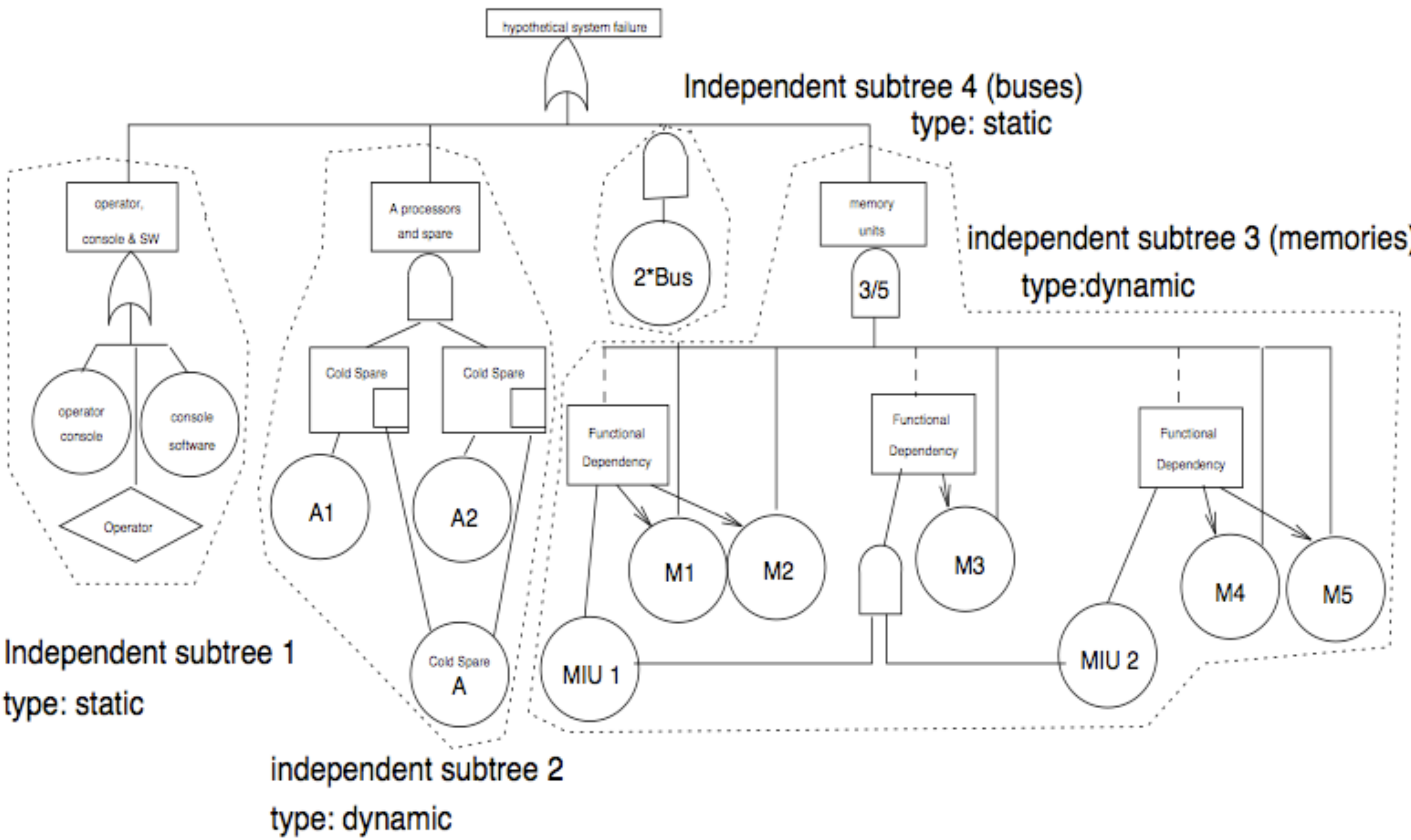


HECS Example



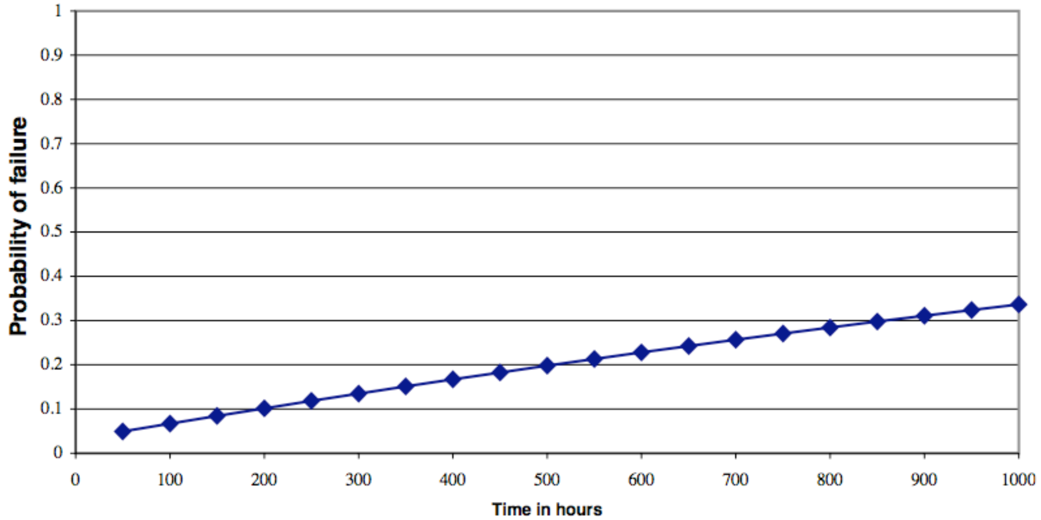


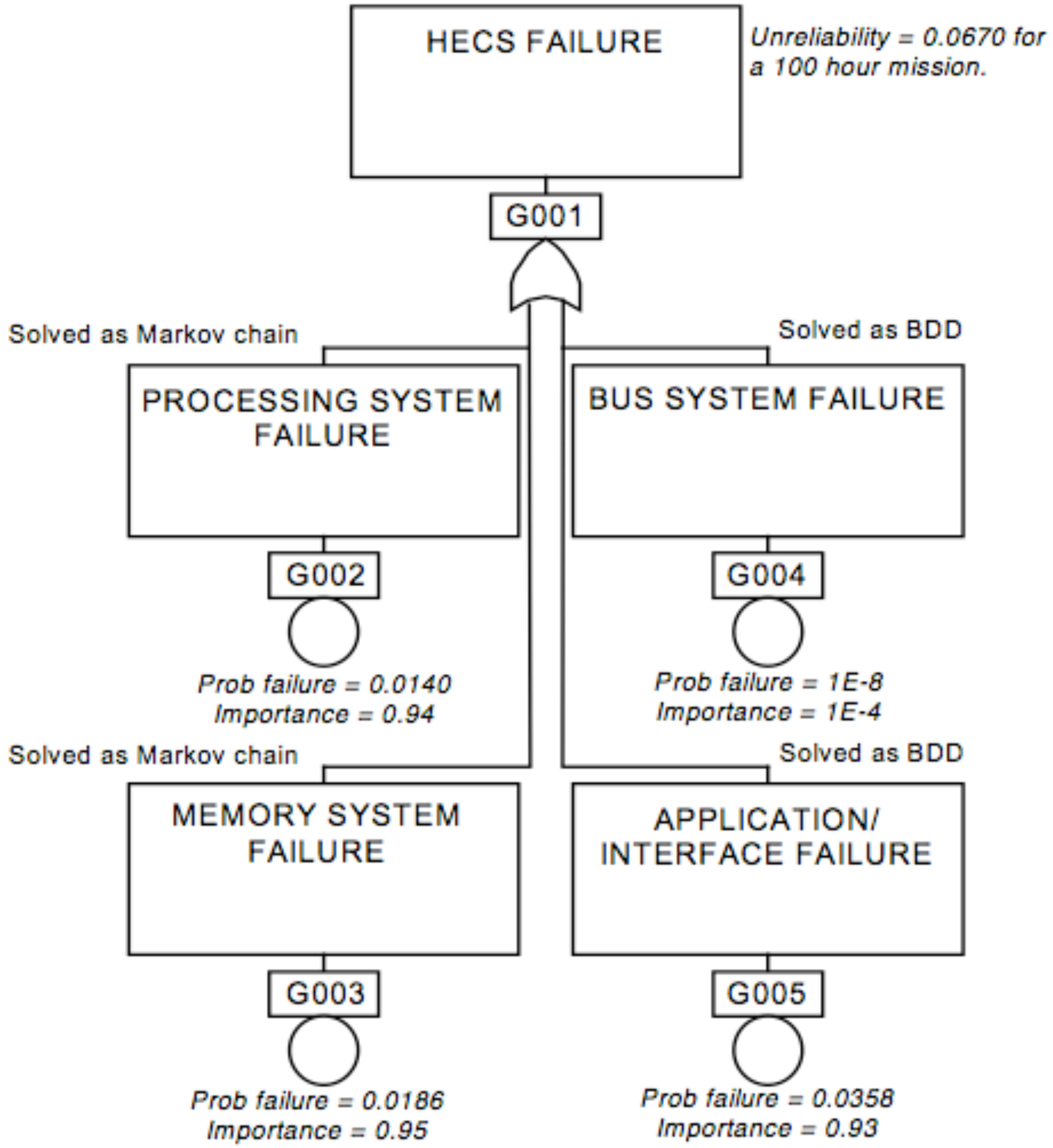
HECS Example



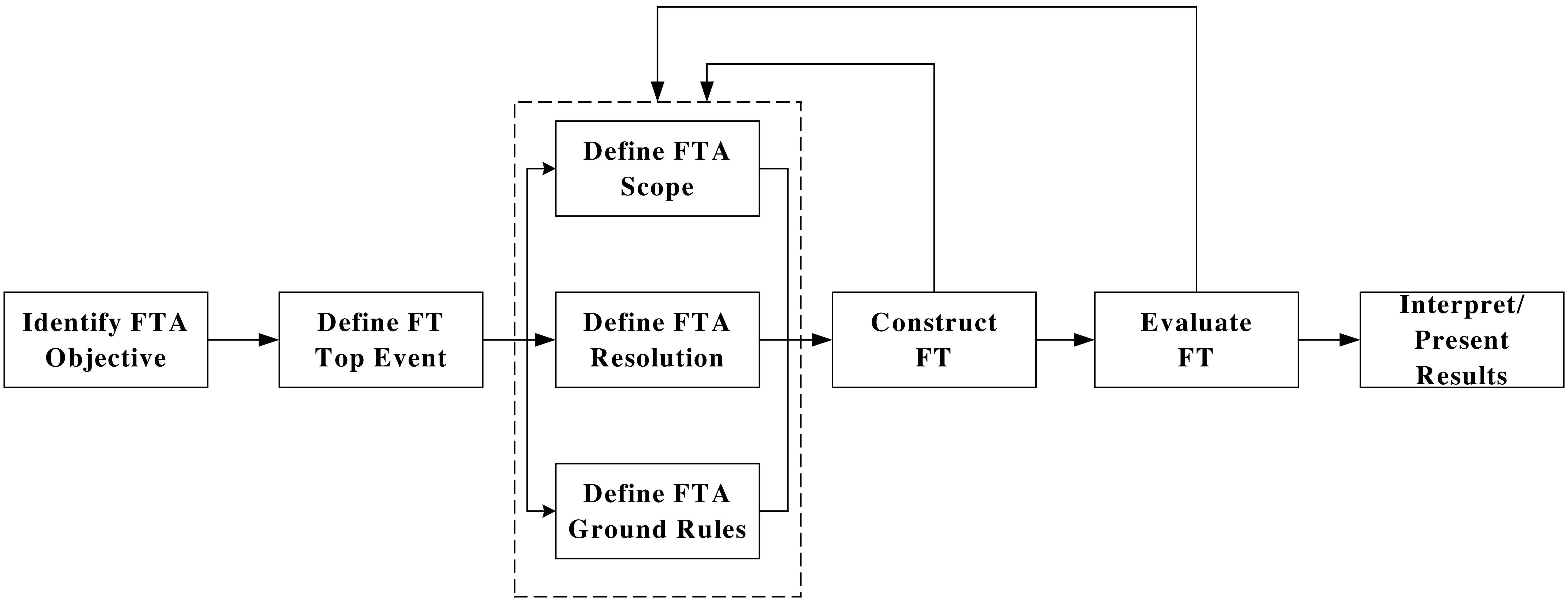
HECS Example

- Analysis with Galileo/ASSAP system for an 100-hour mission
- Processing and memory system analyzed by Markov models
- Importance analysis with Birnbaum method
- Basic assumptions for component failure rates





Fault Tree Construction [NASA]



- Objective should be phrased in terms of a system failure to be analyzed
- Define **scope** (design version, components to be included), **resolution** (based on available probability data) and **ground rules** (naming scheme for events and gates)
- Focus on necessary and sufficient immediate events

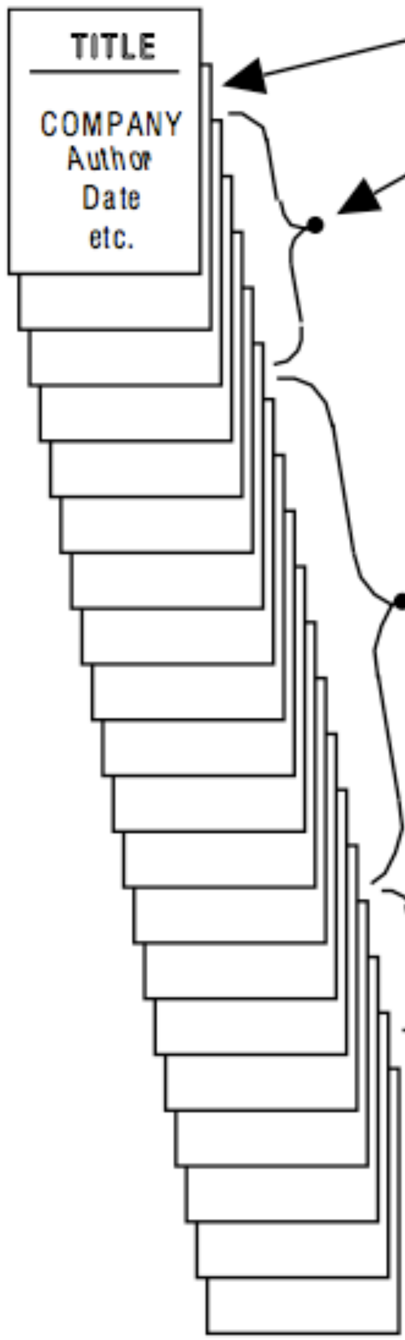
Fault Tree Construction [Misra]

- Step 1: Define the undesired event to be analyzed - what, where, when
- Step 2: Define boundary conditions for the analysis
  - Physical boundaries - What constitutes the system ?
  - Environmental stress boundaries - What is included (earthquake, bombs, ...) ?
  - Level of resolution - How detailed should be the analysis for potential reasons ?
- Step 3: Identify and evaluate fault events
  - Primary failures as basic event, secondary failures as intermediate event
- Step 4: Complete the gates
  - All inputs should be completely defined before further analysis of them
- Complete fault tree level by level

Fault Tree Construction

- Common errors in construction [Misra]
  - *Ambiguous TOP event* - Too general TOP event makes FTA unmanageable, too specific TOP event cannot provide a sufficient system analysis with FTA
  - *Ignoring significant environment conditions* - External stress might be relevant
  - *Inconsistent fault tree event names* - Same name for same fault event or condition
  - *Inappropriate level of resolution* - Detail level of the fault tree should match the detail level of the available information
- Proper and consistent naming is very important (**what** failed and **how**)
- Statistically independent initiators, **immediate** contributors to an event
- Logic can be tested in **success domain** by inverting all statements and gates
- Analyze no further down than is necessary to enter probability data with confidence

FTA Report (Clemens & Sverdrup)



EXECUTIVE SUMMARY (Abstract of complete report)

SCOPE of the analysis…
- Brief system description
- TOP Description/Severity Bounding
- Analysis Boundaries
  - Physical Boundaries
  - Operational Boundaries
  - Operational Phases
  - Human Operator In/Out
  - Interfaces Treated
  - Resolution Limit
  - Exposure Interval
  - Others…

Say what is analyzed and what is not analyzed.

THE ANALYSIS…
- Discussion of Method (Cite Refs.)
- Software Used
- Presentation/Discussion of the Tree
- Source(s) of Probability Data (If quantified)
- Common Cause Search (If done)
- Sensitivity Test(s) (If conducted)
- Cut Sets (Structural and/or Quantitative Importance, if analyzed)
- Path Sets (If analyzed)
- Trade Studies (If done)

Show Tree as Figure. Include Data Sources, Cut Sets, Path Sets, etc. as Tables.

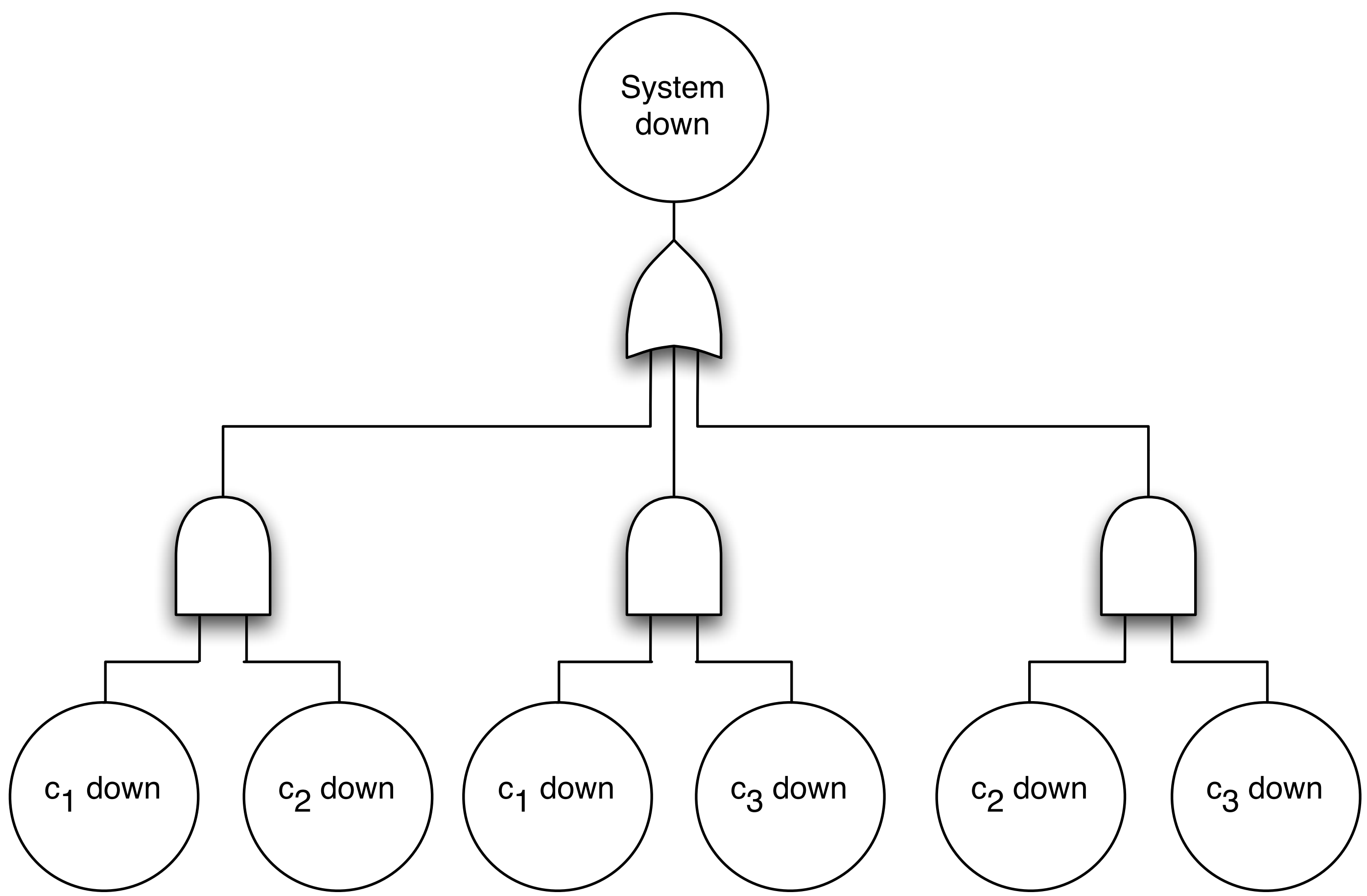
FINDINGS…
- TOP Probability (Give Confidence Limits)
- Comments on System Vulnerability
  - Chief Contributors
  - Candidate Reduction Approaches (If appropriate)

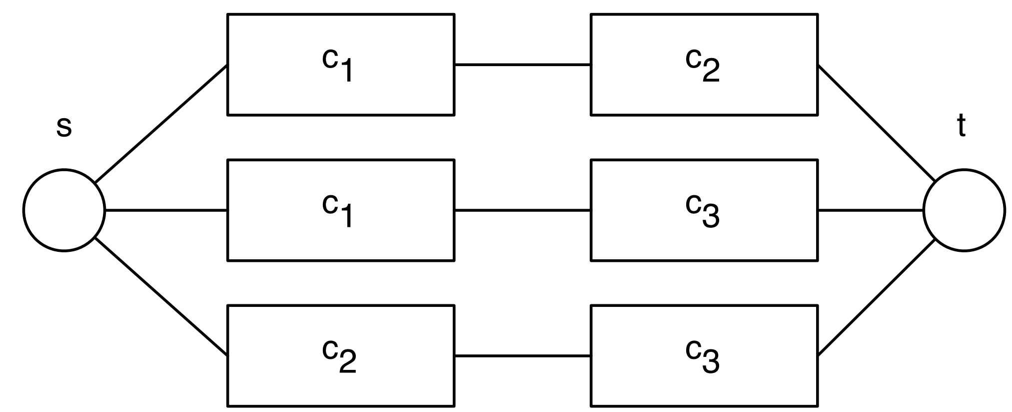
CONCLUSIONS AND RECOMMENDATIONS …
- Risk Comparisons ("Bootstrapping" data, if appropriate)
- Is further analysis needed? …by what method(s)?

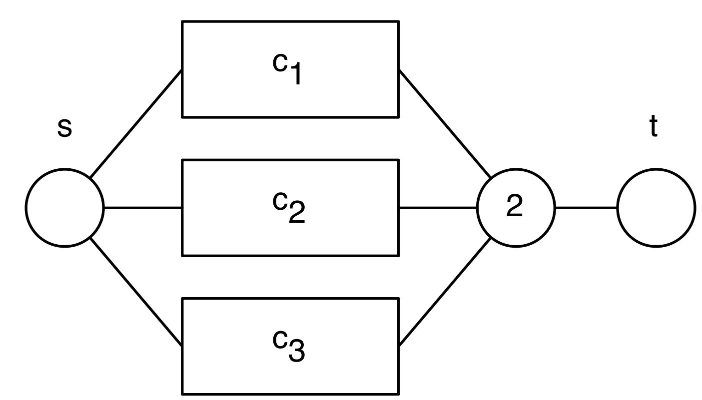
FTA-based Decision Making

- Use FTA to ...
- ... understand the logic leading to the top event, especially in complex systems
- ... prioritize the contributors leading to the top event (typically 10% - 20%)
- ... proactively prevent the TOP event by applying targeted upgrades
- ... monitor the performance of the system by FTA re-evalutation, based on former defects and failures
- ... minimize and optimize resources - identify what is unimportant
- ... assist the system design
- ... diagnose and correct causes of the TOP event

RBD vs. FTA

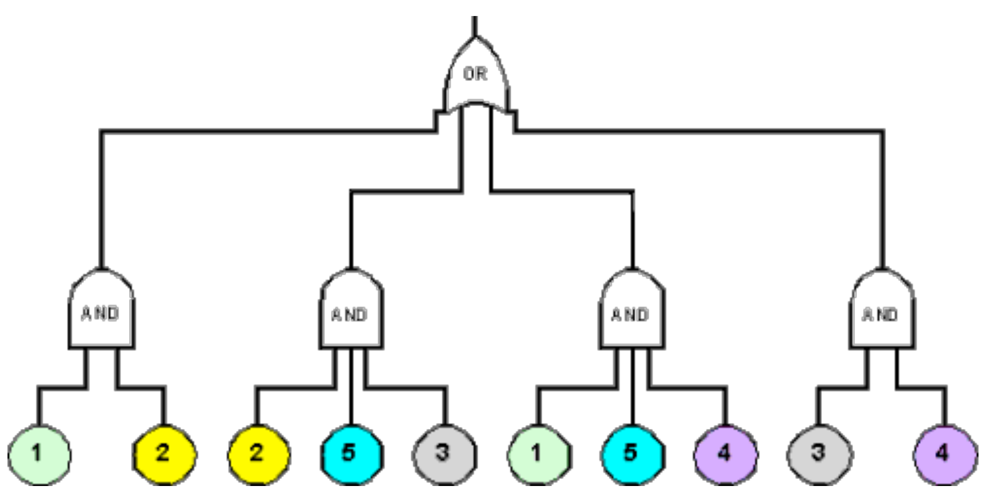

System down
c_1 down
c_2 down
c_1 down
c_3 down
c_2 down
c_3 down

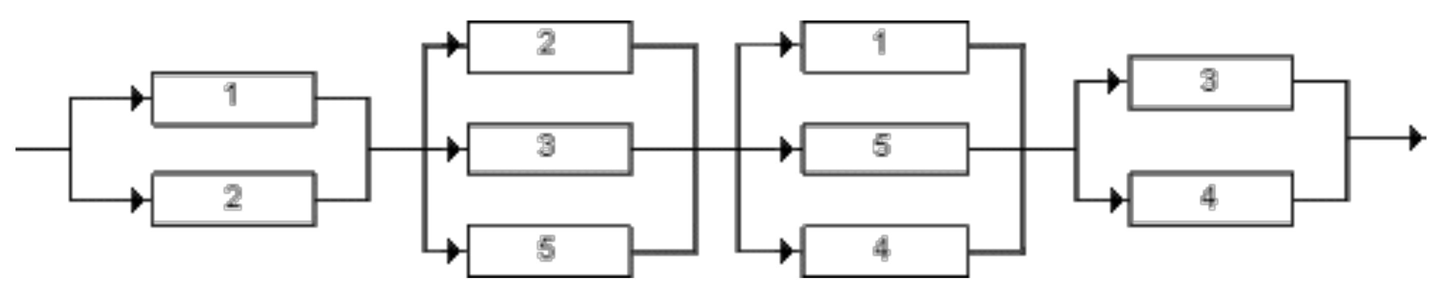

s
t
c_1
c_2
c_1
c_3
c_2
c_3


s
t
c_1
c_2
c_3
2

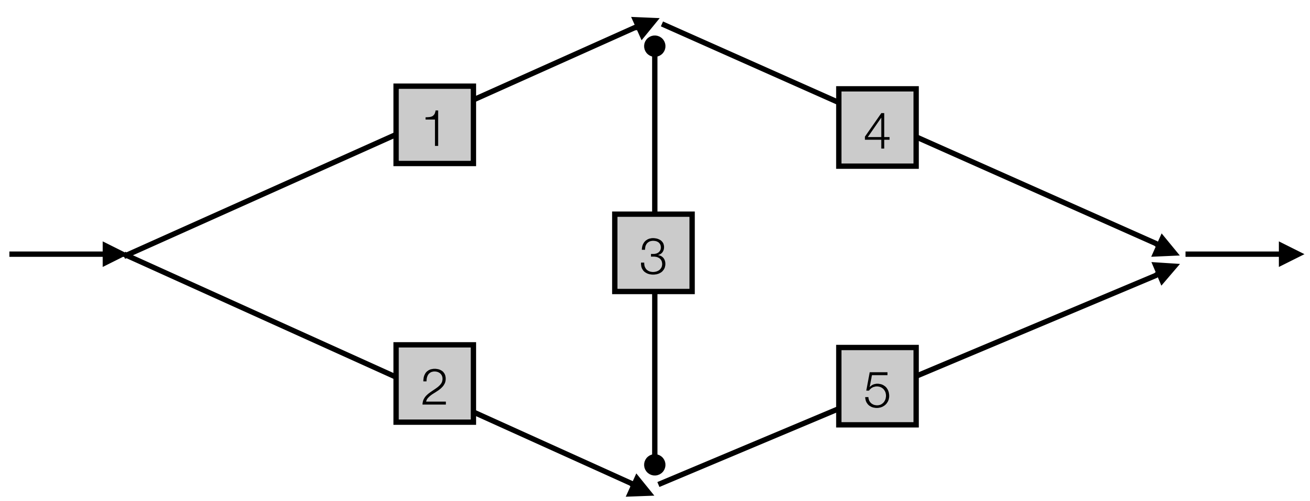
RBD vs. FTA

- Convert fault tree to reliability block diagram
  - Start from TOP event, replace gates successively
  - Logical AND gate <-> parallel structure of the inputs of the gate
  - Logical OR gate <-> serial structure of the inputs of the gate
  - Elements in the fault tree: Failure events, blocks in the RBD: Functioning blocks
- Some FTA and RBD extensions are not convertible
  - Example: Sequence-dependent gates in fault trees

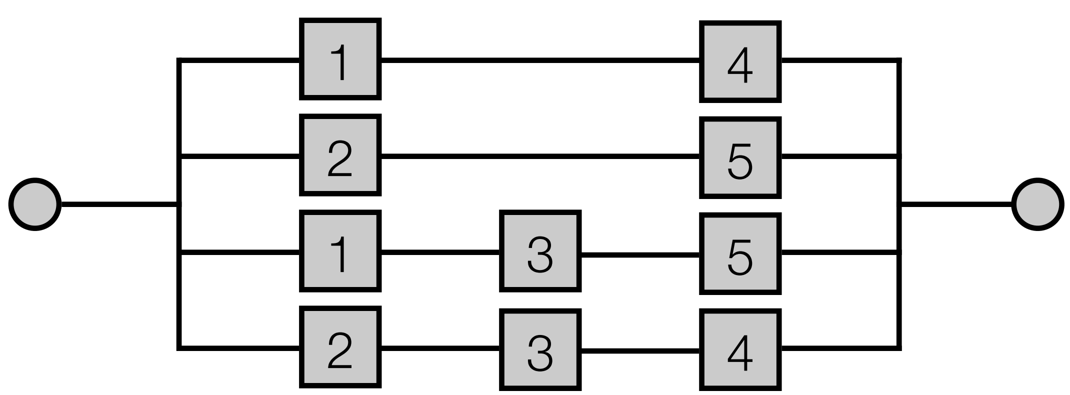




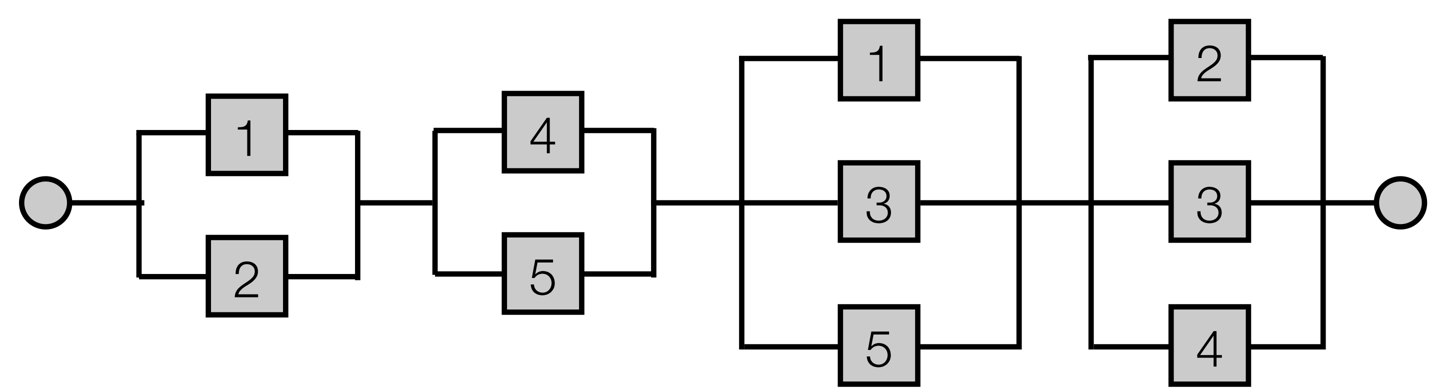
Representing Structures By Paths / Cut Sets [Rausand]



Physical network with a bridge structure



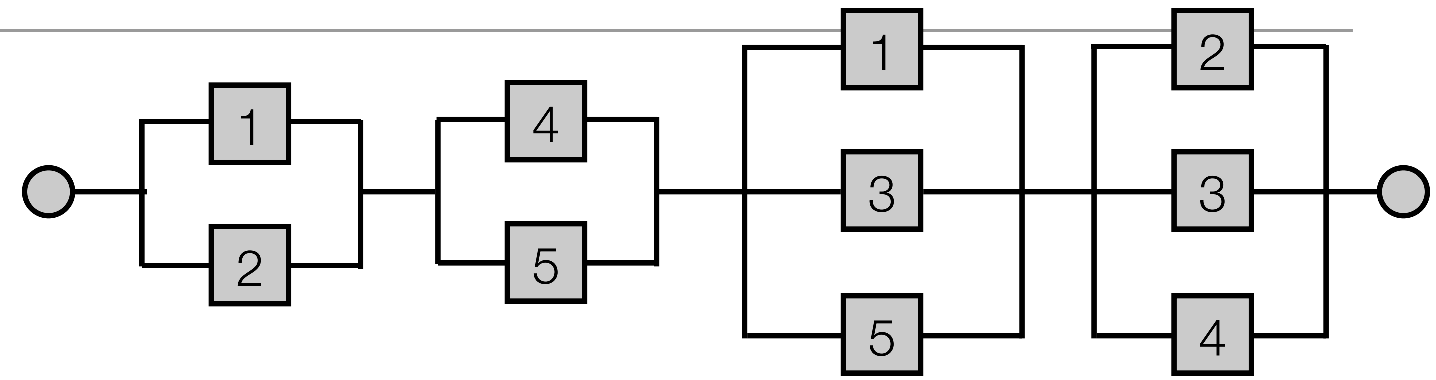
Parallel RBD structure of minimal path series structures



Serial RBD structure of minimal cut parallel structures

Inclusion-Exclusion Principle [Rausand]

- System fails as soon as one of its minimal cut parallel structures fails



- Let E_j denote the event that the minimal cut set structure K_j failed
  - The unreliability Q of the system is: $$Q = Pr\left(\bigcup_{j=1}^{k} E_j\right)$$
  - The general addition theorem gives us:

$$P(A_1 \bigcup A_2 \bigcup \ldots \bigcup A_n) = \sum_{i=1}^{n} P(A_i) - \sum_{i<j} (A_i \bigcap A_j) + \sum_{i<j<k} P(A_i \bigcap A_j \bigcap A_k) -$$
$$\cdots + (-1)^{n-1} P(A_1 \bigcap A_2 \bigcap \ldots \bigcap A_n)$$

-->System unreliability can be computed by determining the probability that one of the minimal cut structures fails

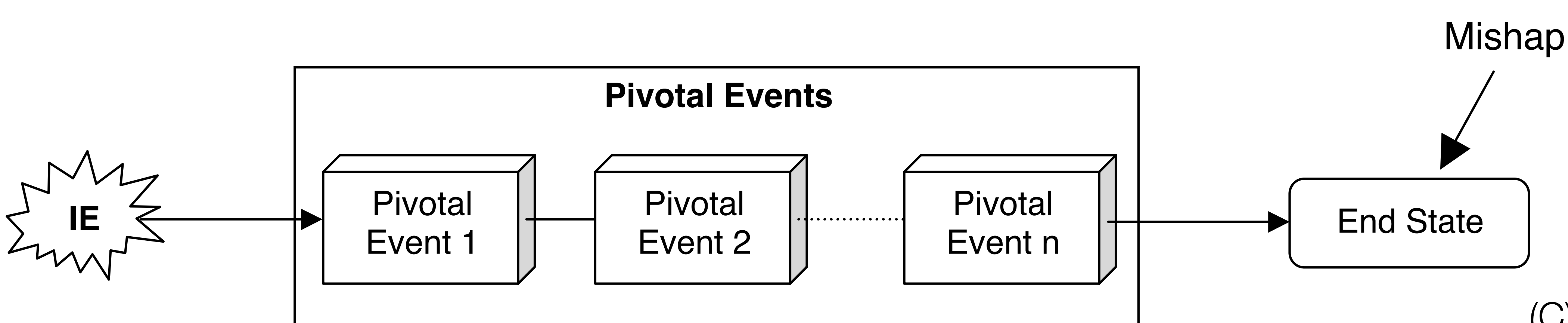
  - Allows exact system unreliability calculation, but inclusion-exclusion principle is very compute- intense (alternatives: ERAC, early term cancellation, ...)

Event Tree Analysis

- Inductive analytical diagram in failure space, based on Boolean logic
- Developed during the WASH-1400 nuclear power plant safety study (1974)
    - Fault trees became to large for proper analysis
    - Condensation of system analysis into a manageable picture
    - Make sure that the accident cases are sufficiently controlled
- Shows event sequences and accident progression in inductive analysis
- Popular approach in nuclear reactor safety engineering
- Starts with specific initiator (critical component failure)
- Companion to **fault tree analysis**, same stochastic foundation

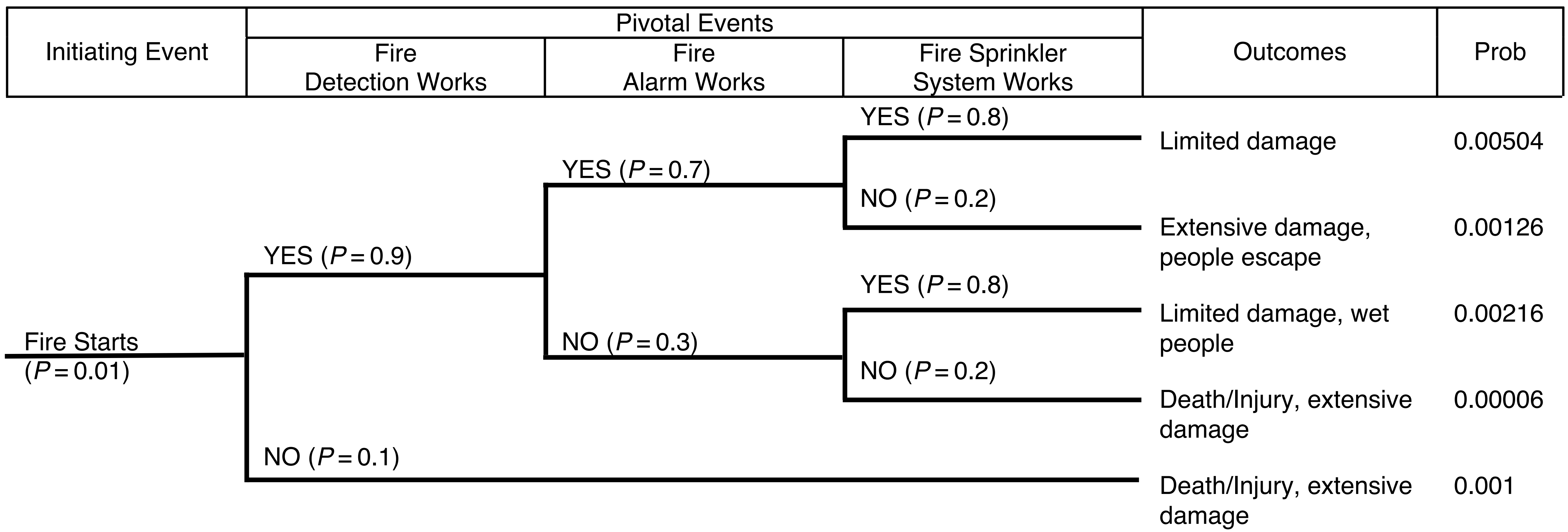
Event Tree Analysis

- **Accident scenario**: Series of events that result in an accident
- **Initiating event**: Technical failure / human error that starts an accident scenario
  - May be identified by other risk analysis technique
  - Often already identified and anticipated in the design phase
- **Pivotal events**: Intermediate events from the safety methods, to stop the accident
  - Split to positive or negative progress, sometimes more than two outcomes
- Frequency of pivotal events in system parts can be obtained from **fault tree analysis**



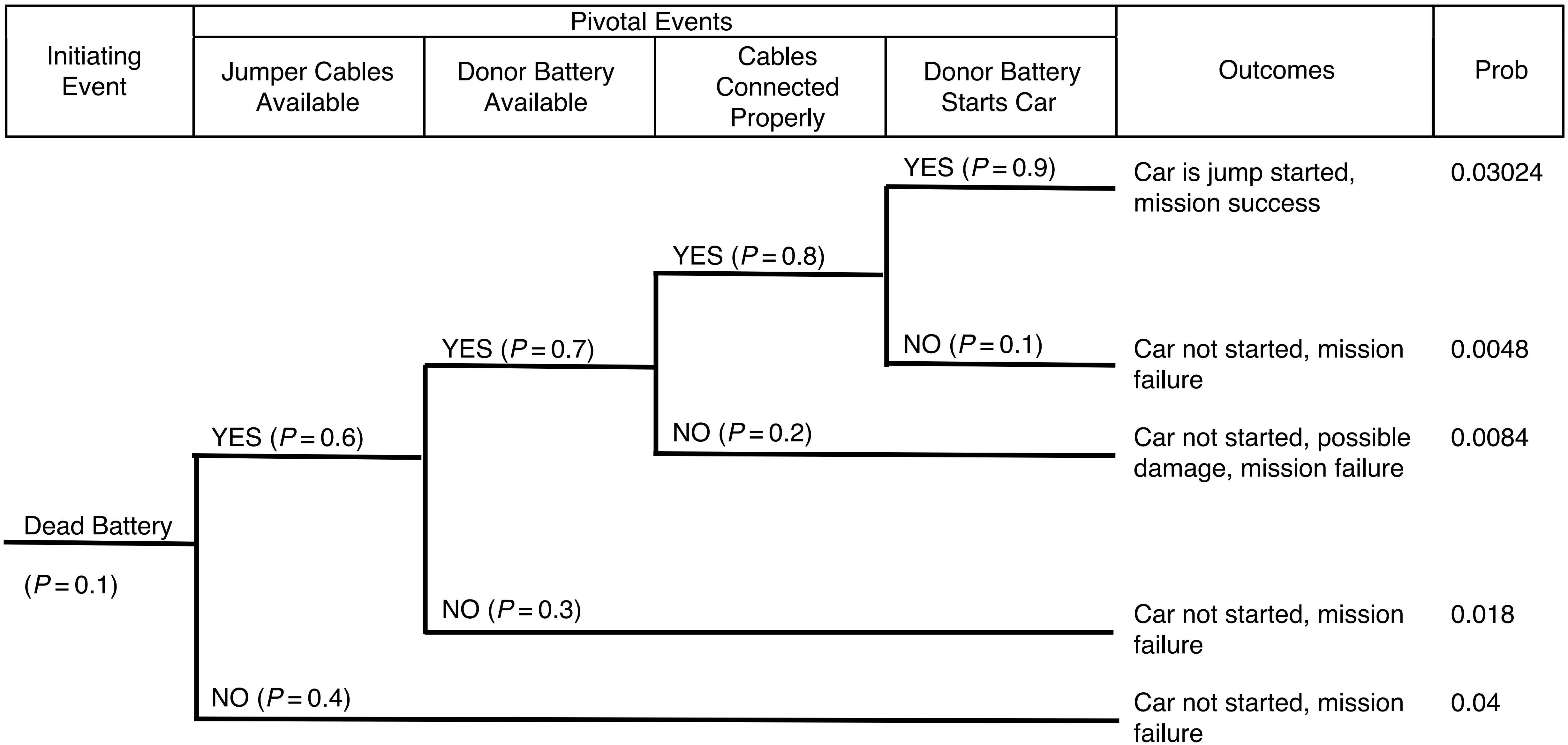
(C) Clifton et al.

Event Tree Analysis



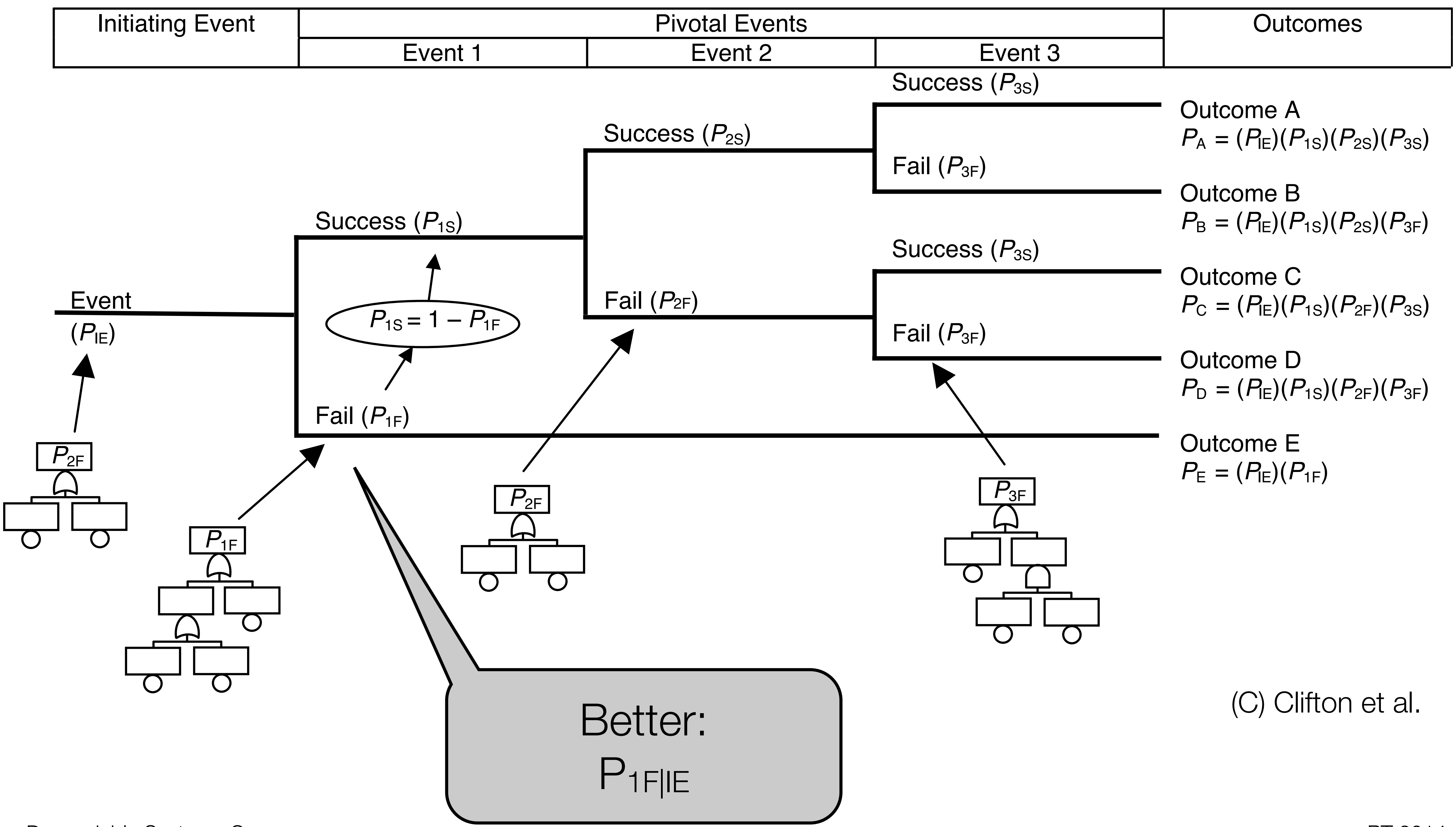
(C) Clifton et al.

Event Tree Analysis



(C) Clifton et al.

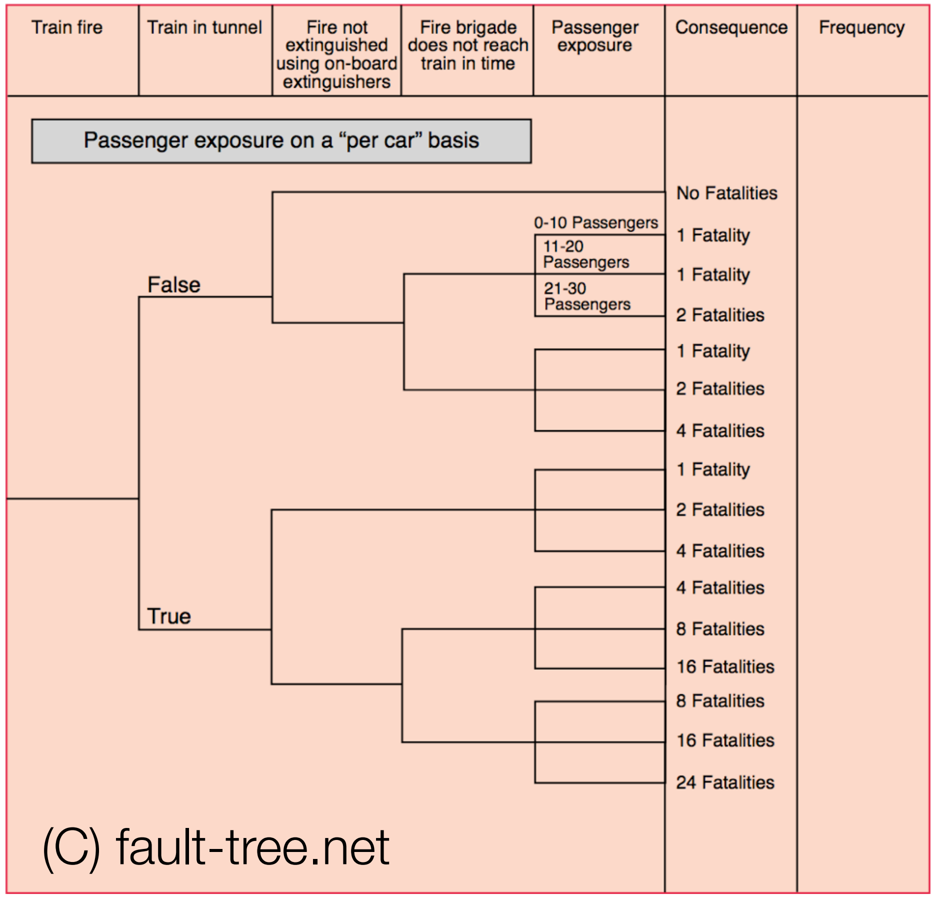
Event Tree Analysis

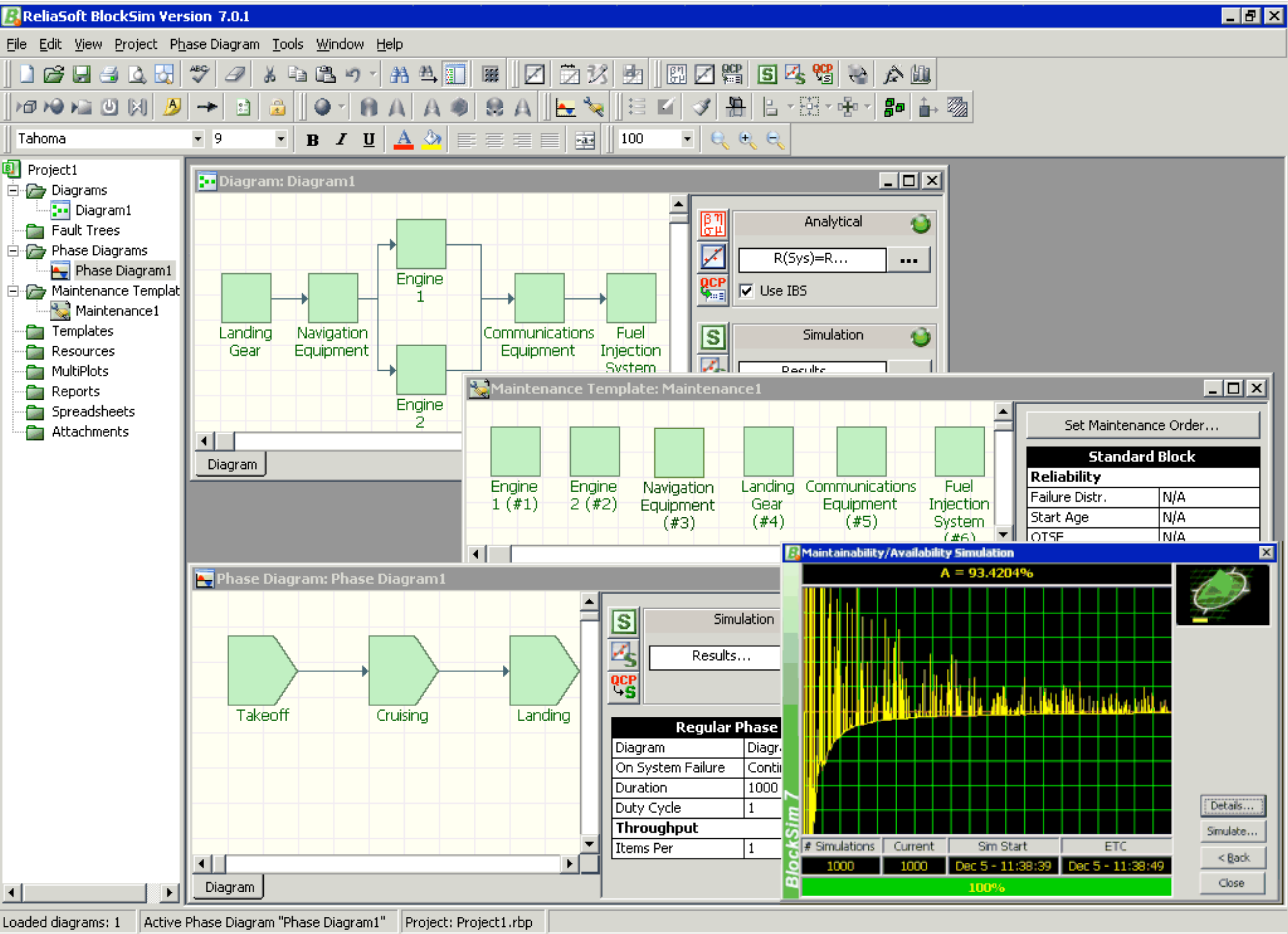


(C) Clifton et al.

Event Tree Analysis

- Possible event chains and the safety functions will be affected by hazard contribution factors
  - Explosion or no explosion, time of the day, wind direction, ...
- For a sequence of n events, there will be 2^n branches
- Possible to split the outcomes into categories, based on severity
  - Outcome frequency, loss of lives, material damage, environmental damage
- Reliability assessment of a safety function comes from FTA or RBD analysis




ReliaSoft BlockSim Version 7.0.1
File Edit View Project Phase Diagram Tools Window Help
Project1
Diagrams
Diagram1
Fault Trees
Phase Diagrams
Phase Diagram1
Maintenance Templat
Maintenance1
Templates
Resources
MultiPlots
Reports
Spreadsheets
Attachments
Diagram: Diagram1
Landing Gear
Navigation Equipment
Engine 1
Engine 2
Communications Equipment
Fuel Injection System
Analytical
R(Sys)=R...
Use IBS
Simulation
Maintenance Template: Maintenance1
Engine 1 (#1)
Engine 2 (#2)
Navigation Equipment (#3)
Landing Gear (#4)
Communications Equipment (#5)
Fuel Injection System
Set Maintenance Order...
Standard Block
Reliability
Failure Distr. N/A
Start Age N/A
Phase Diagram: Phase Diagram1
Takeoff
Cruising
Landing
Simulation
Results...
Regular Phase
Diagram
On System Failure
Duration 1000
Duty Cycle 1
Throughput
Items Per 1
Maintainability/Availability Simulation
A = 93.4204%
# Simulations 1000
Current 1000
Sim Start Dec 5 - 11:38:39
ETC Dec 5 - 11:38:49
100%
Details...
Simulate...
< Back
Close
BlockSim 7
Loaded diagrams: 1
Active Phase Diagram "Phase Diagram1"
Project: Project1.rbp

Tool Support

- Based on modeling fundamentals, existing tools support:
  - Consideration of standby redundancy and the according rate changes
  - Time-dependent analysis
  - Cost / penalty analysis
  - Preventive maintenance planning (replacement time, age replacement policy)
  - Repairable system analysis through simulation
  - Imperfect repairs (restoration factors, resource pools, crew pools)
  - Throughput analysis
  - Automated integration of component reliability databases
  - ...