Automated Analysis of Reliability Architecture

Fondazione Bruno Kessler

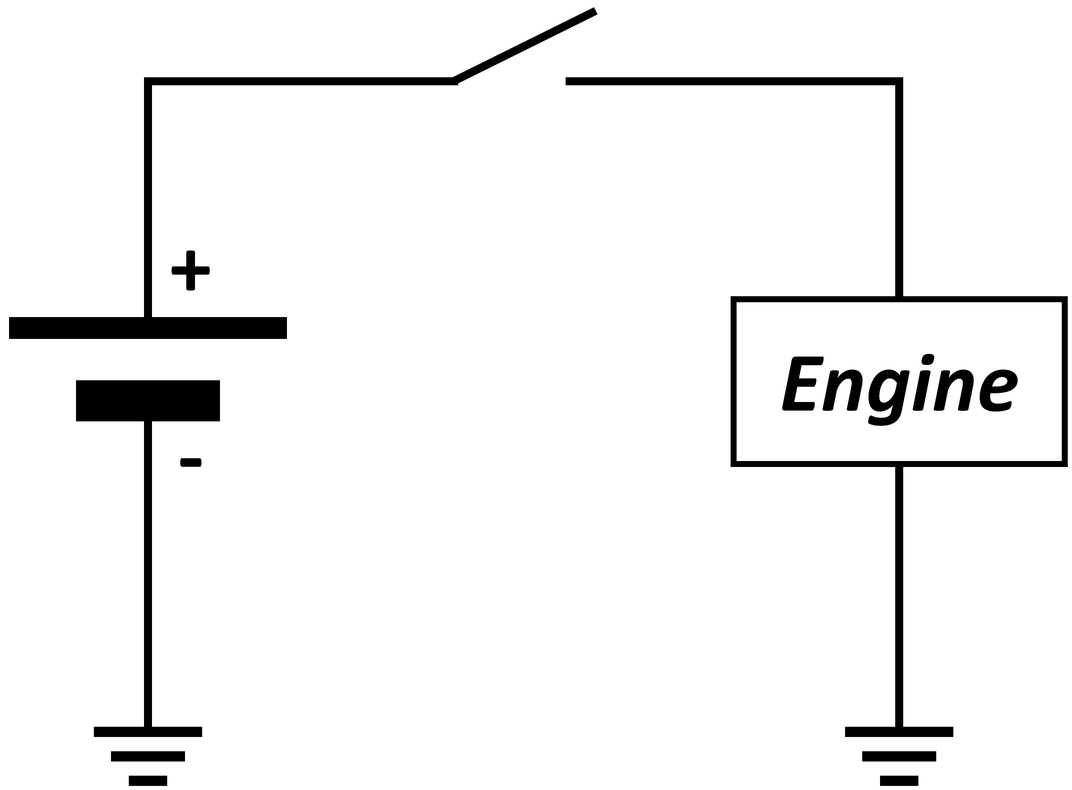
Marco Bozzano, Alessandro Cimatti, and **Cristian Mattarei**

Alpine Verification Meeting, 2013

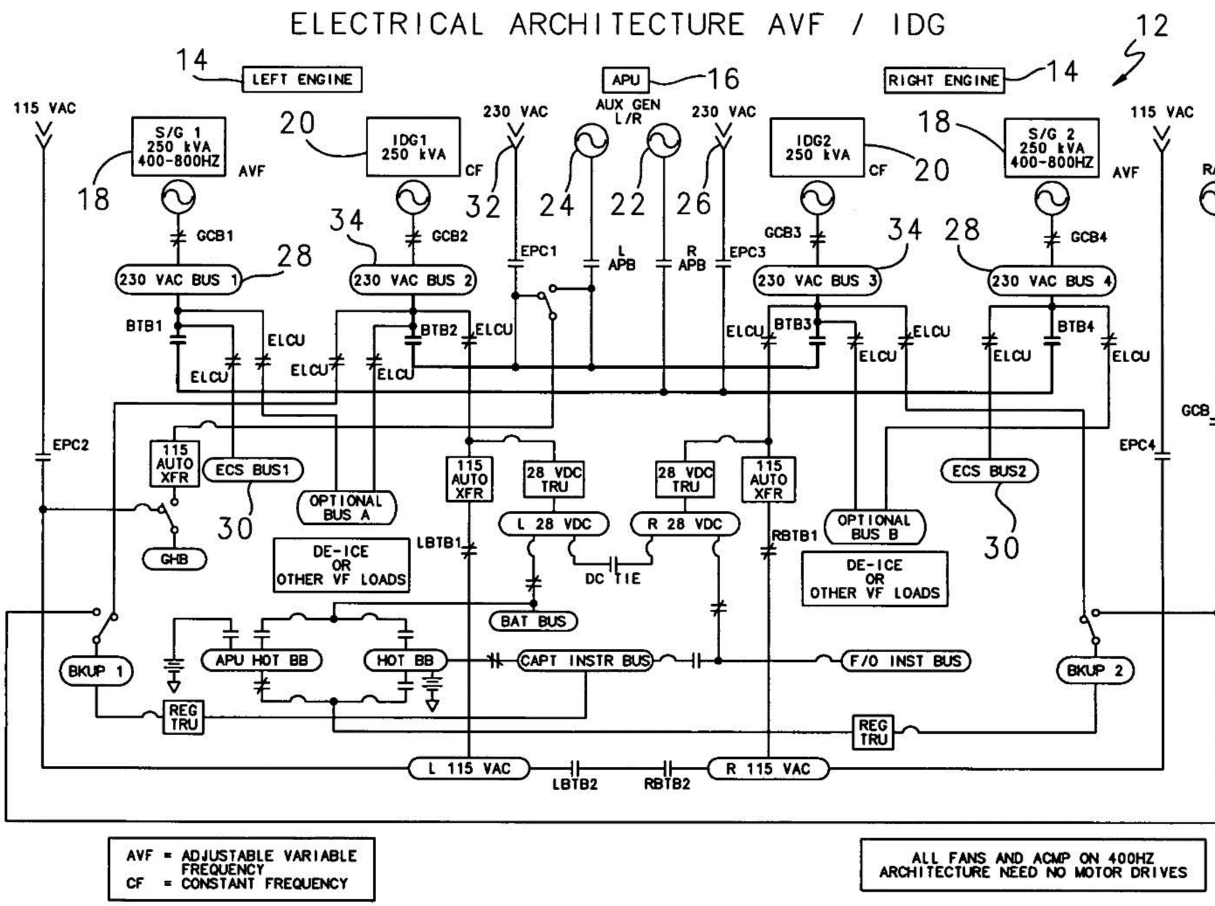
Outline

- Architectural Design in Critical Systems
  - Redundant systems
  - Reliability Analysis
- Automated Approaches
  - EUF modeling and Fault Tree Analysis
  - Efficient Analysis via predicate abstraction
- Conclusion

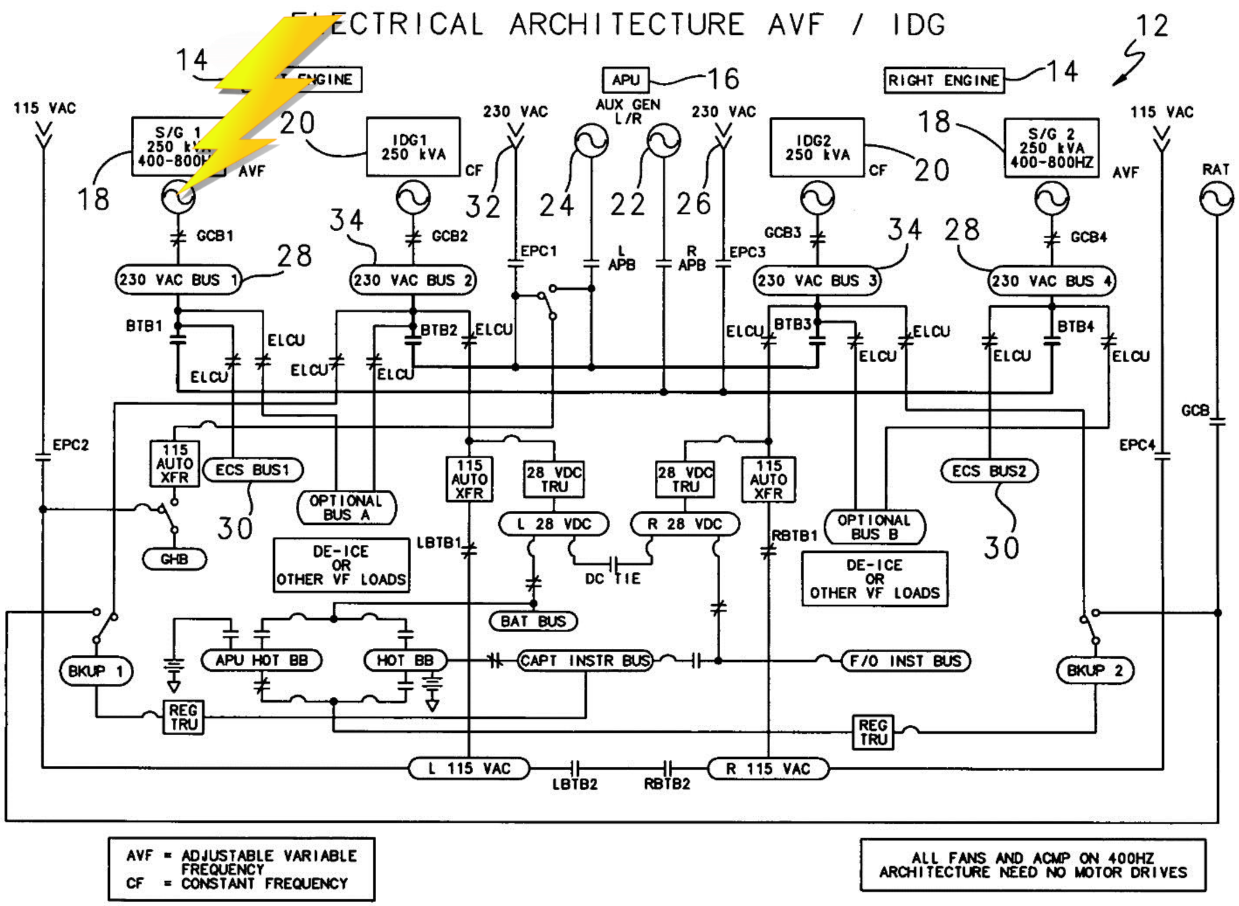
Power system: …in a perfect world



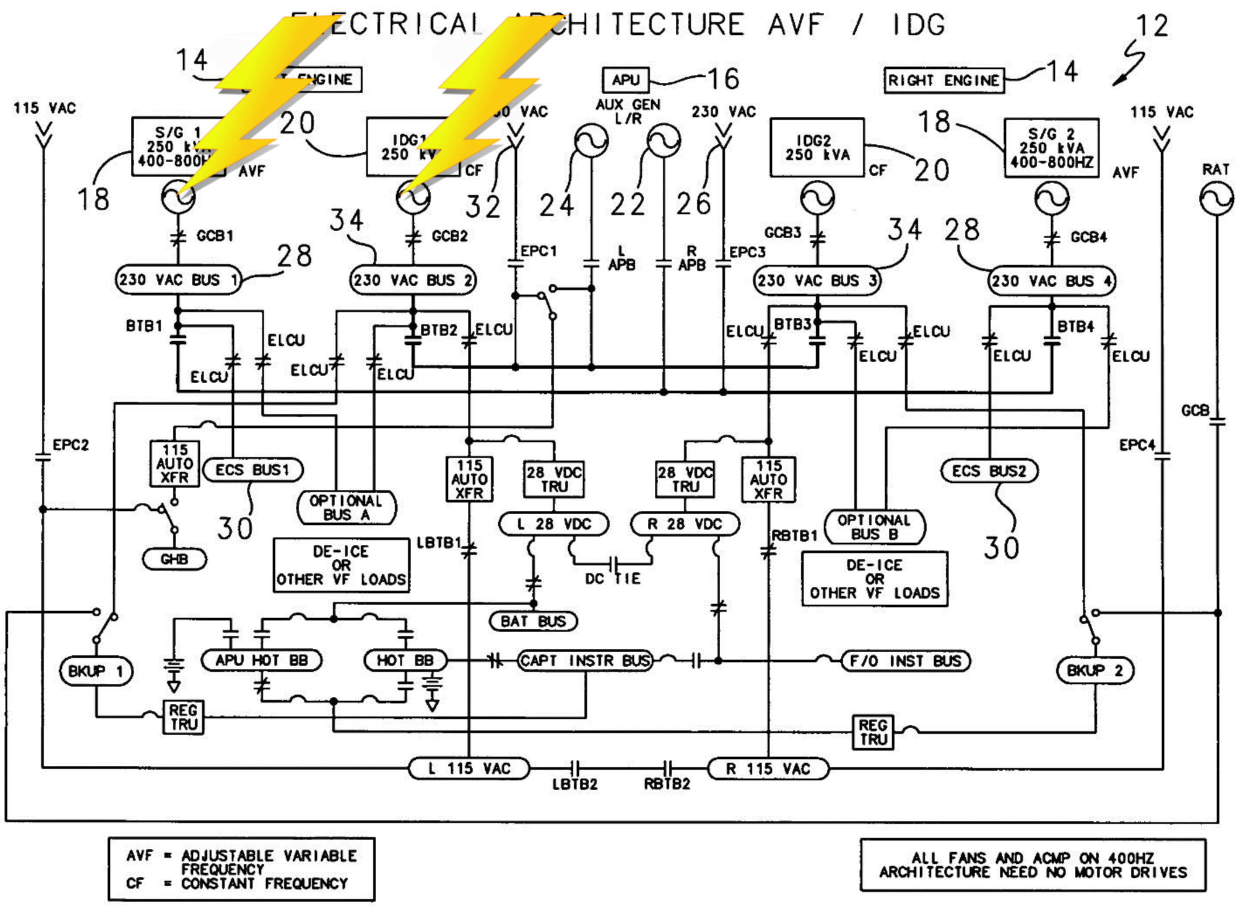
Power system: …in real world



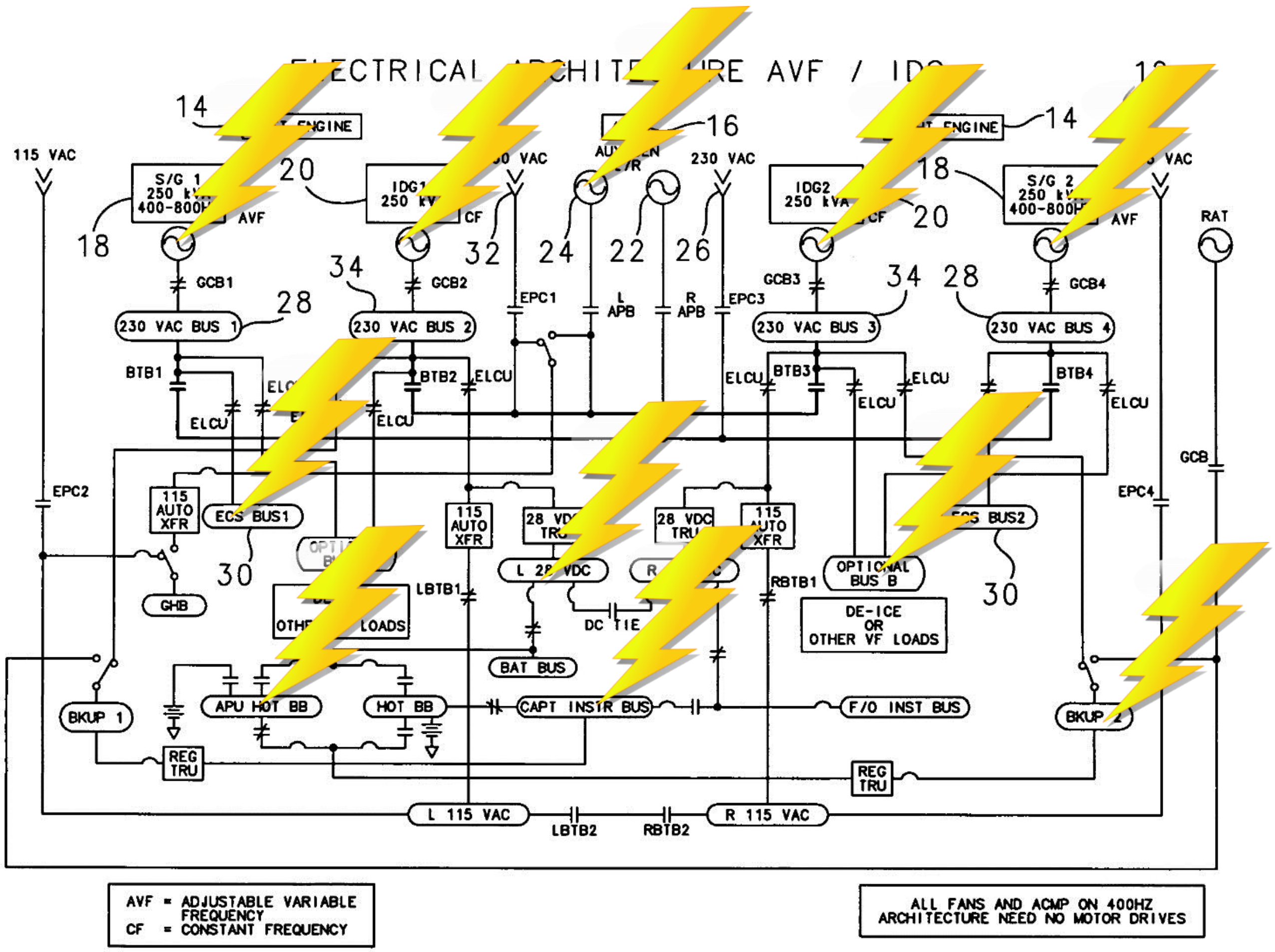
Power system: ...in real world



Power system: …in real world



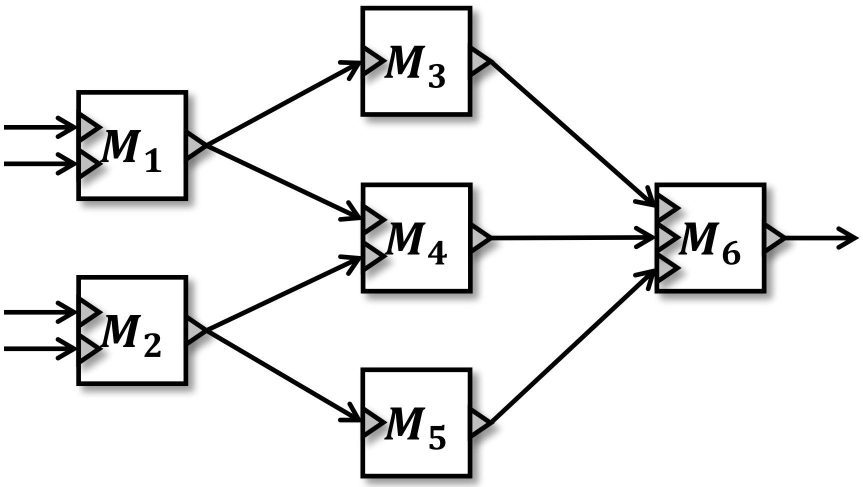
Power system: …in real world



Outline

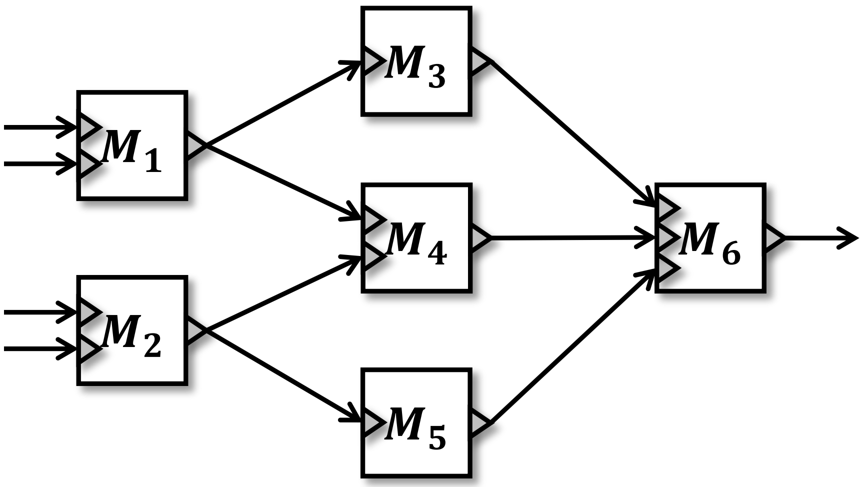
- **Architectural Design in Critical Systems**
  - **Redundant Systems**
  - **Reliability Analysis**
- Automated Approaches
  - EUF modeling and Fault Tree Analysis
  - Efficient Analysis via Predicate Abstraction
- Conclusion

Redundant systems definition: TMR
[Abraham74]

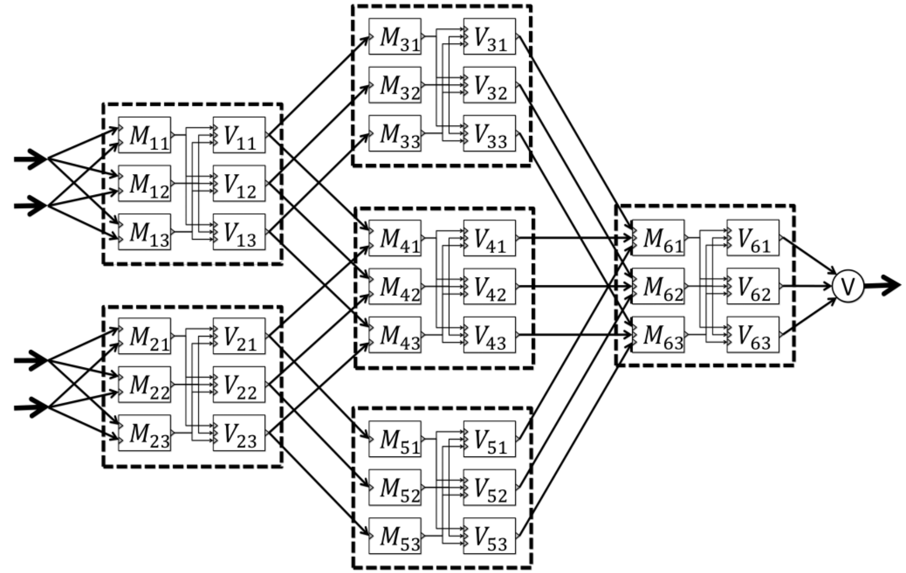


Nominal architecture

Redundant systems definition: TMR
[Abraham74]



Nominal architecture

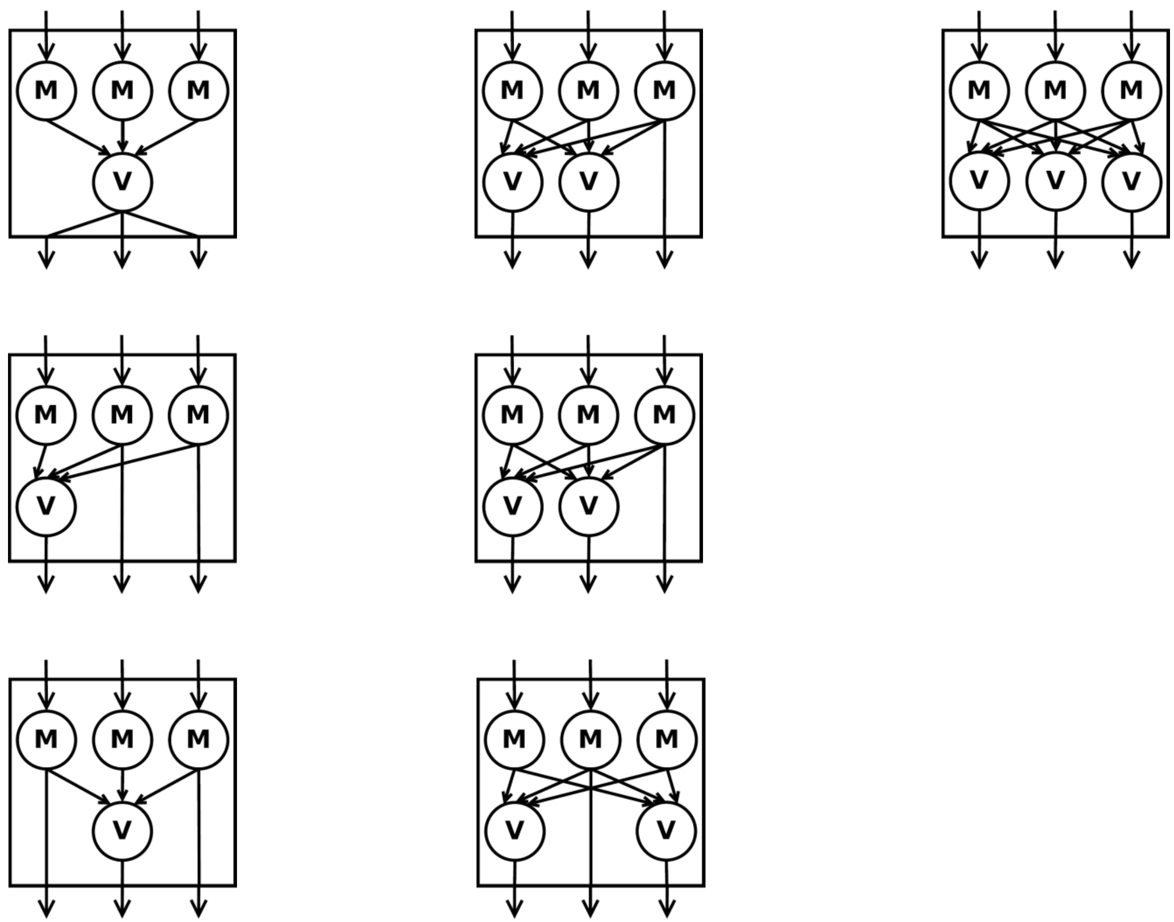


Redundant architecture

- Increase reliability for critical design
- Usage of redundant scheme (e.g. Triple Modular Redundancy)
- Hard to analyze and optimize system reliability

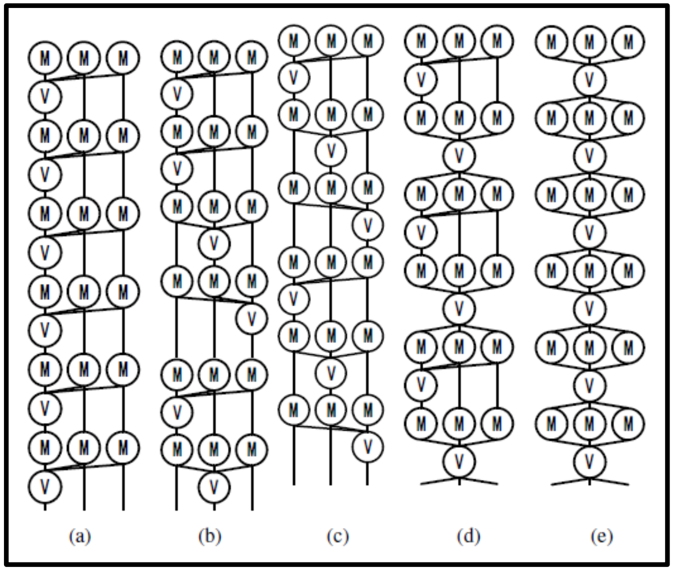
Triple Modular Redundancy patterns

1 voter 2 voters 3 voters

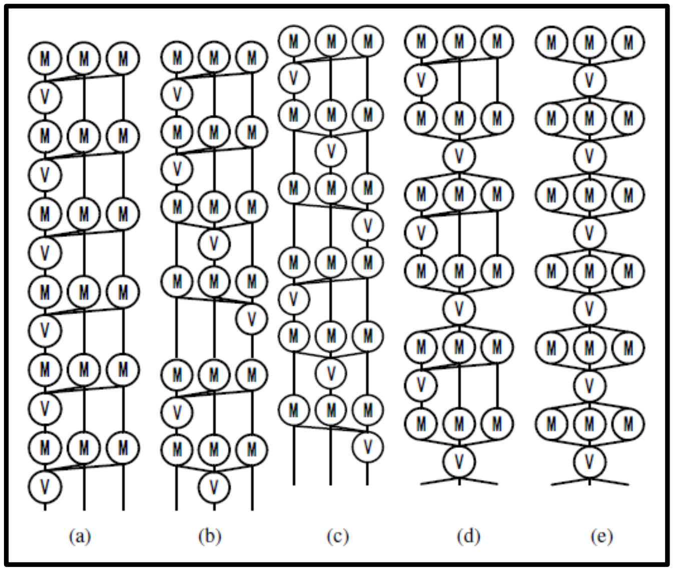


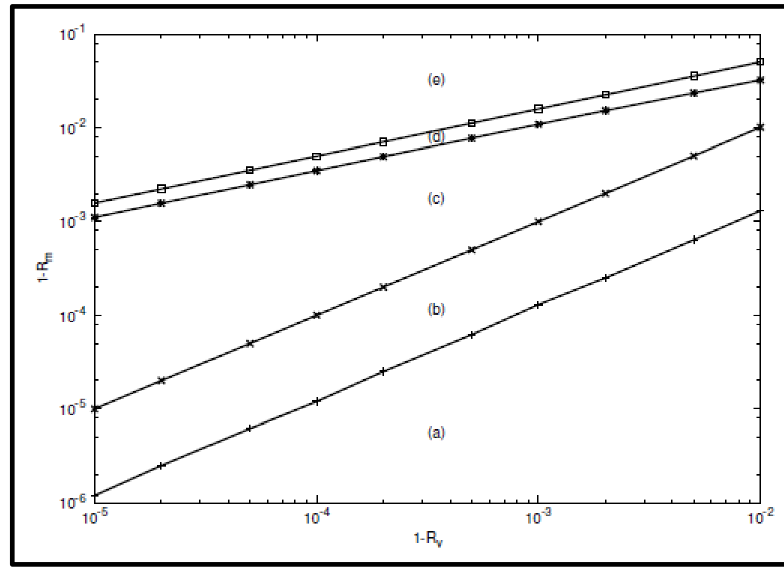
Reliability analysis: manual approach

[Hamamatsu10]



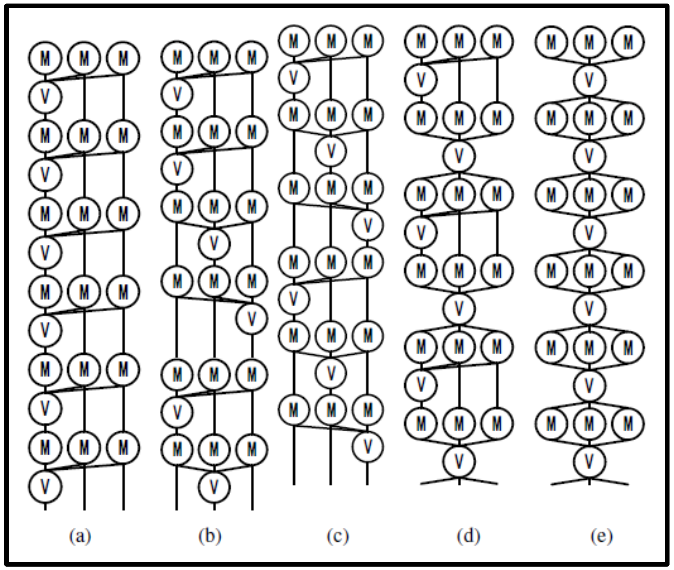
Reliability analysis: manual approach
[Hamamatsu10]

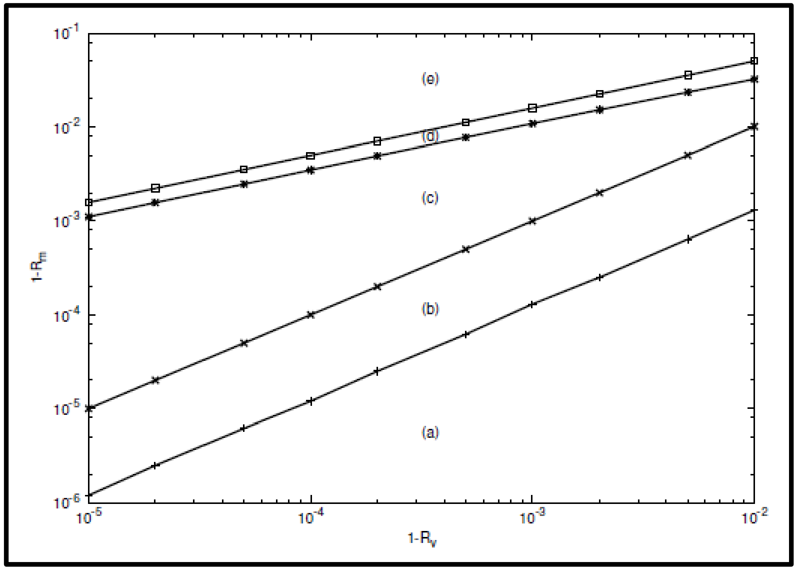




Reliability analysis: manual approach

[Hamamatsu10]



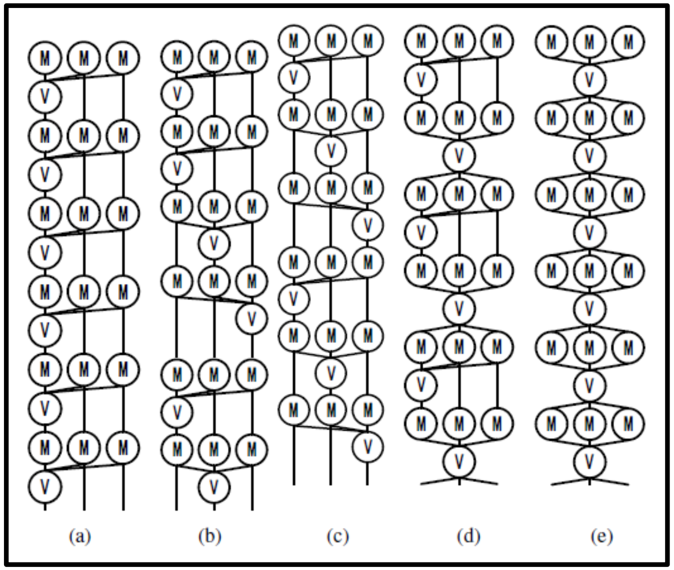


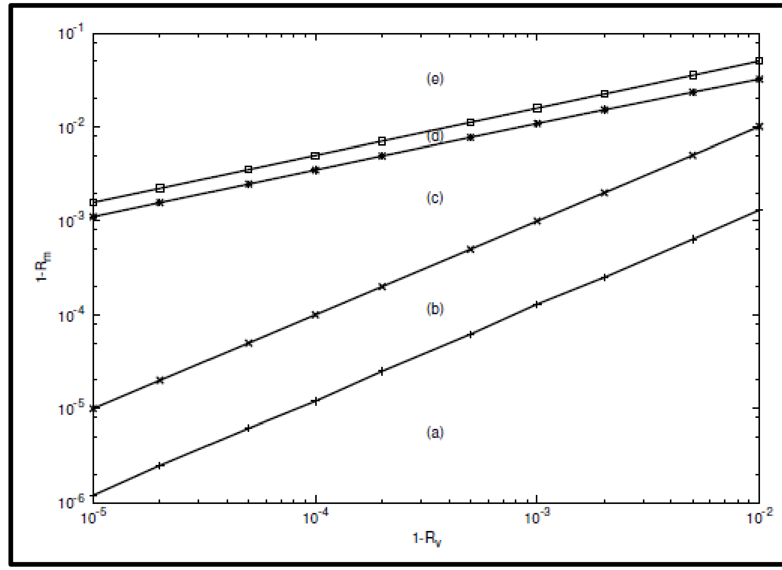
- Stage i, $1 \leq i < n$: For any $s \in \mathcal{S}$,

$$Pr[S_i = s] = \sum_{s' \in \mathcal{S}} \left(Pr[S_i = s \mid S_{i-1} = s'] * Pr[S_{i-1} = s'] \right)$$

Reliability analysis: manual approach

[Hamamatsu10]





- Stage i, $1 \leq i < n$: For any $s \in \mathcal{S}$,

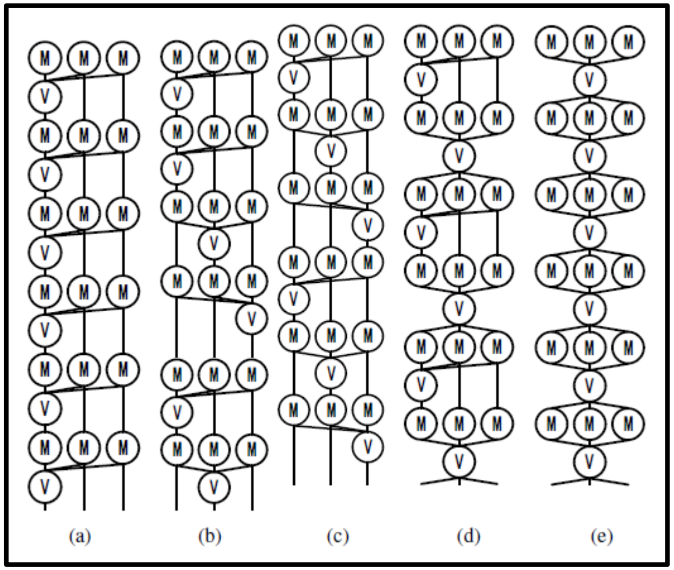
$$Pr[S_i = s] = \sum_{s' \in \mathcal{S}} \left(Pr[S_i = s \mid S_{i-1} = s'] * Pr[S_{i-1} = s'] \right)$$

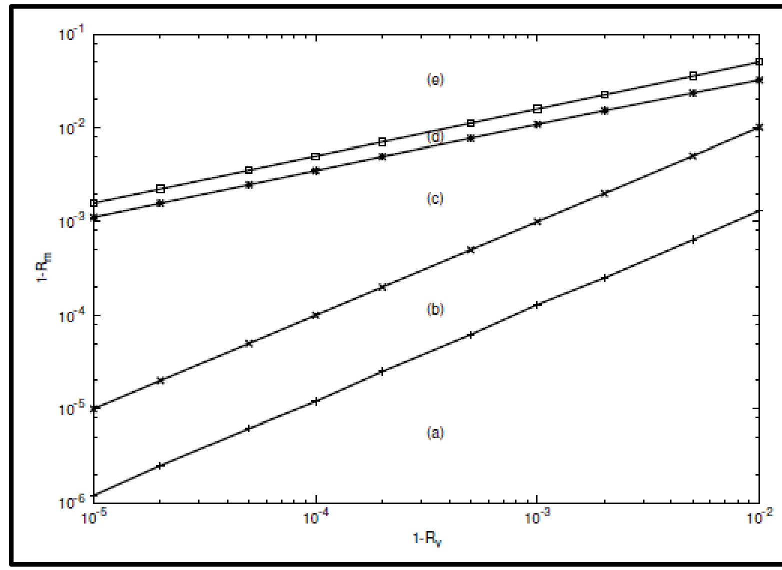
```
{Computing the initial lower bound}
if f((T_0, ..., T_0)) > f((T_1, T_2, T_3, T_1, ...)) then
    t_max := (T_0, ..., T_0)
    R := f((T_0, ..., T_0))
else
    t_max := (T_1, T_2, T_3, T_1, ...)
    R := f((T_1, T_2, T_3, T_1, ...))
end if

{Main}
Search((⊥, ⊥, ..., ⊥))
Return t_max
```

Reliability analysis: manual approach

[Hamamatsu10]

Triple Redundant Module comparison (1 voter)





- Stage i, $1 \leq i < n$: For any $s \in \mathcal{S}$,

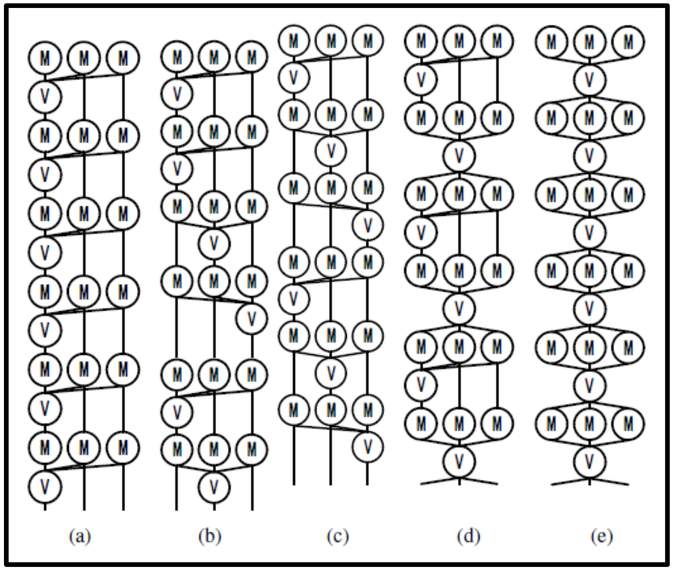
$$Pr[S_i = s] = \sum_{s' \in \mathcal{S}} \left(Pr[S_i = s \mid S_{i-1} = s'] * Pr[S_{i-1} = s'] \right)$$

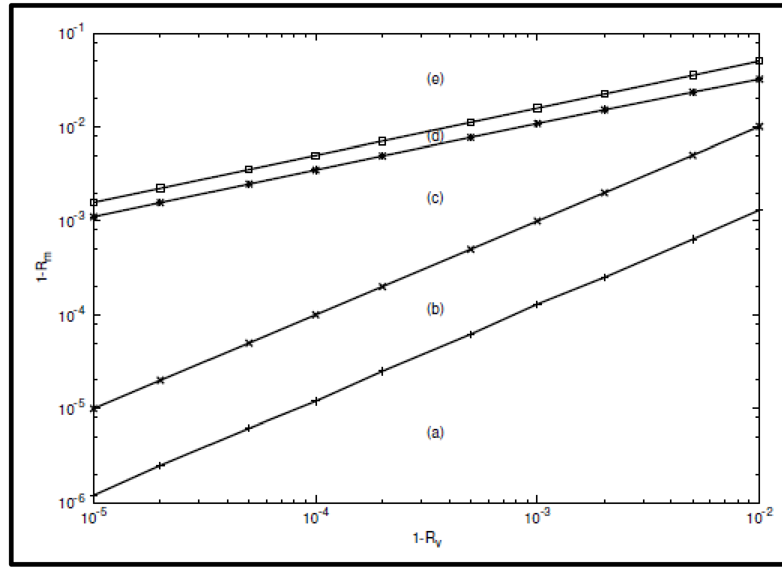
```
{Computing the initial lower bound}
if f((T_0, ··· , T_0)) > f((T_1, T_2, T_3, T_1, ···)) then
    t_max := (T_0, ··· , T_0)
    R := f((T_0, ··· , T_0))
else
    t_max := (T_1, T_2, T_3, T_1, ···)
    R := f((T_1, T_2, T_3, T_1, ···))
end if

{Main}
Search((⊥, ⊥, ··· , ⊥))
Return t_max
```

Reliability analysis: manual approach

[Hamamatsu10]

Triple Redundant Module comparison (1 voter)





- Stage i, $1 \leq i < n$: For any $s \in \mathcal{S}$,

$$Pr[S_i = s] = \sum_{s' \in \mathcal{S}} \left(Pr[S_i = s \mid S_{i-1} = s'] * Pr[S_{i-1} = s'] \right)$$

```
{Computing the initial lower bound}
if f((T_0, ··· , T_0)) > f((T_1, T_2, T_3, T_1, ···)) then
    t_max := (T_0, ··· , T_0)
    R := f((T_0, ··· , T_0))
else
    t_max := (T_1, T_2, T_3, T_1, ···)
    R := f((T_1, T_2, T_3, T_1, ···))
end if

{Main}
Search((⊥, ⊥, ··· , ⊥))
Return t_max
```

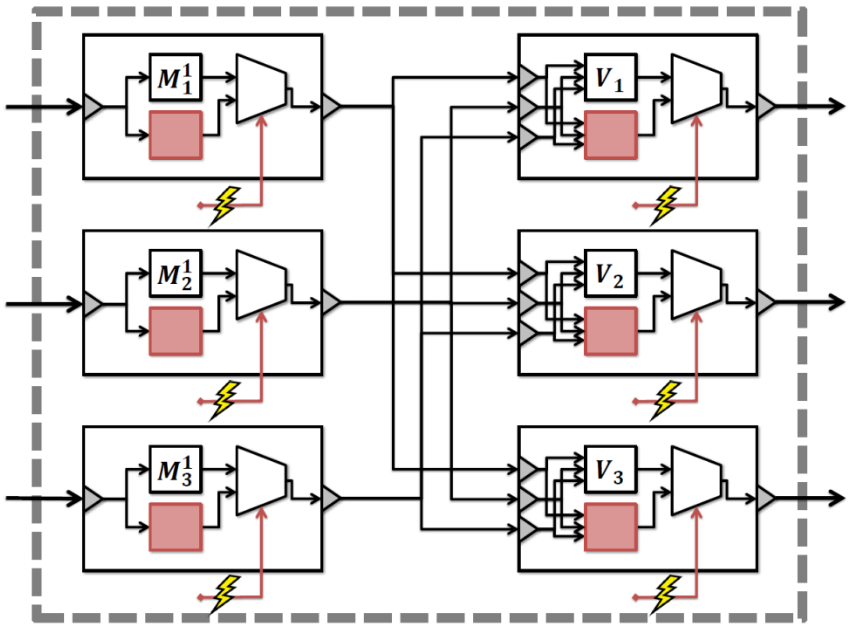
Reliability analysis: manual approach

- Time expensive and error prone reliability computation
- Specific approach for linear structures (not generalizable)
- Needs space discretization

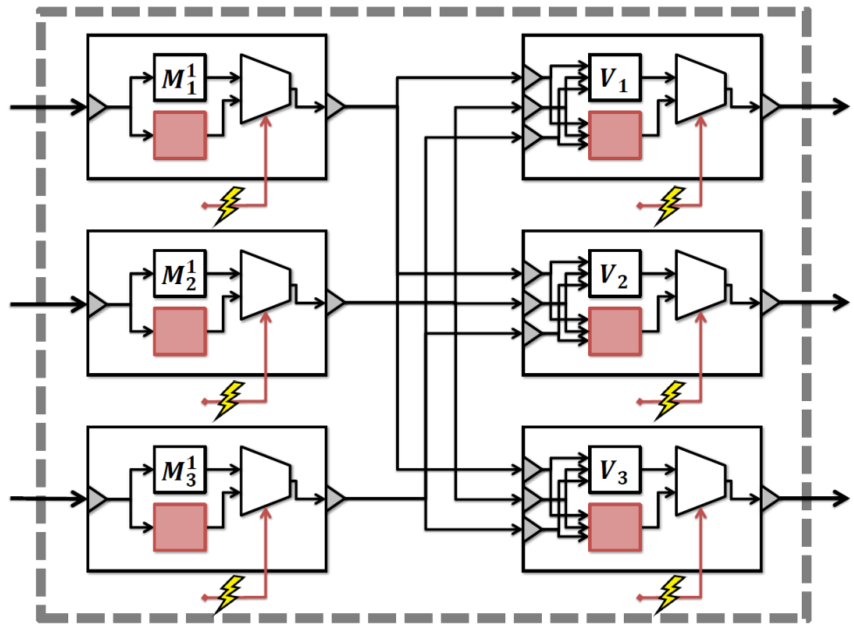
Outline

- Architectural Design in Critical Systems
  - Redundant Systems
  - Reliability Analysis
- **Automated Approaches**
  - **EUF modeling and Fault Tree Analysis**
  - Efficient Analysis via Predicate Abstraction
- Conclusion

Modeling of the extended system

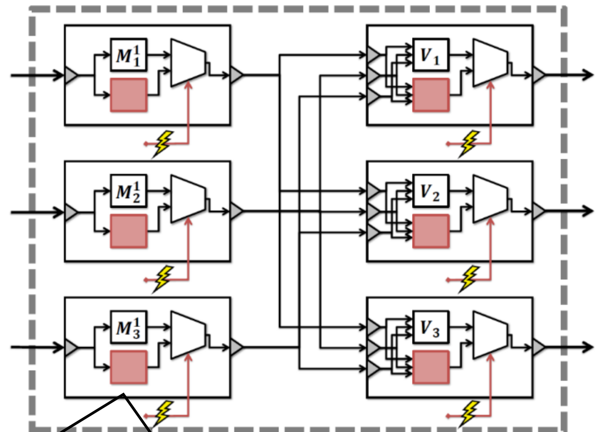


Modeling of the extended system



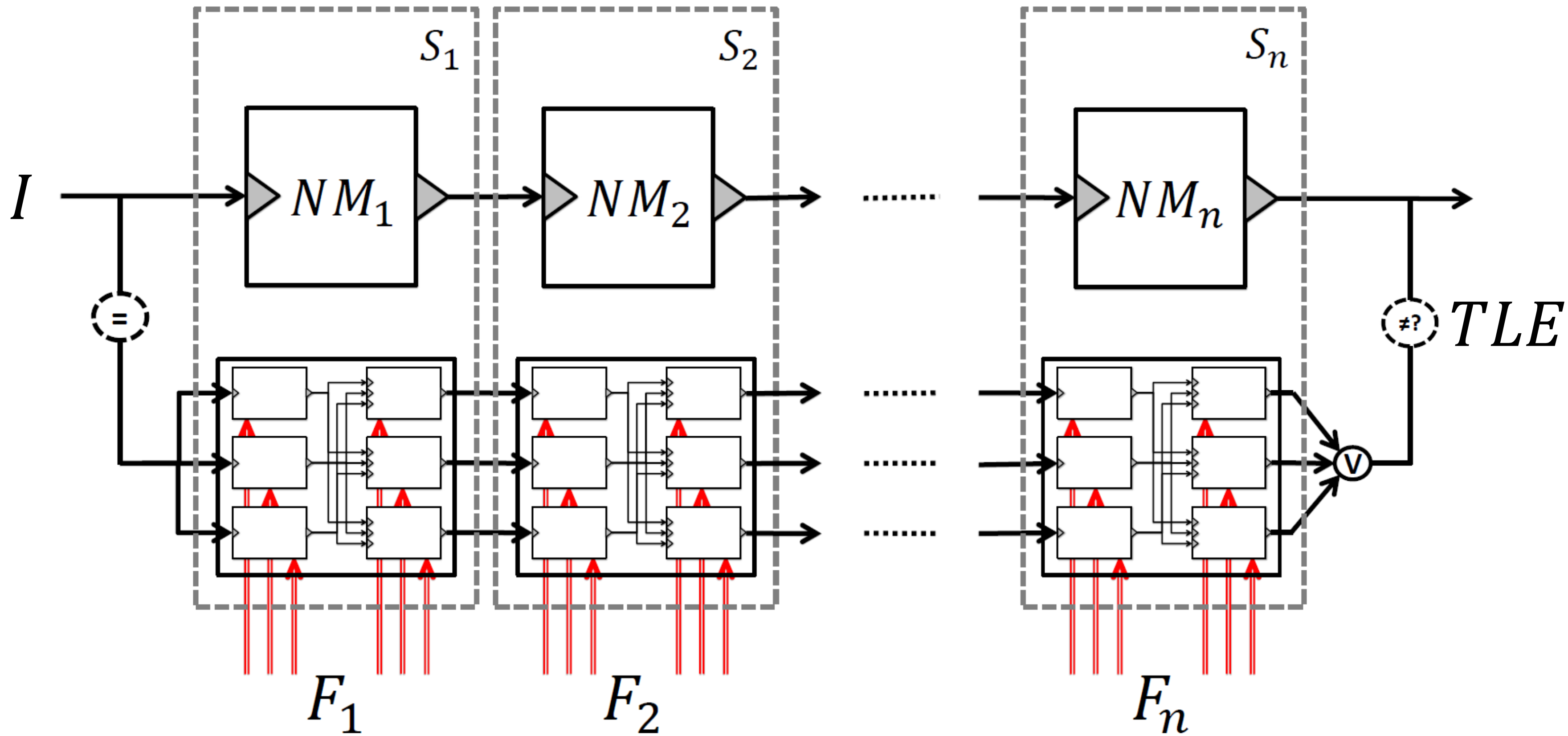
- Duplicate the behavior (nominal and faulty)
- Introduce a multiplexer, triggered by the fault event
- Model the (generic) behavior of components using uninterpreted functions (e.g. $\mathrm{x} = \mathrm{y} \rightarrow f(x) = f(y)$)

Modeling of the extended system




```
MODULE extended_component(nominal_function , fault , input)

VAR
  fault_mode : boolean;

FUN
  faulty_function : real -> real;

ASSIGN
  init(fault_mode) := FALSE;
  next(fault_mode) := fault;

DEFINE
  output: =
    case
      (fault_mode = TRUE) : faulty_function(input);
      TRUE : nominal_function(input);
    esac;
```

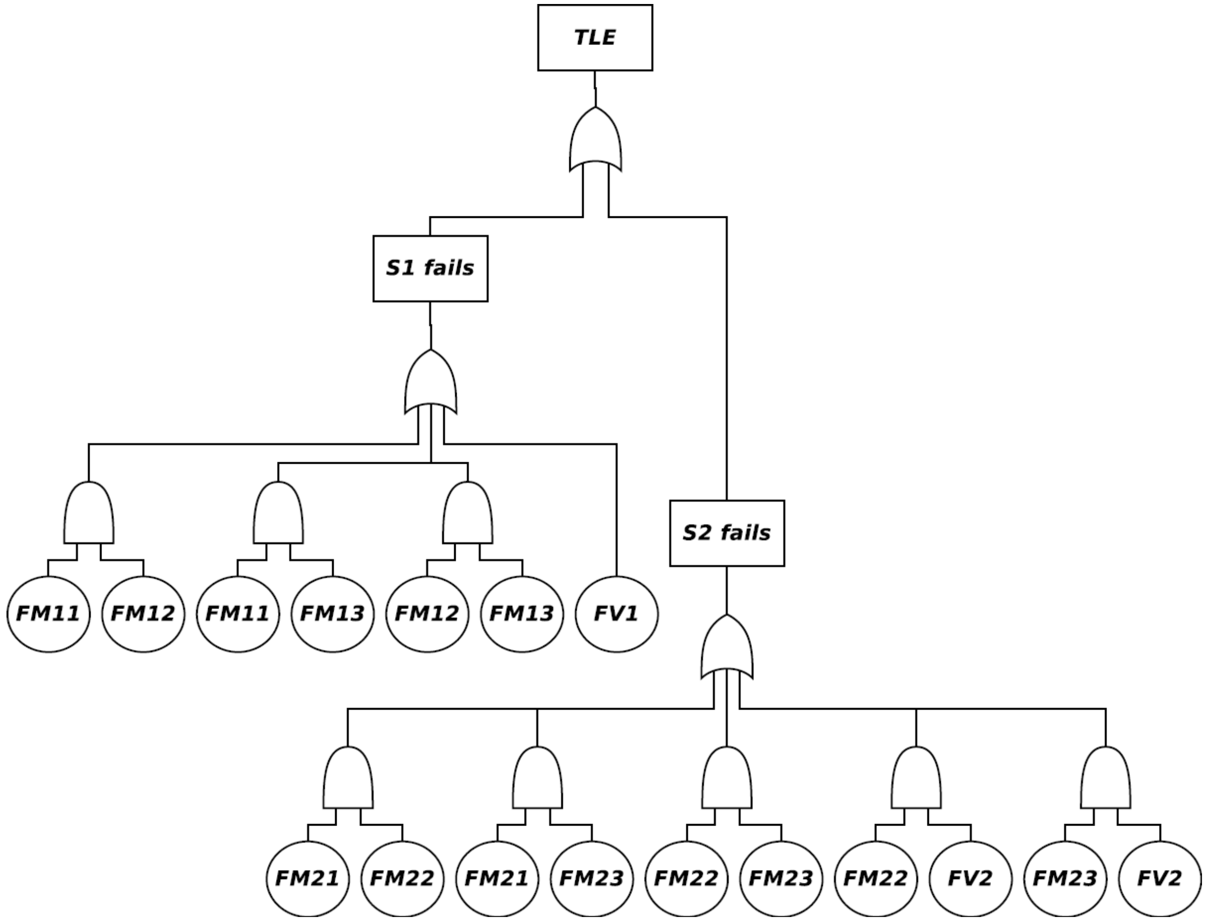
Fault Tree Analysis: equivalence check



$$TLE(I,F) = Nominal(I) \neq Redundant(I,F)$$

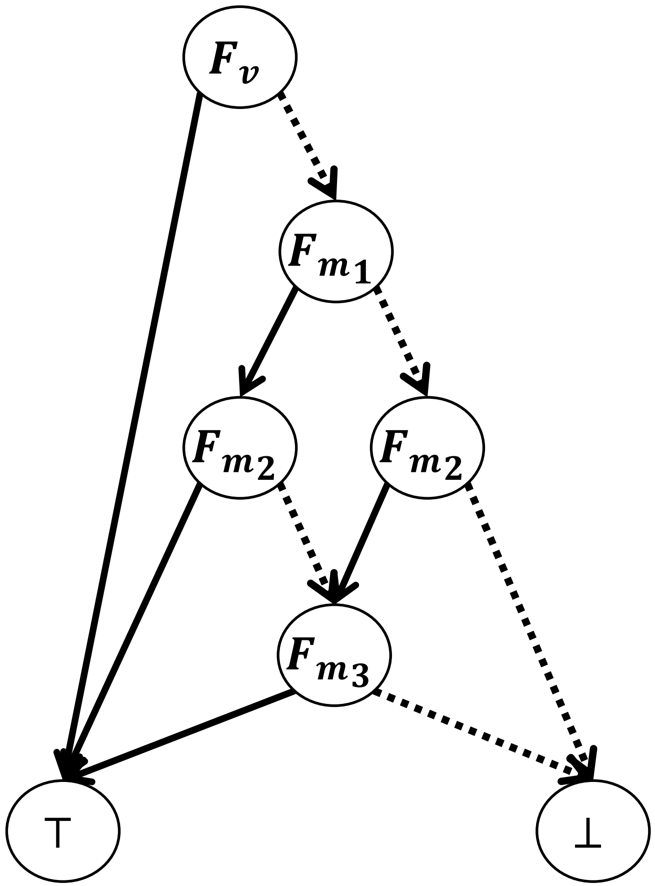
$$FT(F,TLE) = \{f \in 2^F \mid \exists i \in I.\, TLE(i,f) \wedge f \text{ is minimal}\}$$

Fault Tree Analysis: equivalence check



$$FT = (FM_{11} \wedge FM_{12}) \vee (FM_{11} \wedge FM_{13}) \vee \cdots \vee (FM_{23} \wedge FV_2)$$

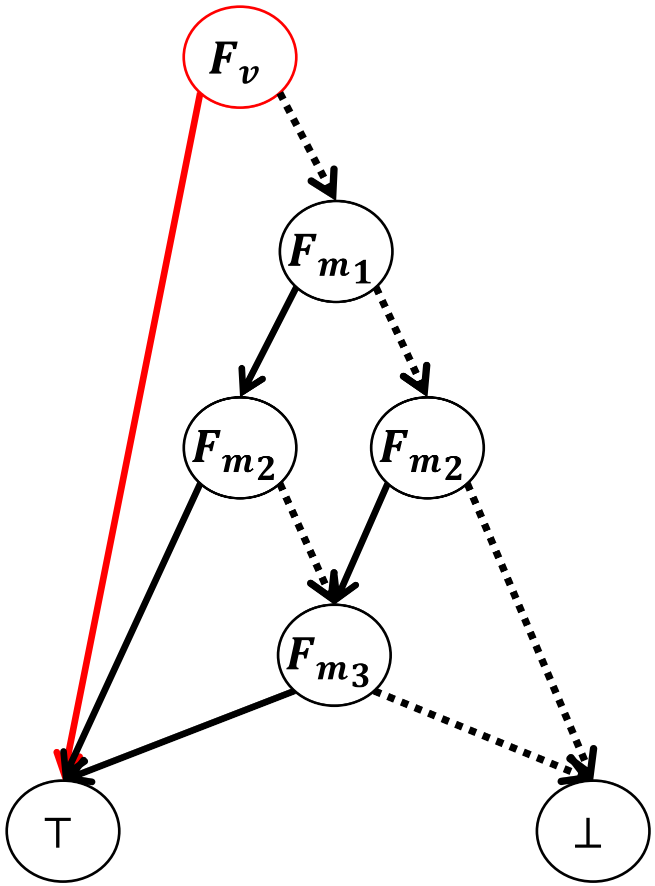
Reliability Function Extraction



$$F_{sys}: \mathbb{R}^{[0,1]} \times \cdots \times \mathbb{R}^{[0,1]} \longmapsto \mathbb{R}^{[0,1]}$$

BDD representation of the Fault Tree

Reliability Function Extraction



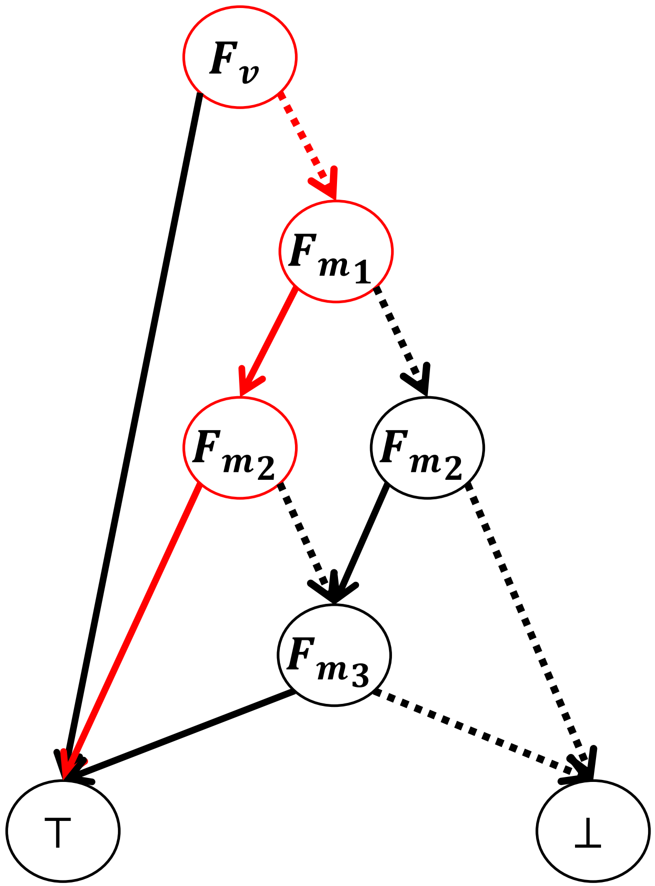
BDD representation of the Fault Tree

$$F_{sys} : \mathbb{R}^{[0,1]} \times \cdots \times \mathbb{R}^{[0,1]} \longmapsto \mathbb{R}^{[0,1]}$$

$$F_{sys}(F_v, F_{m1}, F_{m2}, F_{m3}) =$$

$$\boldsymbol{F_v} +$$

Reliability Function Extraction



BDD representation of the Fault Tree

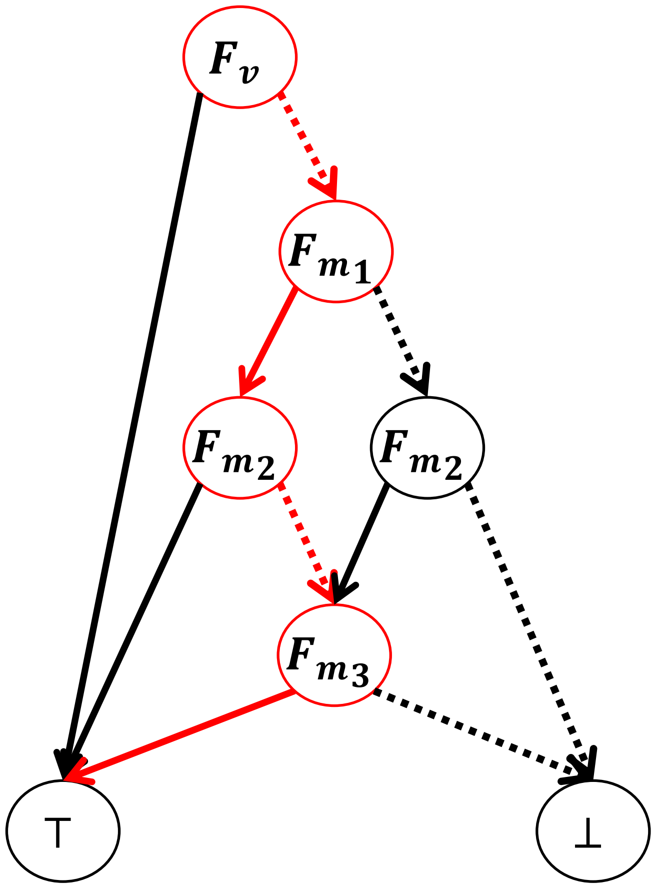
$$F_{sys} : \mathbb{R}^{[0,1]} \times \cdots \times \mathbb{R}^{[0,1]} \longmapsto \mathbb{R}^{[0,1]}$$

$$F_{sys}(F_v, F_{m1}, F_{m2}, F_{m3}) =$$

$$F_v +$$

$$+(\mathbf{1} - \boldsymbol{F_v}) * \boldsymbol{F_{m1}} * \boldsymbol{F_{m2}} +$$

Reliability Function Extraction



BDD representation of the Fault Tree

$$F_{sys} : \mathbb{R}^{[0,1]} \times \cdots \times \mathbb{R}^{[0,1]} \longmapsto \mathbb{R}^{[0,1]}$$

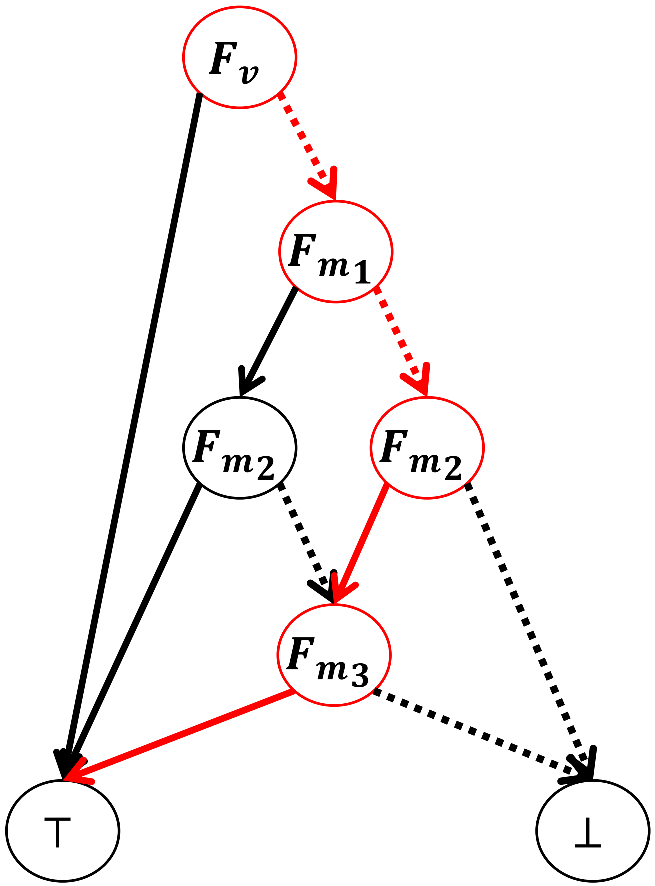
$$F_{sys}(F_v, F_{m1}, F_{m2}, F_{m3}) =$$

$$F_v +$$

$$+(1 - F_v) * F_{m1} * F_{m2} +$$

$$\boldsymbol{+(1 - F_v) * F_{m1} * (1 - F_{m2}) * F_{m3}} +$$

Reliability Function Extraction



BDD representation of the Fault Tree

$$F_{sys} : \mathbb{R}^{[0,1]} \times \cdots \times \mathbb{R}^{[0,1]} \longmapsto \mathbb{R}^{[0,1]}$$

$$F_{sys}(F_v, F_{m1}, F_{m2}, F_{m3}) =$$

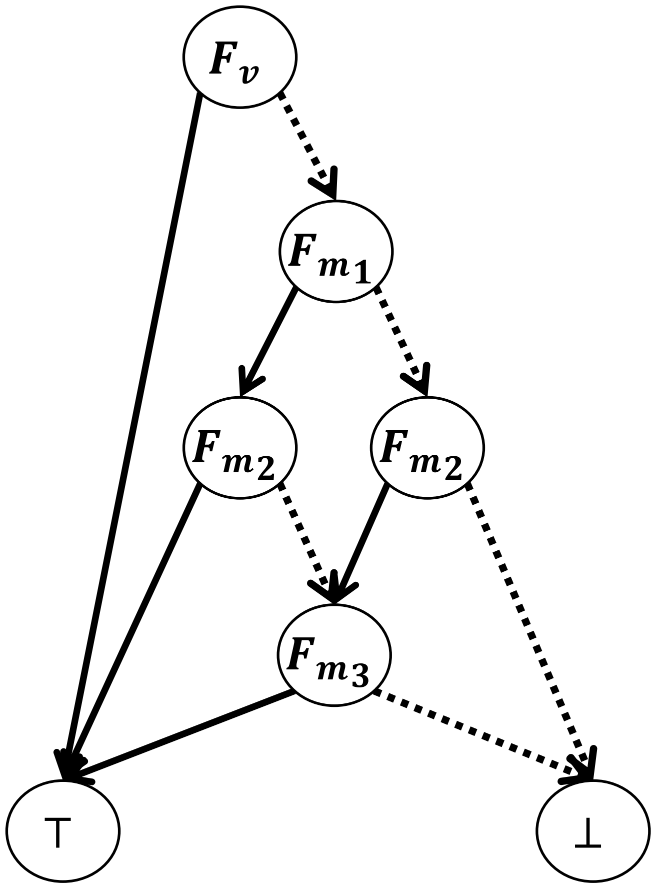
$$F_v +$$

$$+(1 - F_v) * F_{m1} * F_{m2} +$$

$$+(1 - F_v) * F_{m1} * (1 - F_{m2}) * F_{m3} +$$

$$+\boldsymbol{(1 - F_v) * (1 - F_{m1}) * F_{m2} * F_{m3}}$$

Reliability Function Extraction



BDD representation of the Fault Tree

$$F_{sys} : \mathbb{R}^{[0,1]} \times \cdots \times \mathbb{R}^{[0,1]} \longmapsto \mathbb{R}^{[0,1]}$$

$$F_{sys}(F_v, F_{m1}, F_{m2}, F_{m3}) =$$

$$F_v +$$

$$+(1 - F_v) * F_{m1} * F_{m2} +$$

$$+(1 - F_v) * F_{m1} * (1 - F_{m2}) * F_{m3} +$$

$$+(1 - F_v) * (1 - F_{m1}) * F_{m2} * F_{m3}$$

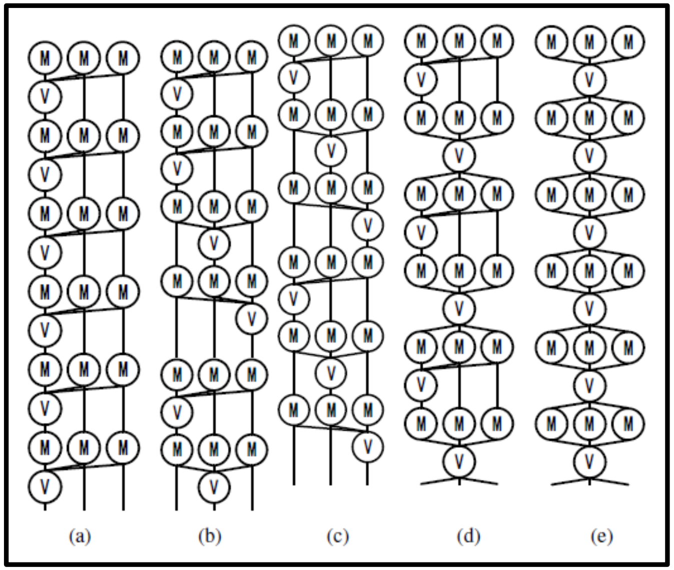
Automated Analysis of Reliability Architecture

1. Model the extended system with uninterpreted functions
2. Perform Fault Tree Analysis
3. Extract Reliability Function, from BDD representation of Fault Tree

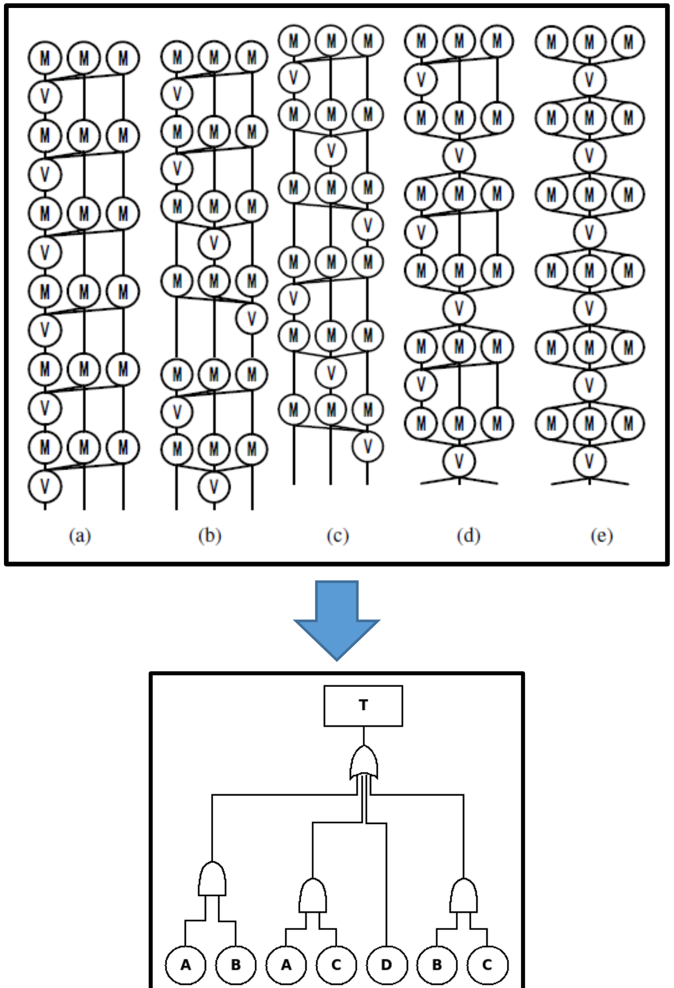
Automated Analysis of Reliability Architecture

1. Model the extended system with uninterpreted functions
2. Perform Fault Tree Analysis
3. Extract Reliability Function, from BDD representation of Fault Tree
4. Evaluate the results with analytical tools (Octave/Matlab)

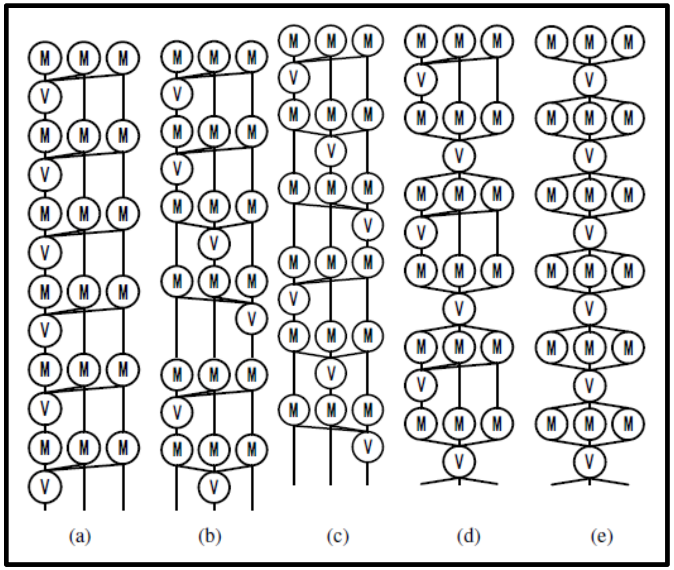
Automated Analysis of Reliability Architecture

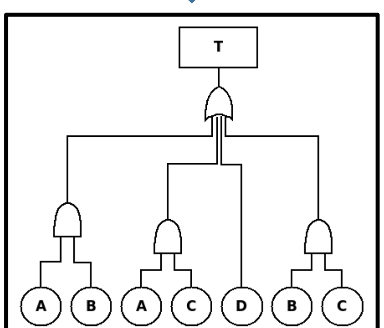


Automated Analysis of Reliability Architecture



Automated Analysis of Reliability Architecture

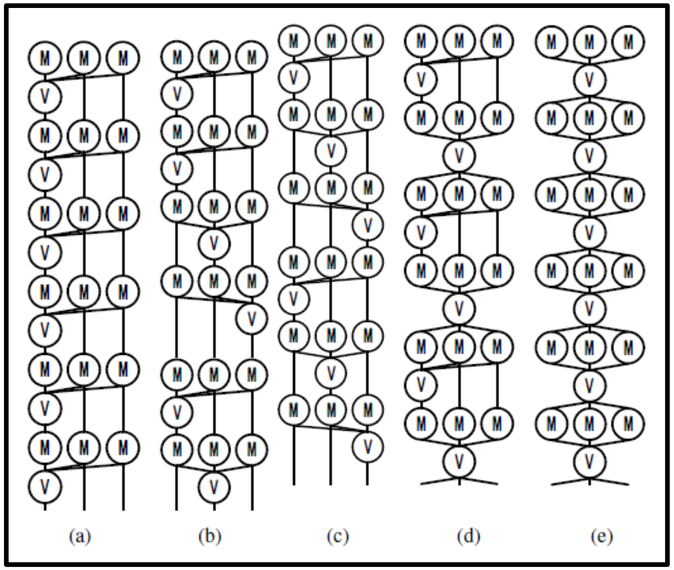


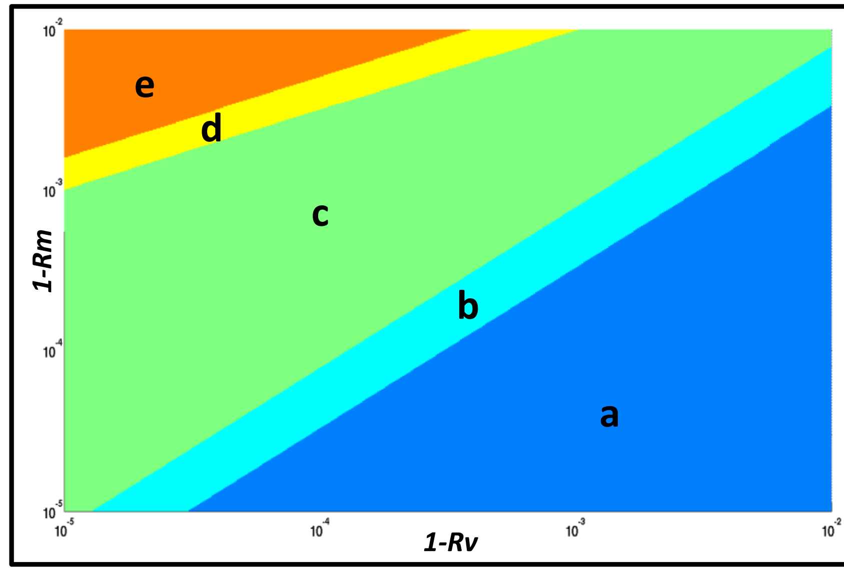


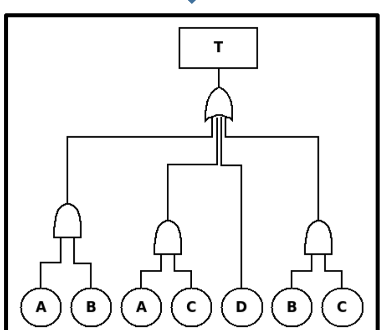
$$F_{sys} = F_v + 3 * F_m^2 - 3 * F_v * F_m^2 \ \ldots$$

Automated Analysis of Reliability Architecture

Triple Redundant Module comparison (1 voter)

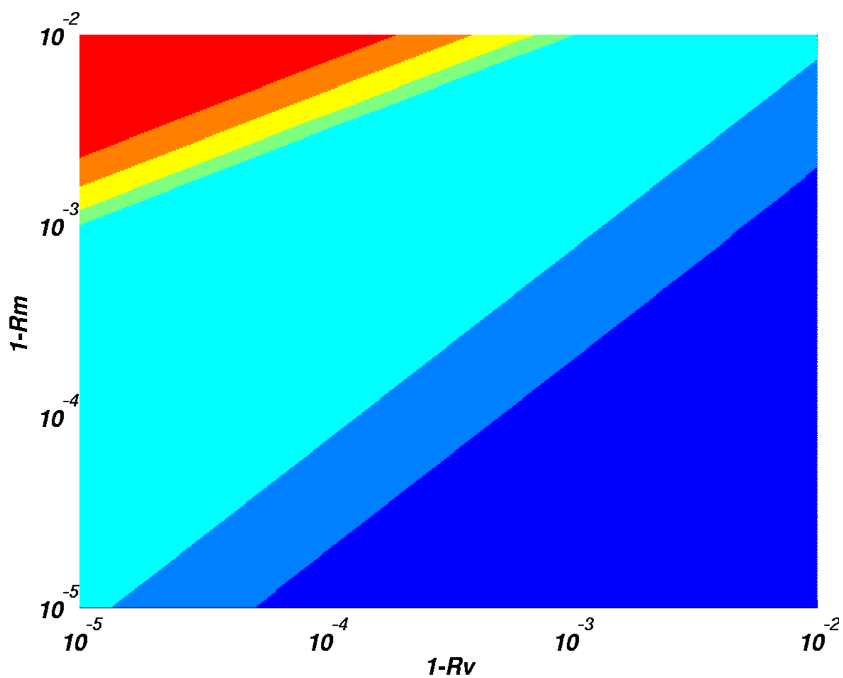




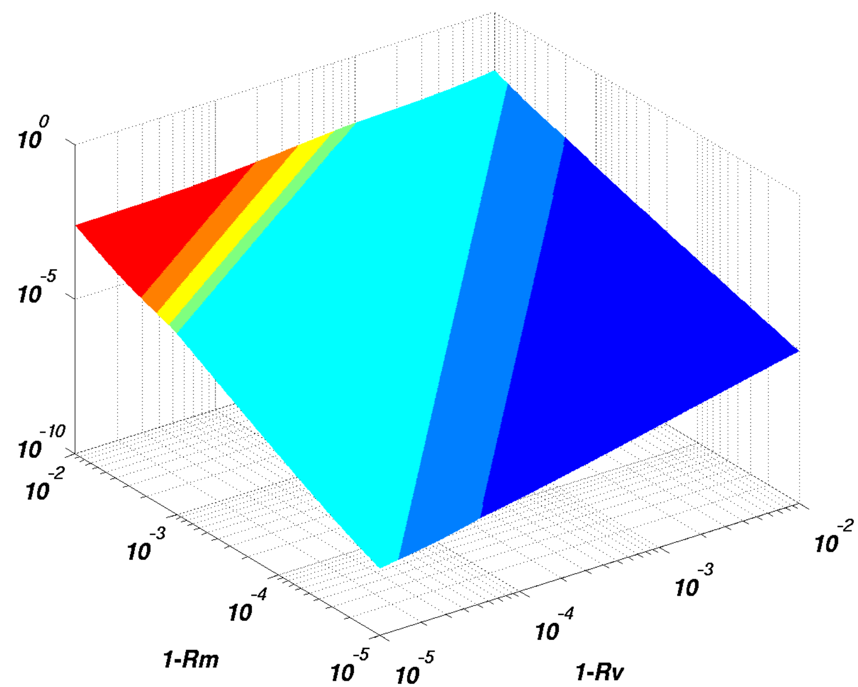


$$F_{sys} = F_v + 3 * F_m^2 - 3 * F_v * F_m^2 \; ...$$

Uniform probability analysis

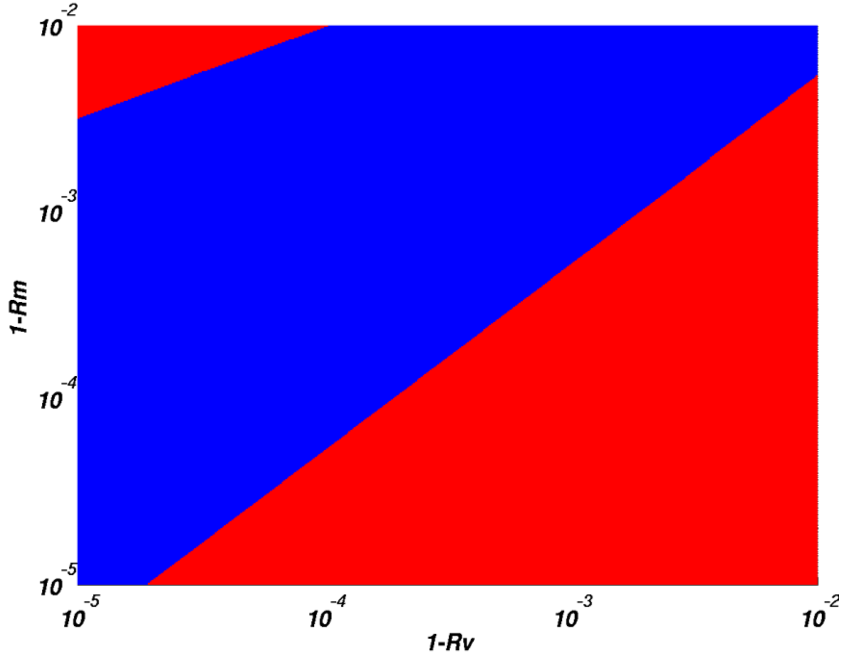


1 voter patterns comparison
(2D)

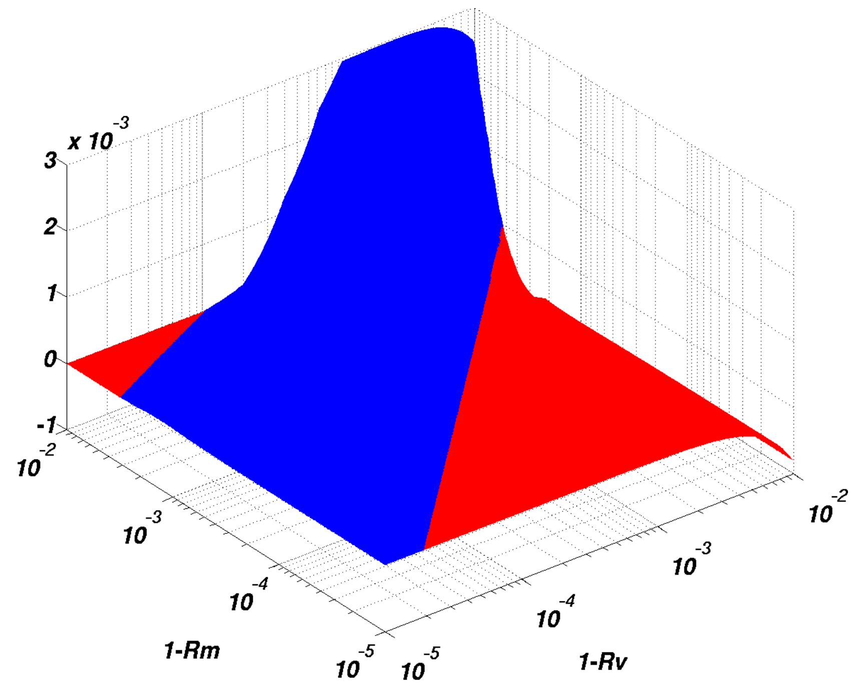


1 voter patterns comparison
(3D)

Uniform probability analysis

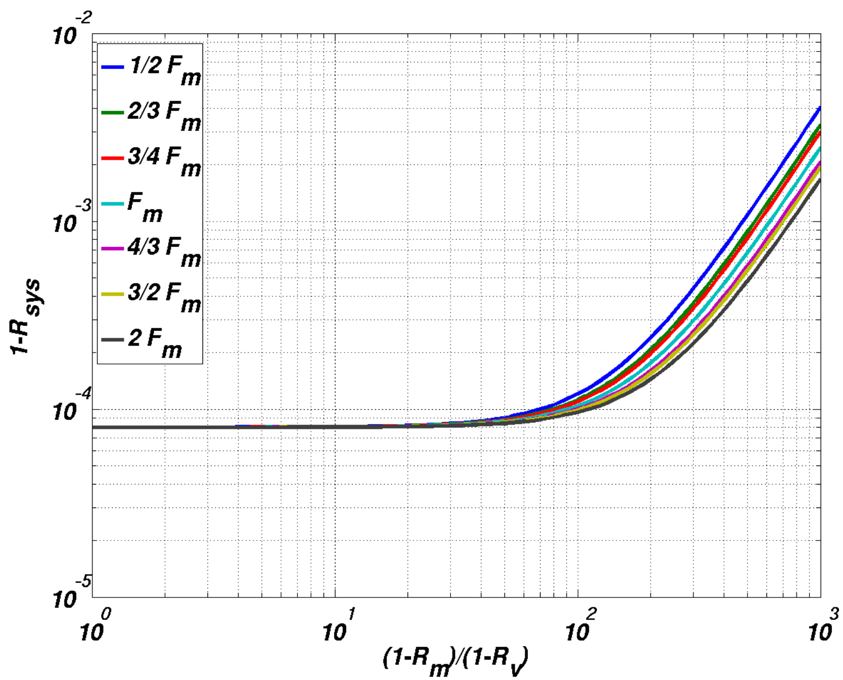


1 vs 2 voters comparison
(2D)

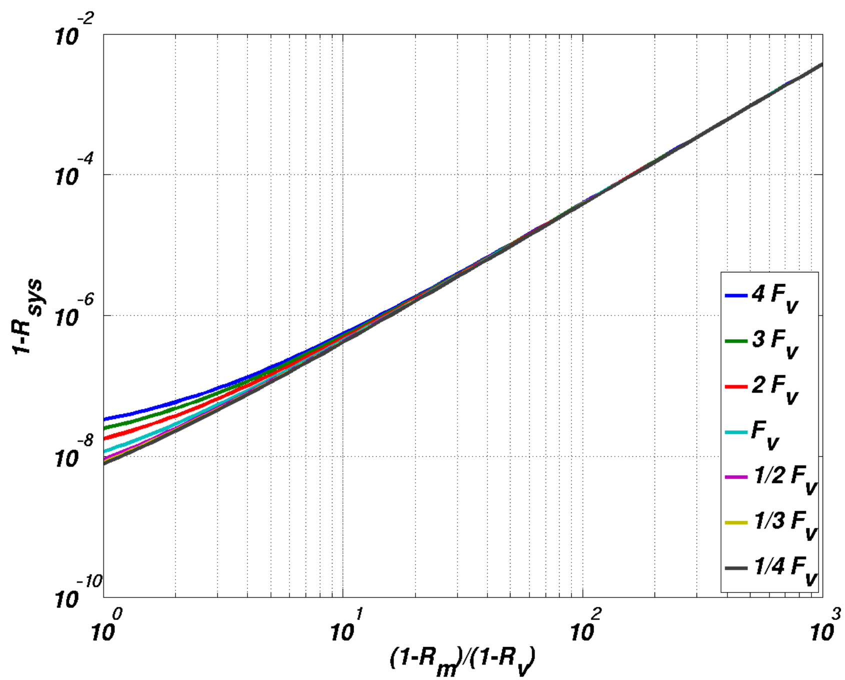


1 vs 2 voters comparison
(3D)

Not uniform probability analysis



Varying F_m for M_1 (1 voter)



Varying F_v for V_1 (2 voters)

Automated Analysis of Reliability Architectures

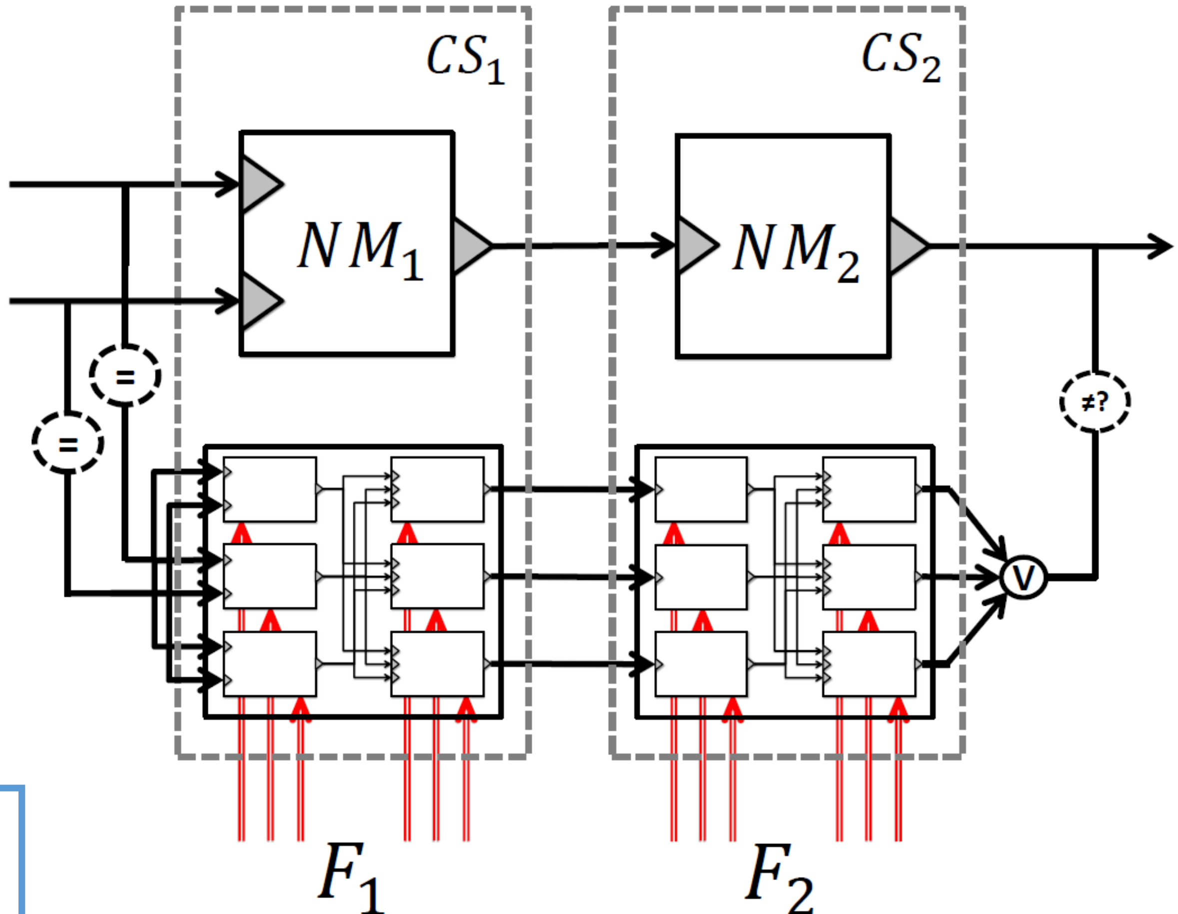
- Full automated technique for the Analysis of Reliability Architecture
- Symbolic technique (it generates the closed form of Reliability function)
- Allows for the reusability of analysis results (i.e. generation of Reliability Functions Libraries)

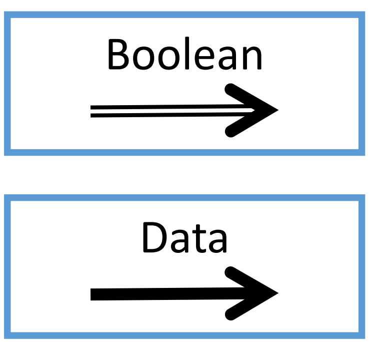
- AllSMT approach: Hard to deal with big system definition (> 10 stages)

Outline

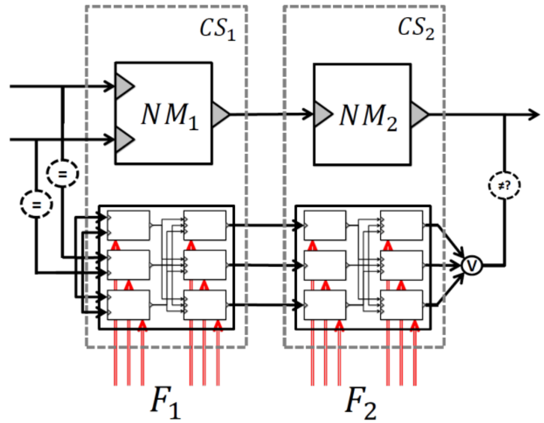
- Architectural Design in Critical Systems
  - Redundant Systems
  - Reliability Analysis
- **Automated Approaches**
  - EUF modeling and Fault Tree Analysis
  - **Efficient Analysis via Predicate Abstraction**
- Conclusion

Modular Abstraction





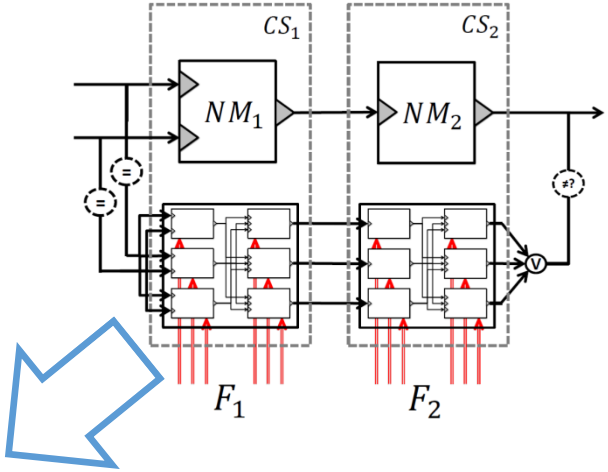
Modular Abstraction

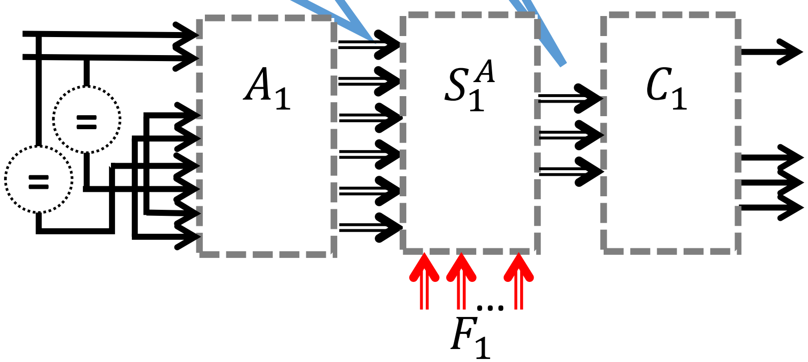


Modular Abstraction

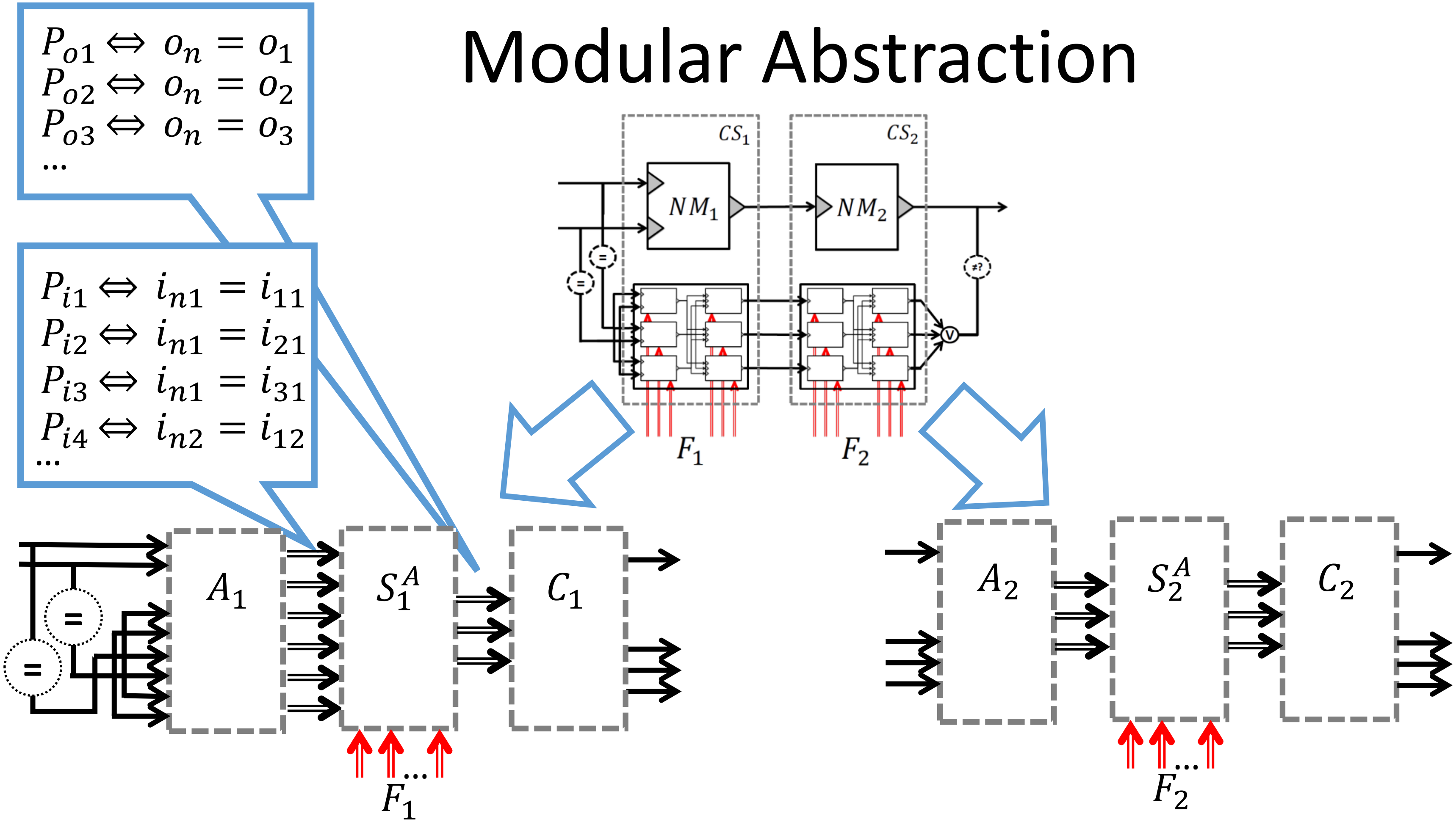
$P_{o1} \Leftrightarrow o_n = o_1$
$P_{o2} \Leftrightarrow o_n = o_2$
$P_{o3} \Leftrightarrow o_n = o_3$
...

$P_{i1} \Leftrightarrow i_{n1} = i_{11}$
$P_{i2} \Leftrightarrow i_{n1} = i_{21}$
$P_{i3} \Leftrightarrow i_{n1} = i_{31}$
$P_{i4} \Leftrightarrow i_{n2} = i_{12}$
...





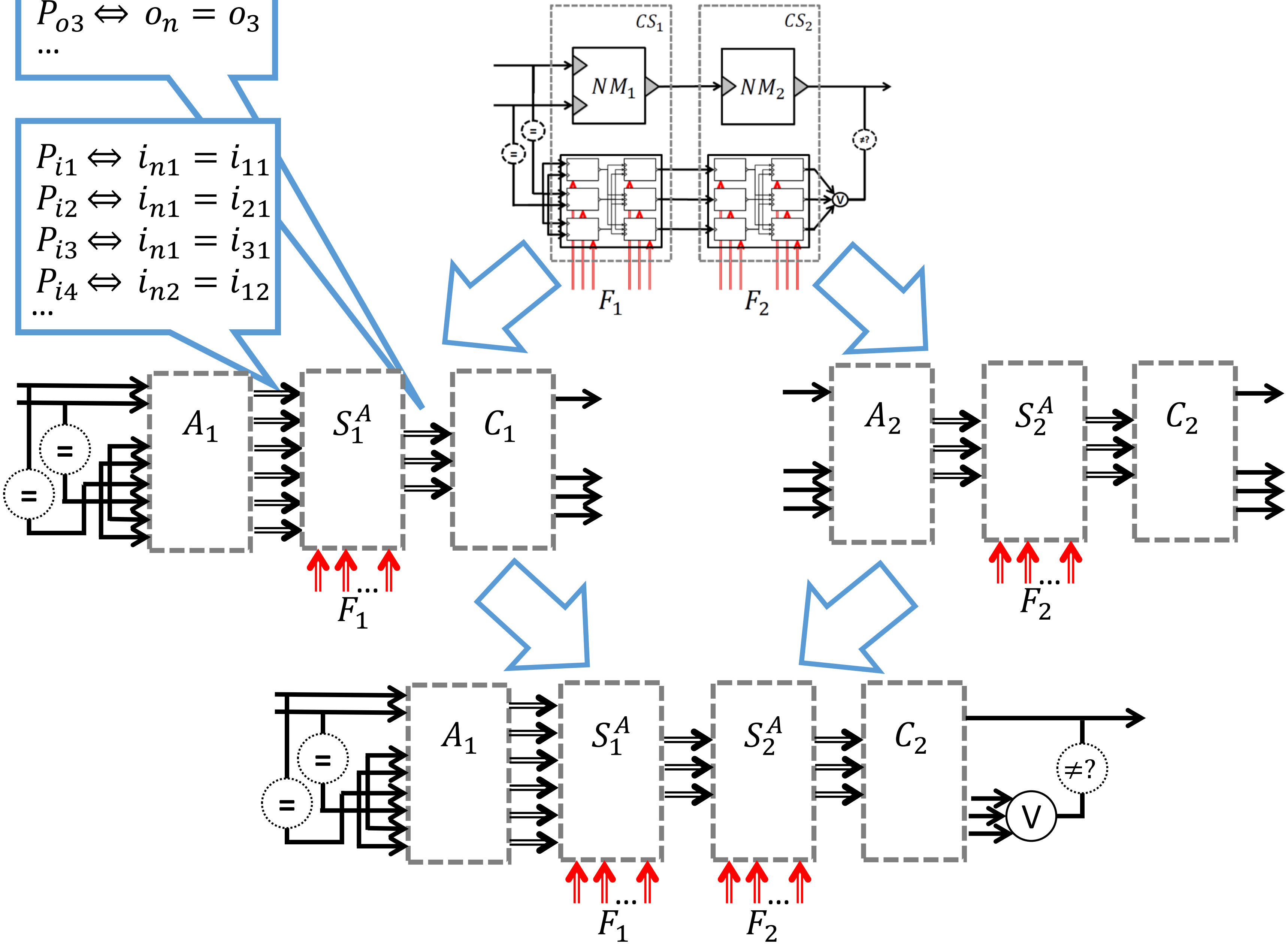
Modular Abstraction



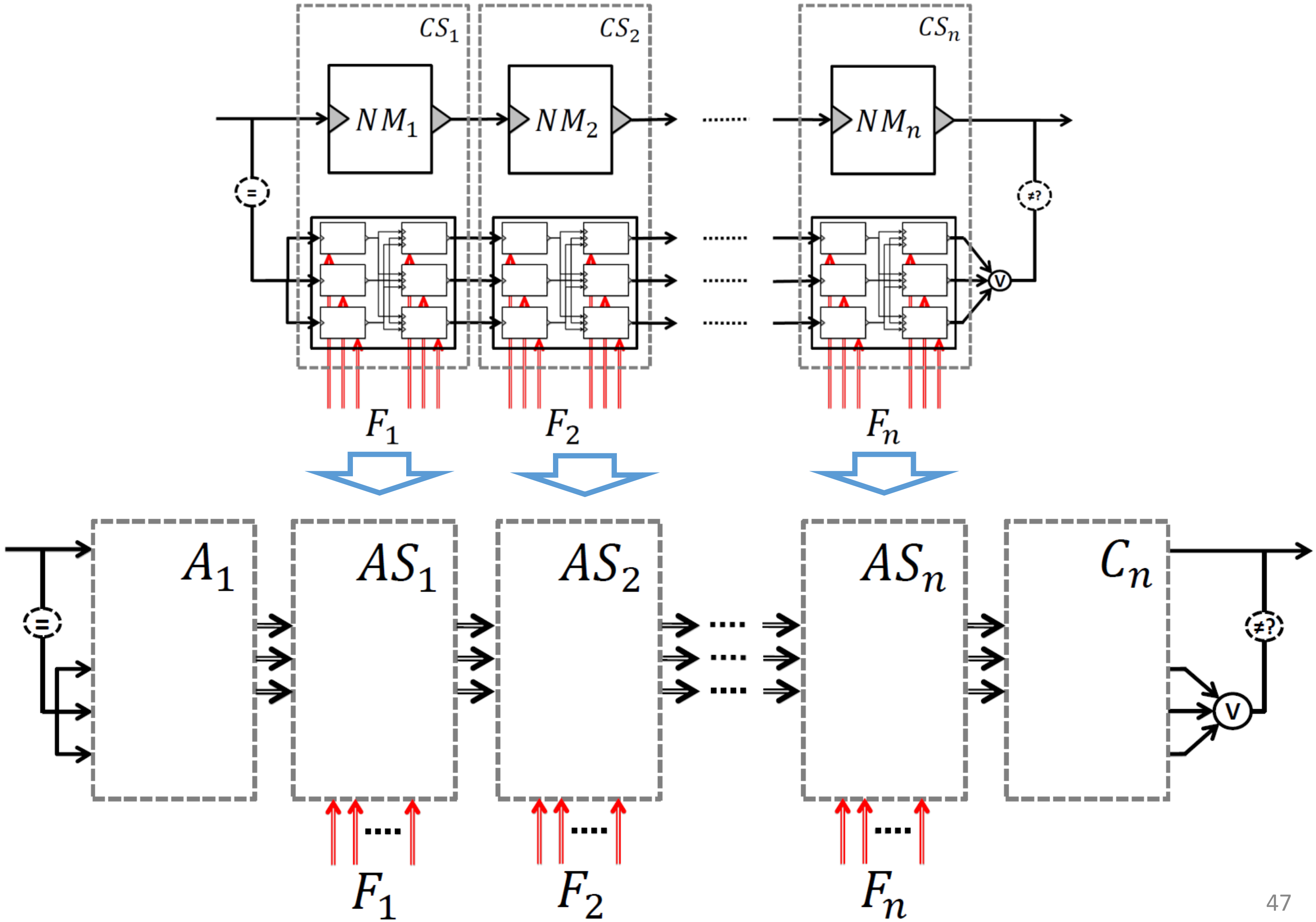
Modular Abstraction

$P_{o1} \Leftrightarrow o_n = o_1$
$P_{o2} \Leftrightarrow o_n = o_2$
$P_{o3} \Leftrightarrow o_n = o_3$
…

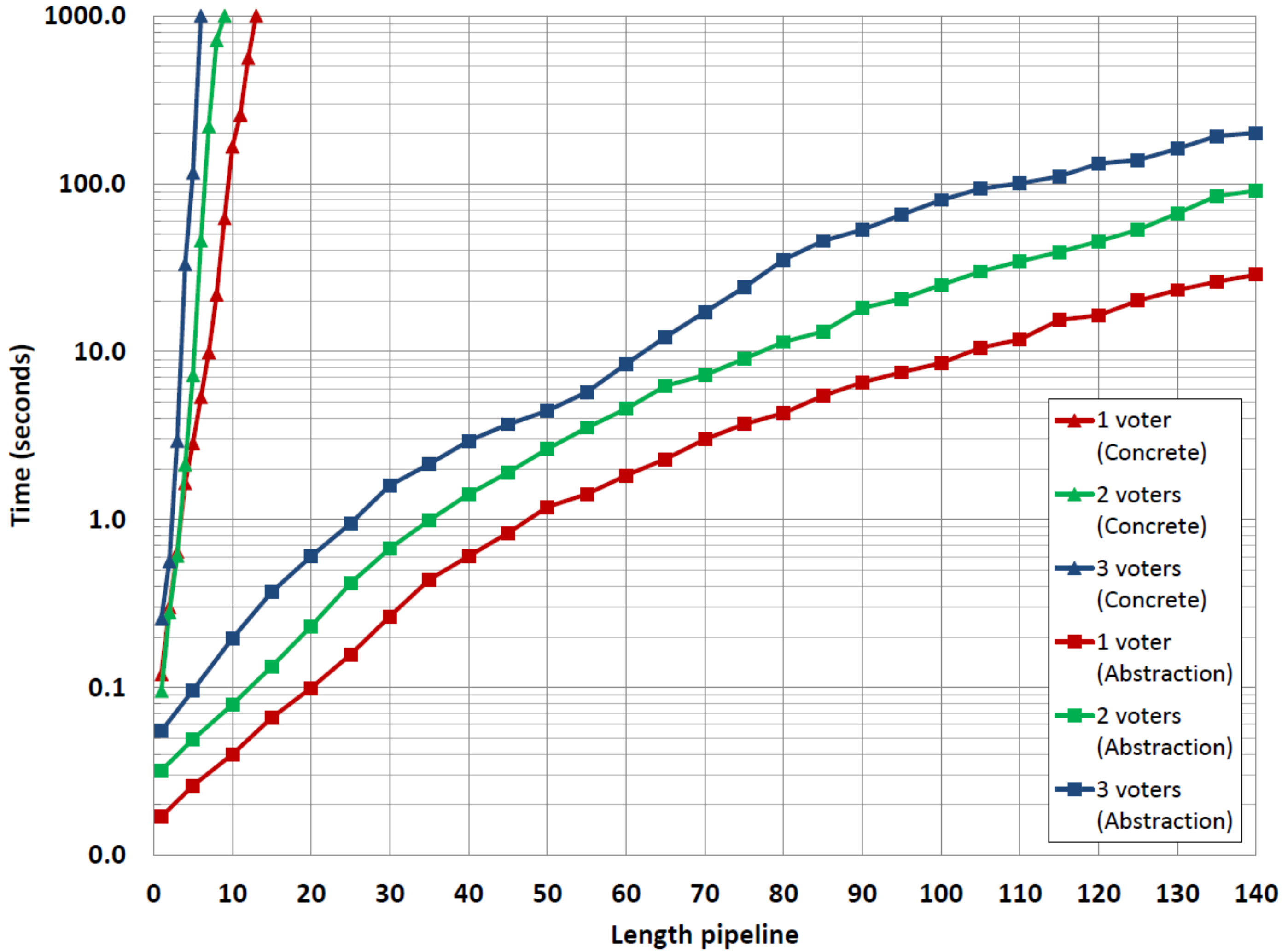
$P_{i1} \Leftrightarrow i_{n1} = i_{11}$
$P_{i2} \Leftrightarrow i_{n1} = i_{21}$
$P_{i3} \Leftrightarrow i_{n1} = i_{31}$
$P_{i4} \Leftrightarrow i_{n2} = i_{12}$
…



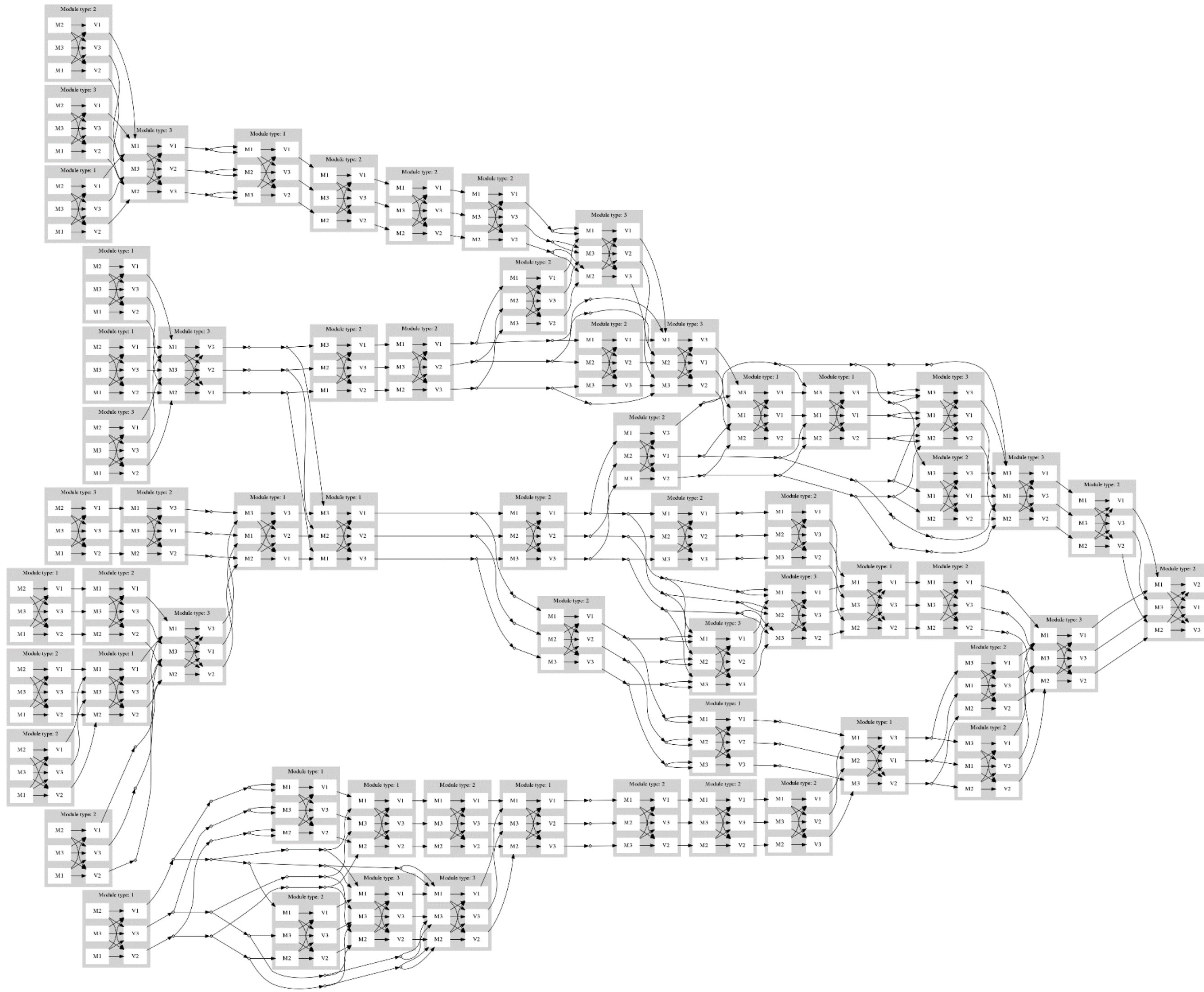
Modular Abstraction



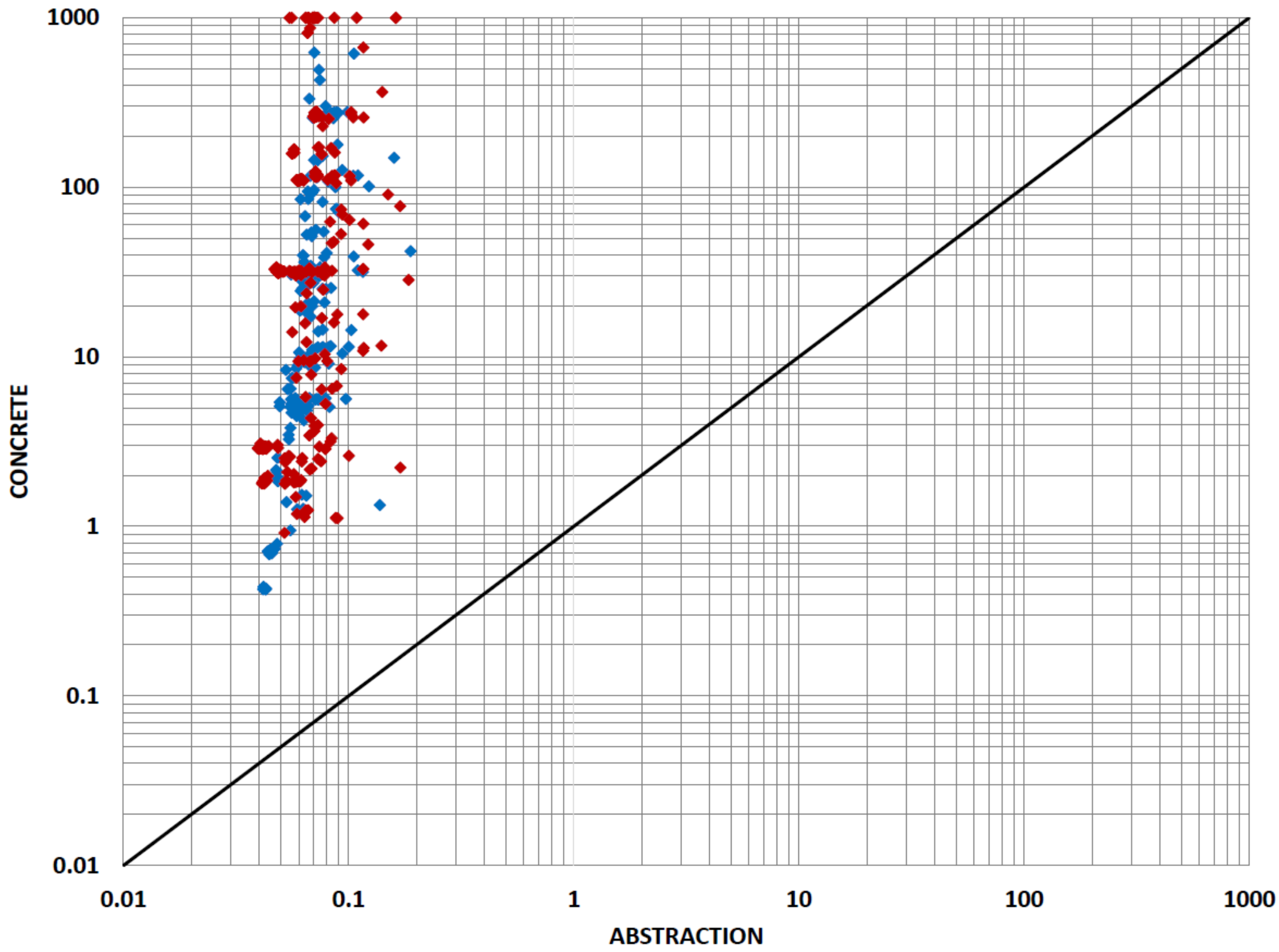
Concrete vs Abstraction: linear



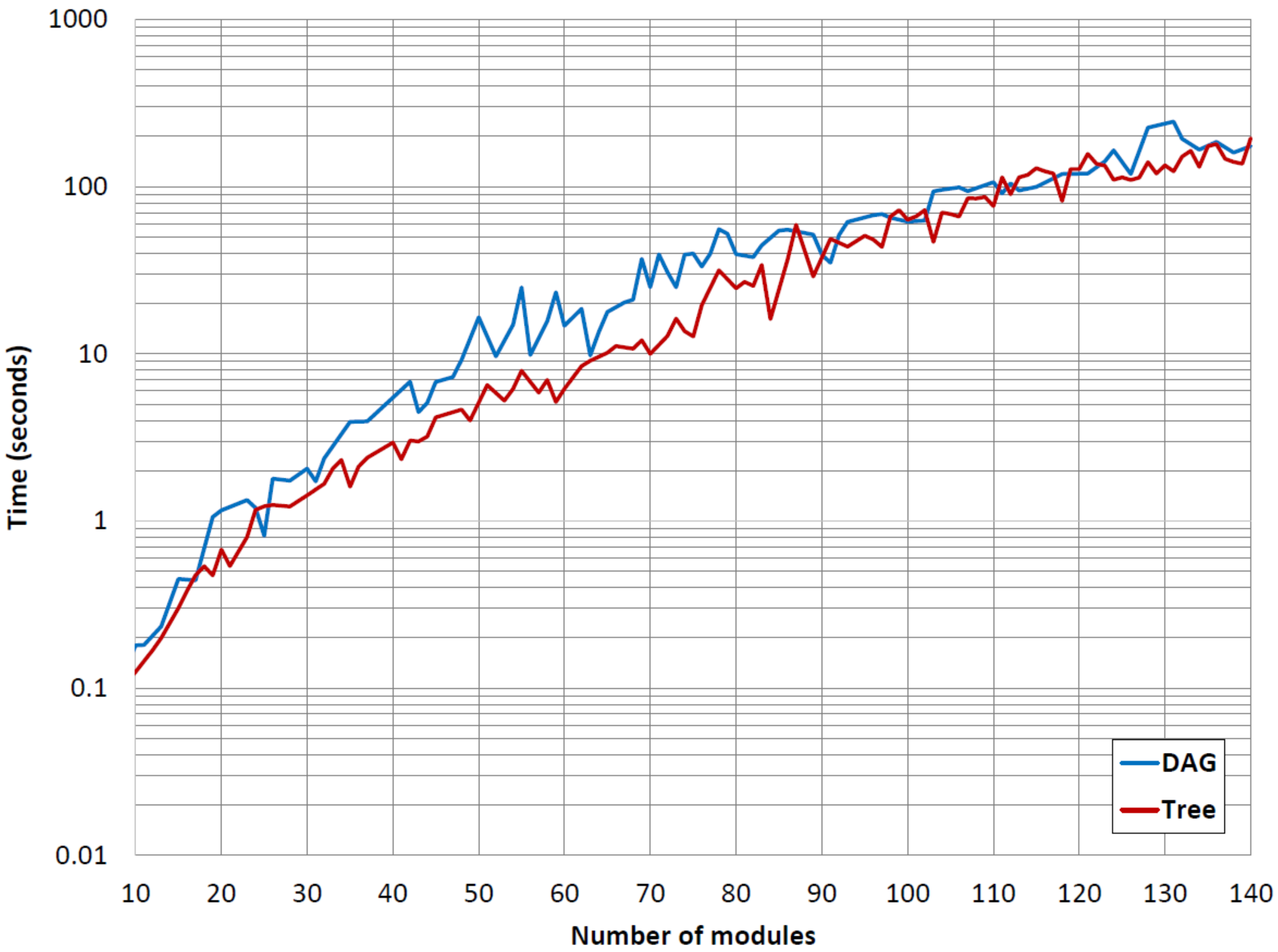
DAG like example with 60 modules



Concrete vs Abstraction: Tree and DAG (< 15 modules)



Abstraction: Tree and DAG



Outline

- Architectural Design in Critical Systems
  - Redundant Systems
  - Reliability Analysis
- Manual Reliability techniques
- Automated Approaches
  - EUF modeling and Fault Tree Analysis
  - Efficient Analysis via Predicate Abstraction
- **Conclusion**

Conclusion

- Automated technique for the analysis of reliability architectures
- Management of linear, Tree and DAG like structures
- Efficient analysis of large systems (> 140 modules) via predicate abstraction

Automated Analysis of Reliability Architectures
Marco Bozzano, Alessandro Cimatti and Cristian Mattarei
In proc. of ICECCS 2013

Efficient Analysis of Reliability Architectures via Predicate Abstraction
Marco Bozzano, Alessandro Cimatti and Cristian Mattarei
Under review of FMCAD 2013

Thank you!

Cristian Mattarei
Fondazione Bruno Kessler
FBK ES-Group
mattarei@fbk.eu