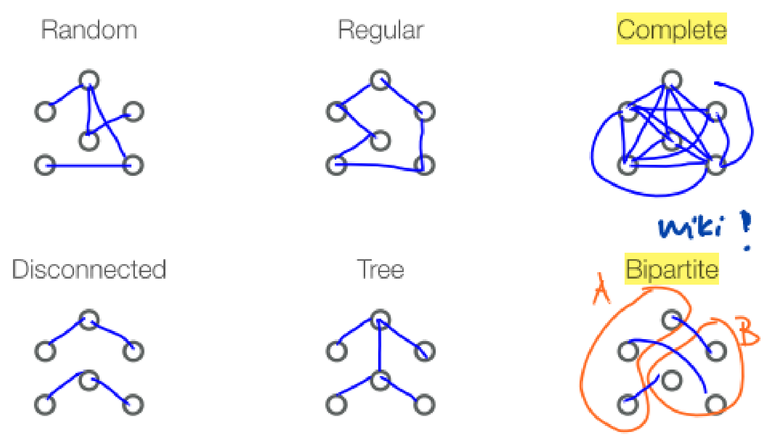
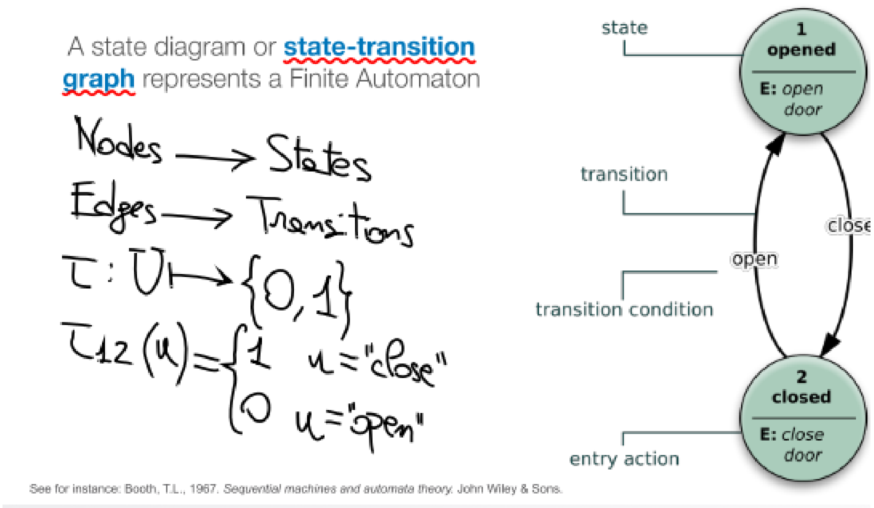
02_Structural Analysis

1. Some Graph Knowledge



2. Finite State Automaton



where τ is some kind of transition activation function

3. Bi-Partite Graph and Regular Graph

Definition: Bi-Partite Graph

A **bipartite graph**, also called a **bigraph**, is a set of graph vertices decomposed into **two disjoint sets** such that **no two graph vertices within the same set are adjacent.**

Definition: Regular Graph

A **regular graph** is a graph where each vertex has the same number of neighbors; i.e. every vertex has the same degree or valency

Matching in Bi-Partite Graph

Definition: Matching

A **matching** in a Bipartite Graph is a set of the edges chosen in such a way that no two edges share an endpoint.

Definition: Maximum Matching

A **maximum matching** is a matching of maximum size (maximum number of edges)

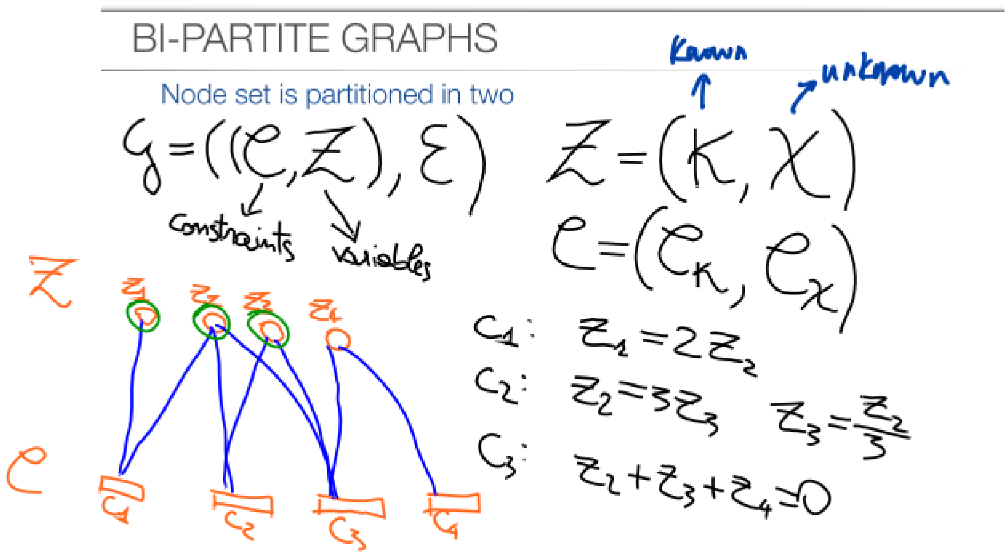
Other Definitions:

- An edge is said to be **weak** with respect to M if it does not belong to M. A vertex is **weak** with respect to M if it is only incident to weak edges
- An M**-augmenting path** is an alternating path whose end vertices are both weak with respect to M

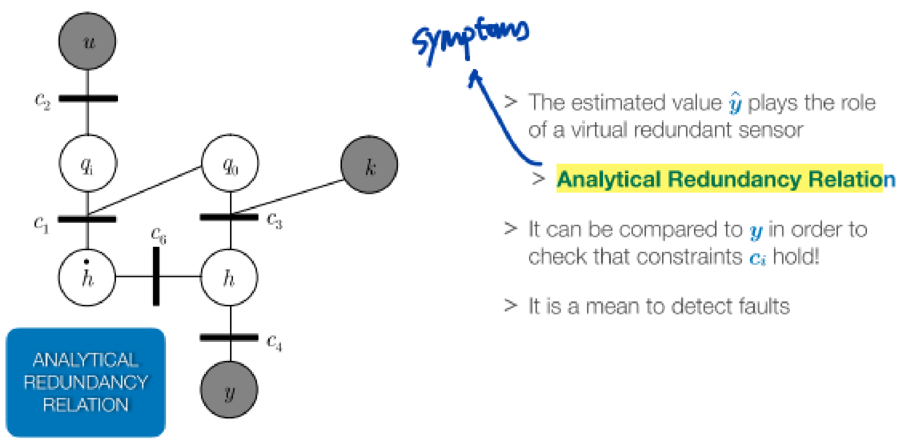
Theorem:

A matching M in a graph G is maximum **if and only if** there exists no M-augmenting path in G

Bi-Partite Usage 1: Define Variable Relations



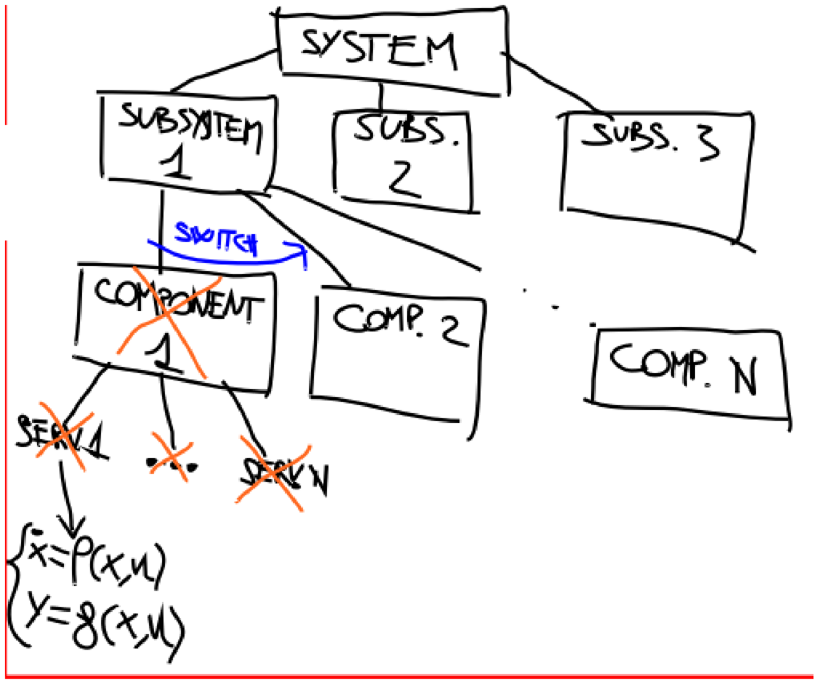
Bi-Partite Usage 2: Discover Analytical Redundancy Relation



The output y is measured, but can also be computed from u and k

3. Components and Services

Components and Service model can be used to **organize knowledge** of the system in a **hierarchical way**



Components

- **Components** provide **services**
- e.g. A tank integrates inflow - outflow to produce a stored mass

Service

- A **service** s_i is described by a 6-tuple

$$s=\{cons, prod, proc, rqst, enable, res\}$$
$$cons=\{q_i, q_o\}$$
$$prod=\{h\}$$
$$proc.=\{\dot h = q_i - q_o \,;\, h=\int \dot h \, dt + h_0\}$$
$$rqst=\{1\}$$
$$enable=\{1\}$$
$$res=\{vessel, pipes\}$$

- Consumed Varibles
- Produced Variables
- Processes
- Request Signals
- Enable Signals
- Resources

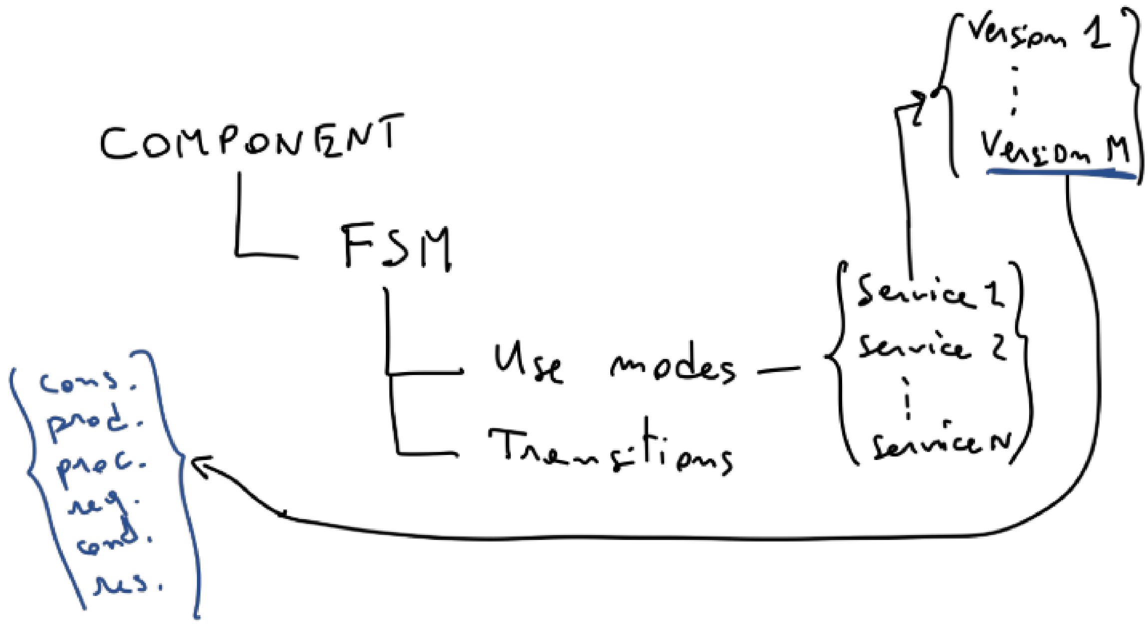
- A services come in different **versions**
- Availability of services depends on component **use mode**

General Component Model

$$< \textit{component } k > \; ::= \; < \textit{state transition graph } G(M(k), \tau(k), m^0(k)) >$$

where

$$\begin{aligned}
< M(k) > \; &::= \; < \textit{set of use-modes } \{m_i(k), i \in I_m(k)\} > \\
< \tau(k) > \; &::= \; < \textit{set of transitions } \{\tau_{ij}(k), \quad i, j \in I_m(k)\} > \\
< m^0(k) > \; &::= \; < \textit{initial use-mode} > \\
< \textit{use-mode } m_i(k) > \; &::= \; < \textit{set of services } S_i(k) \subseteq S(k) > \\
< \textit{service } s_l(k) > \; &::= \; < \textit{pre-ordered versions} \\
&\qquad \left\{ s_l^j(k), \quad j \in J(s_l(k)) \right\} > \\
< \textit{version } s_l^j(k) > \; &::= \; < \textit{consumed vars } cons_l^j(k), \\
&\qquad \textit{produced vars } prod_l(k), \\
&\qquad \textit{procedures } proc_l^j(k), \; \textit{request } rqst_l(k), \\
&\qquad \textit{activation cond. } activ_l^j(k), \\
&\qquad \textit{hardware and software resources } res_l^j(k) > \\
< \textit{transition } \tau_{ij}(k) > \; &::= \; < \textit{condition } c_{ij}(k), \; \textit{origin } m_i(k), \\
&\qquad \textit{destination } m_j(k) > .
\end{aligned}$$



Usage

- For **fault diagnosis**
  - As a simple way to do diagnosis of complex systems where components are described by discrete states (e.g. a component providing a critical service is offline instead than online)
- For fault accommodation via switching of hardware redundant components
  - If another component can provide the same service, you know that you can swap it in

3. FTA: Fault Tree Analysis

A graphical way to analyze **system dependability properties**

Assumed Knowledge

- Components and services, how they are interconnected
- A structural, qualitative knowledge is enough

Fault Tree

Fault Tree Illustration

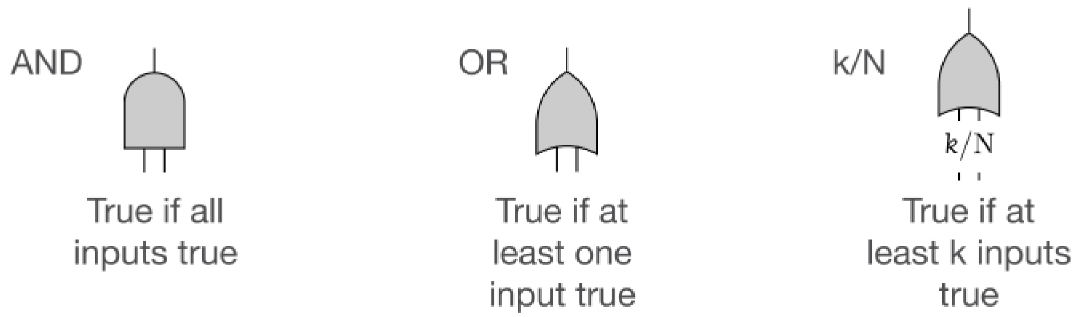
- A **Directed Acyclic Graph**
- **Leaves** represent failure of components **(Basic Events)**

- Components relations are via **Logic Gates**
- **Root** node is the Top Event: **System Failure**
- fault tree is used to **model fault** not normality!

Events

- binary variables: **true means failure, false means healthy**

Gates



Definition: Fault Tree

It is formally defined as a **4-tuple**

$$F = \langle BE, G, T, I \rangle$$

- $\boldsymbol{BE}$ is the set of **basic events** (binary: true means failure, false means healthy)
- $\boldsymbol{G}$ is the set of gates
- $E = BE \cup G$ is the set of events
- $\mathbf{T} : G \to \{\text{AND,OR,NOT,K/N}\}$ describes gate types
- $\boldsymbol{I} : G \to \mathscr{P}(E)$ maps gates to their inputs (basically represents the graph's edges set)
    - $\mathscr{P}(X)$ stands for power set of a set X, i.e. set of all possible subsets of X
    - If $T(g) = k/N$, then $|/(g)| = N$

Fault Tree Analysis

Semantic Function of a Fault Tree

To evaluate a F, a **semantic (i.e. logic) function** is introduced

$$\pi_F : \mathscr{P}(BE) \times E \mapsto \{0, 1\}$$

- $\pi_F(S, e)$ is true if the event e is "failure" when all the basic elements in S are failed

- We use the shorthand notation $\pi_F(S)$ when e is the top event
  in this case the function tells us **whether the system as a whole** will fail assuming a given subset of components are failed

Cut Sets and Minimal Cust Sets

$C \subseteq BE$ is a cut set of FTF if $\pi_F(C) = 1$. A minimal cut set (MCS) is a cut set of which no subset is a cut set, i.e. formally $C \subseteq BE$ is an MCS if $\pi_F(C) = 1 \wedge \forall_{C' \subset C} : \pi_F\left(C'\right) = 0$

Structure Function.

Let us introduce the FT **structure function** f

$$f : \{0,1\}^{|BE|} \mapsto \{0,1\}$$

$f\left(e_1, \ldots e_N\right)$ is **true** when the failure values $e_1, ...e_n$ for the N Basic Elements will result in a failure at the top event

Disjoint Normal Form

A **DNF** is a disjunction of several conjunction, e.g.:

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

Let us express f as a **Disjoint Normal Form, then every conjunction in f is a MCS**

Other Form of Fault Tree

There are many other form of fault trees, such as

Probabilistic Fault Trees

- You **propagate probabilities** of failures rather than deterministic failure events
- Good to estimate system MTTF or reliability

4. FMEA: Failure Mode and Effects Analysis

FMEA is a method to analyze system failure scenarios

- **How** can the failure of components lead to failure of the whole system
- the main difference with the FTA lies in the **"how"**

Assumed Knowledge

- **Components and services**, how they are **interconnected**
- **Fault/failure modes** of each components
- **Effects** of each mode at component level
- **How** each effect will **affect connected components**
- A structural, qualitative knowledge is enough

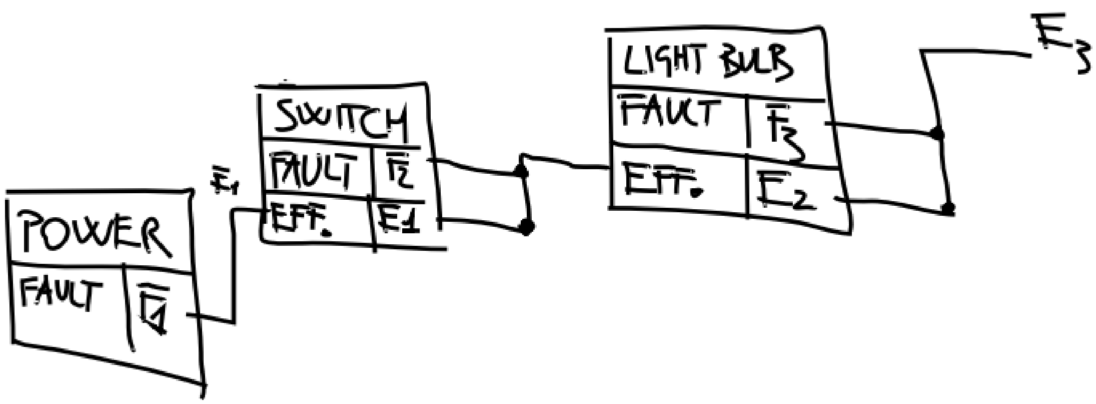
Faults, effects and propagation

Each components can fail to provide its service either because:

- internal faults (basic event in FT)
- **propagation** of faults from components it depends on

Graphical Representation

Graphical Book Representation



Boolean Matrix Representation

Matrix Representation

The **fault propagation** is actually a boolean mapping between fault vector f and effect vector e

- We can formally represent it via a boolean matrix M and a boolean operation $\otimes$

$$e = M \otimes f$$

- The operator $\otimes$ implements a disjoint normal form, as in the FTA

$$e_{(i)} = \left(M_{(i,1)} \wedge f_{(1)}\right) \vee \left(M_{(i,2)} \wedge f_{(2)}\right) \vee \cdots \vee \left(M_{(i,N)} \wedge f_{(N)}\right)$$

- **For hardware redundancy, we need to model it in the fs**

Hierarchical Composition

$$e_{\text{loc}} = M_{\text{loc}} \otimes \begin{bmatrix} f_{\text{loc}} \\ e_{\text{anc}} \end{bmatrix}$$

$$e_{\text{loc}} = \left(M_{\text{loc}} \otimes \begin{bmatrix} I & 0 \\ 0 & M_{\text{anc}} \end{bmatrix}\right) \otimes \begin{bmatrix} f_{\text{loc}} \\ f_{\text{anc}} \end{bmatrix}$$

Inverse Inference

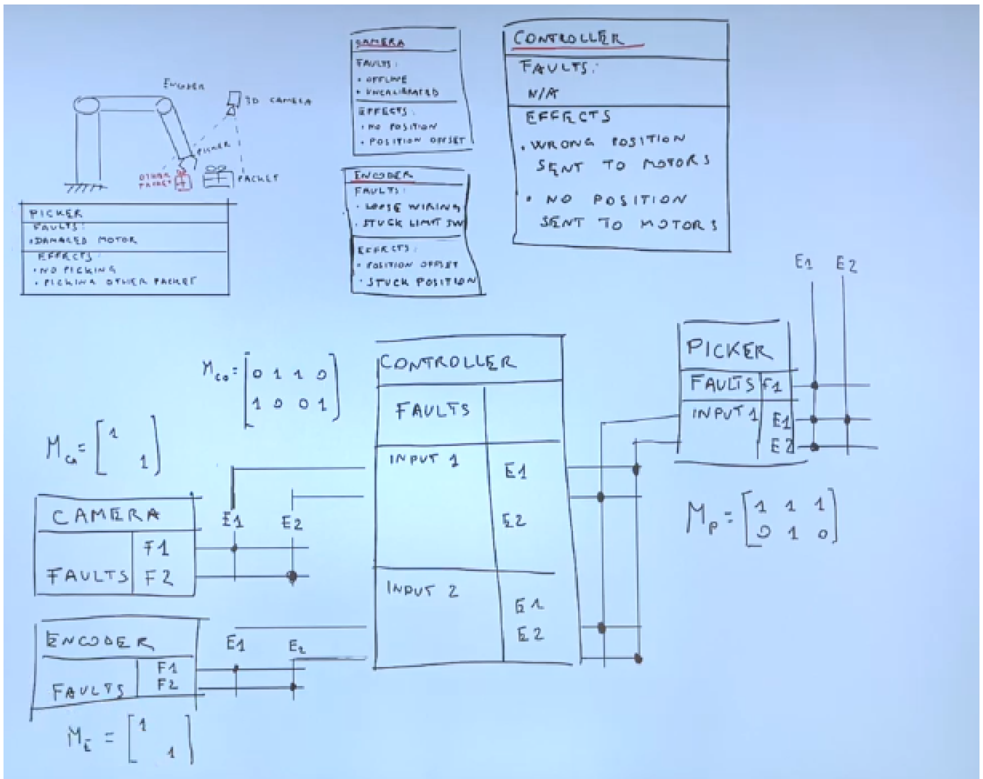
from effects to possible causes

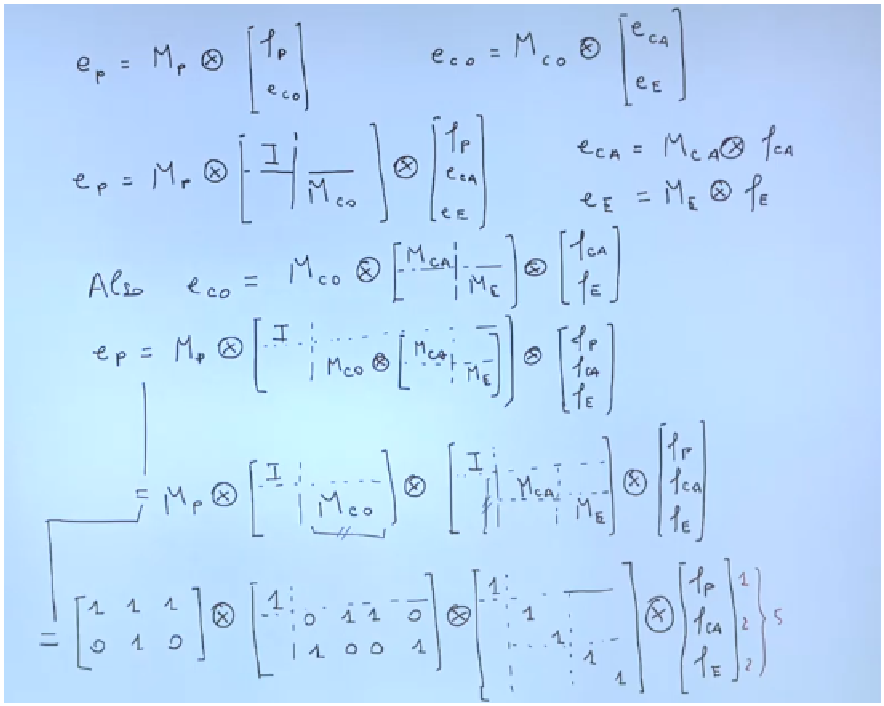
$$f = M^{\top} \odot e$$

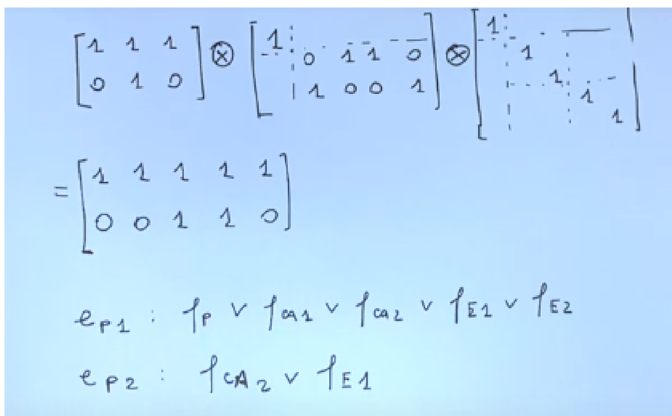
The $\odot$ is defined as:

$$f_{(i)} = \left(M^{\top}_{(i,1)} == e_{(1)}\right) \wedge \left(M^{\top}_{(i,2)} == e_{(2)}\right) \wedge \cdots \wedge \left(M^{\top}_{(I,N)} == e_{(N)}\right)$$

Example:







Summary

- Some Graph Knowledge
  - **Bipartite Graph**
- Finite State Automaton
- Bipartite Graph
  - Define Variable Relations
  - **Discover Analytical Redundancy Relations**
- Components and Service

- **General Component Model**
- Usage: Fault Diagnosis

- FTA: Fault Tree Analysis
  - Fault Tree
  - Cut set and minimal cut set
  - **Fault Tree Analysis: Semantic Model + Structure Function + DNF**
- FMEA: Failure Mode and Effects Analysis
  - **Graphical Representation**
  - **Boolean Matrix Representation + Inverse Inference**