Embedded Systems 25

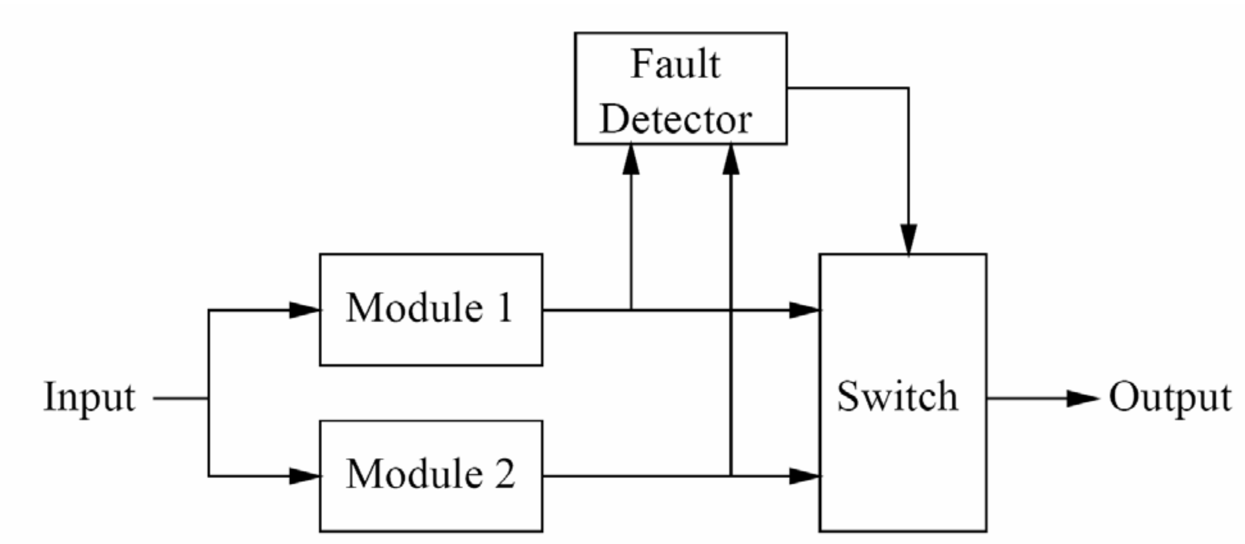
Failure modes of subsystems

REVIEW

- Fail-silent failures
  - subsystem either produces correct results
    or produces (recognizable) incorrect results
    or remains quiet
  - **can be masked as long as at least one system survives**
- Consistent failures
  - If subsystem produces incorrect results all recipients receive same (incorrect) result
  - **can be masked iff the failing systems form a minority**
- Byzantine failures
  - subsystem reports different results to different dependent systems
  - **can be masked iff strictly less than a third of the systems fail**

Dynamic hardware redundancy: standby spare arrangement

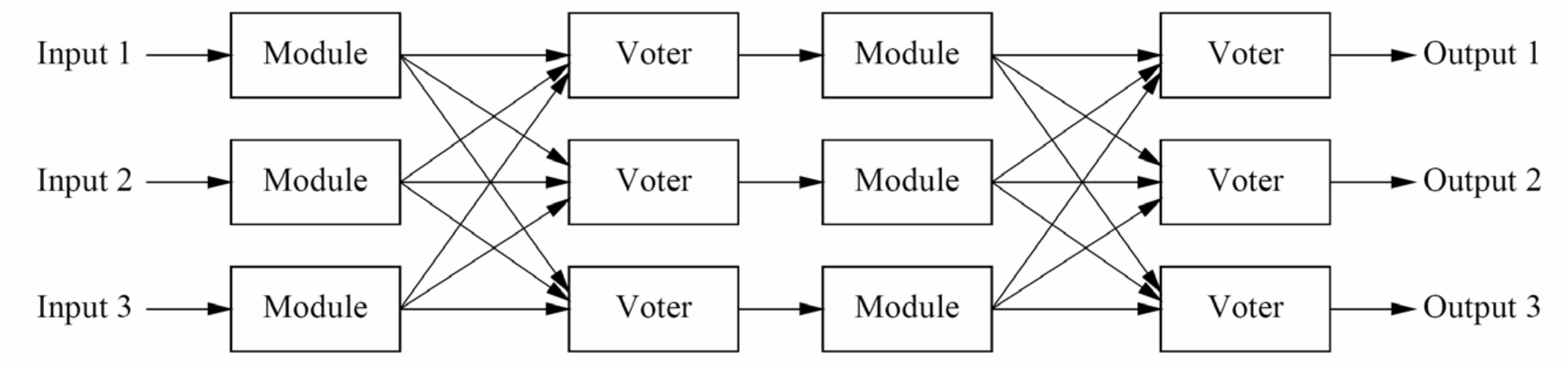
REVIEW



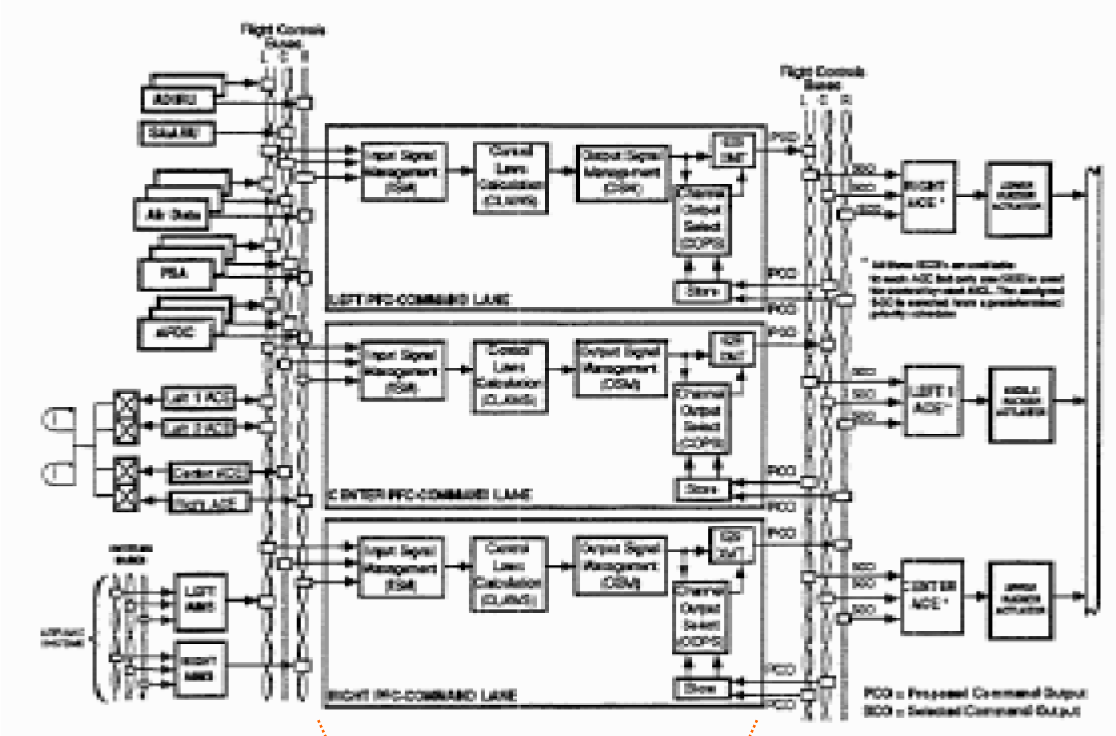
- Fault detection based on outputs (consistency check) not on voting
- Advantage: less redundant hardware
- Disadvantage: fault detection may take time $\Rightarrow$ fault not masked

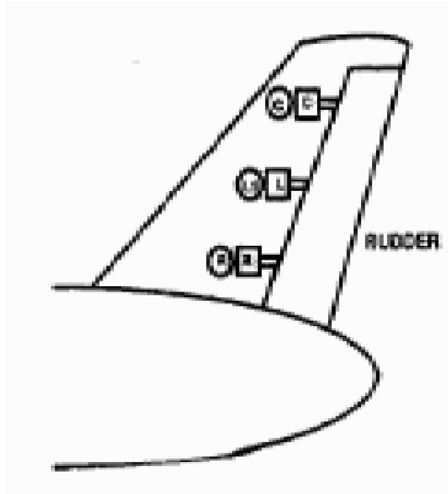
Static hardware redundancy: Multiple Stage TMR

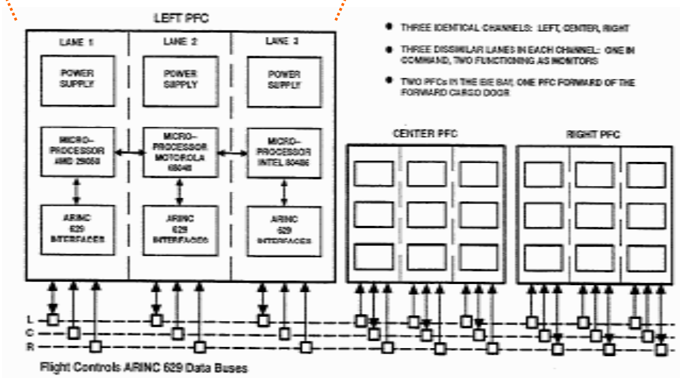
REVIEW



Boeing 777


Flight Controls Buses
L C R
ADIRU
SAARU
Air Data
PSA
AFDC
Left 1 ACE
Left 2 ACE
Center ACE
Right ACE
LEFT AIMS
RIGHT AIMS
Input Signal Management (ISM)
Control Laws Calculation (CLAWS)
Output Signal Management (OSM)
Channel Output Select (COPS)
Store
629 XMT
LEFT PFC COMMAND LANE
CENTER PFC COMMAND LANE
RIGHT PFC COMMAND LANE
PCO
SCO
RIGHT ACE *
LEFT 1 ACE *
CENTER ACE *
PCO = Proposed Command Output
SCO = Selected Command Output


RUDDER
C
L
R


LEFT PFC
LANE 1
LANE 2
LANE 3
POWER SUPPLY
MICRO-PROCESSOR AMD 29050
MICRO-PROCESSOR MOTOROLA 68040
MICRO-PROCESSOR INTEL 80486
ARINC 629 INTERFACES
THREE IDENTICAL CHANNELS: LEFT, CENTER, RIGHT
THREE DISSIMILAR LANES IN EACH CHANNEL: ONE IN COMMAND, TWO FUNCTIONING AS MONITORS
TWO PFCs IN THE EE BAY, ONE PFC FORWARD OF THE FORWARD CARGO DOOR
CENTER PFC
RIGHT PFC
L
C
R
Flight Controls ARINC 629 Data Buses

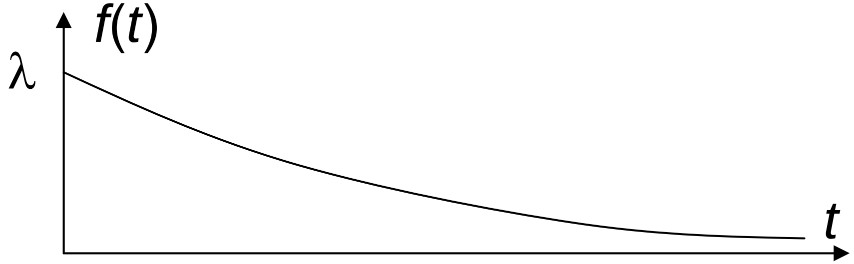
Reliability: f(t), F(t)

REVIEW

- Let T: time until first failure, T is a random variable
- Let $f(t)$ be the density function of T

Example: Exponential distribution

$f(t)=\lambda e^{-\lambda t}$



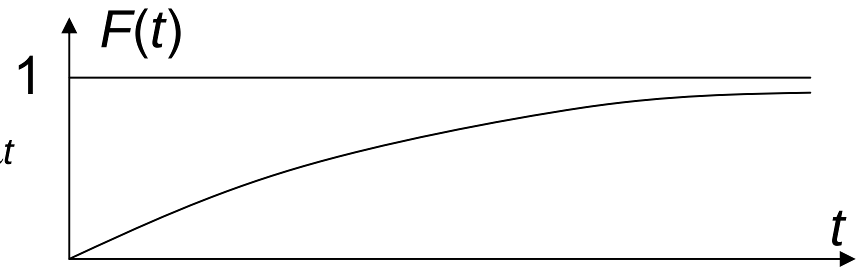
- $F(t)$ = probability of the system being faulty at time t:

$F(t) = \Pr(T \le t)$

$$F(t) = \int_0^t f(x)\,dx$$

Example: Exponential distribution

$$F(t) = \int_0^t \lambda e^{-\lambda x} dx = -[e^{-\lambda x}]_0^t = 1 - e^{-\lambda t}$$



Reliability: *R(t)* REVIEW

- **Reliability** $R(t)$ = probability that the time until the first failure is larger than some time t:

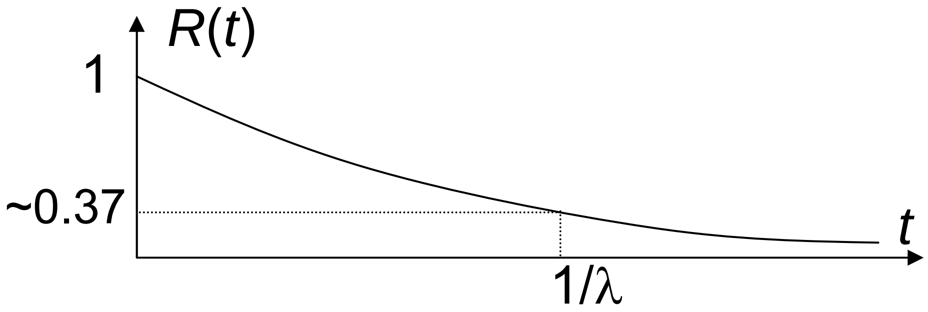
$R(t)=\Pr(T>t),\ t\geq 0$ $$R(t)=\int_t^{\infty} f(x)dx$$

$$F(t)+R(t)=\int_0^t f(x)dx+\int_t^{\infty} f(x)dx=1$$

$$R(t)=1-F(t)$$

<u>Example</u>: Exponential distribution

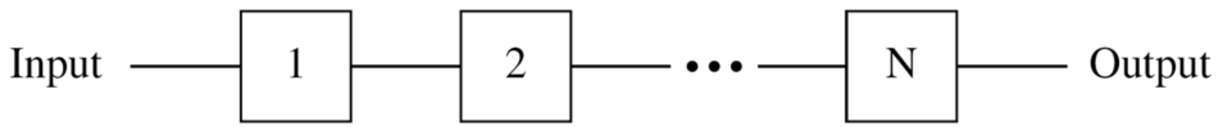
$R(t)=e^{-\lambda t};$



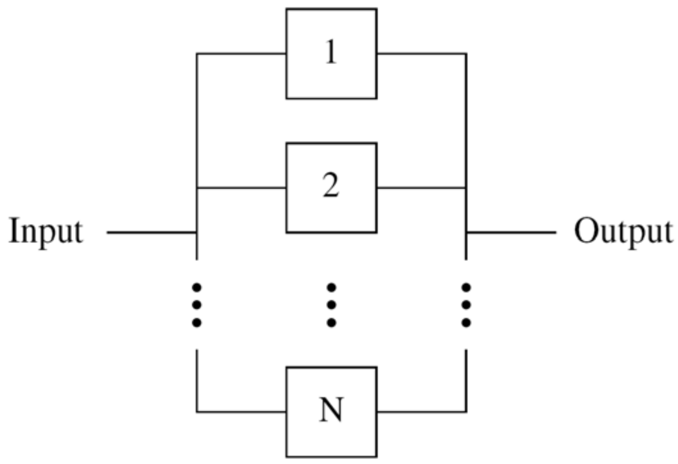
Reliability block analysis

REVIEW

- **Goal:** compute reliability of a system from the reliability of its components
- **Serial composition**



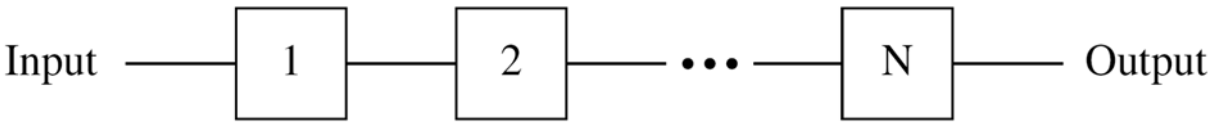
- **Parallel composition**



Inductive computation of reliability **REVIEW**

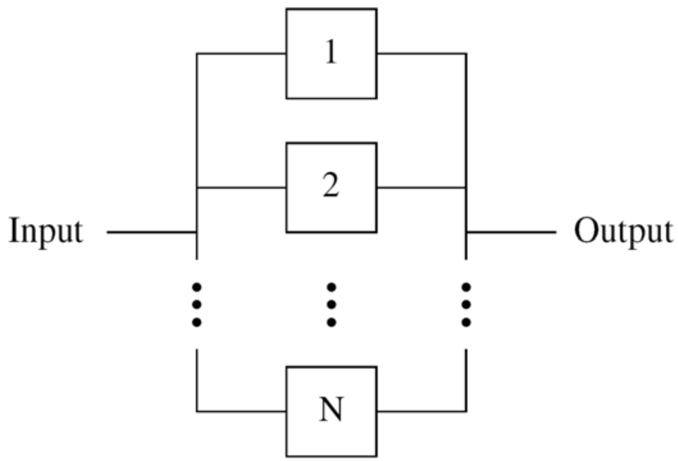
- **Assumption:** failures of the individual components are independent
- **Serial composition**

$$\prod_{i=1}^{N} R_i(t)$$



- **Parallel composition**

$$1 - \prod_{i=1}^{N} (1 - R_i(t))$$



Approximation: Minimal Cuts

REVIEW

- A **minimal cut** is a minimal set of components such that their simultaneous failure causes a system failure

- $$1 - \sum_{j \in MinimalCuts} \prod_{i \in j} [1 - R_i(t)]$$

  is a lower bound for the reliability R(t) of the full system.

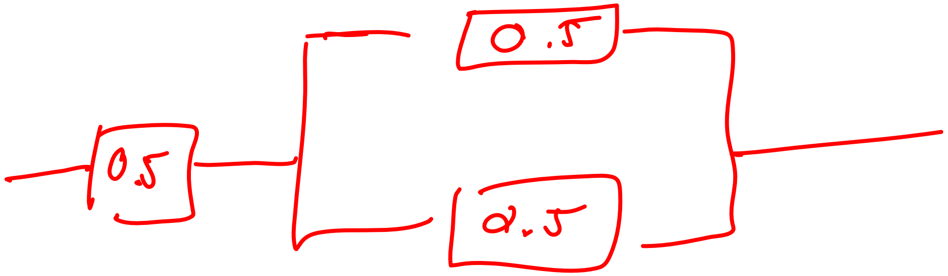
- Minimal cuts with a single component are called *single point failures*.

Approximation: Minimal Tie Sets

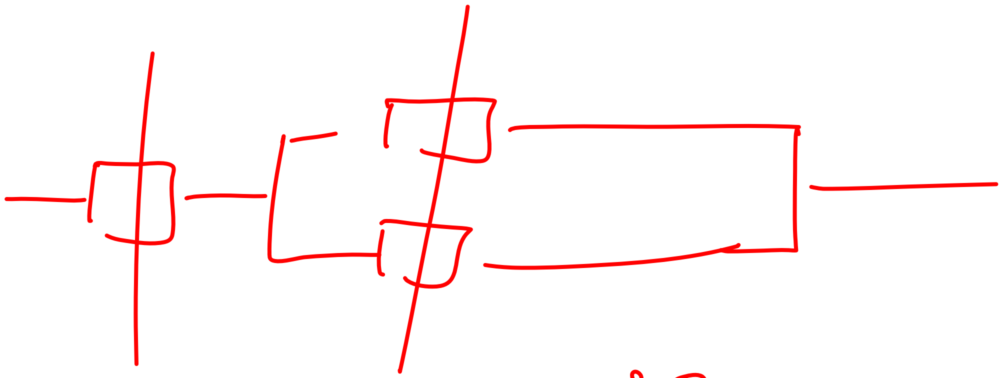
- A **minimal tie set** is a minimal set of components such that their simultaneous functioning guarantees the functioning of the system
- $$\sum_{j \in MinimalTies} \prod_{i \in j} R_i(t)$$

  is an upper bound for the reliability R(t) of the full system.

Example

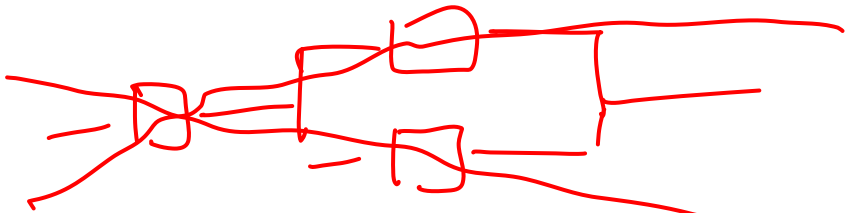


Minimal cuts:



$$R \geq 1 - [(1-0.5) + (1-0.5)^2] = 1 - [0.5 + 0.25] = 0.25$$

Minimal ties:

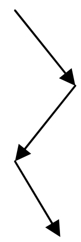


$$R \leq 0.5 \cdot 0.5 + 0.5 \cdot 0.5 = 0.5$$

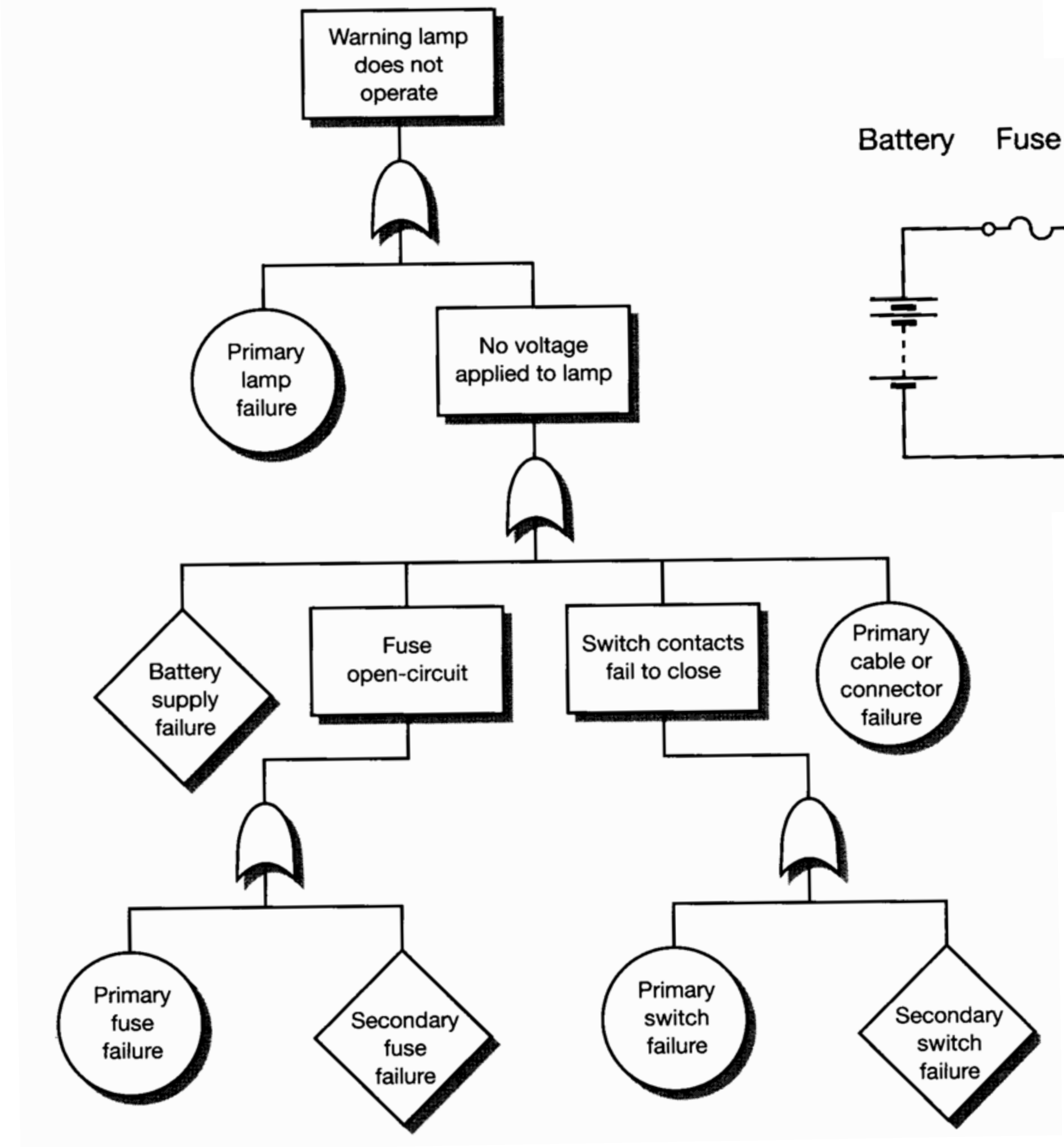
Exact: $0.5 \cdot (1 - (1-0.5)^2) = 0.5 \cdot 0.75 = 0.375$

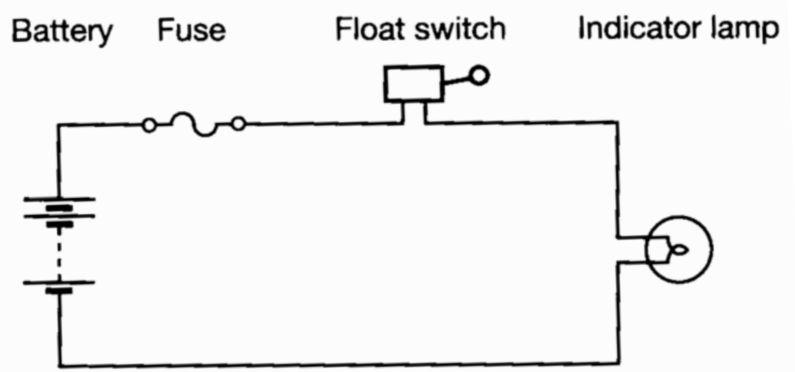
Fault tree Analysis (FTA)

- FTA is a top-down method of analyzing risks. Analysis starts with possible damage, tries to come up with possible scenarios that lead to that damage.
- FTA typically uses a graphical representation of possible damages, including symbols for AND- and OR-gates.
- OR-gates are used if a single event could result in a hazard.
- AND-gates are used when several events or conditions are required for that hazard to exist.



Example: Brake fluid warning lamp





Neil Storey:
Safety-critical computer systems

Direct Analysis

$$1 - \sum_{\vec{p} \in \{0,1\}^n} \left(FT(\vec{p}) \cdot \prod_{i=1}^{n} (1 - R_i(t))^{p_i} \cdot R_i(t)^{1-p_i} \right)$$

where

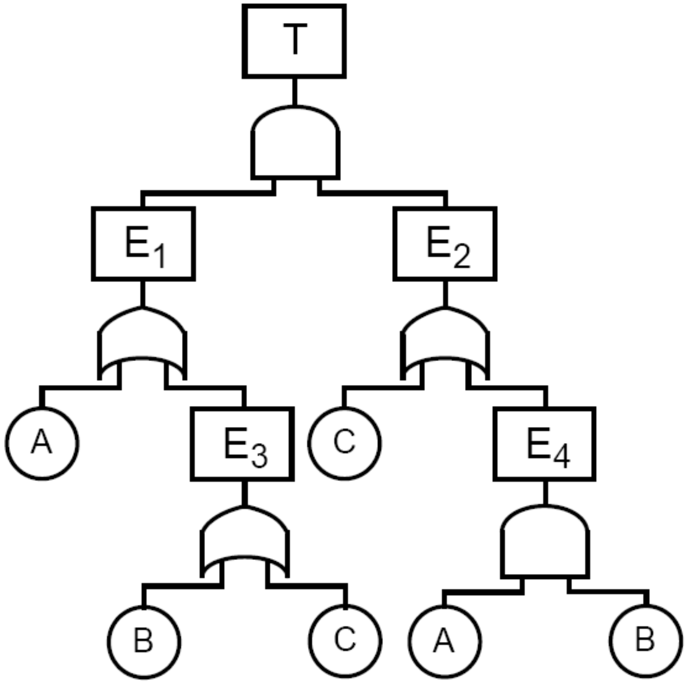
$\vec{p} = (p_1, \ldots, p_n)$ denotes the occurrence of the base events, and

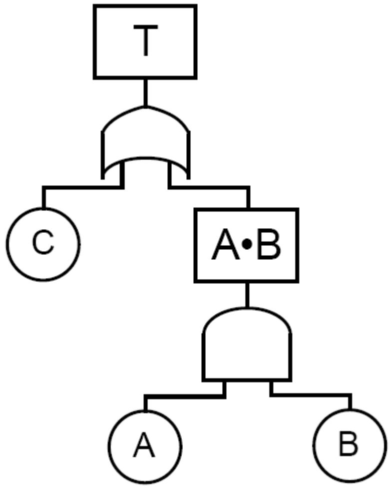
$FT(\vec{p})$ denotes the value of the top event

Problem: combinatorial explosion!

Equivalence

- Two fault trees are equivalent if the associated logical formulas are equivalent.
- E.g., $(A \vee (B \vee C) \wedge (C \vee (A \wedge B)))$ $\equiv$ $(C \vee (A \wedge B))$

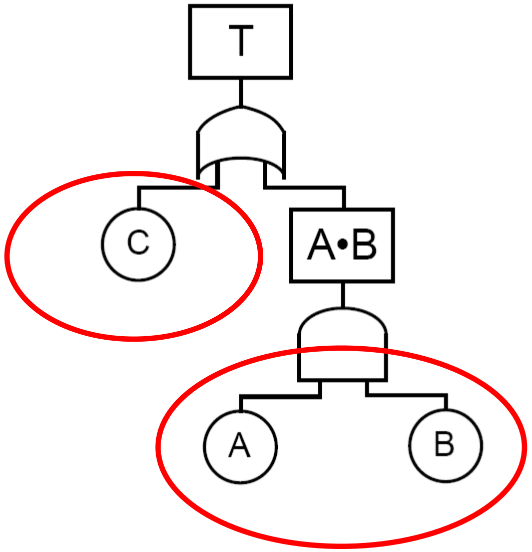




Minimal cut sets

Minimal cut set = “smallest set of basic events which, in conjunction, cause the top level event to occur”.

Logically: Disjunctive Normal Form (DNF) =
disjunction of conjunctions of basic events.



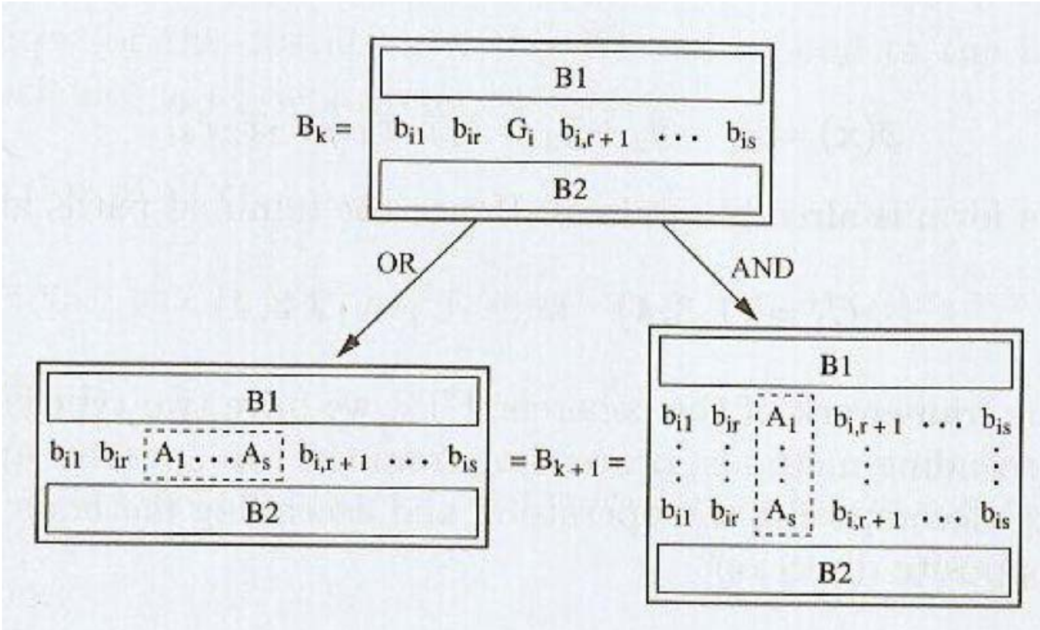
Example:

C (single point of failure) and

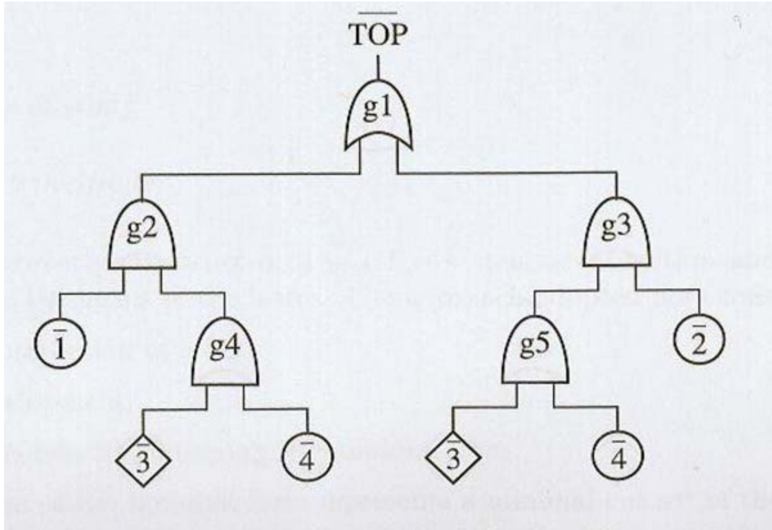
$A \wedge B$.

Mocus Algorithm (1972) „Method of Obtaining Cut Sets“

- Initialize the first element of a matrix with the top event operator
- As long as there is still an operator in the matrix:
  - If it is an AND operator, replace it with its inputs in the column
  - If it is an OR operator, replace it with its inputs in the row.
- Each column corresponds to a cut set; reduce to obtain minimal cut sets.



Example



$$[g_1] \longrightarrow [g_2 \;\; g_3] \longrightarrow \begin{bmatrix} 1 & g_3 \\ g_4 & g_3 \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} 1 & 2 \\ 1 & g_5 \\ g_4 & g_3 \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ g_4 & g_3 \end{bmatrix} \longrightarrow \cdots \longrightarrow \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 3 & 2 \\ 3 & 3 \\ 3 & 4 \\ 4 & 2 \\ 4 & 3 \\ 4 & 4 \end{bmatrix}$$

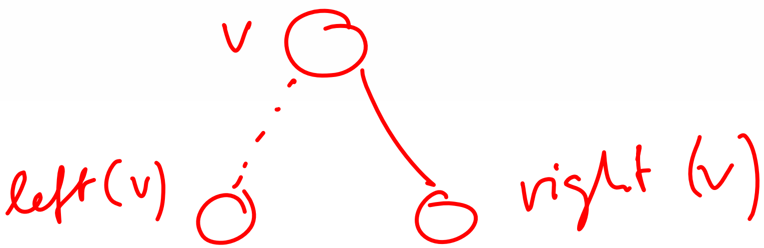
$\Rightarrow$ cut sets: $\{1,3,4\}$, $\{2,3,4\}$.

Binary decision trees

- Let X be a set of boolean variables and $<$ a total order on X
- Binary decision tree (BDT) is a complete binary tree over $\langle X, < \rangle$
  - each leaf v is labeled with a boolean value $val(v) \in \mathbb{B}$
  - non-leaf v is labeled by a boolean variable $Var(v) \in X$
  - such that for each non-leaf v and vertex w:

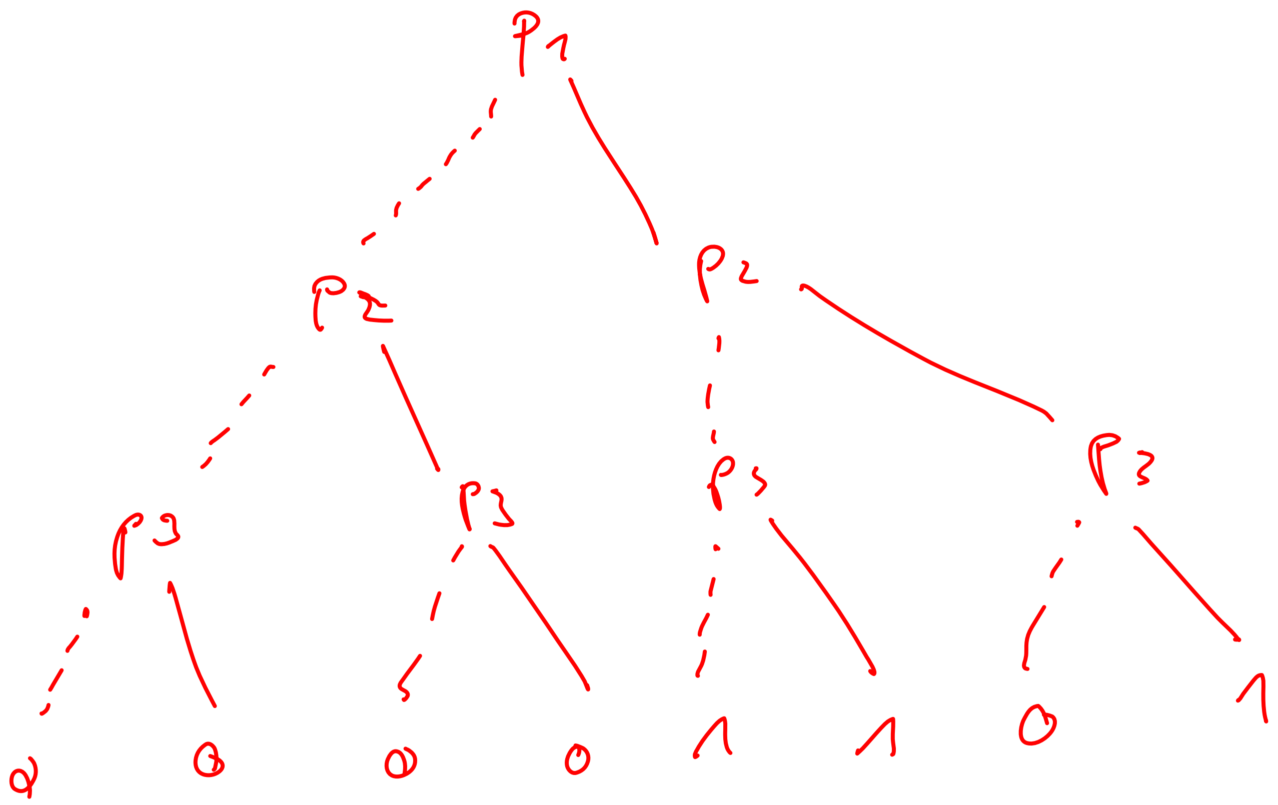
$$w \in \{ \, left(v), right(v) \, \} \;\Rightarrow\; (\, Var(v) < Var(w) \;\vee\; w \text{ is a leaf})$$

$\Rightarrow$ On each path from root to leaf, variables occur in the same order



Example

BDT = $p_1 \wedge (\neg p_2 \vee p_3)$



Shannon Expansion

- Each boolean function $f : \mathbb{B}^n \longrightarrow \mathbb{B}$ can be written as:

$$f(x_1, \ldots, x_n) = (x_i \ \wedge \ f[x_i := 1]) \ \vee \ (\neg\, x_i \ \wedge \ f[x_i := 0])$$

  - where $f[x_i := 1]$ stands for $f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$
  - and $f[x_i := 0]$ is a shorthand for $f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$

- The boolean function $f_{\mathsf{B}}(v)$ represented by vertex v in BDT B is:
  - for v a leaf: $f_{\mathsf{B}}(v) = \mathit{val}(v)$
  - otherwise:

$$f_{\mathsf{B}}(v) = (\mathit{Var}(v) \ \wedge \ f_{\mathsf{B}}(\mathit{right}(v))) \ \vee \ (\neg\, \mathit{Var}(v) \ \wedge \ f_{\mathsf{B}}(\mathit{left}(v)))$$

- $f_{\mathsf{B}} = f_{\mathsf{B}}(v)$ where v is the root of B

Considerations on BDTs

- BDTs are not compact
  - a BDT for boolean function $f : \mathbb{B}^n \rightarrow \mathbb{B}$ has 2^n leafs

  $\Rightarrow$ they are as space inefficient as truth tables!

$\Rightarrow$ BDTs contain quite some redundancy

  - all leafs with value one (zero) could be collapsed into a single leaf
  - a similar scheme could be adopted for isomorphic subtrees

- The size of a BDT does not change if the variable order changes

Ordered Binary Decision Diagrams

share equivalent expressions [Akers 76, Lee 59]

- Binary decision diagram (OBDD) is a directed graph over $\langle X, < \rangle$ with:
  - each leaf v is labeled with a boolean value *val*$(v) \in \{0, 1\}$
  - non-leaf v is labeled by a boolean variable *Var*$(v) \in X$
  - such that for each non-leaf v and vertex w:

$$w \in \{ \mathit{left}(v), \mathit{right}(v) \} \;\Rightarrow\; (\mathit{Var}(v) < \mathit{Var}(w) \;\vee\; w \text{ is a leaf})$$

$\Rightarrow$ An OBDD is acyclic

- f_{B} for OBDD B is obtained as for BDTs

Example

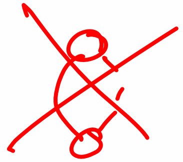
$p_1 \land (\neg p_2 \lor p_3)$

Reduced OBDDs

OBDD B over $\langle X, < \rangle$ is called *reduced* iff:

1. for each leaf v, w: $(\mathit{val}(v) = \mathit{val}(w)) \;\Rightarrow\; v = w$

   $\Rightarrow$ identical terminal vertices are forbidden



2. for each non-leaf v: $\mathit{left}(v) \neq \mathit{right}(v)$

   $\Rightarrow$ non-leafs may not have identical children

3. for each non-leaf v, w:

$$(\mathit{Var}(v) = \mathit{Var}(w) \;\wedge\; \mathit{right}(v) \cong \mathit{right}(w) \;\wedge\; \mathit{left}(v) \cong \mathit{left}(w)) \;\Rightarrow\; v = w$$

   $\Rightarrow$ vertices may not have isomorphic sub-dags

this is what is mostly called BDD; in fact it is an ROBDD!

Canonicity

[Fortune, Hopcroft & Schmidt, 1978]

For ROBDDs B and B′ over $\langle X, < \rangle$ we have:

$(f_{\mathsf{B}} = f_{\mathsf{B'}})$ implies B and B′ are isomorphic

$\Rightarrow$ *for a fixed variable ordering, any boolean function can be uniquely represented by an ROBDD (up to isomorphism)*

MakeNode(var,v_1,v_2)

If H(var,v_1,v_2) $\neq$ empty then return H(var,v_1,v_2);

lookup in hashtable

If (v_1=v_2) then return v_1

res := new node(var,v_1,v_2);

H(var,v_1,v_2) := res;

memorize result

return res;

Computing AND and OR

- Shannon expansion for binary operations:

$$f \text{ op } g = (x_1 \wedge (f[x_1 := 1] \text{ op } g[x_1 := 1])) \vee (\neg x_1 \wedge (f[x_1 := 0] \text{ op } g[x_1 := 0]))$$

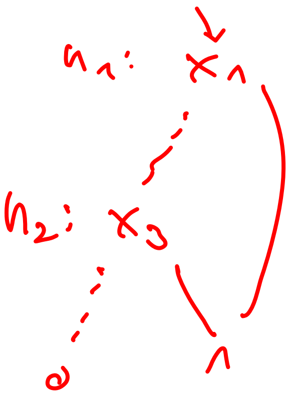
- A top-down evaluation scheme using the Shannon's expansion:
  - let v be the variable highest in the ordering occurring in B_f orB_g
  - split the problem into subproblems for $v := 0$ and $v := 1$, and solve recursively
  - at the leaves, apply the boolean operator *op* directly
  - reduce afterwards to turn the resulting OBDD into an ROBDD

- Efficiency gain is obtained by *dynamic programming*
  - the time complexity of constructing the ROBDD of $\mathrm{B}_{f \text{ op } g}$ is in $\mathcal{O}(|\mathrm{B}_f| \cdot |\mathrm{B}_g|)$

Apply(op,v_1,v_2)

if $G(v_1, v_2) \neq$ empty **then return** $G(v_1, v_2)$ **fi**; lookup in hashtable
if (v_1 and v_2 are terminals) **then** *res* := *val*(v_1) *op* *val*(v_2) **fi**;
else if (v_1 is terminal and v_2 is nonterminal)
    then *res* := $MakeNode$(*Var*(v_2), APPLY(*op*, v_1, *left*(v_2)), APPLY(*op*, v_1, *right*(v_2)));
else if (v_1 is nonterminal and v_2 is terminal)
    then *res* := $MakeNode$(*Var*(v_1), APPLY(*op*, *left*(v_1), v_2), APPLY(*op*, *right*(v_1), v_2));
else if (*Var*(v_1) = *Var*(v_2))
    then *res* := $MakeNode$(*Var*(v_1), APPLY(*op*, *left*(v_1), *left*(v_2)), APPLY(*op*, *right*(v_1), *right*(v_2)));
else if (*Var*(v_1) $<$ *Var*(v_2))
    then *res* := $MakeNode$(*Var*(v_1), APPLY(*op*, *left*(v_1), v_2), APPLY(*op*, *right*(v_1), v_2));
else (* *Var*(v_1) $>$ *Var*(v_2) *)
    res := $MakeNode$(*Var*(v_2), APPLY(*op*, v_1, *left*(v_2)), APPLY(*op*, v_1, *right*(v_2)));
$G(v_1, v_2)$:= *res*; memorize result
return *res*

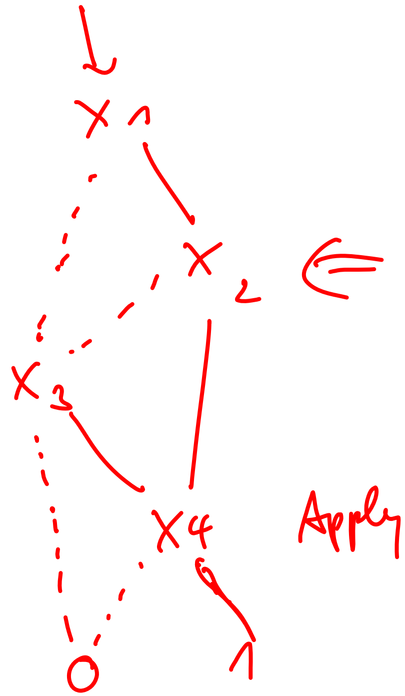
Example



$\wedge$

- Apply($\wedge$, n_1, m_1)
  - $\rightarrow$ Apply($\wedge$, n_2, m_1) ✓
    - $\rightarrow$ Apply($\wedge$, n_2, m_2) ✓
      - $\rightarrow$ Apply($\wedge$, 0, 0)
      - $\rightarrow$ Apply($\wedge$, 1, m_3)
    - $\rightarrow$ Apply($\wedge$, n_2, m_3)
      - $\rightarrow$ Apply($\wedge$, 0, m_3) = 0
      - $\rightarrow$ Apply($\wedge$, 1, m_3)
  - $\rightarrow$ Apply($\wedge$, 1, m_1)

$\Longrightarrow$

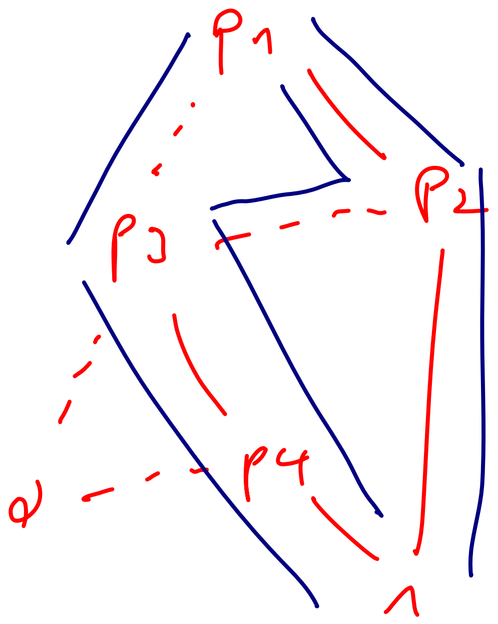


Apply($\wedge$, 1, m_3)

ROBDDs of Fault Trees

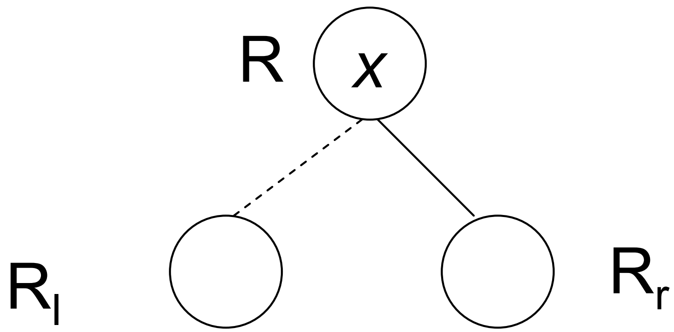
- Each path through the BDD from the root to a leaf node represents a disjoint combination of component failures and non-failures
- A path with a leaf node labeled with a 1 leads to system failure
- Probabilities associated with arcs on each path are either (1-$R(t)$) (component failure probability) for the right branch or $R(t)$ for the left branch
- System unreliability is given by the sum of the probabilities for all paths from the root to a leaf node labeled 1

Example



$$\Rightarrow R(t) = (1 - R_1(t)) \cdot (1 - R_2(t))$$
$$+ (1 - R_1(t)) \cdot R_2(t) \cdot (1 - R_3(t)) \cdot (1 - R_4(t))$$
$$+ R_1(t) \cdot (1 - R_3(t)) \cdot (1 - R_4(t))$$

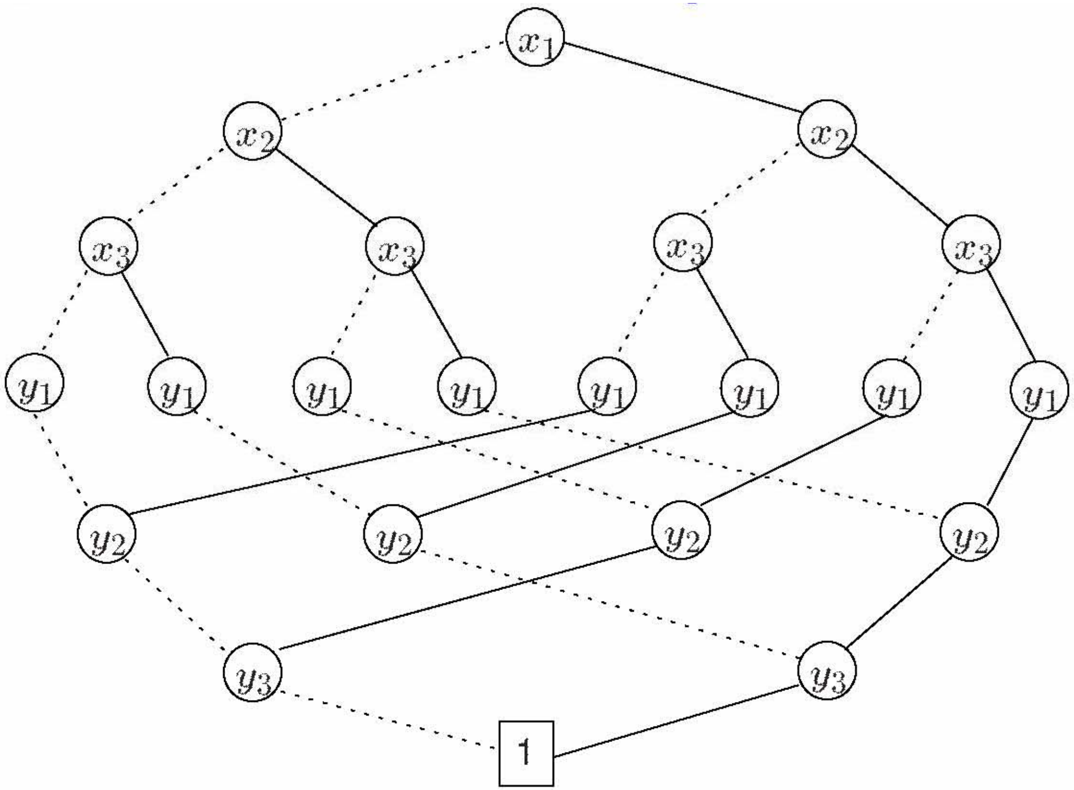
Recursive BDD evaluation



$$R(t) = R_x(t) * R_l(t) + (1-R_x(t)) * R_r(t)$$

$$R_1(t)=1$$
$$R_0(t)=0$$

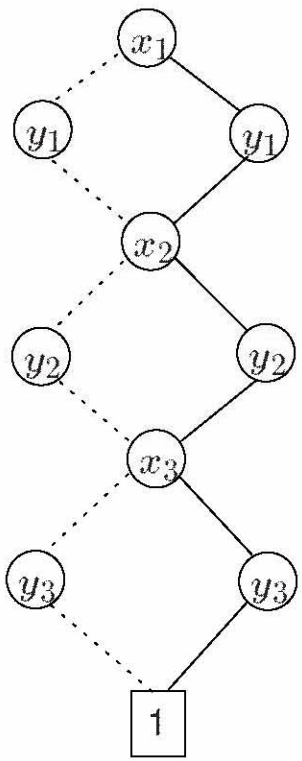
ROBDDs can be exponentially large



The ROBDD of $f_{stab}(\overline{x}, \overline{y}) = (x_1 \leftrightarrow y_1) \ \wedge \ \ldots \ \wedge \ (x_n \leftrightarrow y_n)$

has $3 \cdot 2^n - 1$ vertices under ordering $x_1 < \ldots < x_n < y_1 < \ldots < y_n$

Alternative variable ordering



The ROBDD of $f_{stab}(\overline{x}, \overline{y}) = (x_1 \leftrightarrow y_1) \ \wedge \ \ldots \ \wedge \ (x_n \leftrightarrow y_n)$

has $3 \cdot n + 2$ vertices under ordering $x_1 < y_1 < \ldots < x_n < y_n$

Optimal variable ordering

- The size of ROBDDs is dependent on the variable ordering
- Is it possible to determine $<$ such that the ROBDD has minimal size?
  - the optimal variable ordering problem for ROBDDs is NP-complete
  - polynomial reduction from the 3SAT problem (Bollig & Wegener, 1996)
- There are many boolean functions with large ROBDDs
  - for almost all boolean functions the minimal size is in $\Omega(\frac{2^n}{n})$
- How to deal with this problem in practice?
  - guess a variable ordering in advance
  - rearrange the variable ordering during the manipulations of ROBDDs

Sifting Algorithm (1993)

Dynamic variable reordering using variable swapping

1. Select some variable x_i
2. By successive swapping determine position where the ROBDD has least size
3. Shift to its optimal position
4. Go back to 1 until no more improvement.

Often only yields local optimum, but works well in practice.

Limitations of combinatorial models

- Assumption that failure probability is independent of the system state is often wrong.

Example: cold-spare redundancy

- Failure during standby is unlikely
- Failure during activation is likely

$\Rightarrow$ state-based models are required