Next Generation Fault Tree Analysis Methods

(D^2T^2 - Dynamic and Dependent Tree Theory)

University of Nottingham
UK | CHINA | MALAYSIA

John Andrews

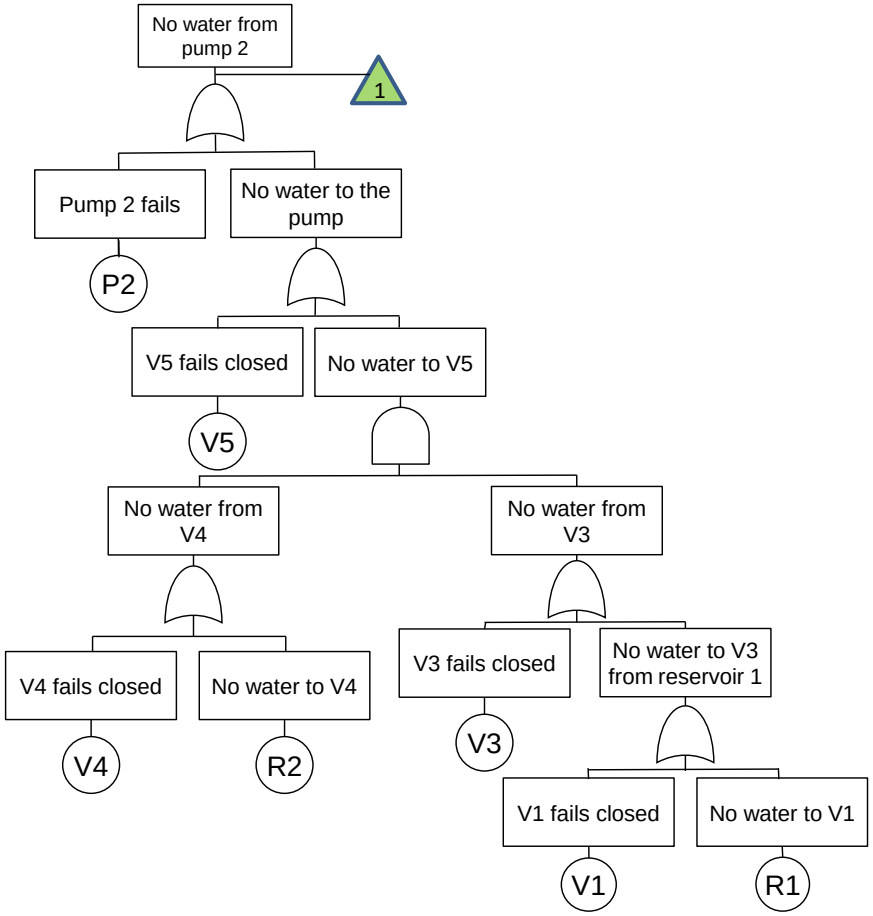
Silvia Tolo

Content

- Review of Fault Tree Analysis
  - Binary Decision Diagrams
- Dependencies and Complexities in Engineering Systems
- Modelling Dependencies
  - Markov Models
  - Petri Nets
  - Characteristics of the methods
- NxGen Requirements
  - Code structure
- Modularisation Methods
- D^2T^2 Dynamic and Dependent Fault Tree Analysis
- Case Study
  - Plant Cooling System
- Summary / Conclusions

Review of Fault Tree Analysis

Fault Tree Analysis



Independent basic events

Component failure models

- Limited maintenance process detail

No Repair: $Q(t) = F(t) = 1 - e^{-\lambda t}$

Revealed: $Q(t) = \frac{\lambda}{\lambda + \nu}\left(1 - e^{-(\lambda+\nu)t}\right)$

Unrevealed: $Q_{AV} = \lambda\left(\frac{\theta}{2} + \tau\right)$

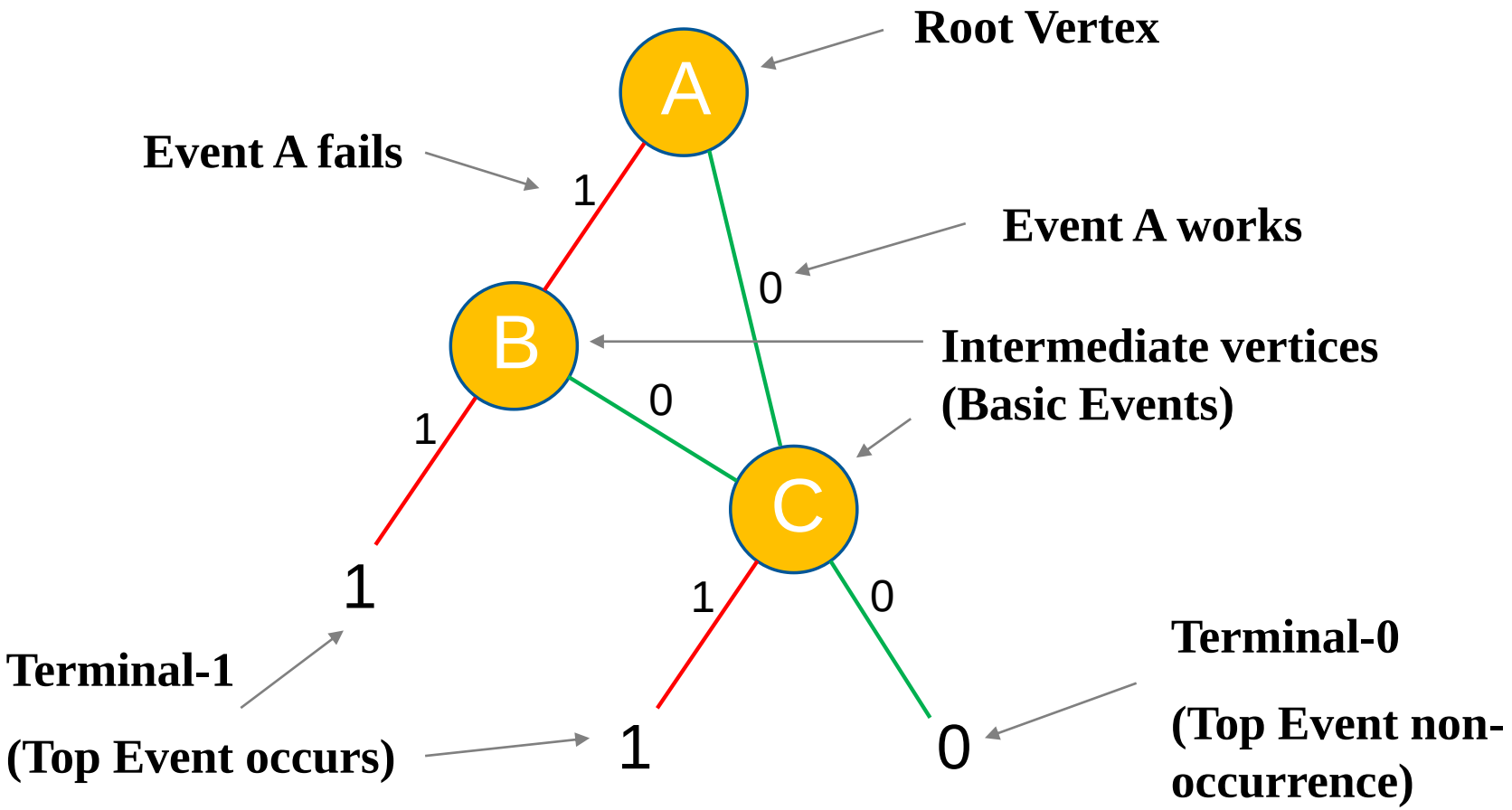
- Constant failure and repair rates
- Snap-shot in time

The Fundamental Elements:

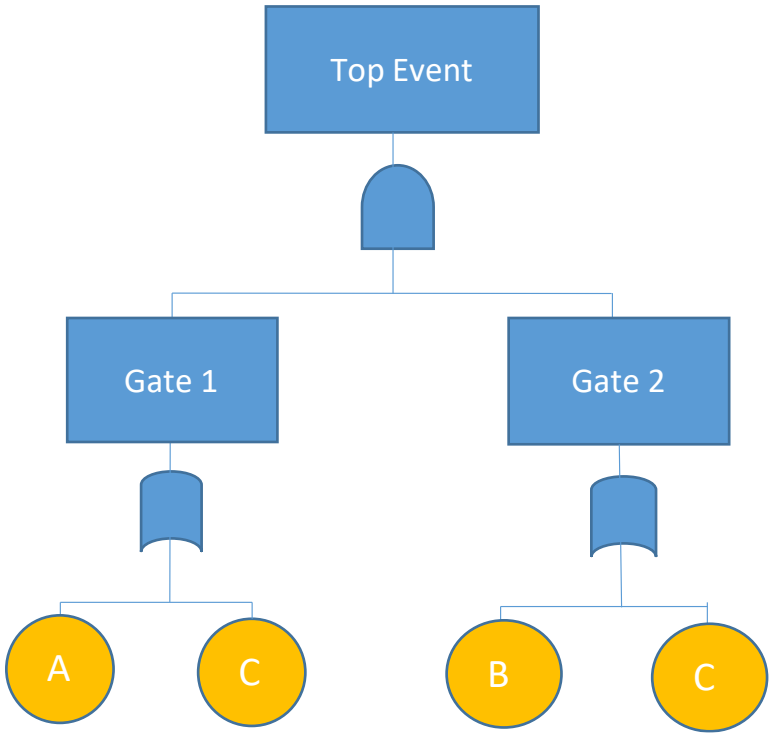
Efficient FTA
Binary Decision Diagrams

Binary Decision Diagrams (BDDs)

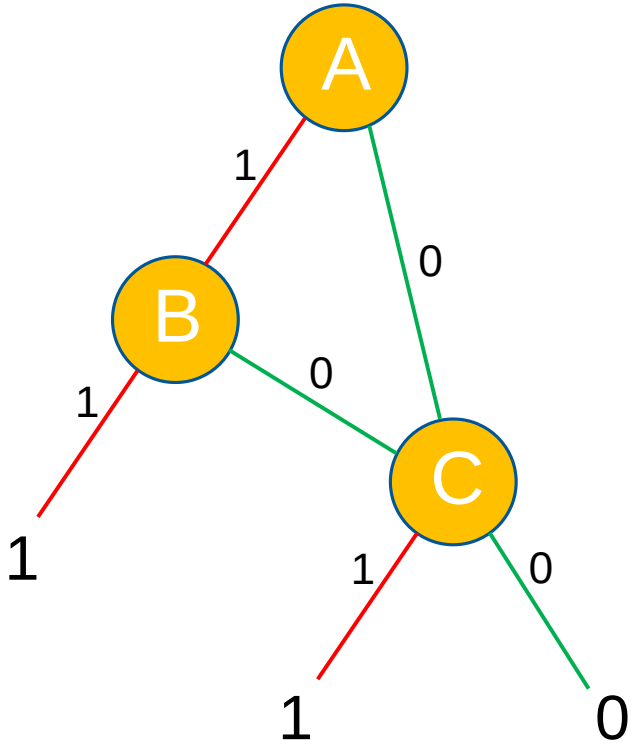
Ordering of Basic Events: A < B < C



Binary Decision Diagram



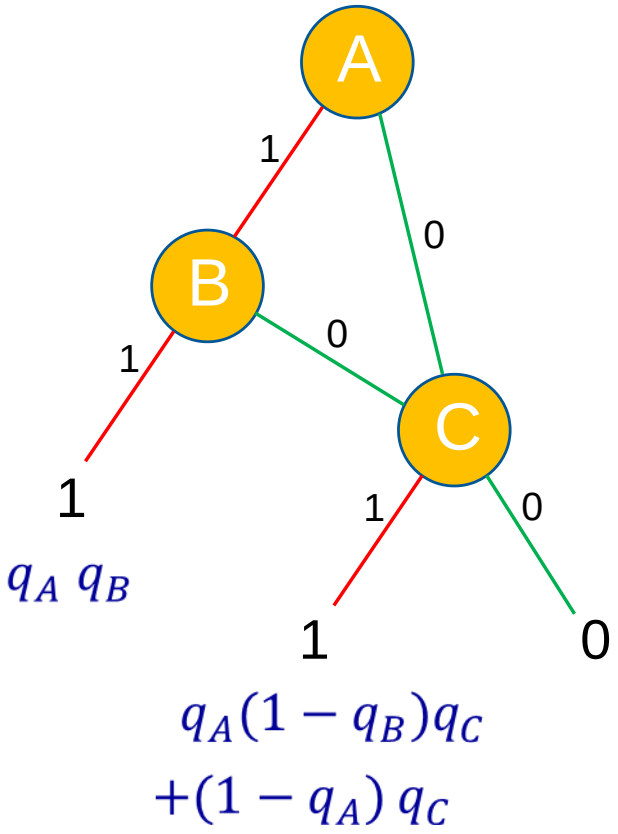
ORDERING A < B < C



Min Cut Sets: {C}, { A, B}

$$Q_{SYS} = q_A\, q_B + q_C - q_A\, q_B\, q_C$$

Top Event Probability



$$TOP = A.B + A.\bar{B}.C + \bar{A}.C$$

\+ OR

. AND

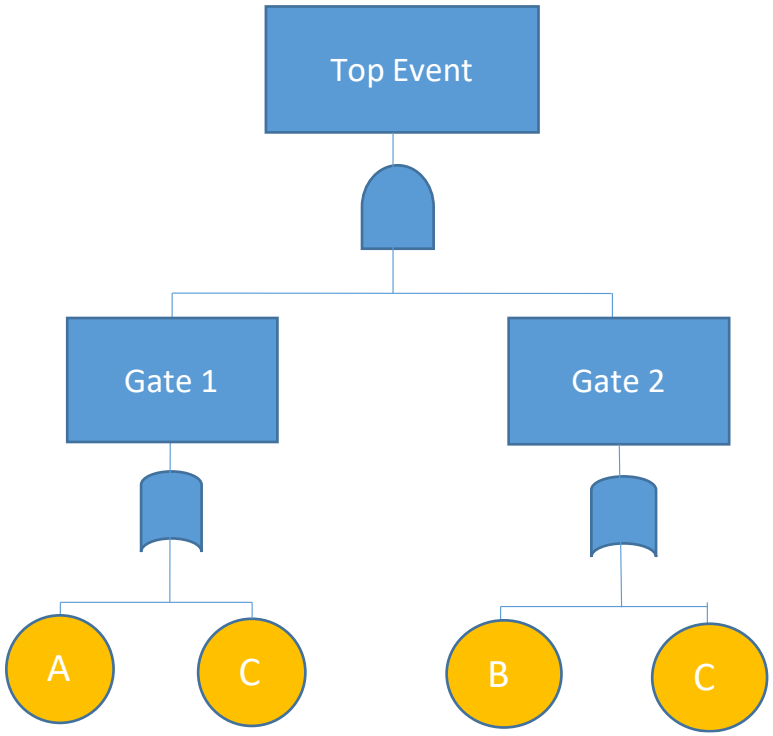
$$Q_{SYS} = q_A\ q_B + q_A(1-q_B)q_C + (1-q_A)\ q_C$$

$$= q_A\ q_B + q_C - q_A\ q_B\ q_C$$

- Exact
- Fast
- Efficient

No need to derive the Min Cut Sets as an intermediate step

Fault Tree to BDD Conversion



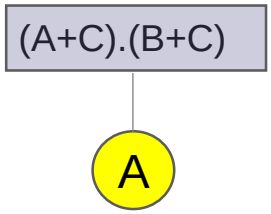
Basic Event Ordering: $A < B < C$

Generate the Logic Equation represented by the Fault Tree

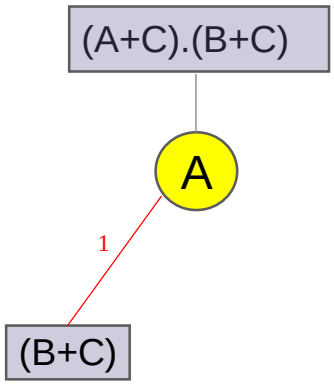
$$TOP = Gate1 \,.\, Gate2$$

$$= (A + C) \,.\, (B + C)$$

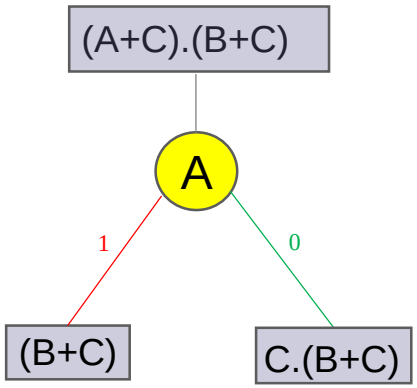
BDD Construction From Logic Equation



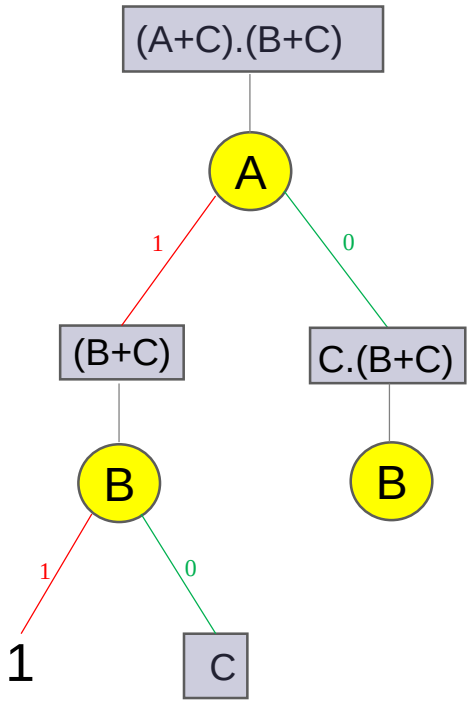
BDD Construction From Logic Equation



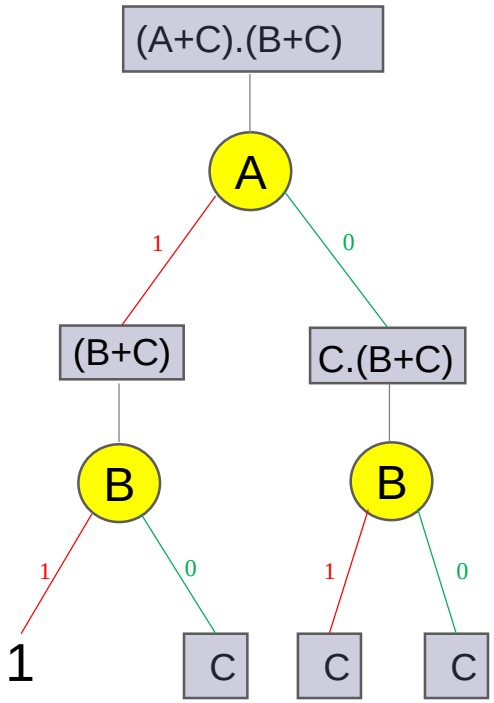
BDD Construction From Logic Equation



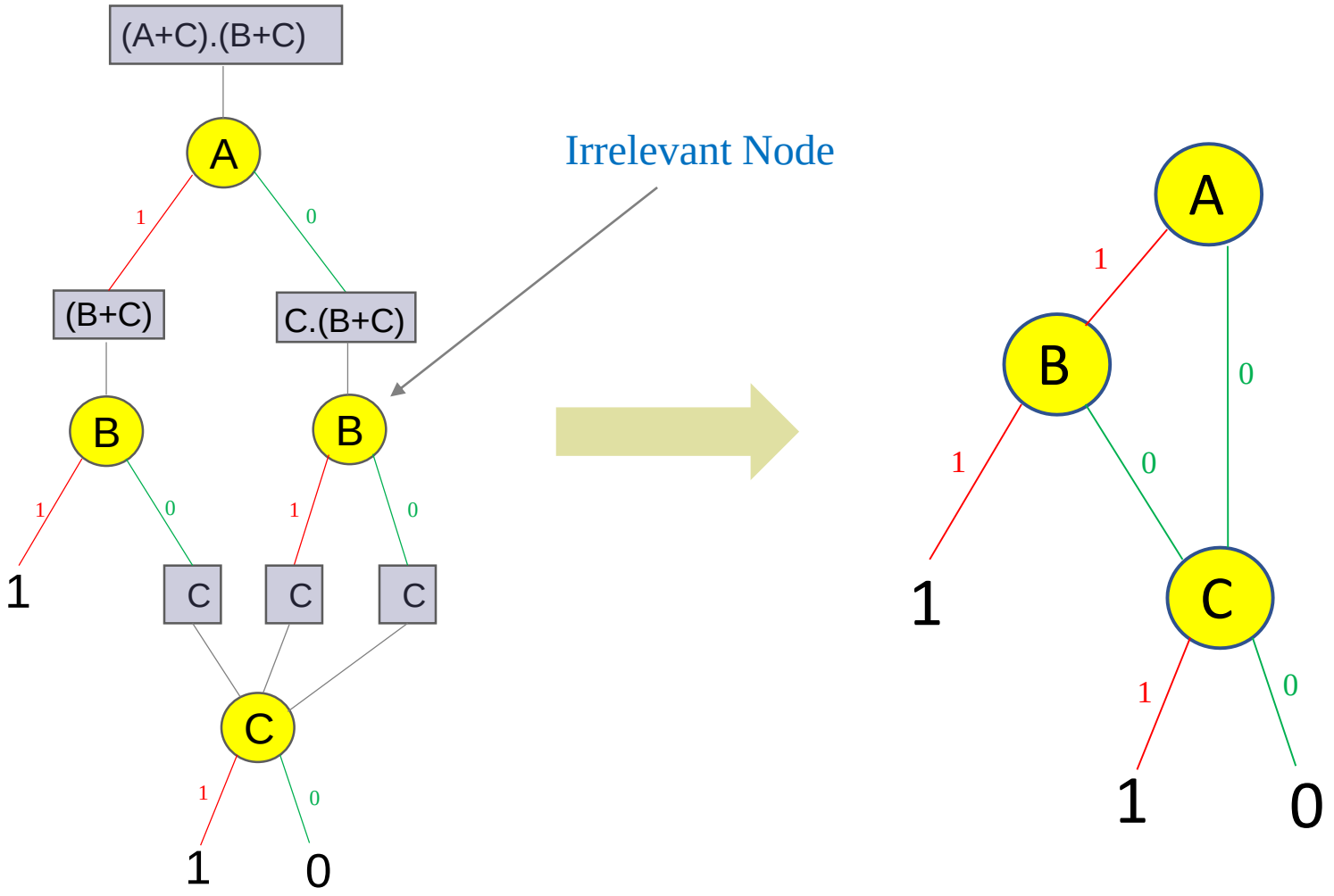
BDD Construction From Logic Equation



BDD Construction From Logic Equation



BDD Construction From Logic Equation



if-then-else (ite) Notation

For node A

if A (=1) then

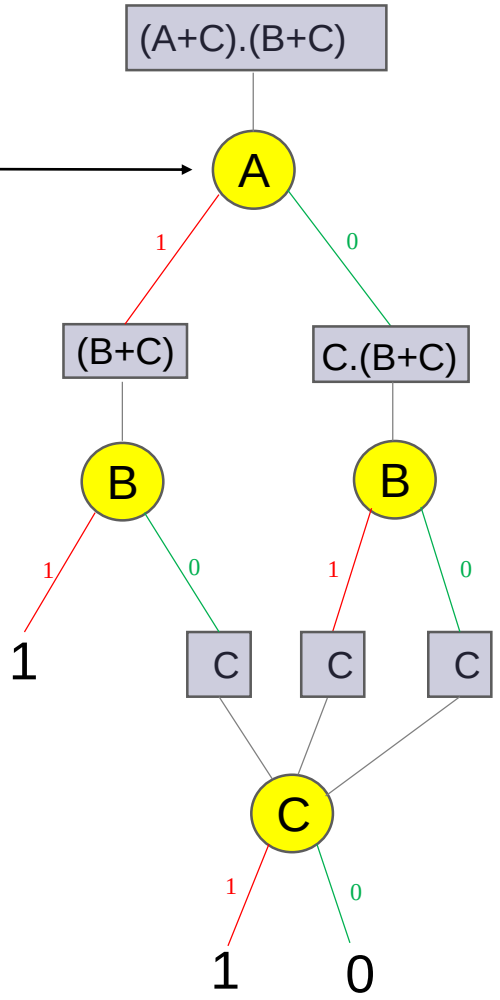
consider function f1=B+C

else

consider function f2=C.(B+C)

<u>ite notation</u>

TOP = ite(A, f1, f2)



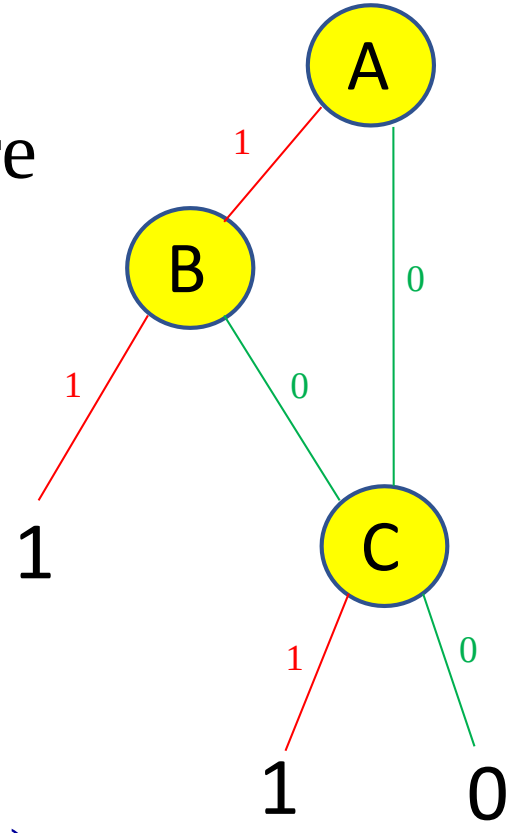
if-then-else (ite) BDD Definition

Defining each node as an ite structure

TOP=ite(A, f1, f2)

f1=ite(B, 1, f2)

f2=ite(C, 1, 0)



TOP=ite(A, ite(B, 1, f2) , f2)

TOP=ite(A, ite(B, 1, ite(C, 1, 0)) , ite(C, 1, 0))

BDD Generation Using ite Rules

- Define all Basic Events

  e.g. A = ite(A, 1, 0)

- If G=ite(X, g1, g2) and H=ite(Y, h1, h2)

  then:

$\oplus$ = AND or OR

$$G \oplus H = \begin{cases} ite(X, g1 \oplus H, g2 \oplus H) & \text{if } X<Y \\ ite\ (X, g1 \oplus h1, g2 \oplus h2) & \text{if } X=Y \end{cases}$$

- Apply bottom-up to each gate in the fault tree

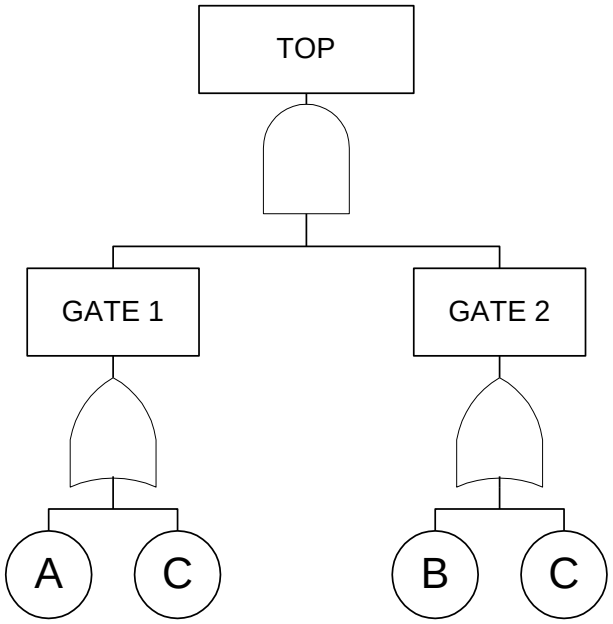
- Use simplification rules

$G + 1 = 1$ $\quad G + 0 = G$

$G \cdot 1 = G$ $\quad G \cdot 0 = 0$

$ite(X, f1, f1) = f1$

Example



Ordering A < B < C

Basic Events:

A=ite(A, 1, 0)

B=ite(B, 1, 0)

C=ite(C, 1, 0)

GATE1 = A + C

= ite(A,1 ,0)+ ite(C,1 ,0)

= ite(A, 1+ ite(C,1 ,0) , 0+ite(C,1 ,0))

= ite(A, 1, ite(C,1 ,0))

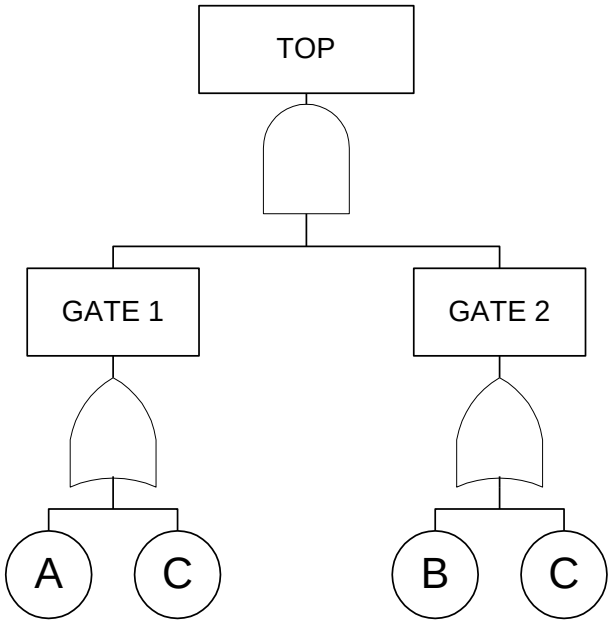
if $G = ite(X, g1, g2)$

$H = ite\ (Y, h1, h2)$

then:

$$G \oplus H = \begin{cases} ite(X, g1 \oplus H, g2 \oplus H) & if\ X < Y \\ ite(X, g1 \oplus h1, g2 \oplus h2) & if\ X = Y \end{cases}$$

Example



GATE1 = B + C

= ite(B,1 ,0) + ite(C,1 ,0)

= ite(B, 1+ ite(C,1 ,0) , 0+ite(C,1 ,0))

= ite(B, 1, ite(C,1 ,0))

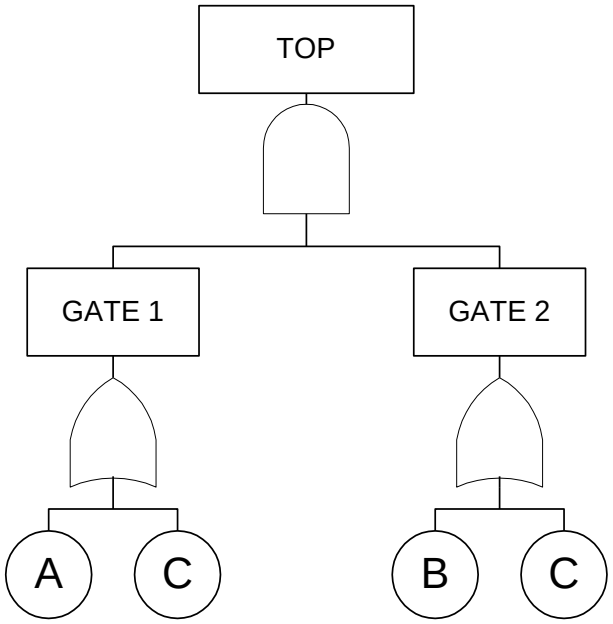
if $G = ite(X, g1, g2)$

$H = ite\ (Y, h1, h2)$

then:

$$G \oplus H = \begin{cases} ite(X, g1 \oplus H, g2 \oplus H) & if\ X < Y \\ ite(X, g1 \oplus h1, g2 \oplus h2) & if\ X = Y \end{cases}$$

Example



TOP = GATE1 . GATE2

= ite(A, 1, ite(C,1 ,0)).ite(B, 1, ite(C,1 ,0))

= ite(A, 1. ite(B, 1, ite(C,1 ,0) ,
ite(C,1 ,0) .ite(B, 1, ite(C,1 ,0))

= ite(A, ite(B, 1, ite(C,1 ,0) ,
ite(B, 1. ite(C,1 ,0) , ite(C,1 ,0). ite(C,1 ,0))

= ite(A, ite(B, 1, ite(C,1 ,0) ,
ite(B, ite(C,1 ,0) , ite(C,1 ,0))

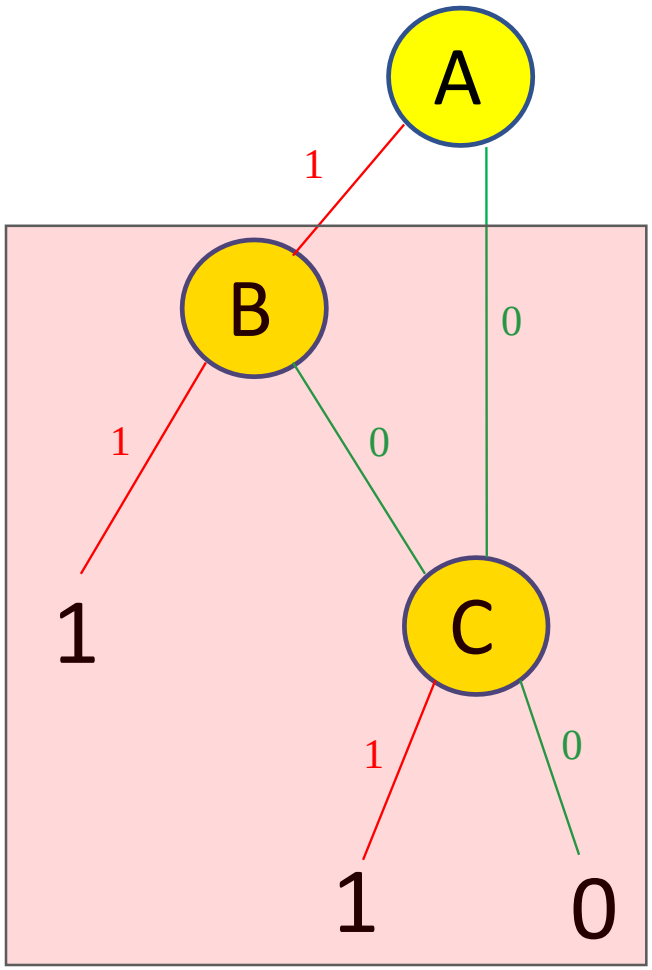
= ite(A, ite(B, 1, ite(C,1 ,0) , ite(C,1 ,0))

if $G = ite(X, g1, g2)$
$H = ite\ (Y, h1, h2)$

then:

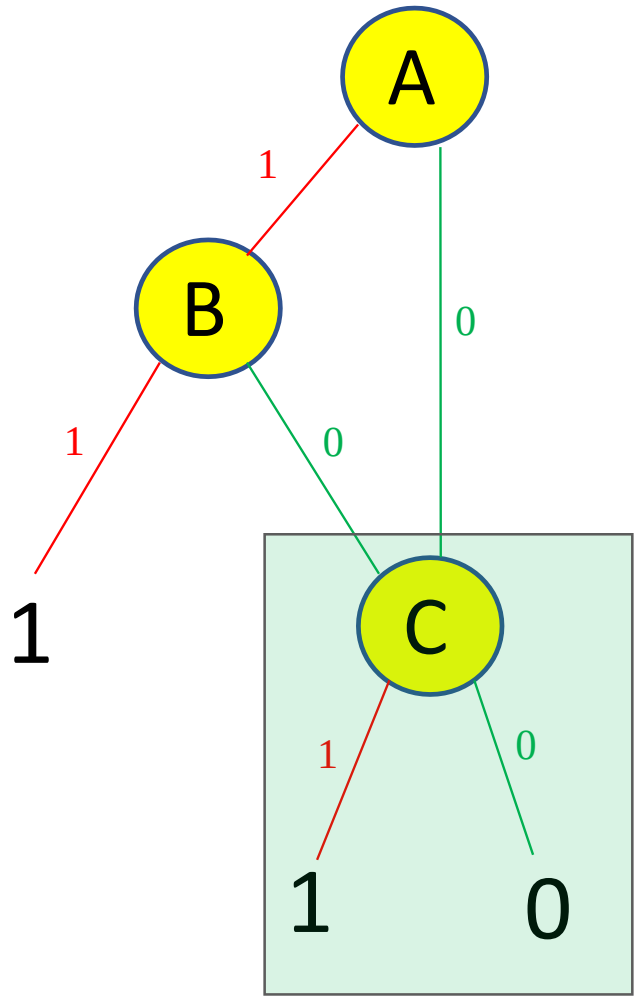
$$G \oplus H = \begin{cases} ite(X, g1 \oplus H, g2 \oplus H) & if\ X < Y \\ ite(X, g1 \oplus h1, g2 \oplus h2)\ if\ X = Y \end{cases}$$

Example - cont



TOP = ite(A, ite(B, 1, ite(C, 1 ,0)), ite(C, 1 ,0))

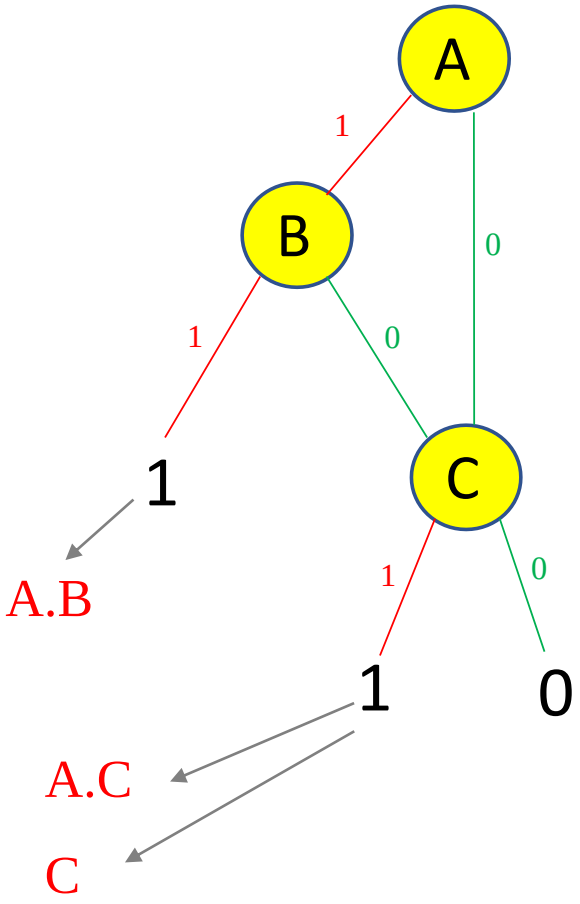
Example - cont



TOP = ite(A, ite(B, 1, ite(C, 1 ,0) , ite(C, 1 ,0))

Minimal Cut Sets

Causes of Failure

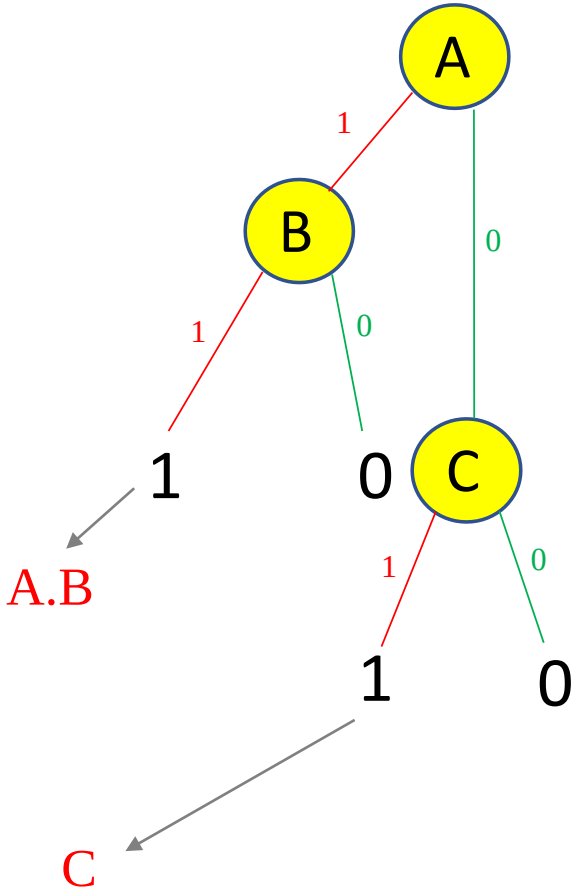


The paths deliver Cut Sets:

The list of component failed states which result in system failure.

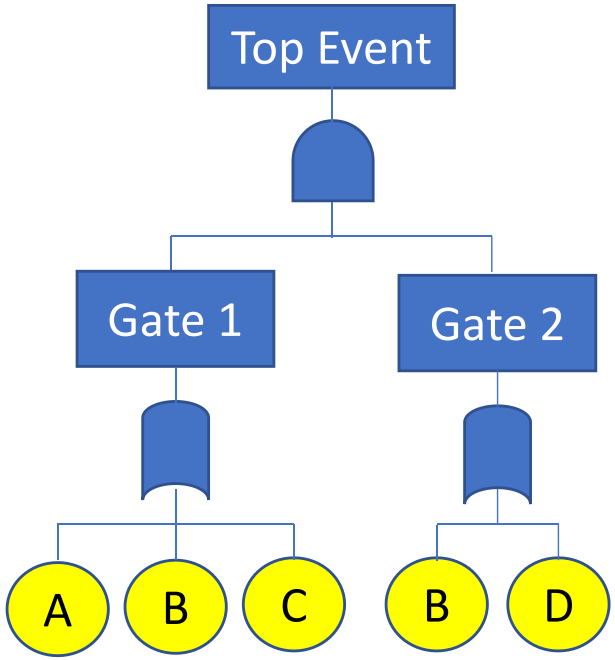
The BDD can be processed to deliver a BDD (Zero-suppressed BDD) which encodes only the Minimal Cut Sets:

The list of component failures which is necessary and sufficient to cause system failure.



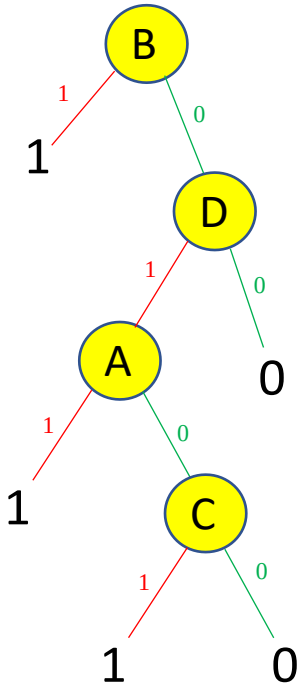
Variable Ordering Methods

Ordering Example



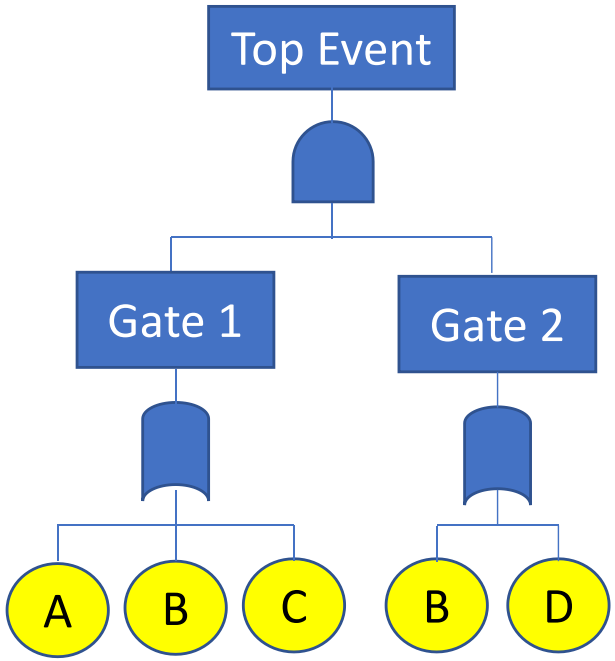
TOP = (A + B + C) . (B + D)
= B + A.D + C.D

Ordering: B < D < A < C



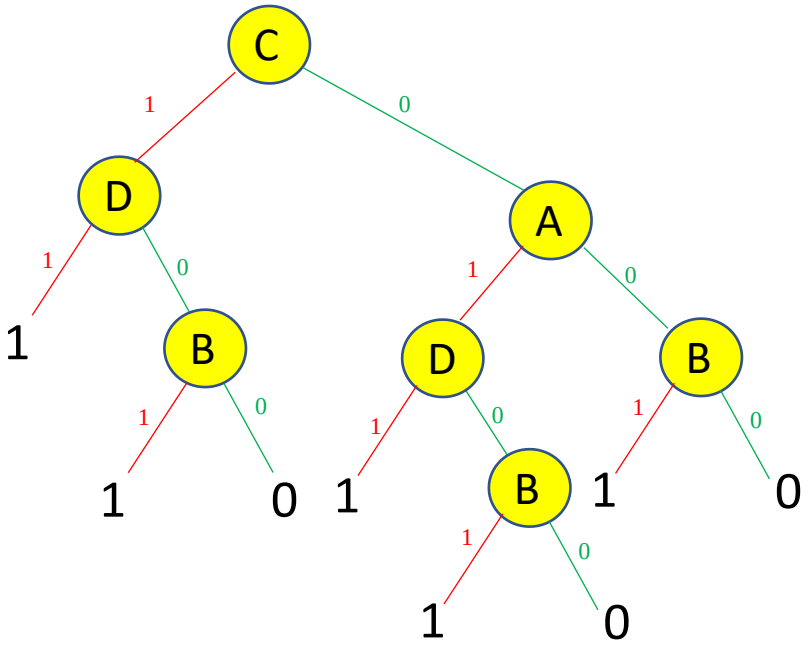
- 3 paths through the BDD
- 3 minimal cut sets

Ordering Example



TOP = (A + B + C) . (B + D)
= B + A.D + C.D

Ordering: C < A < D < C



- 5 paths through the BDD
- 3 minimal cut sets

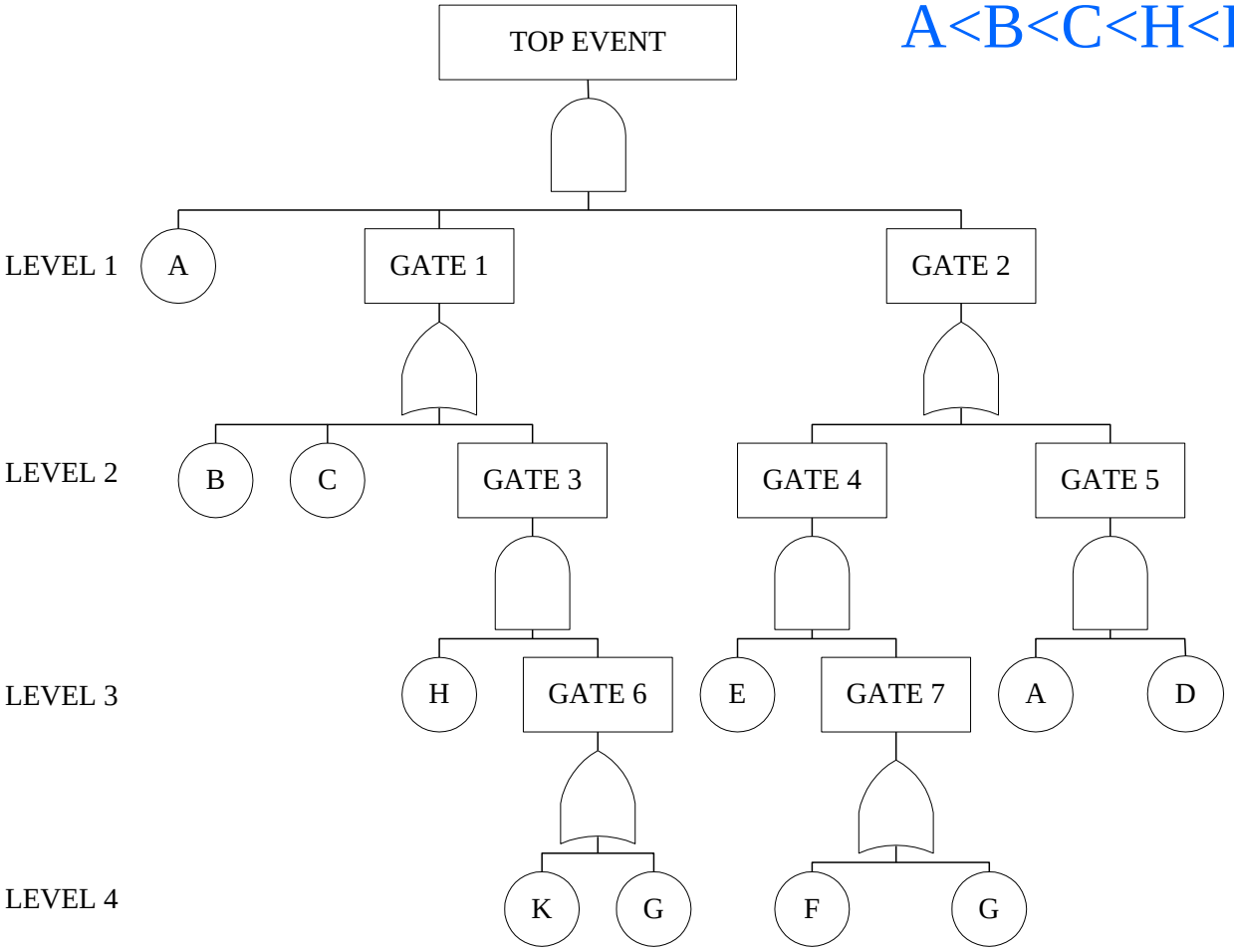
Variable Ordering

- Ordering scheme selected can have a dramatic effect on the analysis.
  - Good gives an efficient analysis
  - Bad can make problem intractable

- A common approach is a systematic traversal of the fault tree structure such as:
  - Top-down, left-right

Ordering Heuristics

TOP-DOWN, LEFT-RIGHT:



A<B<C<H<E<D<K<G<F

A

B, C

H, E, D

K, G, F

Variable Ordering Schemes

- Many other ordering schemes can be used
- Alternatives to these ‘neighbourhood’ methods are based on ‘structural importance’.
- ‘Structural importance’ methods allow nodes to be selected from anywhere in the tree structure. Nodes are allocated a ‘weighting’ which indicates their contribution to the top event. Highest ‘weightings’ ordered first.
- Neural Network selection methods.

System Failure Frequency

$$w_{SYS}(t) = \sum_{\substack{i \\ initiators}} G_i(\boldsymbol{q}).w_i\,(t)$$

The Criticality Function, $G_i(q)$, is the probability that the system is in a critical state for component i such that the failure of component i causes system failure.

$w_i(t)$ is the failure intensity of component i.

$$G_i(\boldsymbol{q}) = \frac{\partial Q_{SYS}}{\partial q_i} = Q_{SYS}(1_i, \boldsymbol{q}) - Q_{SYS}(0_i, \boldsymbol{q})$$

$Q_{SYS}(1_i, \boldsymbol{q})$ probability that the system fails with component i failed

$Q_{SYS}(0_i, \boldsymbol{q})$ probability that the system fails with component i working

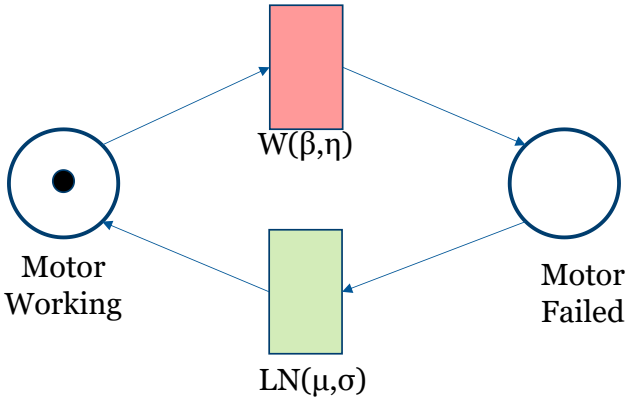
Dependencies and Complexities in Engineering Systems

Complexity – non-constant rates

Non-constant Failure / Repair Rates

Failure Time Distribution

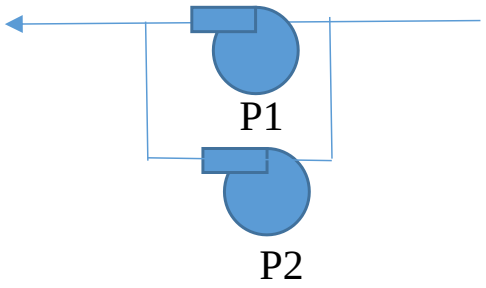
- Component experiences wear-out
- Systems operating beyond their design life
- Weibull failure time distributions are common



Repair Time Distribution

- Repair is not a random process
- Lognormal repair time distributions are common

Dependency - Standby



Standby System

- Pump P1 operational.
- When P1 fails P2 takes over the duty

Hot Standby

Both pumps are operational but the fluid is just driven by P1. On failure of P1, the fluid now passes through P2

P1 & P2 Independent

Warm Standby

Pump P2 is not operational in standby. It becomes operational when P1 fails. It can fail in standby but with a lower rate than when operational.

P1 & P2 Dependent

Cold Standby

Pump P2 is not operational in standby. It becomes operational when P1 fails. It cannot fail in standby.

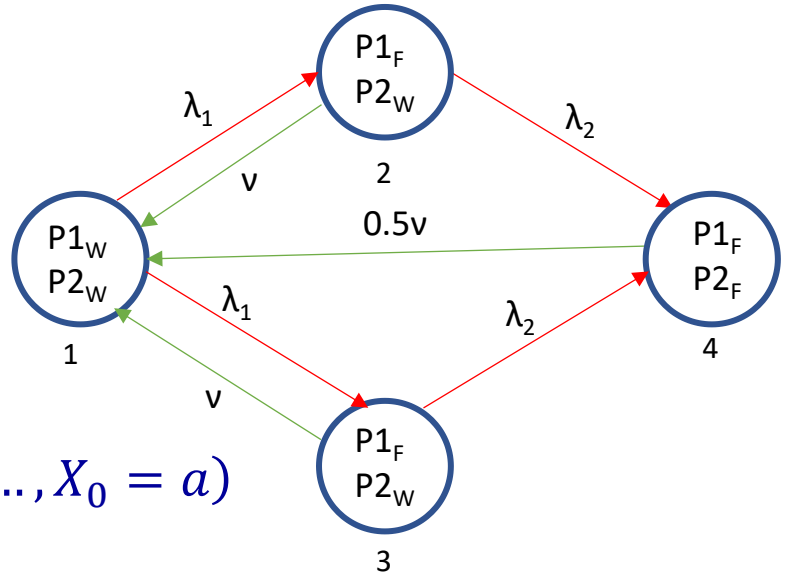
P1 & P2 Dependent

Dependency Examples

Type	Description	Example
Secondary Failure	When one component fails it increases the load on a second component which then experiences an increased failure rate	Two pumps both operational and sharing the load. Each pump has the capability to deliver the full demand should the other pump fail
Opportunistic Maintenance	A component fails which causes a system shutdown or the requires specialist equipment for the repair. The opportunity is taken to do work on a second component which has not failed but is in a degraded state	Components on a circuit board. Components in a sub-sea production module
Common Cause	When one characteristic (eg materials, manufacturing, location, operation, installation maintenance) causes the degraded performance in several components	Incorrect maintenance done on several identical sensors Impact breaks the circuit on cables routed in the same way to different redundant channels
Queueing	Failed components all needing the same maintenance resource are queued. Then repaired in priority order	Limited number of maintenance teams, equipment or spares

Modelling Dependencies and Complexities in Engineering Systems

Markov Analysis



Characteristics

- State – based method
  - States represent the system states
- Memoryless property

$$P(X_{t+dt} = k \mid X_t = j, X_{t-dt} = i, X_{t-2dt} = h, \dots, X_0 = a)$$

$$= P(X_{t+dt} = k \mid X_t = j)$$

- Exponential distribution for state residence times (constant transition rates)

Solution

- Numerical Methods

Outputs

- The probability of being in each state at time t.

$$(\dot{P_1}, \dot{P_2}, \dot{P_3}, \dots, \dot{P_n}) = (P_1, P_2, P_3, \dots, P_n) \begin{bmatrix} -\lambda_{1,1} & \cdots & \lambda_{1,n} \\ \vdots & \ddots & \vdots \\ \lambda_{n,1} & \cdots & -\lambda_{n,n} \end{bmatrix}$$

Markov Modelling Procedure

Model Development - Markov State Transition Diagram

- Identify all possible states.
- List all transitions between states (failures/repairs).

Model Analysis

- Develop one equation for each state on the diagram (state equations).
- Solve equations to find probability of being in each state.

Single Component Failure Model

States:

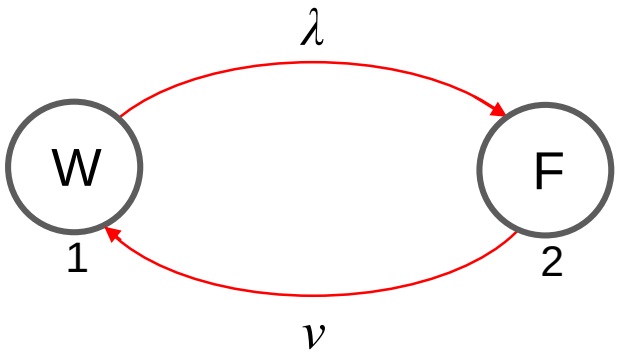
Working (W)

Failed (F)

Transitions:

Failure (W → F)

Repair (F → W)



Outputs:

$P_F(t)$ = probability of component failed at time t unavailability

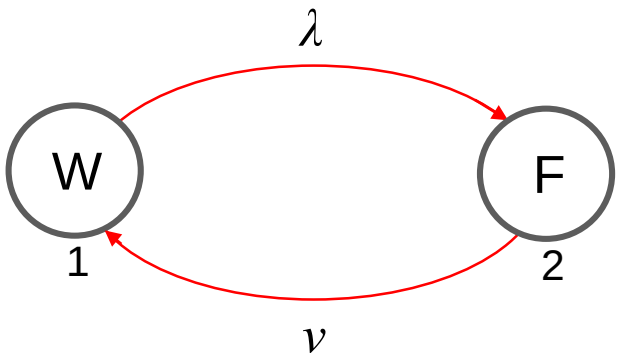
$P_W(t)$ = probability of component working at time t availability

Derive the Transition Rate Matrix

Rate of change of state i probability =

- (rate of leaving state i) x P(residing in state i)

$$+\sum_{\substack{j=1 \\ j\neq i}}^{n}(rate\ of\ arriving\ in\ state\ i\ from\ state\ j)\ \times P(residing\ in\ state\ j)$$



$$\frac{dP_W(t)}{dt} = -\lambda P_W(t) + \nu P_F(t)$$

$$\frac{dP_F(t)}{dt} = \lambda P_W(t) - \nu P_F(t)$$

Derive the Transition Rate Matrix (A)

$$\frac{dP_W(t)}{dt} = -\lambda P_W(t) + \nu P_F(t)$$

Denote: $\frac{dP_W(t)}{dt}$ $by\ \dot{P_W}$

$$\frac{dP_F(t)}{dt} = \lambda P_W(t) - \nu P_F(t)$$

$\frac{dP_F(t)}{dt}$ $by\ \dot{P_F}$

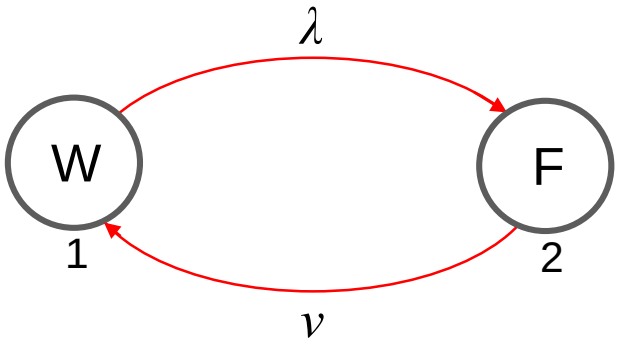
Therefore in Matrix form:

$$[\dot{P}_W(t) \quad \dot{P}_F(t)] = [P_W \quad P_F].\begin{bmatrix} -\lambda & \lambda \\ \nu & -\nu \end{bmatrix}$$

$$\dot{\boldsymbol{P}} = \boldsymbol{P}.[\boldsymbol{A}]$$

$$[\boldsymbol{A}] = \begin{bmatrix} -\lambda & \lambda \\ \nu & -\nu \end{bmatrix}$$

Transition Rate Matrix



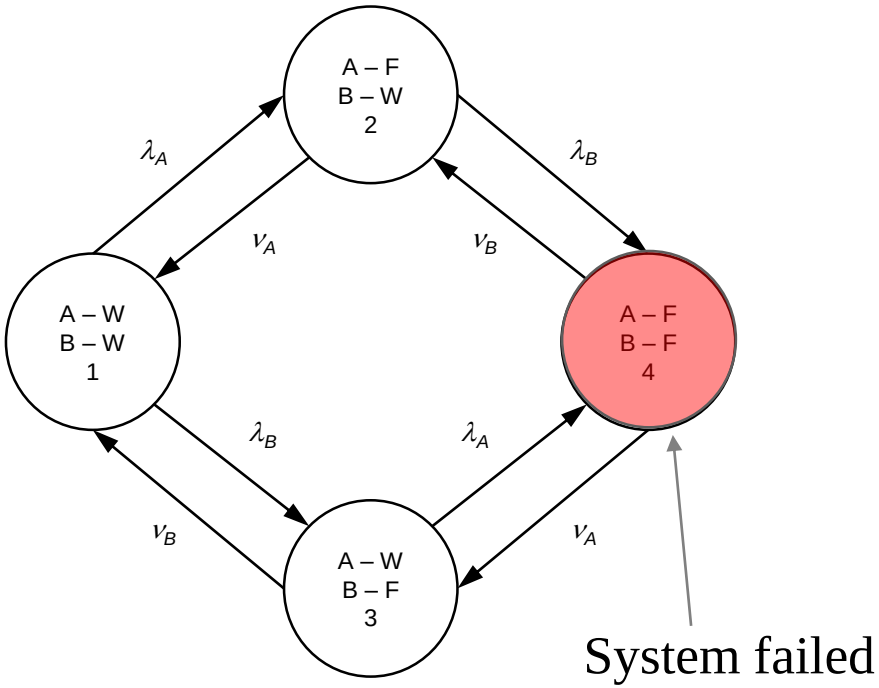
$$[\boldsymbol{A}] = \begin{bmatrix} -\lambda & \lambda \\ \nu & -\nu \end{bmatrix}$$

Rules:

- The dimension of the matrix is equal to the number of states.
- Element *i*, *j* (*i*th row, *j*th col) represents the transition rate from state *i* to state *j*.
- A diagonal element *i*,*i* is the total transition rate out of state *i* (always negative). (All rows sum to zero).

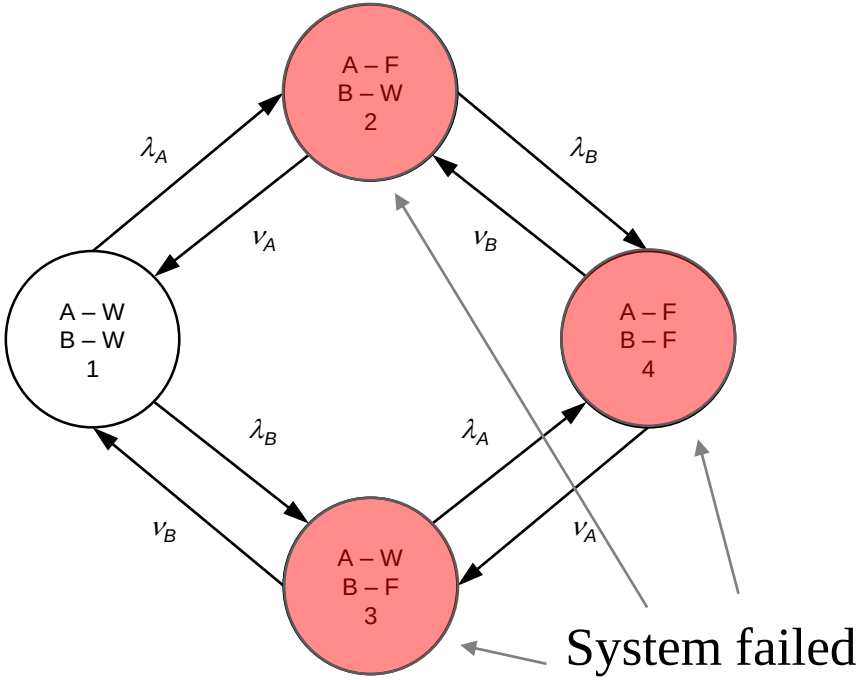
Example – 2 Component System

Two component parallel system (availability model)



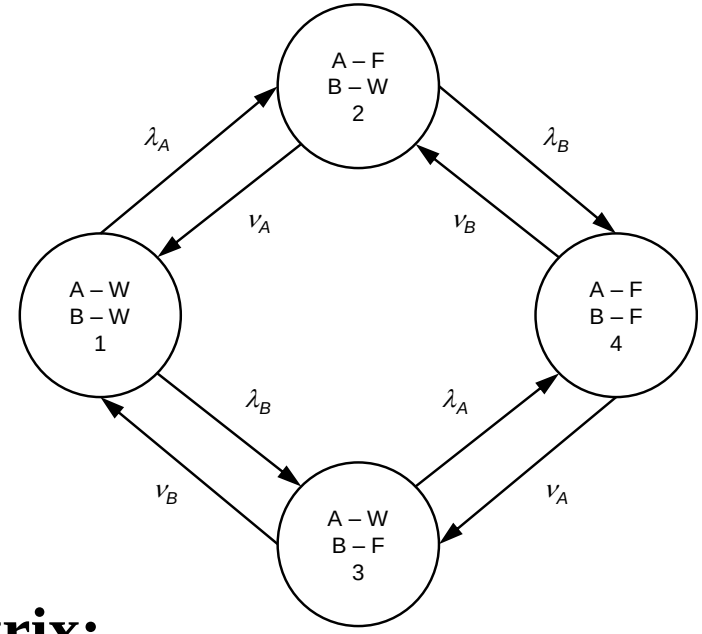
$Q_{SYS} = P_4$

Two component series system (availability model)



$Q_{SYS} = P_2 + P_3 + P_4$

Example – Availability Model

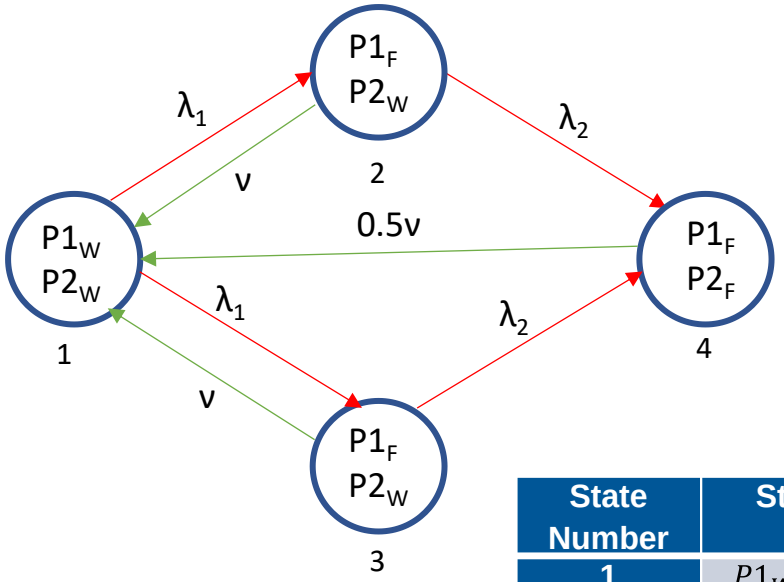


Transition rate matrix:

$$[A] = \begin{array}{c} \text{from:} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \end{array} \begin{array}{c} \text{to:} \\ \begin{bmatrix} -(\lambda_A + \lambda_B) & \lambda_A & \lambda_B & 0 \\ \nu_A & -(\lambda_B + \nu_A) & 0 & \lambda_B \\ \nu_B & 0 & -(\lambda_A + \nu_B) & \lambda_A \\ 0 & \nu_B & \nu_A & -(\nu_A + \nu_B) \end{bmatrix} \end{array}$$

Dependency Example

Pumps P1 and P2 operate together to provide a flow. Should one pump fail then the second can deliver the required flow on its own. However, when one fails it puts an extra load on the other and increases its failure rate from λ_1 to λ_2.



Pump Failure:

$\lambda_1 = 2.0 \times 10^{-5}$ per hour Normal Load

$\lambda_2 = 5.0 \times 10^{-3}$ per hour Full Load

Pump Repair:

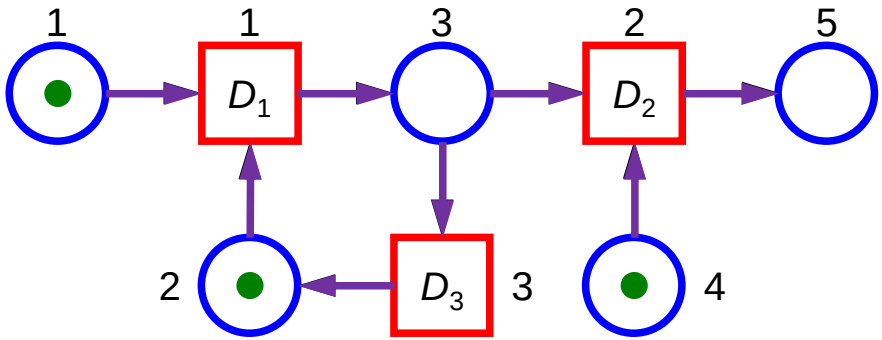
$\nu = 0.041667$ (MTTF = 24hrs)

$\nu_2 = 0.5\ \nu$.

State Number	State	State Probability	Intensity Expression	State Intensity
1	$P1_W P2_W$	0.99743518	$w1 = (Q2 + Q3).\nu + Q4.0.5\nu$	7.12456×10^{-5}
2	$P1_F P2_W$	0.00042747	$w2 = Q1.\lambda_1$	1.99487×10^{-5}
3	$P1_W P2_F$	0.00042747	$w3 = Q1.\lambda_1$	1.99487×10^{-5}
4	$P1_F P2_F$	0.00170988	$w4 = (Q2 + Q3).\lambda_2$	4.2747×10^{-6}

The Fundamental Elements: Petri Nets

Petri Net Basics and Definitions



Places

Conditions, available resources, counters

Tokens

Mark places
Represent the current status of the system

Transitions

D_j

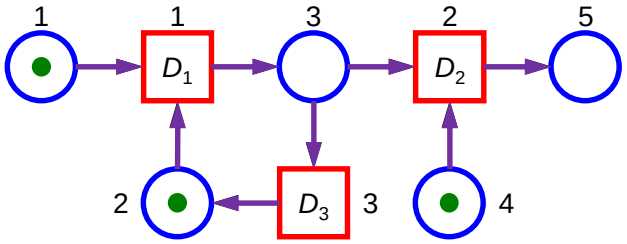
- *Time delay D_j at which transitions occur*
- *Immediate $D_j = 0$*
- *Timed $D_j > 0$*

Edges

- *Input edges*
  - place to transition
- *Output edges*
  - transition to place

Petri Net Modelling

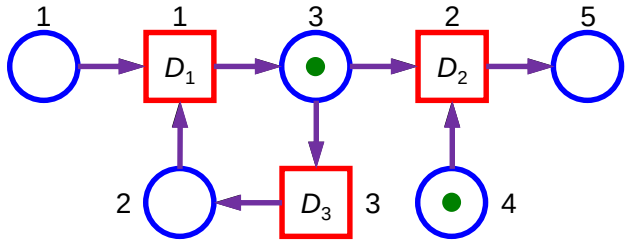
If all input places of a transition are marked by at least one token then this transition is called **enabled**.



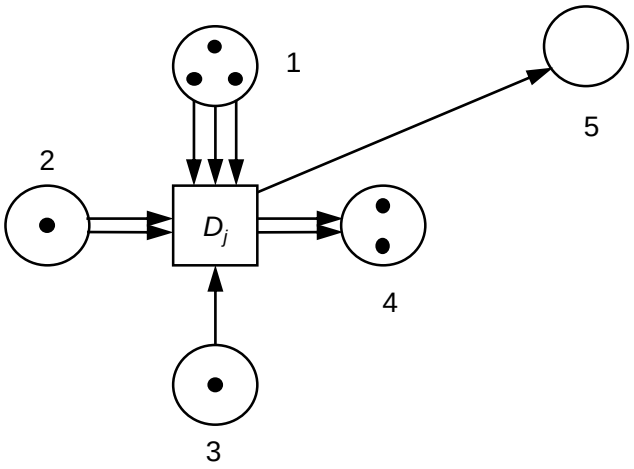
After a delay $D \geq 0$ the transition **fires**.

- removes one token from each of its input places
- adds one token to each of its output places.

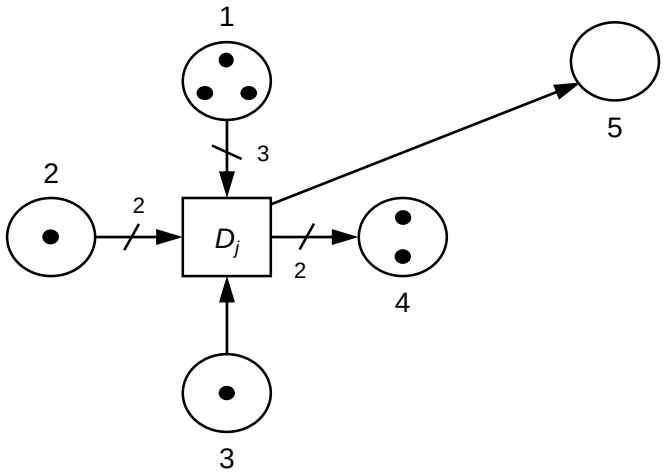
After D_1



Weighted Edges



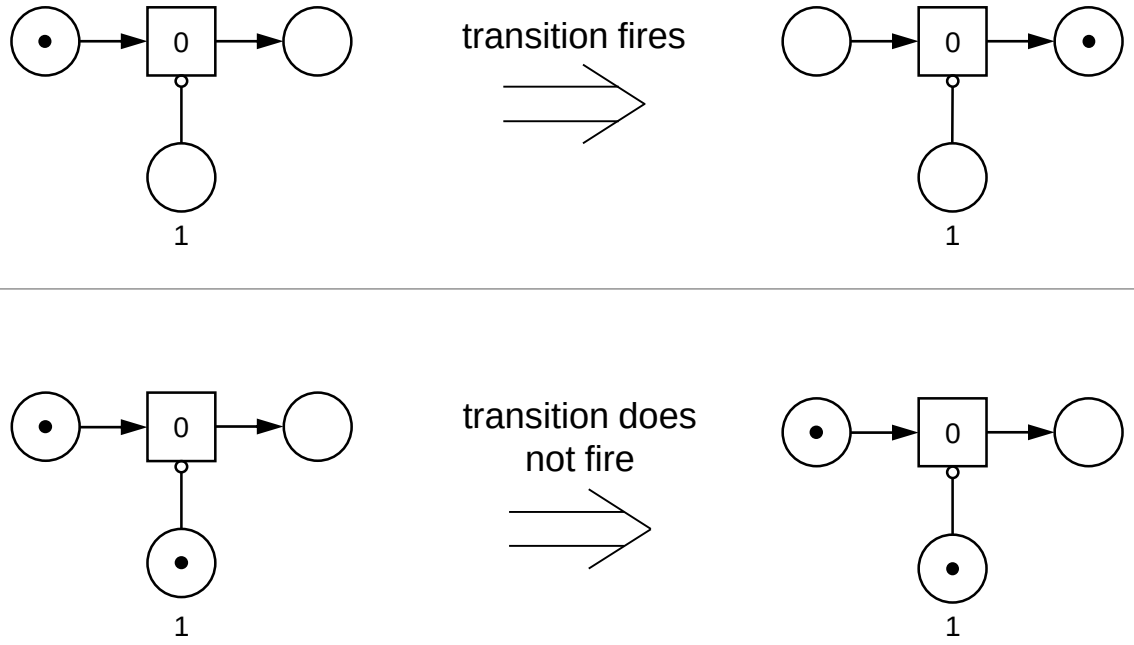
Multigraph



Weighted Edge

Inhibit Edges

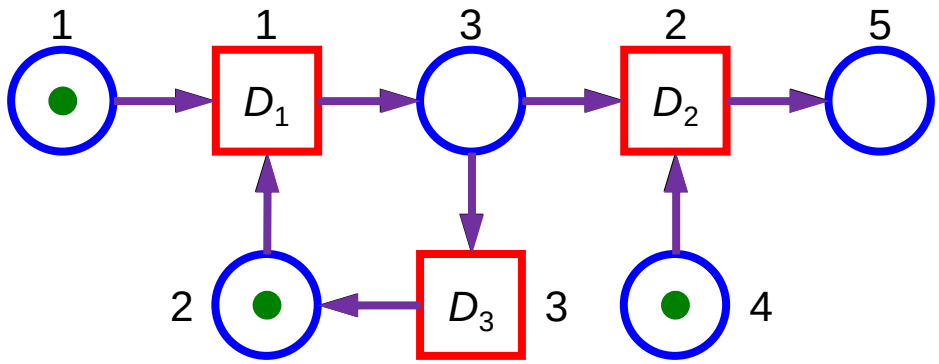
Blocks a transition when the input place is marked.



Petri Net Modelling

Characteristics

- Any distribution of times to transition
- Capable of modelling very complex maintenance strategies
- Concise structure



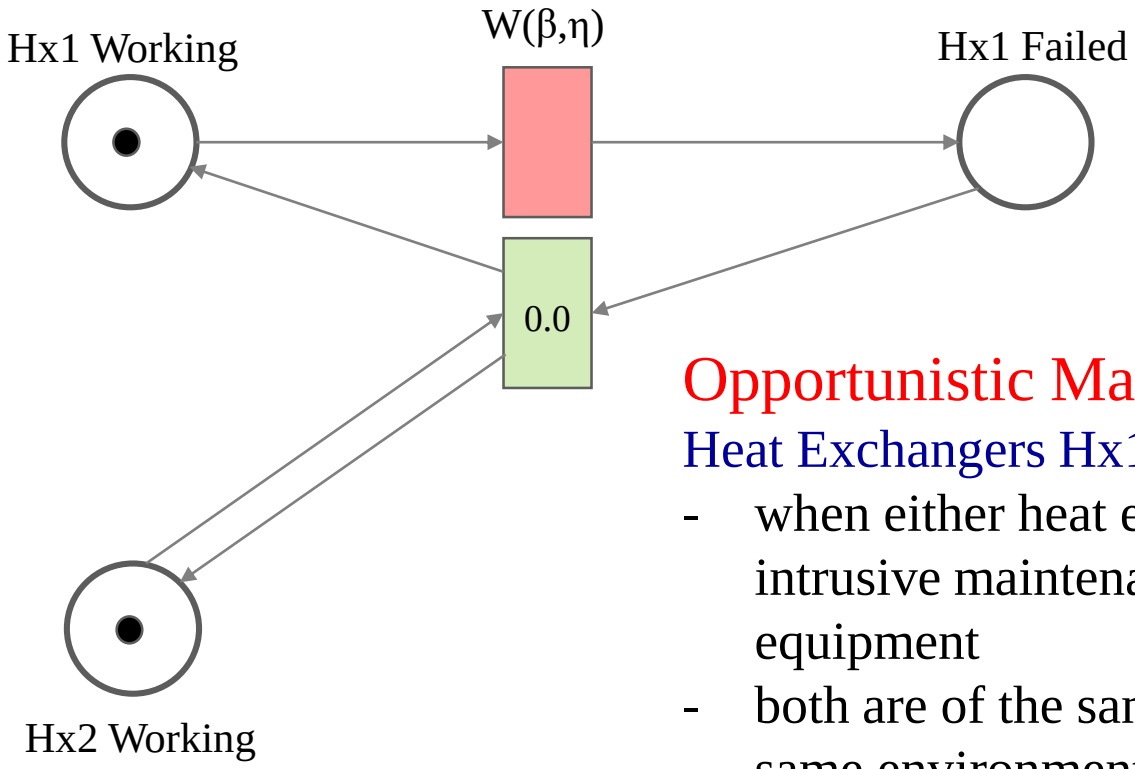
Solution

- Monte Carlo Simulation

Outputs

- Produces distributions of:
  - duration in any state
  - no of incidences of entering any state

Dependency Example

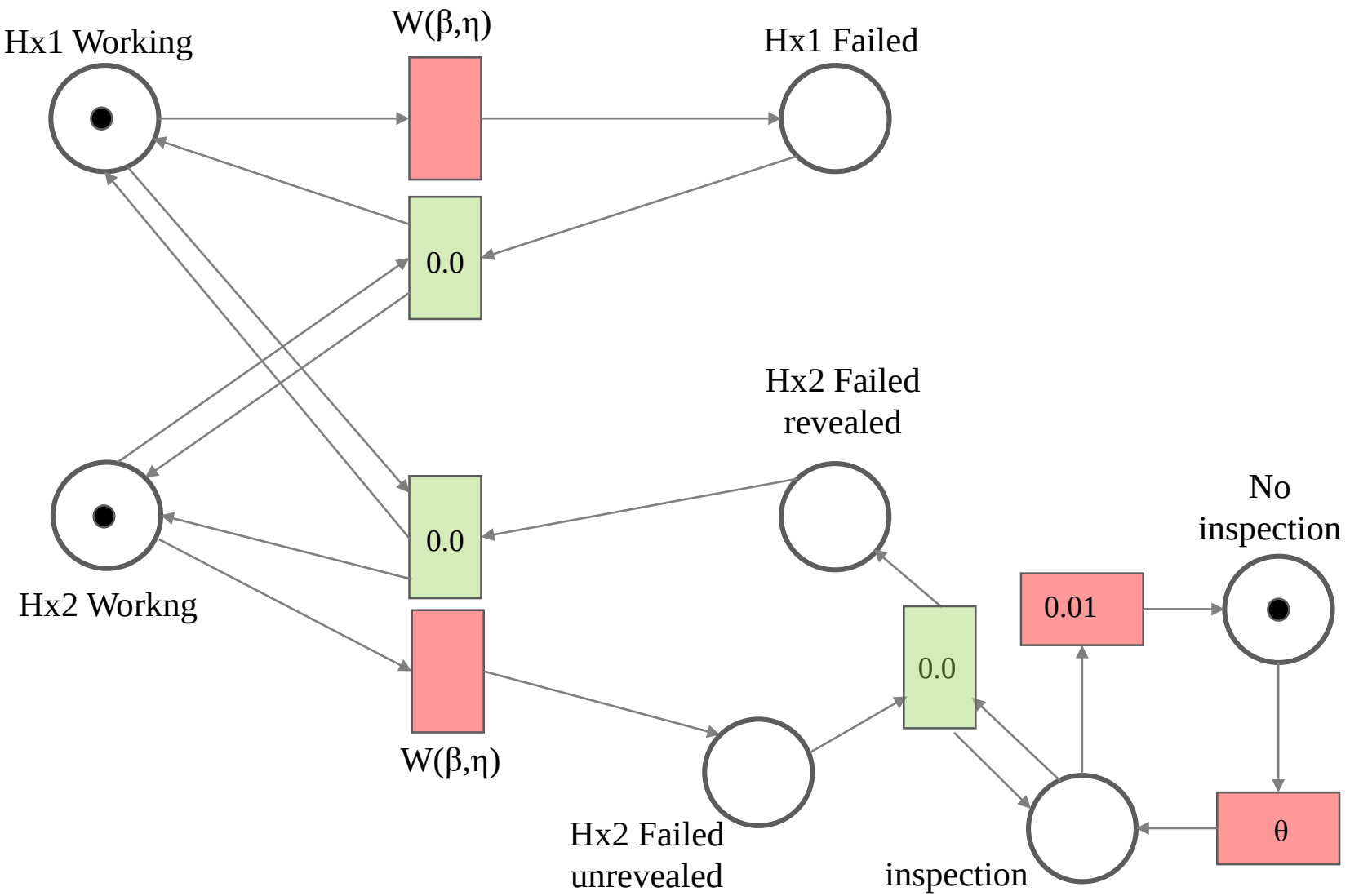


Opportunistic Maintenance Dependency

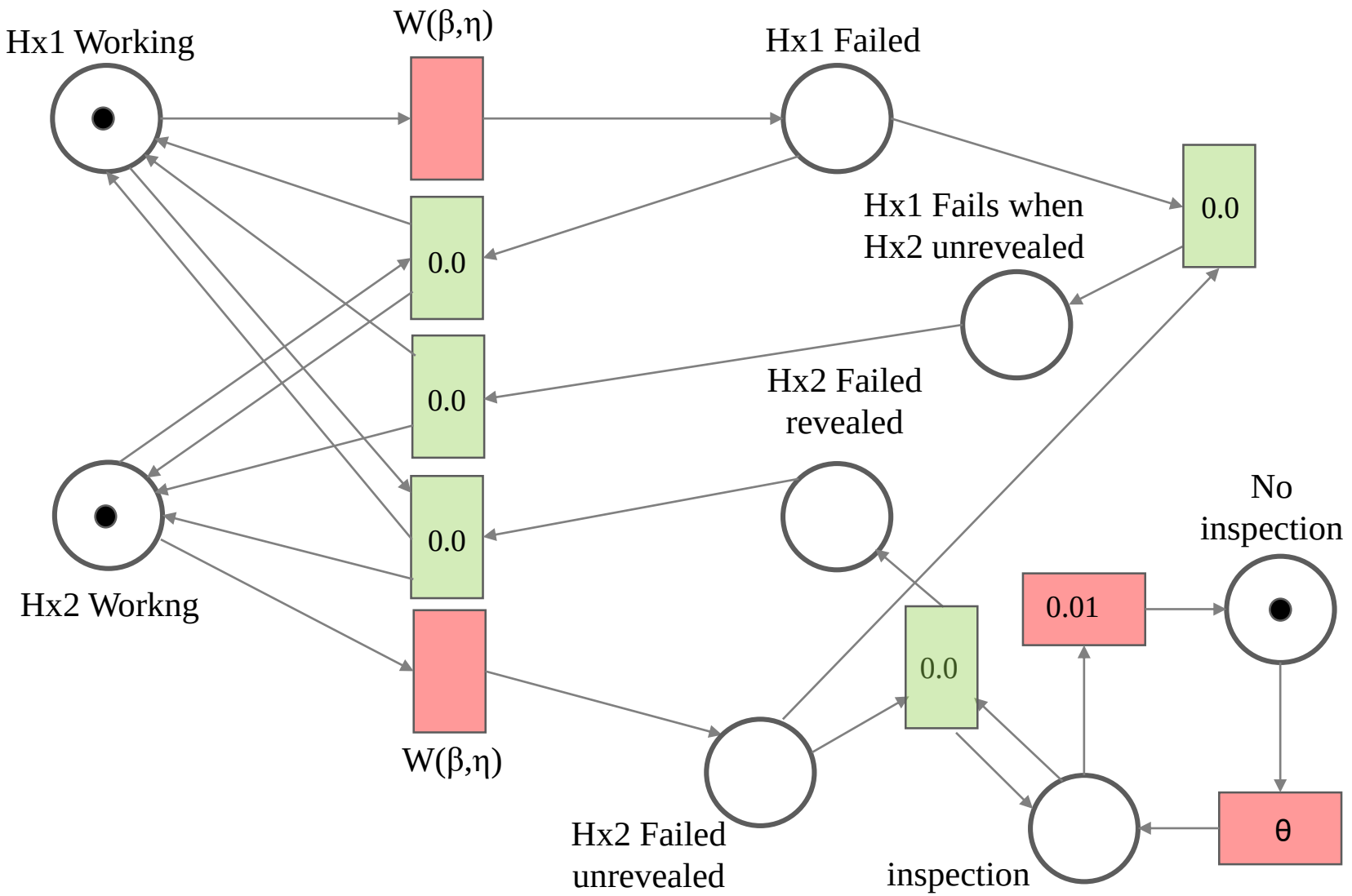
Heat Exchangers Hx1 & Hx2

- when either heat exchanger fails it needs intrusive maintenance requiring specialist equipment
- both are of the same age and operate in the same environment
- the second will fail in the not too distant future
- repair both at the same time
- Hx1 – initiator, Hx2 - enabler

Dependency Example



Dependency Example

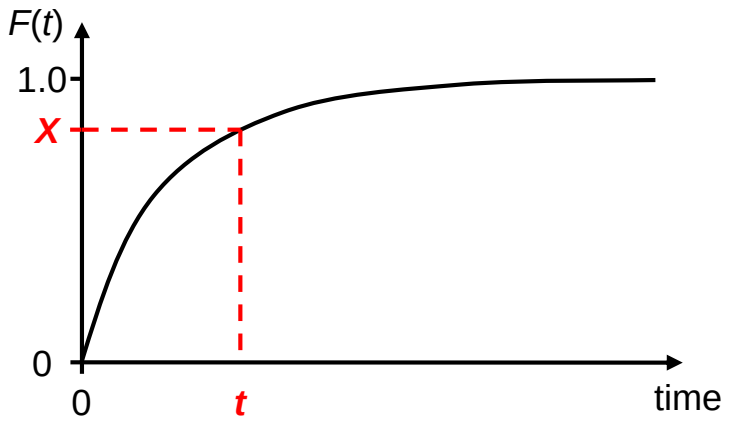


Petri Net Analysis - Simulation

Monte Carlo Simulation

Sampling from Distributions

- Inverse Transform Technique
- $F(t)$ has the same range and properties as the $U(0,1)$ distribution
- $U(0,1)$ can be generated by Random Numbers (X).



Exponential Distribution

$$f(t) = \frac{1}{\mu} e^{-\frac{t}{\mu}}$$

μ – mean time to failure

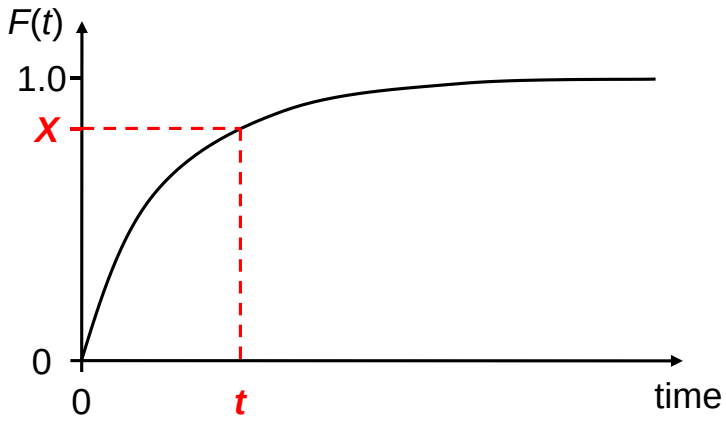
$$F(t) = \int_0^t f(u)du = 1 - e^{-\frac{t}{\mu}}$$

Generate a random, X

$$F(t) = X = 1 - e^{-\frac{t}{\mu}}$$

$$t = -\mu \ln(1 - X)$$

$$t = -\mu \ln(X)$$



Weibull Distribution

$$f(t) = \frac{\beta t^{\beta-1}}{\eta^{\beta}} e^{-\left(\frac{t}{\eta}\right)^{\beta}} \qquad F(t) = 1 - e^{-\left(\frac{t}{\eta}\right)^{\beta}}$$

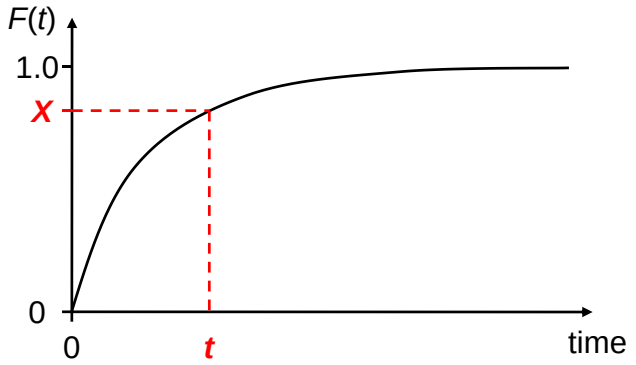
$$t \geq 0, \qquad \beta \geq 1, \qquad \eta \geq 1$$

Generate a random number, X

$$F(t) = X = 1 - e^{-\left(\frac{t}{\eta}\right)^{\beta}}$$

$$e^{-\left(\frac{t}{\eta}\right)^{\beta}} = 1 - X$$

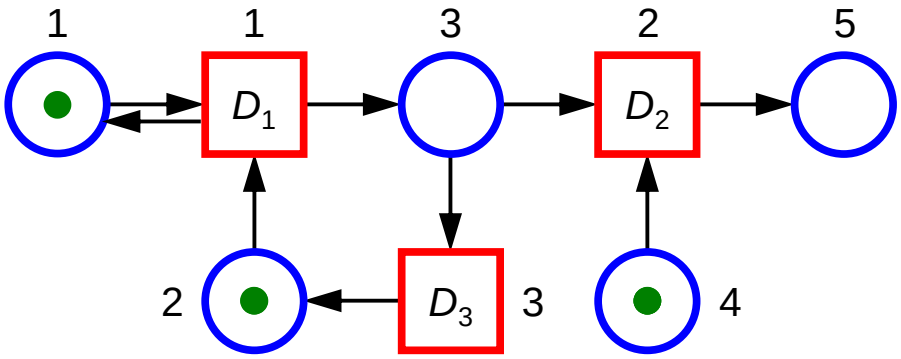
$$\left(\frac{t}{\eta}\right)^{\beta} = -ln(1 - X)$$



$$t = \eta[-ln(1 - X)\ ]^{\frac{1}{\beta}}$$

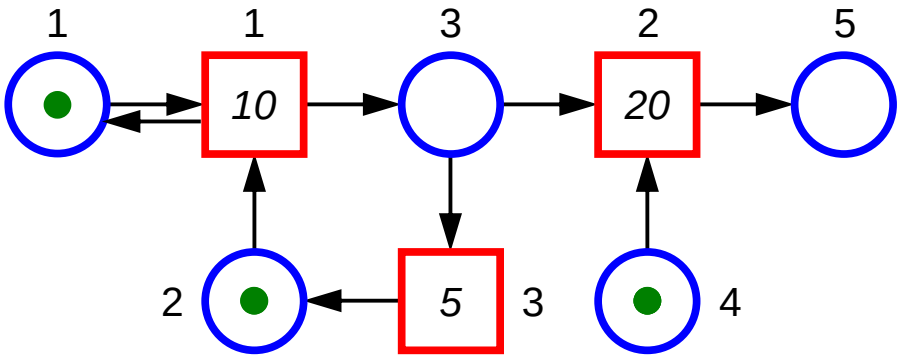
$$t = \eta[-ln(X)\ ]^{\frac{1}{\beta}}$$

Petri Net Simulation Example



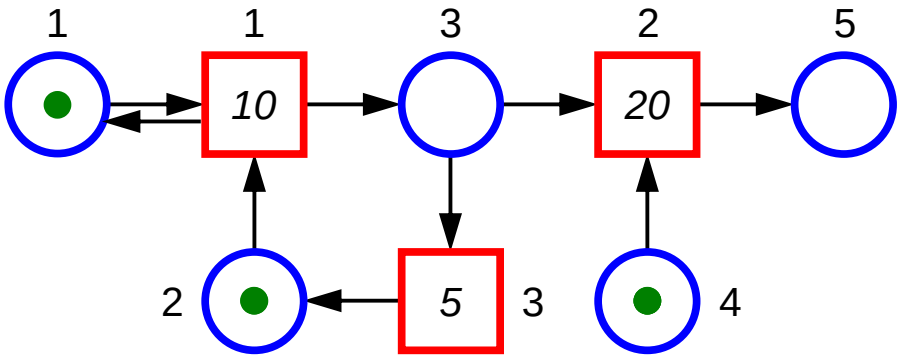
Generate random samples from
the transition distributions

Simulation



Generate random samples from
the transition distributions

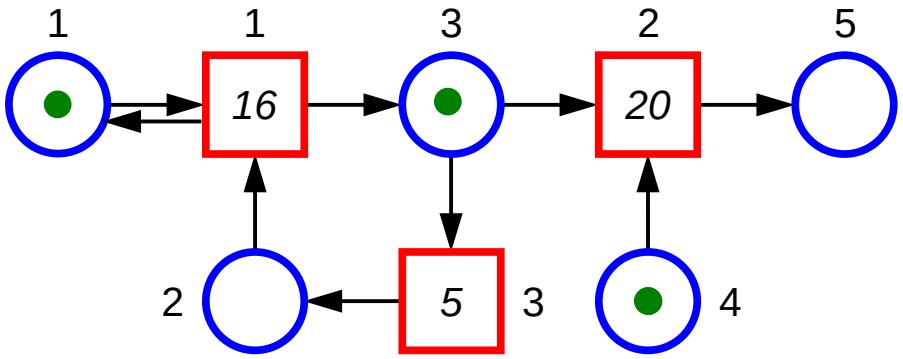
Simulation



Generate random samples from the transition distributions

Transition 1 fires at **time=10**
(Generate next random sample for **t1 = 16**)

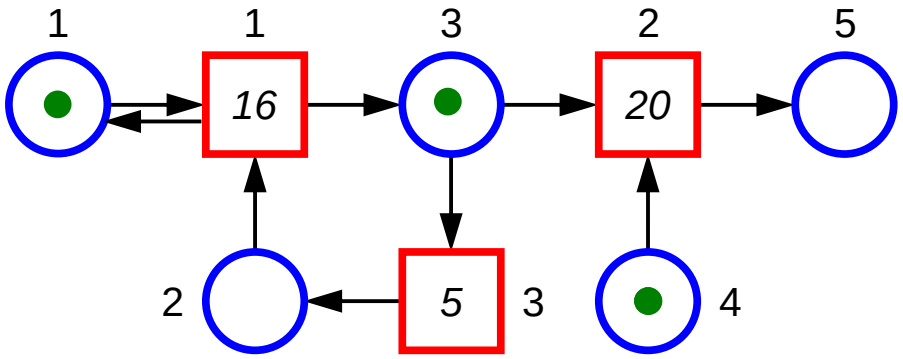
Simulation



Generate random samples from the transition distributions

Transition 1 fires at **time=10**
(Generate next random sample for **t1 = 16**)

Simulation

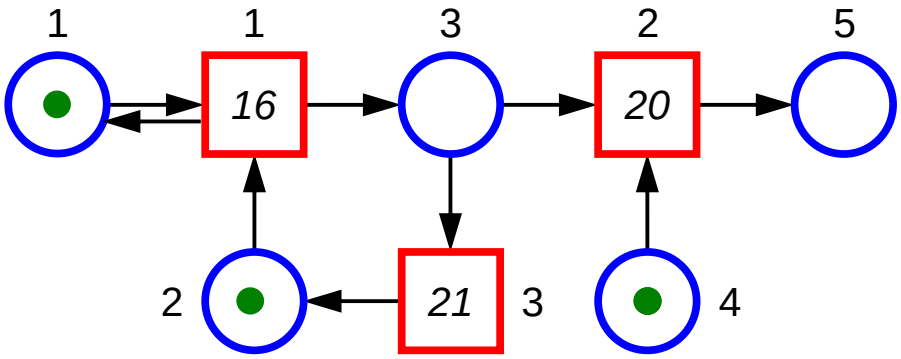


Generate random samples from the transition distributions

Transition 1 fires at **time=10**
(Generate next random sample for t1 = 16)

Transition t2 and t3 enabled.
t3 fires at **time = 15 (10+5)**
Generate next random sample for **t3=21**

Simulation

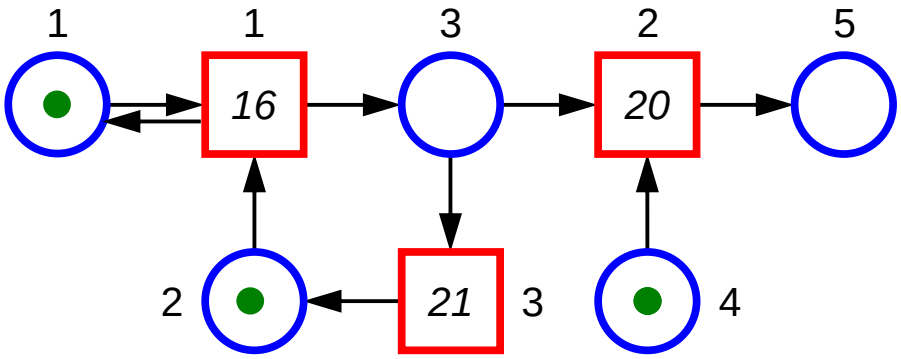


Generate random samples from the transition distributions

Transition 1 fires at **time=10**
(Generate next random sample for t1 = 16)

Transition t2 and t3 enabled.
t3 fires at **time = 15**
Generate next random sample for t3=21

Simulation



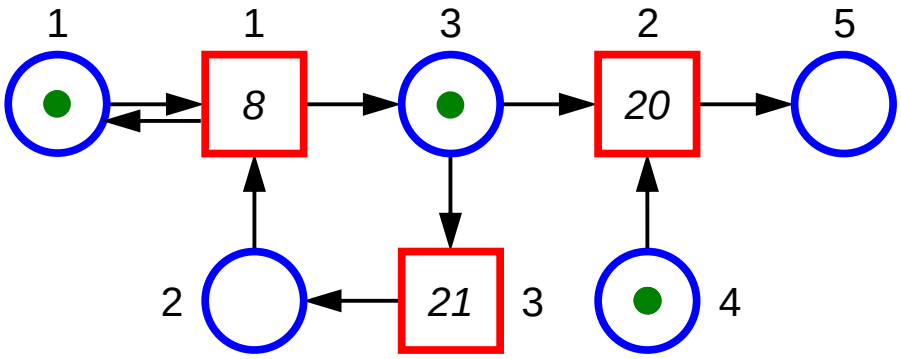
Generate random samples from the transition distributions

Transition 1 fires at **time=10**
(Generate next random sample for t1 = 16)

Transition t2 and t3 enabled.
t3 fires at **time = 15**
Generate next random sample for t3=21

Transition t1 enabled and fires at **time = 31 (15+16)**
Generate next random sample **for t1 = 8**

Simulation



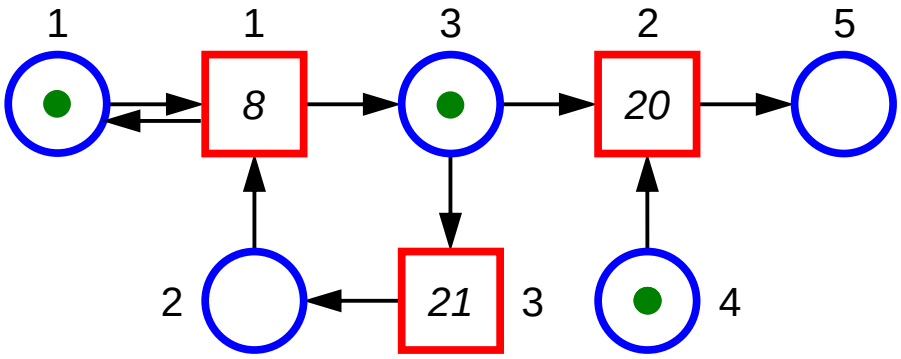
Generate random samples from the transition distributions

Transition 1 fires at **time=10**
(Generate next random sample for t1 = 16)

Transition t2 and t3 enabled.
t3 fires at **time = 15**
Generate next random sample for t3=21

Transition t1 enabled and fires at
time = 31
Generate next random sample
for t1 = 8

Simulation



Generate random samples from the transition distributions

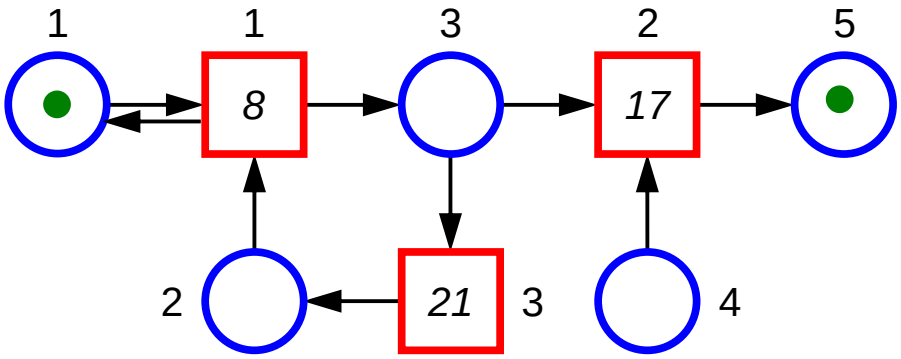
Transition 1 fires at **time=10**
(Generate next random sample for t1 = 16)

Transition t2 and t3 enabled.
t3 fires at **time = 15**
Generate next random sample for t3=21

Transition t1 enabled and fires at **time = 31**
Generate next random sample for t1 = 8

Transitions t2 and t3 enabled.
t2 fires at **time=51 (31+20)**
Generate next random sample for **t2=17**

Simulation



Generate random samples from the transition distributions

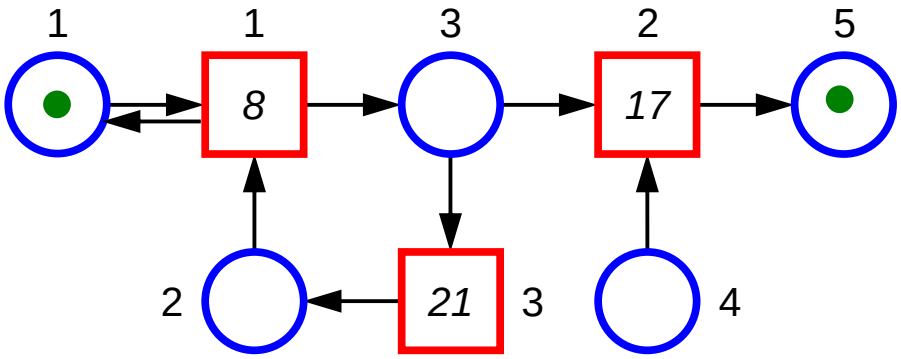
Transition 1 fires at **time=10**
(Generate next random sample for t1 = 16)

Transition t2 and t3 enabled.
t3 fires at **time = 15**
Generate next random sample for t3=21

Transition t1 enabled and fires at **time = 31**
Generate next random sample for t1 = 8

Transitions t2 and t3 enabled.
t2 fires at **time=51 (31+21**
Generate next random sample for **t2=17**

Simulation



Generate random samples from the transition distributions

Transition 1 fires at **time=10**
(Generate next random sample for t1 = 16)

Transition t2 and t3 enabled.
t3 fires at **time = 15**
Generate next random sample for t3=21

Transition t1 enabled and fires at **time = 31**
Generate next random sample for t1 = 8

Transitions t2 and t3 enabled.
t2 fires at **time=51**
Generate next random sample for t2=17

Statistics of system performance obtained by recording the time duration in each place or the number of transitions to each place

Complexity Example

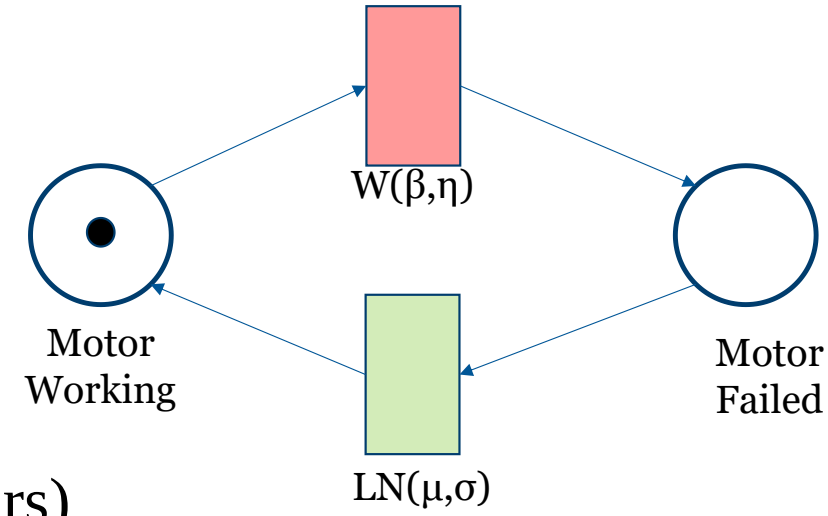
C1 = {MOTOR}

Failure time distribution

Weib(β=2.1,η=1200 hours)

Repair time distribution

LogN(μ=24.0 hours,σ=4.8 hours)



STATE	Probability	Frequency (per hour)
Motor Failed	0.0058389642	8.686868×10^{-5}

Dependency Example

State Probabilities:

$P(Hx1_W, Hx2_W)=0.98646987828725829$

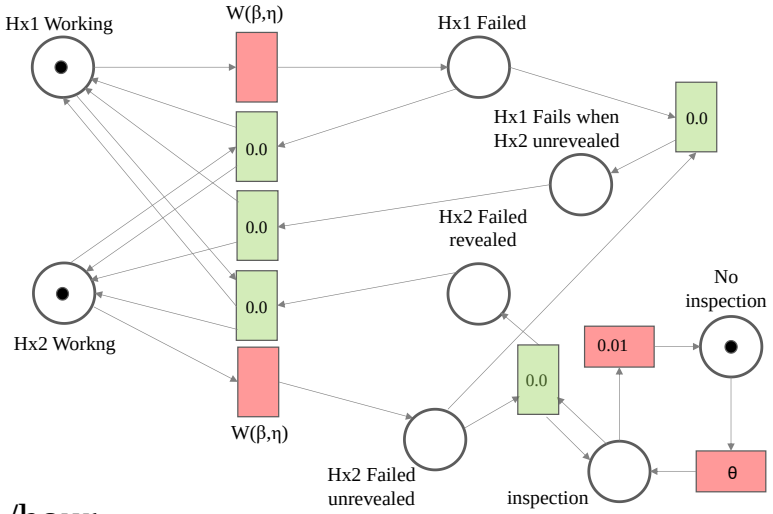
$P(Hx1_W, Hx2_F)=0.0135301$

$P(Hx1_F, Hx2_F)=0.0$

$P(Hx1_F)=0.0$

$P(Hx2_F| Hx1_F)=0.0$

$P(Hx2_F| Hx1_W)= 0.0135301$



State Failure Intensities

$w(Hx1_F, Hx2\_unrevealed)=3.1709792 \times 10^{-07}$ /hour

$w(Hx1_F, Hx2_W)=1.8161063 \times 10^{-05}$ /hour

$w(Hx1_F)=1.8478161 \times 10^{-05}$ /hour

Characteristics

Whole system modelling can be challenging:

Model Size

- Models can become large for full system analysis
  - State-space explosion for Markov models

Model Solution Times

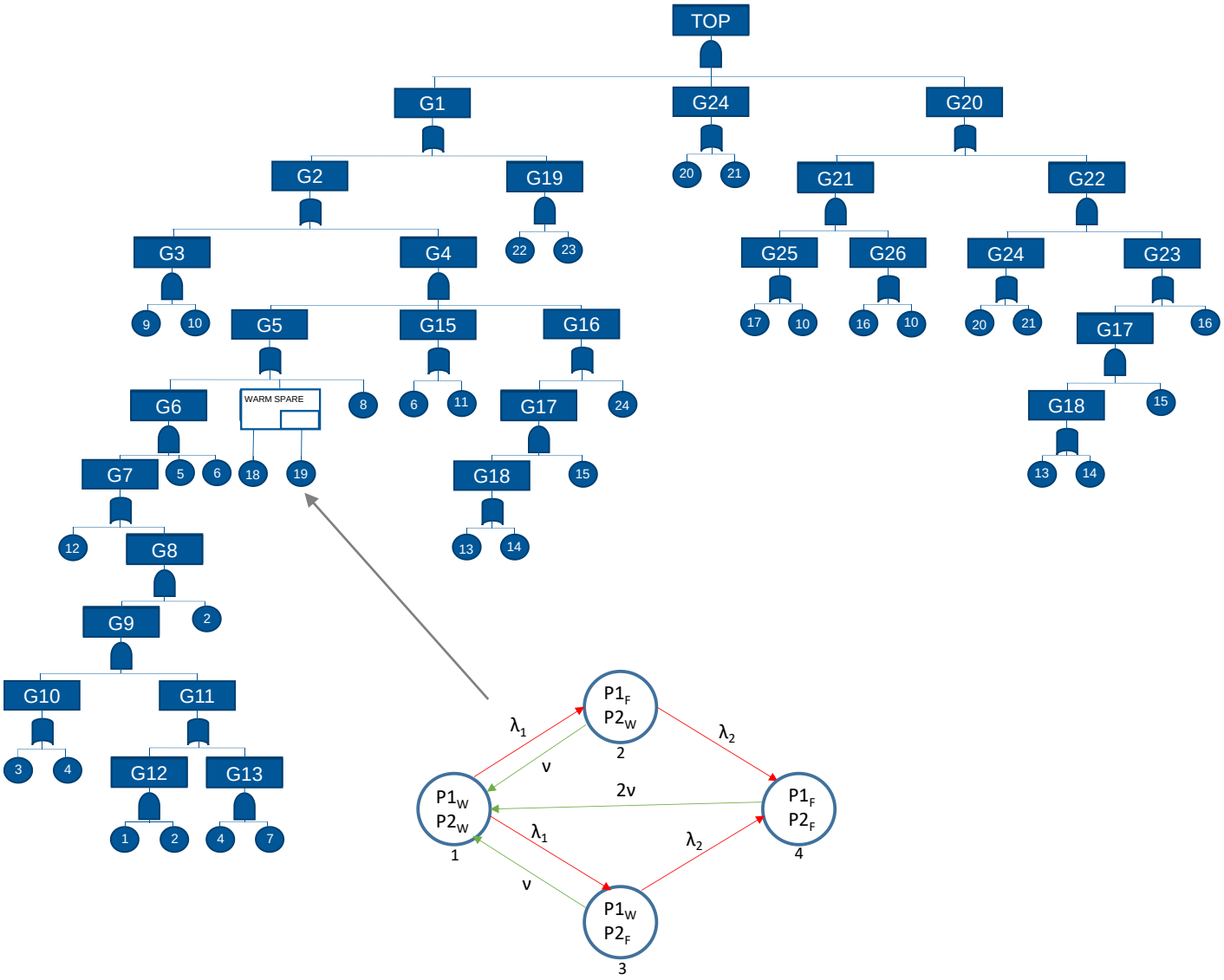
- Models solution can be computationally intensive
  - Monte Carlo Simulation analysis for Petri Nets can have long convergence times when systems are large or system failures are rare

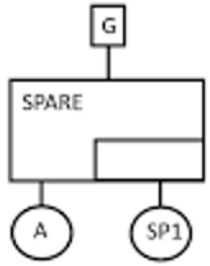
Auditability

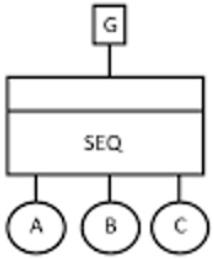
- Lack the causality structure of Fault Trees
  - Peer review and auditing difficult for regulators

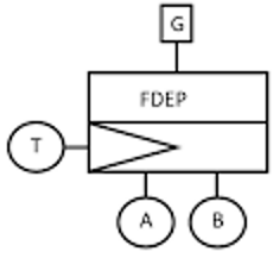
FTA Approaches to Modelling Complexities and Dependencies

Dynamic Fault Trees

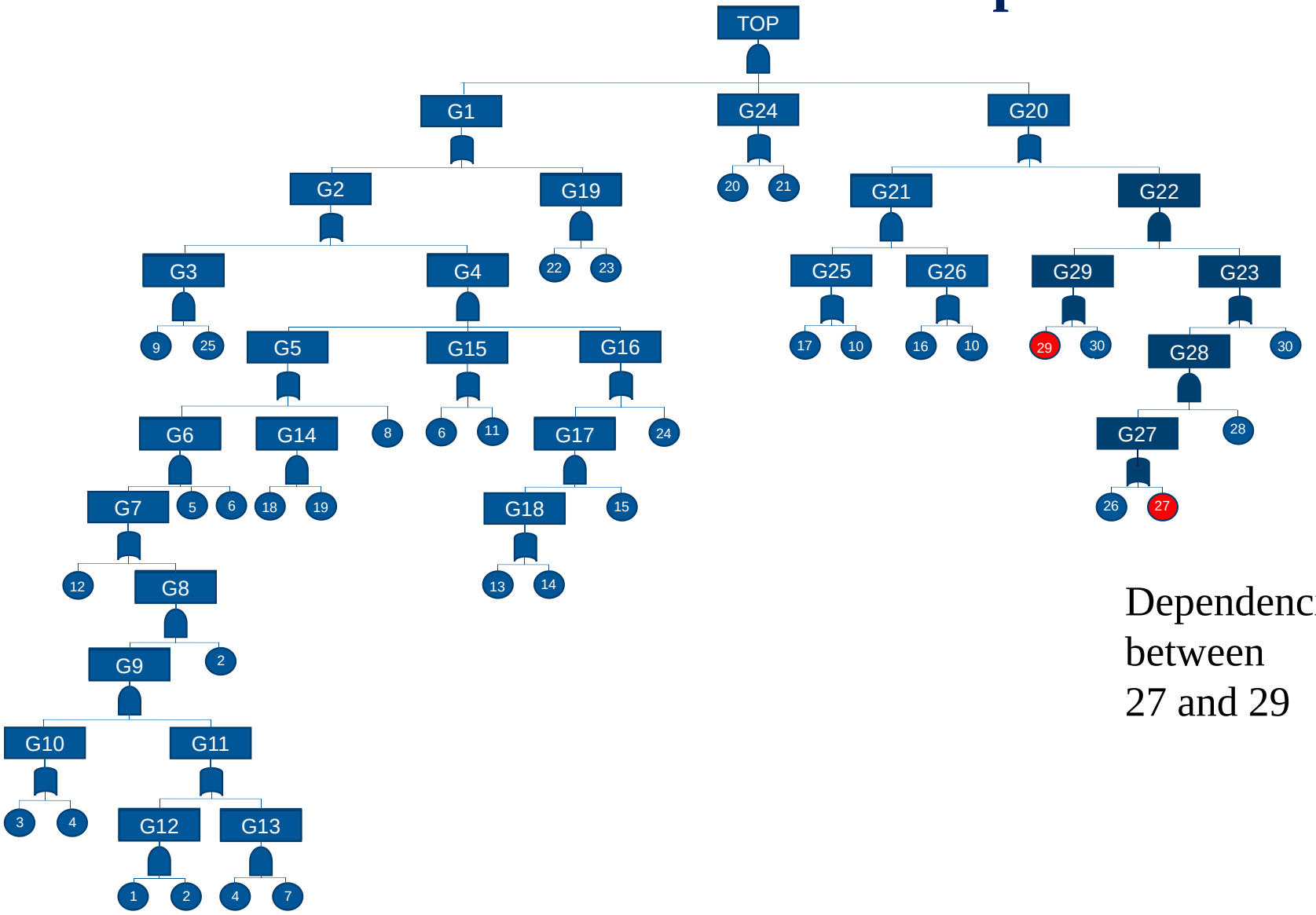




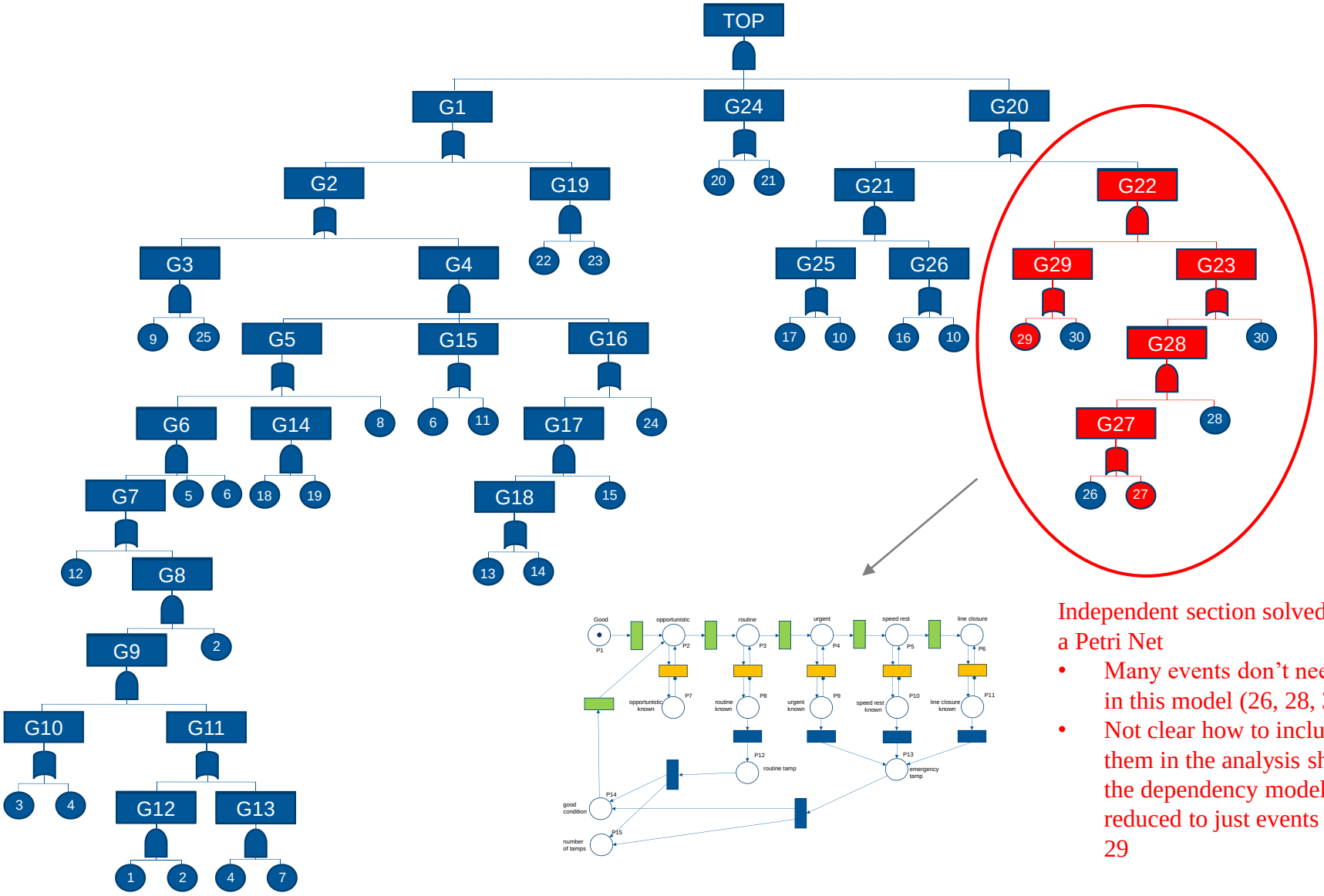




Independent Modules



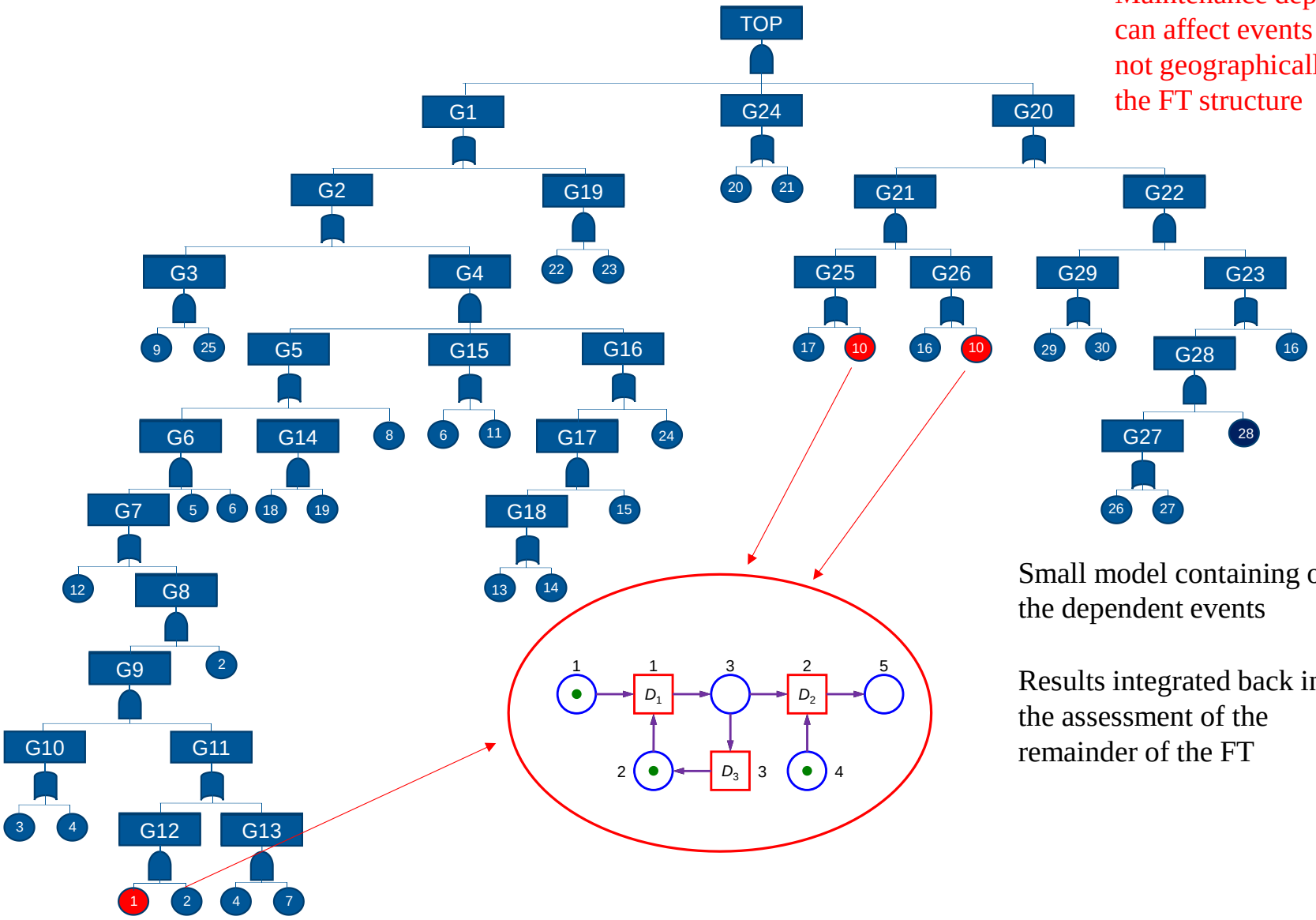
Dependencies between 27 and 29



Independent section solved using a Petri Net

- Many events don't need to be in this model (26, 28, 30)
- Not clear how to include them in the analysis should the dependency model be reduced to just events 27 and 29

Maintenance dependency's can affect events which are not geographically close in the FT structure



Small model containing only the dependent events

Results integrated back into the assessment of the remainder of the FT

Dynamic and Dependent Tree Theory
D^2T^2

Modelling Requirements

Model Requirements

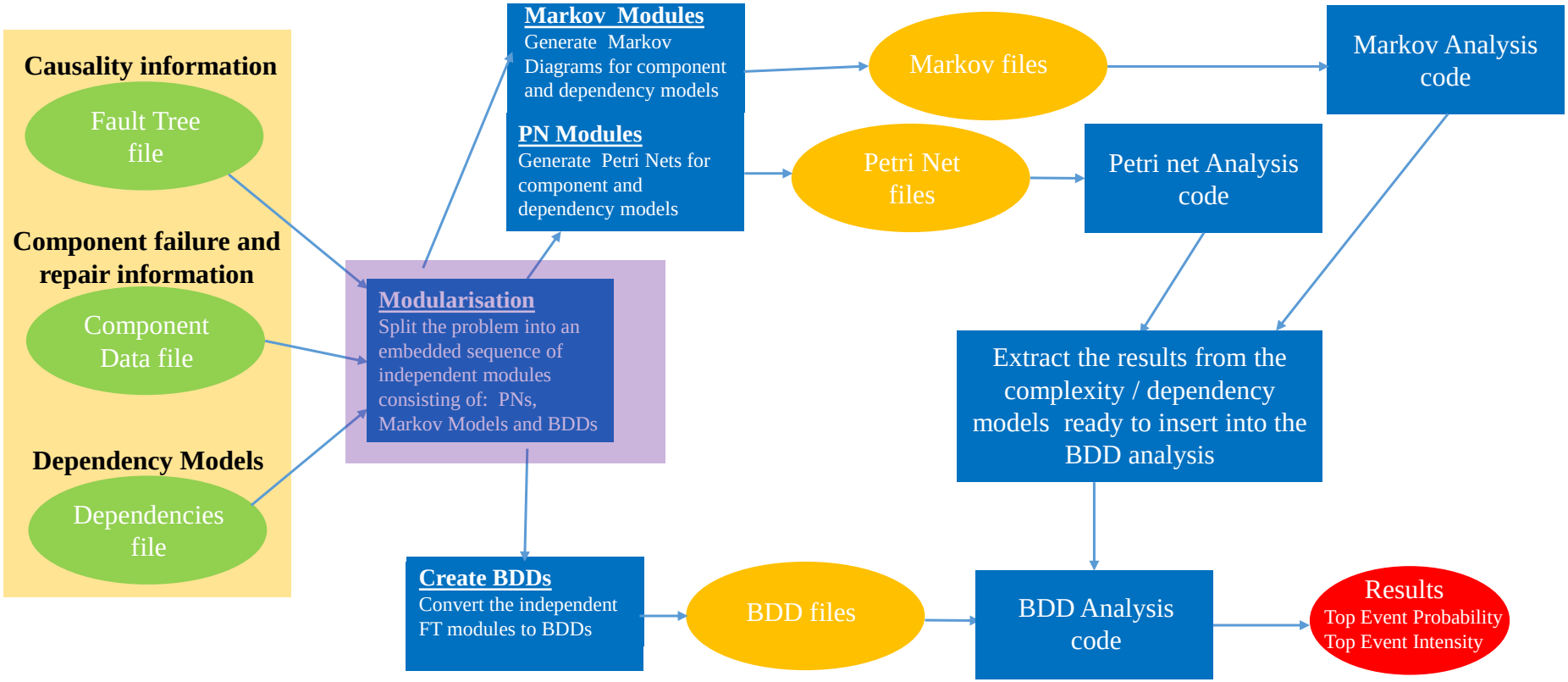
Dependencies

- Model the dependencies and complexities using Petri Nets or Markov models
  - Always use the *simplest dependency model*

Binary Decision Diagrams

- Dependencies are just required to be considered on each path
- Path numbers can be very high so every effort needs to be made to minimise these - indirectly by *minimise the size of the BDD*
  - minimise the fault tree size using an effective modularisation
  - effective variable ordering

Basic Structure of the Code



Modularisation

Faunet Methods

Repeatedly Apply

- Contraction

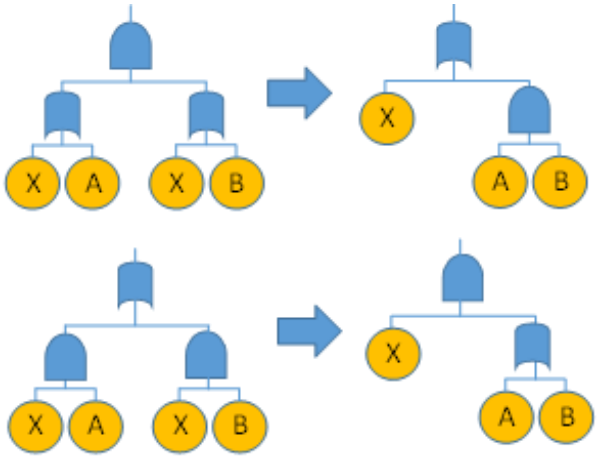
  Subsequent gates of the same type are contracted into a single gate

- Factorisation

  Extracts factors expressed as groups of events that always occur together in the same gate type. The factors can be any number of events if they satisfy the following:
  - All events in the group are independent and initiators
  - All events in the group are independent and enablers.
  - All events in the group feature a dependency and contain all events in the same dependency group.

- Extraction

  Restructure:



Quantification of Factors

For combinations formed from independent events

OR combinations, $Cf_i = x_1 + x_2 + \cdots x_n$

$$Q_{Cfi} = 1 - \prod_{j=1}^{n} \left(1 - q_{x_j}\right)$$

If the factor contains only initiating events:

$$w_{Cfi} = \sum_{j=1}^{n} w_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \left(1 - q_{x_k}\right)$$

AND combinations, $Cf_i = x_1 . x_2 . \ldots . x_n$

$$Q_{Cfi} = \prod_{j=1}^{n} q_{x_j}$$

$$w_{Cfi} = \sum_{\substack{j=1 \\ initiators}}^{n} \left(w_j \prod_{\substack{k=1 \\ k \neq j}}^{n} q_{x_k} \right)$$

Quantification of Factors

For combinations of events from a dependency group

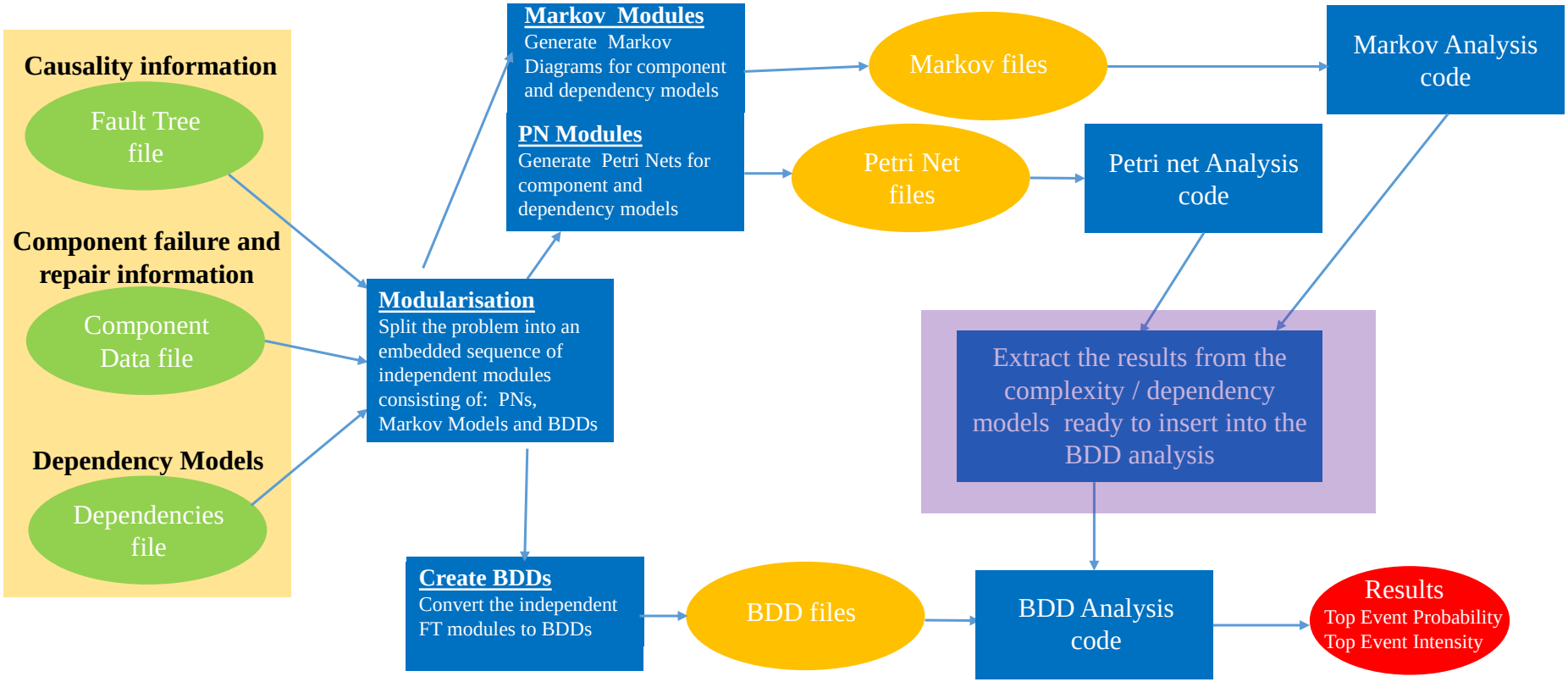
OR combinations, $Cf_i = x_1 + x_2 + \cdots x_n$

AND combinations, $Cf_i = x_1.x_2.\ldots.x_n$

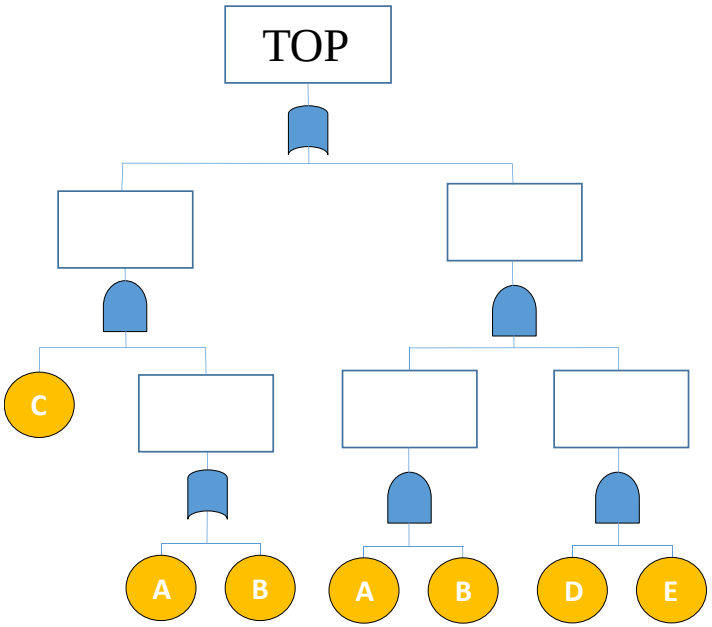
Q_{Cfi}, w_{Cfi} are extracted from the PN / Markov model

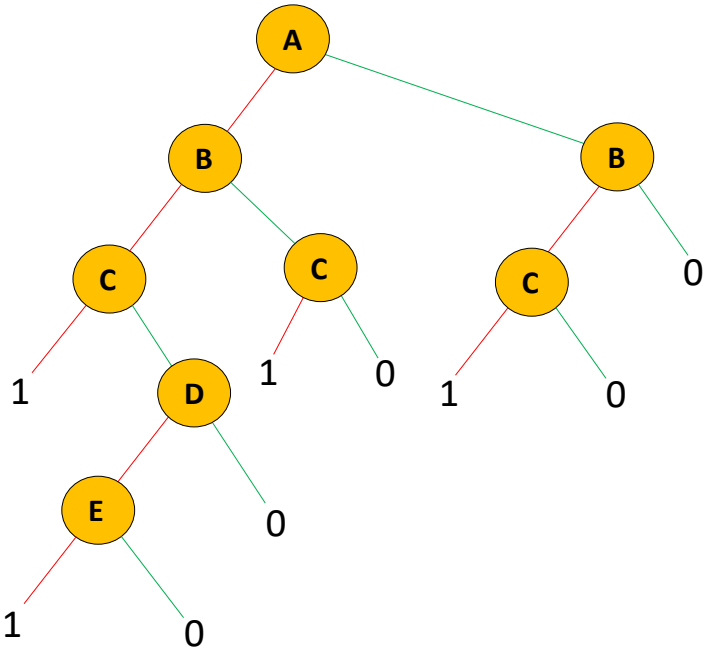
Top Event Quantification for Dependent Events

Basic Structure of the Code



Example





Dependency groups

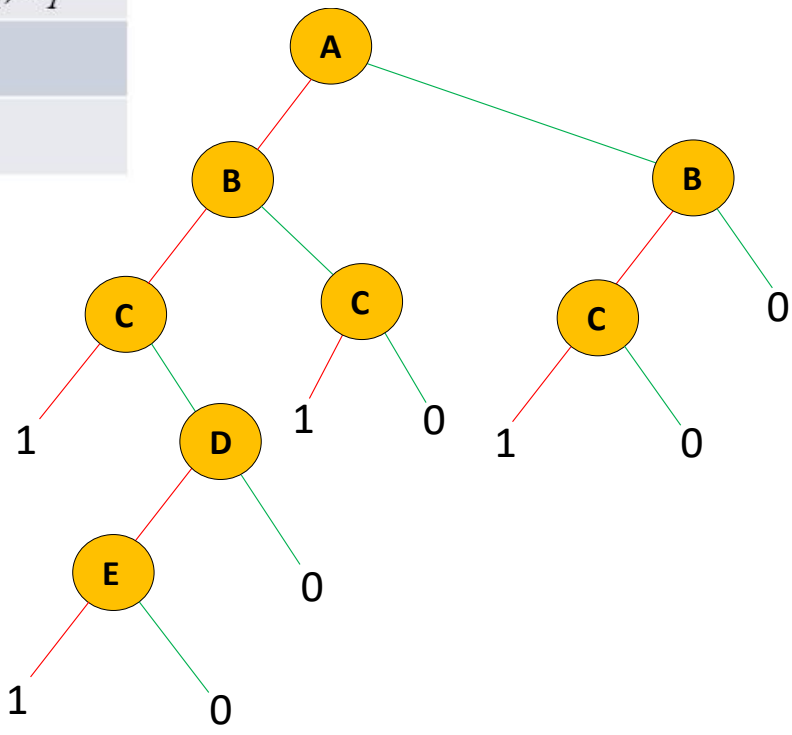
D1 = { B, C }

D2 = { D, E }

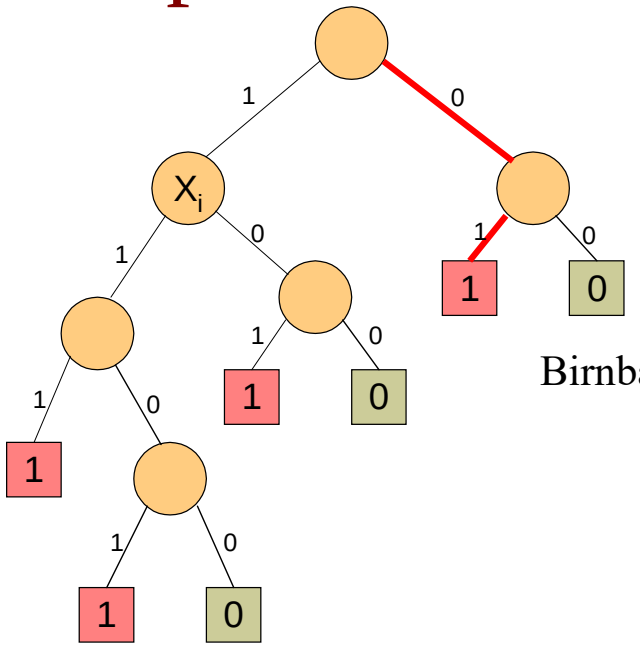
Top Event Probability

j	$path_j$	$Ipath_j$	$Dpath_j^1$	$Dpath_j^2$
1	a_1, b_1, c_1	a_1	b_1, c_1	
2	a_1, b_1, c_0, d_1, e_1	a_1	b_1, c_0	d_1, e_1
3	a_1, b_0, c_1	a_1	b_0, c_1	
4	a_0, b_1, c_1	a_0	b_1, c_1	

$$Q_{SYS} = \sum_{j=0}^{npath} \left[P(Ipath_j) . \prod_{k=1}^{ndep} P(Dpath_j^k) \right]$$



Top Event Intensity



$$w_{SYS}(t) = \sum_{\substack{i \\ initiators}} G_i(\boldsymbol{q}).w_i\,(t)$$

Birnbaum's Measure of Importance / Criticality Function

$$G_i(\boldsymbol{q}) = Q_{SYS}(1_i, \boldsymbol{q}) - Q_{SYS}(0_i, \boldsymbol{q})$$

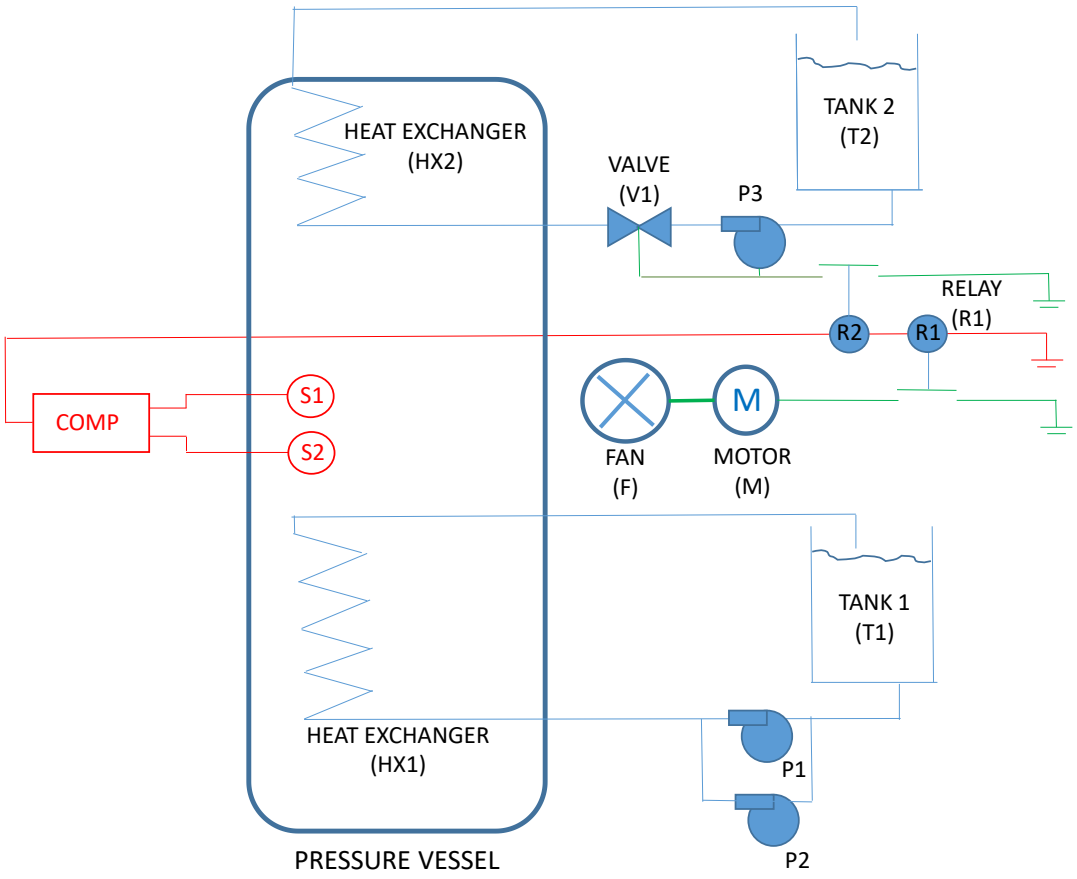
$$Q_{SYS}\left(1_i, \underline{q}\right) = \sum_{x_{i_1} \in path_j} P\left(path_j - x_{i_1}\right) + \sum_{x_i \notin path_j} P(path_j | x_i = 1)$$

$$Q_{SYS}\left(0_i, \underline{q}\right) = \sum_{x_{i_0} \in path_j} P\left(path_j - x_{i_0}\right) + \sum_{x_i \notin path_j} P(path_j | x_i = 0)$$

Case Study Example

Plant Cooling System

Plant Cooling System - Features



P1, P2, P3 and M – common power supply PoW

Sub-Systems

Primary Cooling Water System

- Tank (T1), Pumps (P1,P2), Heat Exchanger (Hx1), Power Supply (PoW)

Detection System

- Sensors (S1,S2), Computer (Comp)

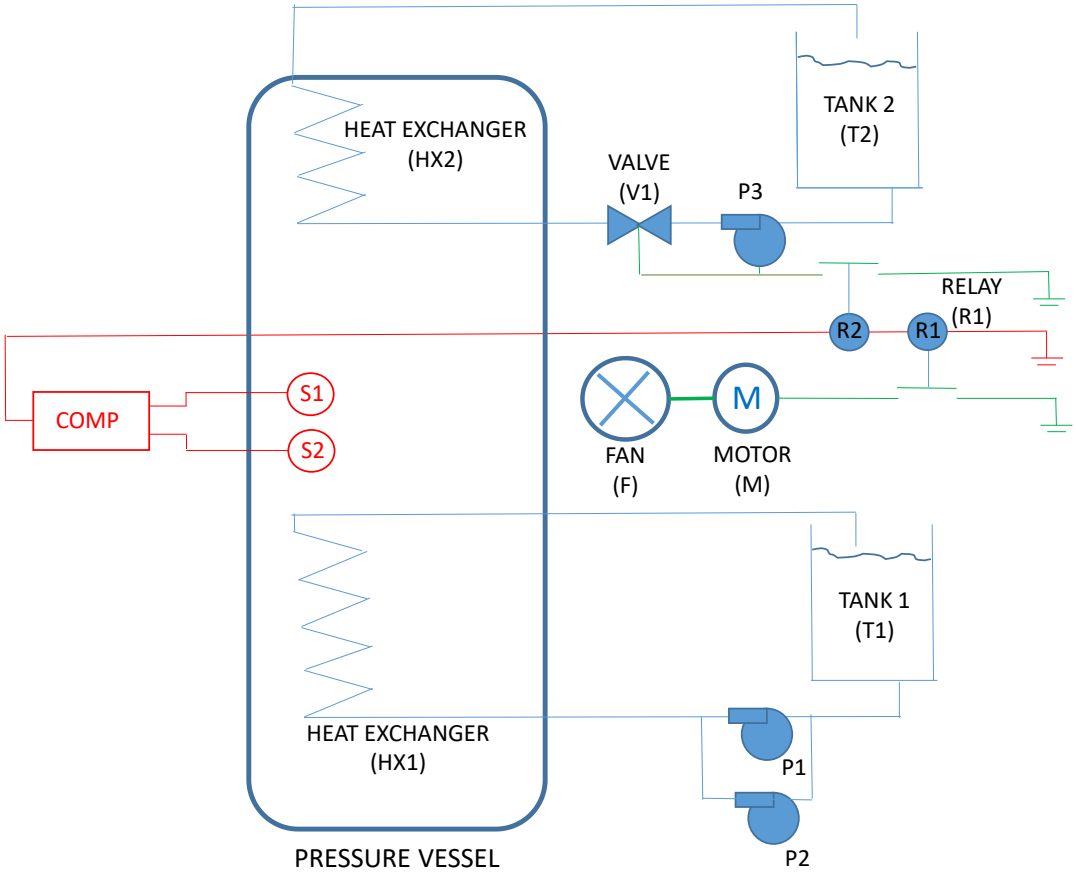
Secondary Cooling Water System

- Tank(T2), Pump (P3), Heat Exchanger (Hx2), Valve (V1), Relay (R2), Power Supply (PoW)

Secondary Cooling Fan System

- Fan (F), Motor (M), Relay (R1)

Plant Cooling System - Features



Complex Features

Non-constant failure / repair rates

- Motor M - Weibull failure time distribution and a lognormal repair time distribution

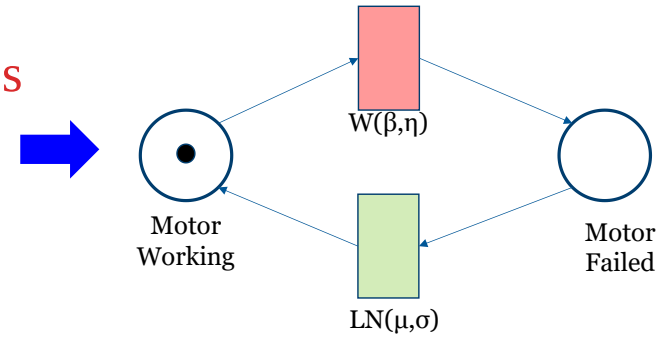
Dependencies

- Pumps P1 & P2 – if one fails it puts increased load (and increases the failure rate) of the other
- Heat Exchangers Hx1 & Hx2 - when one needs replacement – needs specialist equipment and both are replaced
- Pump P3 - two events P3S and P3R are clearly dependent

Complexity and Dependency Models

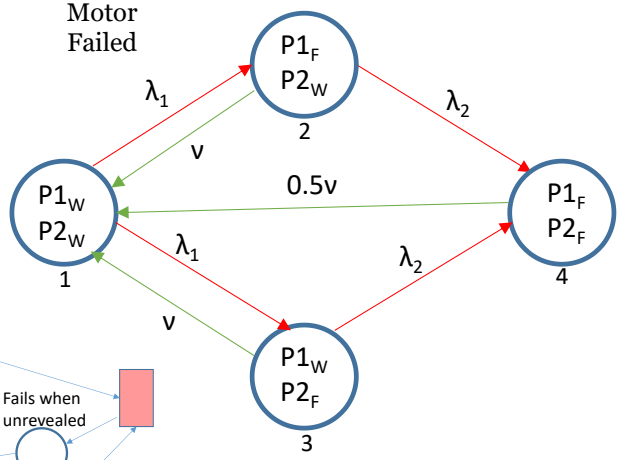
Non-constant failure / repair rates

- Motor M - Weibull failure time distribution and a lognormal repair time distribution

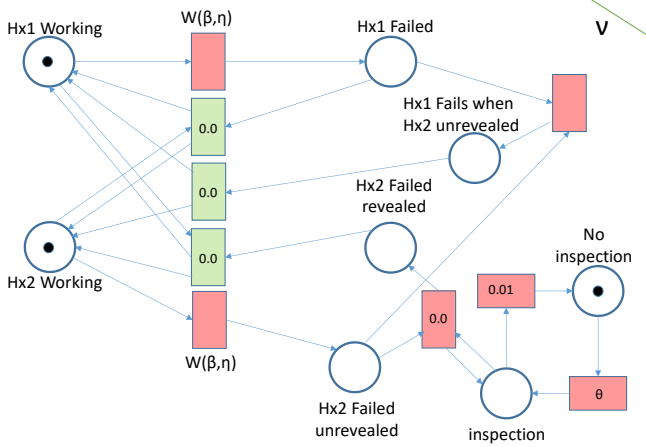


Dependencies

- Pumps P1 & P2 – if one fails it puts increased load (and increases the failure rate) of the other



- Heat Exchangers Hx1 & Hx2 - when one needs replacement – needs specialist equipment and both are replaced



- Pump P3 - two events P3S and P3R are clearly dependent

$$q_{P3} = q_{P3S} + (1.0 - q_{P3S})\lambda_{P3R} t_{period}$$

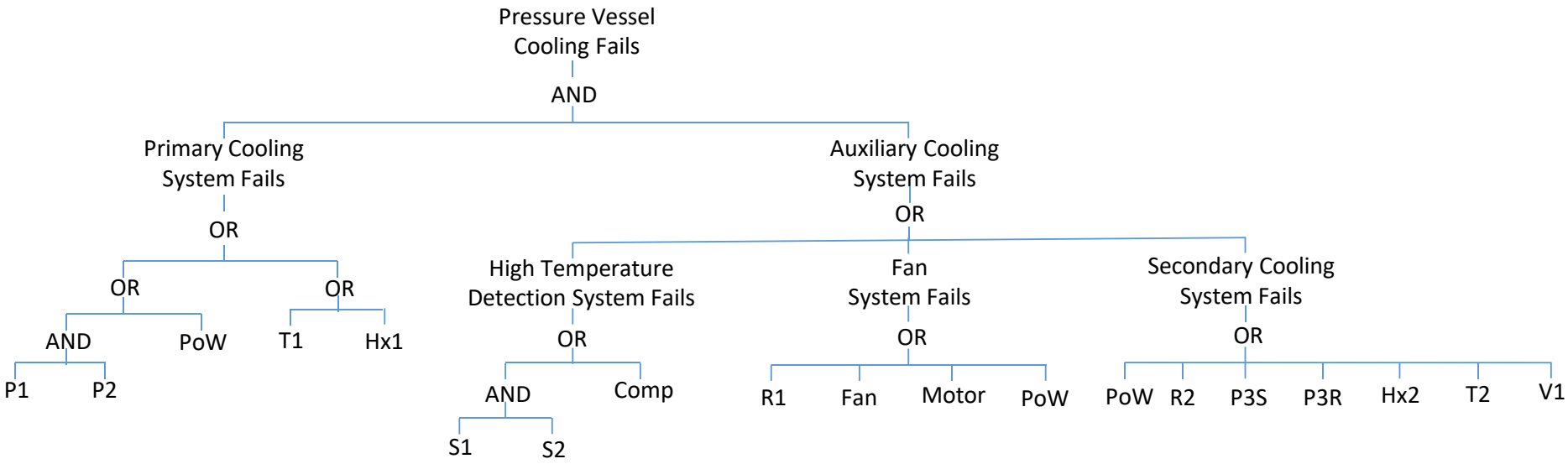
$$= 0.05 + 0.095 \times 10^{-4} \times 30$$

$$= 0.05285$$

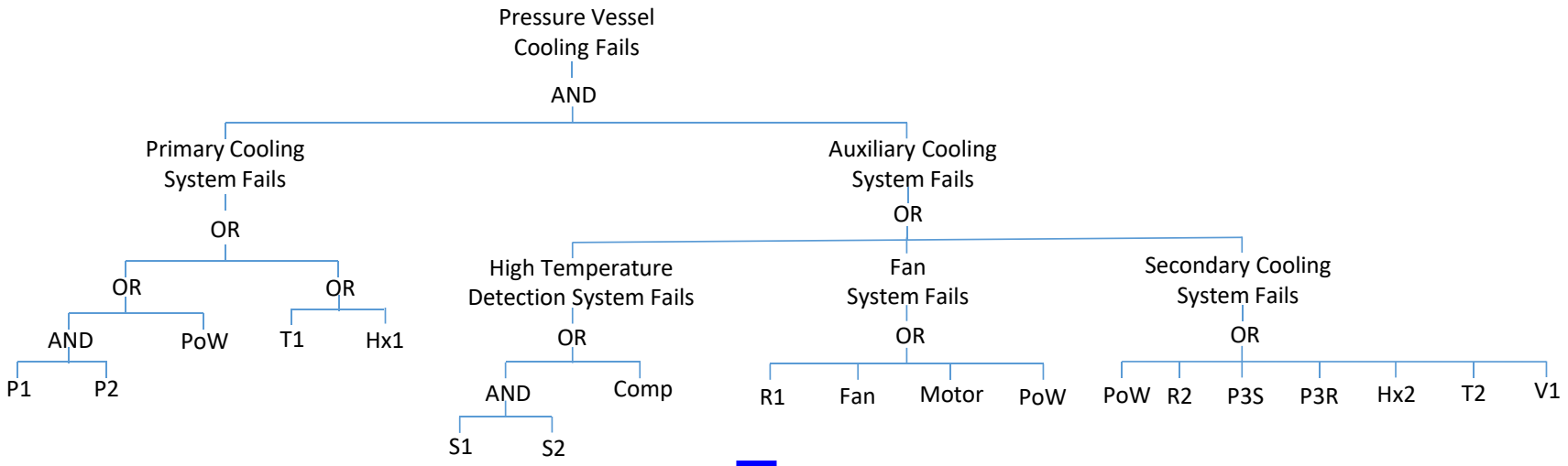
Component Data

Event Code	Description	I / E	D-Group	Failure rate (/hour)	Mean time to repair (hours)	Inspect interval (hours)	q	w
P1, P2	Pumps fail when running	I	D1	Failure rate $\lambda_1 = 2 \times 10^{-5}$ /h under normal load $\lambda_2 = 5 \times 10^{-3}$/h under full load Repair rate $\nu = 0.041667$ (MTTF = 24hrs)				
T1	Water Supply failure	I		$1x10^{-5}$	24		$2.4x10^{-5}$	$9.99976 x10^{-6}$
Hx1	Heat Exchanger fails	I	D2	Failure time = W(β=2.5, η=30,000h) The system is shut down when the repair is undertaken				
PoW	Power supply failure	I		$1x10^{-4}$	10		$1x10^{-3}$	$9.99 x10^{-5}$
S1, S2	Sensor fails to detect a high temperature	E		$5x10^{-4}$	5	730	0.185	
Comp	Computer fails to process sensor signals	E		$5x10^{-5}$	5	2190	0.055	
R1 / R2	Relay contacts fail to close	E		$1x10^{-5}$	24	2190	0.0112	
Fan	Fan fails	E		$2x10^{-6}$	8	2190	$2.206 x10^{-3}$	
Motor	Fan motor fails	E	C1	Failure time = W(β=1.5, η=12,000h) Repair time = LogN(μ=24hrs, σ=4.8h)				
P3S	Pump fails to activate	E	D3				0.05	
P3R	Pump fails when running	E	D3	$1x10^{-4}$				
T2	Water Supply failure	E		$1x10^{-5}$	24	2190	0.0112	
Hx2	Heat Exchanger fails	E	D2	Failure time = W(β=2.5, η=30,000h) The system is shut down when the repair is undertaken				
V1	Valve fails to open	E		$5x10^{-5}$	30	2190	0.05625	

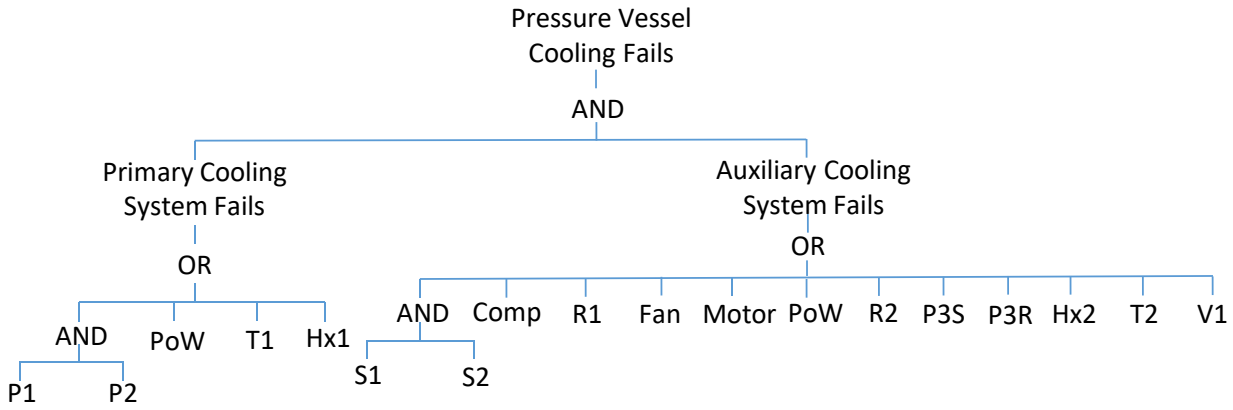
Fault Tree Structure



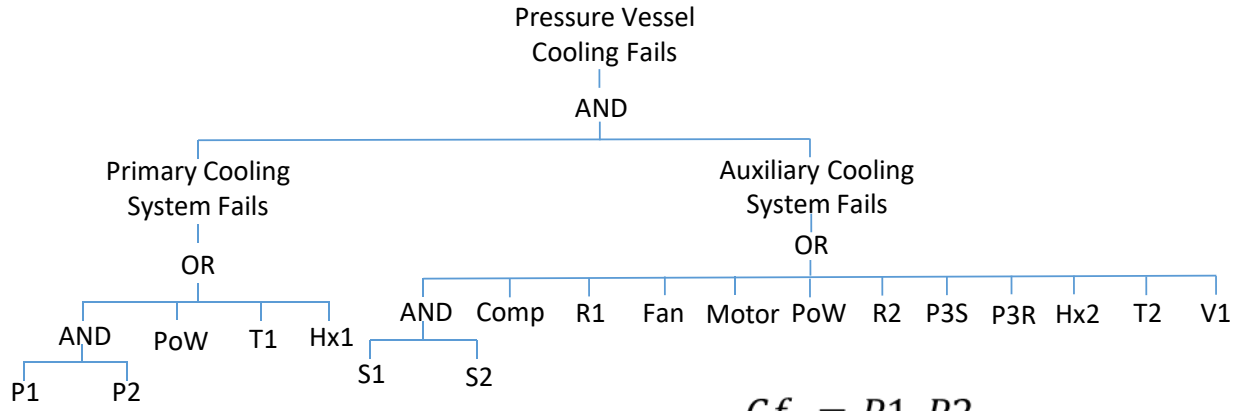
Modularisation (1)



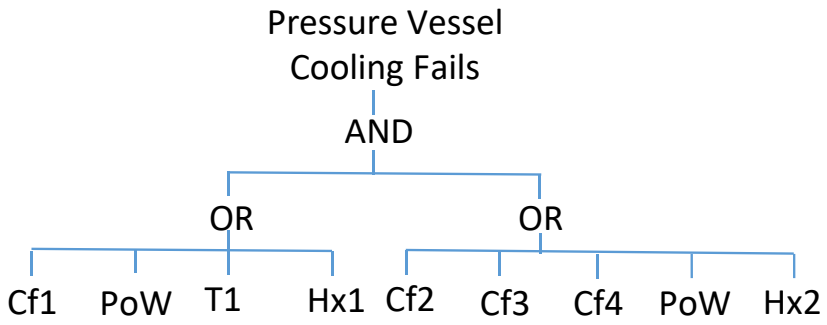
Contraction 1



Modularisation (2)



Factorisation 1



$$Cf_1 = P1.P2$$

(dependency group D1 – initiators)

$$Cf_2 = S1.S2$$

(independent enablers)

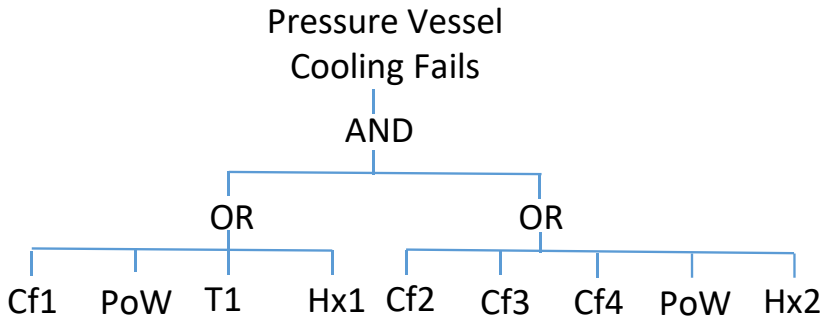
$$Cf_3 = Comp + R1 + Fan + Motor + R2 + T2 + V1$$

(independent enablers)

$$Cf_4 = P3S + P3R$$

(dependency group D3 – enablers)

Modularisation (3)



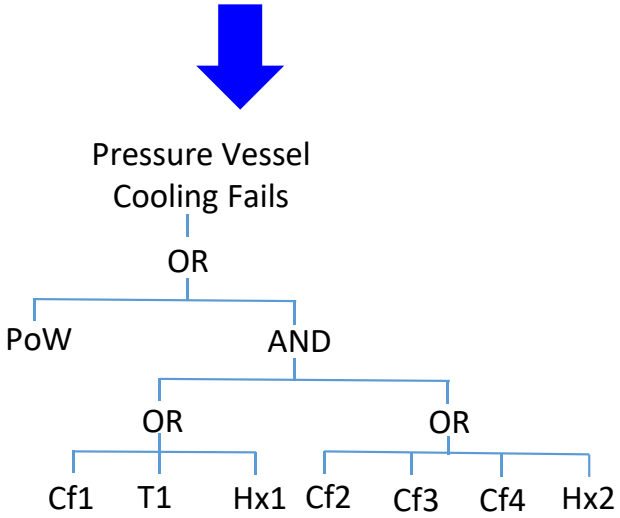
$$Cf_1 = P1.P2$$

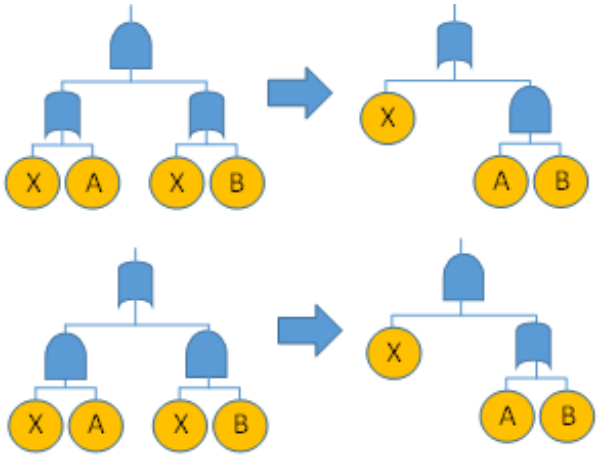
$$Cf_2 = S1.S2$$

$$Cf_3 = Comp + R1 + Fan + Motor + R2 + T2 + V1$$

$$Cf_4 = P3S + P3R$$

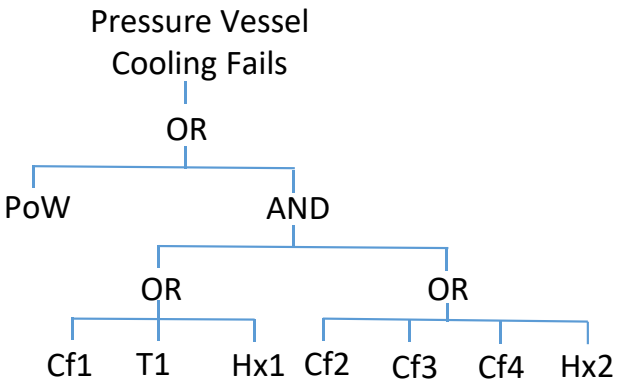
Extraction 1





Contraction 2 - No Change

Modularisation (4)



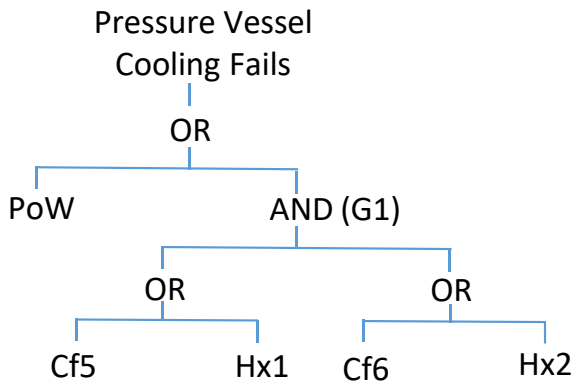
$$Cf_1 = P1.P2$$

$$Cf_2 = S1.S2$$

$$Cf_3 = Comp + R1 + Fan + Motor + R2 + T2 + V1$$

$$Cf_4 = P3S + P3R$$

Factorisation 2

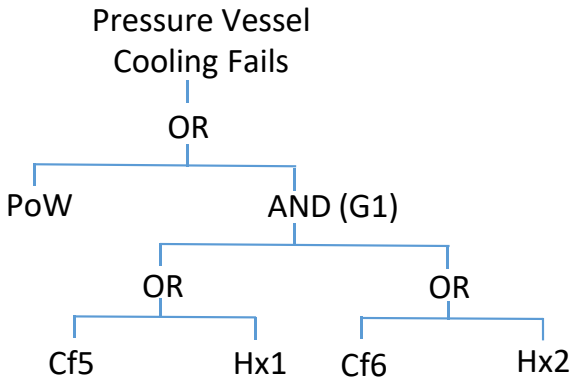


$$Cf_5 = Cf_1 + T1$$

$$Cf_6 = Cf_2 + Cf_3 + Cf_4$$

Simplest possible Faunet representation

Modularisation (5)



$$Cf_1 = P1.P2$$

$$Cf_2 = S1.S2$$

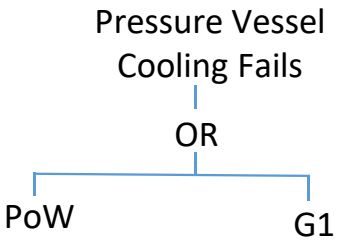
$$Cf_3 = Comp + R1 + Fan + Motor + R2 + T2 + V1$$

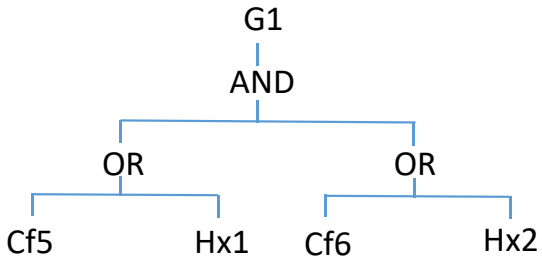
$$Cf_4 = P3S + P3R$$

$$Cf_5 = Cf_1 + T1$$

$$Cf_6 = Cf_2 + Cf_3 + Cf_4$$

Applying the Rauzy & Dutuit algorithm gives independent section Top and G1





$$Cf_7 = PoW + G1$$

Top Event Probability

Event Code	Description	I / E	D-Group	q
Cf1	P1. P2	I	D1	0.00170988
Cf2	S1. S2	E		0.00034225
Cf3	Comp + R1 + Fan + Motor + R2 + T2 + V1	E		0.6035094
Cf4	P3S + P3R	E	D3	0.05285
Cf5	Cf_1 + T1	E		0.0017338
Cf6	Cf_2 + Cf_3 + Cf_4	E		0.6246519
G1	BDD	I		
Cf7	PoW + G1			

From dependency model

Direct from component results

From factor calculations

Note - Motor from complexity model

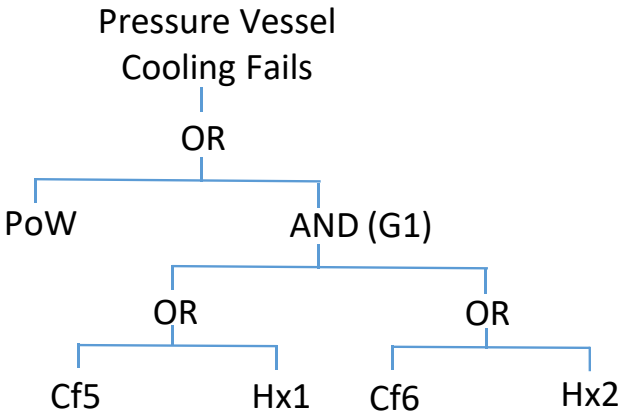
From dependency model

From factor calculations

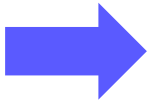
From factor calculations

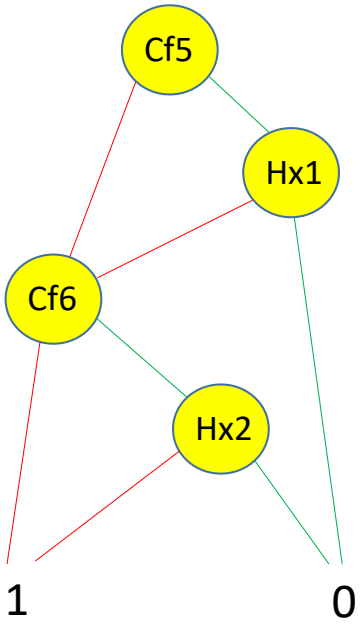
$$Q_{Cfi} = 1 - \prod_{j=1}^{n} \left(1 - q_{x_j}\right)$$

BDD Analysis for G1

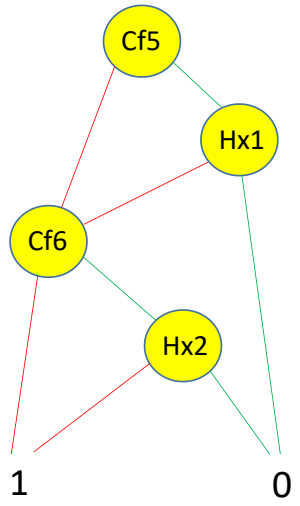


Variable Ordering
Cf5 < Hx1 < Cf6 < Hx2





BDD Probability Analysis



j	$path_j$	$Ipath_j$	$Dpath_j^1$
1	$Cf5_1, Cf6_1$	$Cf5_1, Cf6_1$	
2	$Cf5_1, Cf6_0, Hx2_1$	$Cf5_1, Cf6_0$	$Hx2_1$
3	$Cf5_0, Hx1_1, Cf6_1$	$Cf5_0, Cf6_1$	$Hx1_1$
4	$Cf5_0, Hx1_1, Cf6_0, Hx2_1$	$Cf5_0, Cf6_0$	$Hx1_1, Hx2_1$

$$Q_{SYS} = \sum_{j=0}^{npath} \left[P(Ipath_j). \prod_{k=1}^{ndep} P(Dpath_j^k) \right]$$

$Q_{path1} = P(Cf5_1). P(Cf6_1) = 0.0010830$

$Q_{G1} = 0.00109175$

$Q_{path2} = P(Cf5_1). (1 - P(Cf6_1)). P(Hx2_1) = 8.8052957e\text{-}06$

$Q_{path3} = (1 - P(Cf5_1)). P(Cf6_1). P(Hx1_1) = 0.0$

$Q_{path4} = (1 - P(Cf5_1)). (1 - P(Cf6_1)). P(Hx1_1, Hx2_1) = 0.0$

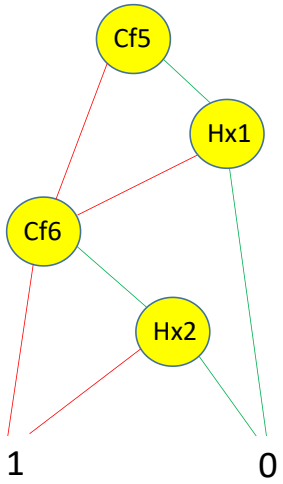
Top Event Probability

Event Code	Description	I / E	D-Group	q
Cf1	P1. P2	I	D1	0.00170988
Cf2	S1. S2	E		0.00034225
Cf3	Comp + R1 + Fan + Motor + R2 + T2 + V1	E		0.6035094
Cf4	P3S + P3R	E	D3	0.05285
Cf5	Cf_1 + T1	E		0.0017338
Cf6	Cf_2 + Cf_3 + Cf_4	E		0.6246519
G1	BDD	I		0.001091749
Cf7	PoW + G1			0.0020906577

$$Q_{Cfi} = 1 - \prod_{j=1}^{n} \left(1 - q_{x_j}\right)$$

Top Event Probability = 0.0020906577

Top Event Failure Intensity



$$w_{SYS}(t) = \sum_{\substack{i \\ initiators}} G_i(\boldsymbol{q}).w_i\,(t)$$

Birnbaum's Measure of Importance / Criticality Function

$$G_i(\boldsymbol{q}) = Q_{SYS}(1_i, \boldsymbol{q}) - Q_{SYS}(0_i, \boldsymbol{q})$$

Initiators:
T1, Hx1, P1, P2, PoW

$$Q_{SYS}\left(1_i, \underline{q}\right) = \sum_{x_{i_1} \in path_j} P\left(path_j - x_{i_1}\right) + \sum_{x_i \notin path_j} P(path_j | x_i = 1)$$

$$Q_{SYS}\left(0_i, \underline{q}\right) = \sum_{x_{i_0} \in path_j} P\left(path_j - x_{i_0}\right) + \sum_{x_i \notin path_j} P(path_j | x_i = 0)$$

Top Event Failure Intensity

Variable	Q(var=F)	Q(var=W)	G_i(var)	$G_{i\ (var)}$ w
Hx1				$1.152147238 \times 10^{-5}$
T1	0.6300421	0.0020756	0.6279665	6.2795143×10^{-6}
P1	0.5042367	0.1268205	0.3774162	8.3356331×10^{-6}
P2	0.5042367	0.1268205	0.3774162	8.3356331×10^{-6}
PoW	1.0	0.0010918	0.9989082	9.979093×10^{-5}

Top Events Failure Intensity

$$w_{SYS}(t) = 1.342632 \times 10^{-4} / hour$$

Conclusions

- Dynamic and Dependent Tree Theory, D^2T^2, enables the evaluation of fault trees which are not limited by the restrictions which apply to conventional fault trees solved by Kinetic Tree Theory.
- The analysis algorithm utilises BDDs, Petri Nets and Markov Models.
- Retains the familiar and popular fault tree causality structure.
- The Petri net and Markov models dedicated to solve the complexities and dependencies are minimal in size.
- Modularisation of the fault tree minimises the size of the BDD utilised in the system evaluation (and therefore the number of paths).

The End

Any Questions?

Professor John Andrews
Faculty of Engineering
University of Nottingham
Nottingham, NG7 2RD
England

Email: john.andrews@nottingham.ac.uk

Dr Silvia Tolo
Faculty of Engineering
University of Nottingham
Nottingham, NG7 2RD
England

Email: silvia.tolo@nottingham.ac.uk