Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Laboratory for Safety Analysis
INSTITUTE FOR ENERGY TECHNOLOGY

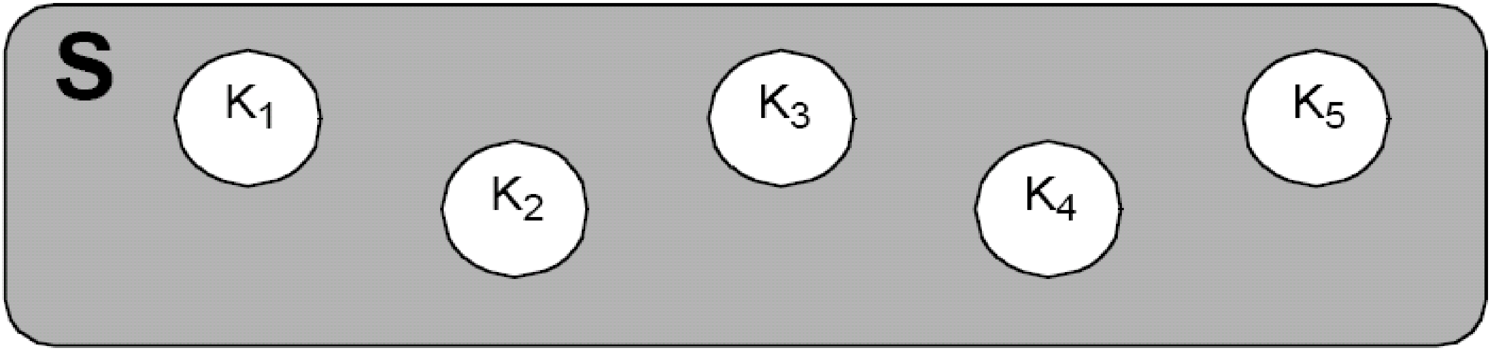
Reliability of Technical Systems

Main Topics

1. Short Introduction, Reliability Parameters: Failure Rate, Failure Probability, etc.
2. Some Important Reliability Distributions
3. Component Reliability
4. Introduction, Key Terms, Framing the Problem
5. System Reability I: Reliability Block Diagram, Structure Analysis (Fault Trees), State Model.
6. System Reability II: State Analysis (Markovian chains)
7. System Reability III: Dependent Failure Analysis
8. Static and Dynamic Redundancy
9. Advanced Methods for Systems Modeling and Simulation (Petri Nets, network theory, object-oriented modeling)
10. Software Reliability, Fault Tolerance
11. Human Reliability Analysis
12. Case study: Building a Reliable System

Component Reliability and System Reliability

system $S = \{K_1, \dots, K_n\}$,   number of components: n



Conclusions if the system structure is known:
Reliability of $K_1, \dots, K_n$ $\Leftrightarrow$ **Reliability of S** (both directions)

Assumption:
Failures of the components are **stochastically independent**.

- The calculation of the system reliability is of particular importance when it cannot be measured directly – for example, when a new system has to be designed from existing components.

- It must be noticed that the presented methods only apply if the failures of the components are stochastically independent. In reality, it is approximately the case if the main fault mechanisms are independent from each other. However, when faults propagate from a component to another one, the independence assumption is strongly violated.

System Function

We will use the component identifiers $K_1, \ldots, K_n$ and the system identifier S **also** as binary **random variables**, indicating whether the respective component is faultless or not.

We write the binary values in one of the following forms:
true = 1 = faultless,
false = 0 = faulty.

The (Boolean) **system function** f expresses the relationship between the component state (0 or 1) and the system state (0 or 1):

$S = f(K_1, \ldots, K_n)$

Example: $S = K_1 \wedge (K_2 \vee K_3)$ denotes a system which works properly as long as K_1 and at least one of the components K_2 or K_3 are faultless.

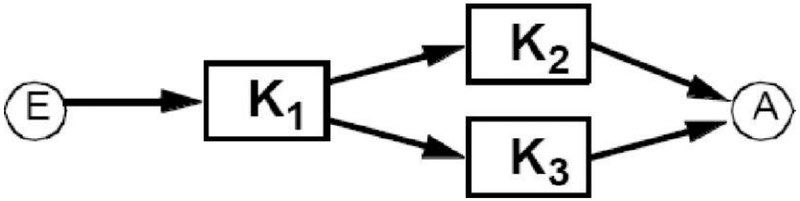
Reliability Block Diagram

Directed graph:

- Exactly one starting node E, exactly one terminal node A.
- Other nodes represent the binary random variable of a component (stating whether "faultless" or "faulty").

  Notice that multiple nodes are allowed for a single component.
- Additional virtual nodes H help to simplify the representation.

Semantics: The system is faultless if and only if there exists a path from E to A solely via faultless components.



A reliability diagram illustrates a system function.

In principle, its modeling power is somewhat smaller than the Boolean representation of the system function, because special monotony conditions must be satisfied (see next slide).

In the reliability diagram there is no means to express negation. However, we can learn from the motivation of the monotony conditions, nearly all reasonable systems can be modeled by a system function without negation.

Some properties of the modeling by a reliability diagram are:

- Arranging several components along a path means connecting them by an "and" operation
- Arranging several components in parallel paths represents an "or" operation.
- If a component appears in several expressions of the Boolean system function, it must be put in a corresponding number of parallel paths of the reliability diagram.

Required Monotony

In the reliability diagram negation cannot be expressed. Hence for a system $S = \{K_1, \ldots , K_n\}$ a monotonic system function is required:

- A system is faultless, if all its components are faultless:
  $K_1 \wedge \ldots \wedge K_n \Rightarrow S$
- A system is faulty, if all its components are faulty:
  $\neg K_1 \wedge \ldots \wedge \neg K_n \Rightarrow \neg S$
- A faulty system remains faulty, if an additional component fails:
  $K \wedge \neg S \Rightarrow (\neg K \Rightarrow \neg S)$
- A faultless system remains faultless, if one of its components makes the transition from faulty to faultless:
  $\neg K \wedge S \Rightarrow (K \Rightarrow S)$

In the system function of a non-redundant system, all components are connected by an "and" operation. All components must be faultless to form a faultless system. In the example the "or" operator between components K_2 and K_3 indicated redundancy, which can be interpreted in various ways:

- The components form static redundancy.
- The components form dynamic redundancy, where K_2 is the primary and K_3 is the spare component.

Rules for Boolean Functions : φ(x)

If we use this method to calculate the system function probability, then the probability of "or"-connected "and" expressions must be determined. Thereby the following rules have to be obeyed:

$$\varphi(\bar{X}) = 1 - \varphi(X)$$

$$\varphi(X \wedge Y) = \varphi(X) \cdot \varphi(Y)$$ in case of stochastical independence

$$\varphi(X \vee Y) = \varphi(X) + \varphi(Y) - \varphi(X \wedge Y)$$

$$\varphi(X_1 \wedge \ldots \wedge X_r) = \prod_{i=1}^{r} \varphi(X_r)$$ in case of stochastical independence

The rules for the "and" operations only apply in case of stochastically independence.

As an example, two expressions may be not independent if they contain the same variable, Therefore, the following calculation for $\varphi(X) = 0.9$ is not valid:

$$\varphi(X \wedge \bar{X}) = \varphi(X) \cdot \varphi(\bar{X}) = \varphi(X) \cdot (1 - \varphi(X)) = 0{,}9 \cdot 0{,}1 = 0{,}09$$

Correct is the following:

$$\varphi(X \wedge \bar{X}) = \varphi(0) = 0$$

The rule for the "and" operation is also part of the rule for the "or" operation. It must be applied accordingly there.

The rule for the "or" operation can cause a very high computation overhead. In this method, this rule must be applied for the “or “ connection of the cut expression.

Function Probability

The function probability $\varphi(K)$ or. $\varphi(S)$, respectively, denotes (either time-depent or time-independent)

- either the **reliability** R (probability to survive)
- or the **availability** V

of a component K or the system S, respectively.

For a component X or a system X we write:

$\bar{X} = \neg X$	X is not functioning
$\varphi(\bar{X}) = 1 - \varphi(X) = \bar{\varphi}(X)$	failure probability

Fault Case Probability

A **fault case** $C \in \{true, false\}^n$ is a binary vector with n elements.

Fault case C occurs, if for the vector $(K_1, \dots, K_n)$
the equality $C = (K_1, \dots, K_n)$ holds

The **fault case probability** (the probability that C occurs) is:

$$\gamma(C) = \prod_{i=1}^{n} \begin{cases} \varphi(K_i) & \text{if } K_i \\ 1 - \varphi(K_i) & \text{if } \overline{K_i} \end{cases}$$

Example: For a system $S = \{K_1, K_2, K_3\}$ with
$\varphi(K_1) = 0.9$, $\varphi(K_2) = 0.8$ and $\varphi(K_3) = 0.7$
the fault case probability of fault case C = (1, 0, 1) is:

$$\gamma((1, 0, 1)) = 0{,}9 \cdot (1 - 0{,}8) \cdot 0{,}7 = 0{,}126$$

A fault case is a combination of the possible values of the binary random variables indicating which components are faultless and which are not.

For a system consisting of n components there are exactly 2^n fault cases.

For convenience the completely faultless case C = (true,…., true) is also called a fault case in this context.

It should be noticed that two fault cases are always disjoint. Two different fault cases cannot occur at a time.

System Function Probability

Since fault cases are disjoint the equality holds:
system function probability = sum of the probabilities of the fault cases in which the system as a whole is faultless.

For the system $S = \{K_1, \ldots, K_n\}$ with system function f we obtain:

$$\varphi(S) = \sum_{C \in \{true, false\}^n} \begin{cases} \gamma(C) & \text{if } f(C) \\ 0 & \text{if } \neg f(C) \end{cases}$$

By this formula the first method to determine the system function probability is defined: **complete distinction of fault cases**.

- advantage: easy to understand
- disadvantage: high computation overhead because of a typically very high number of fault cases

1st Method: Complete Distinction of Fault Cases (Example)

Let be $S = K_1 \wedge (K_2 \vee K_3)$

$\varphi(K_1) = 0.9 \quad \varphi(K_2) = 0.8 \quad \varphi(K_3) = 0.7$

This system is faultless in the following three fault cases (out of a total of $2^3 = 8$ fault cases):

$C_1 = (1, 0, 1), \quad C_2 = (1, 1, 0), \quad C_3 = (1, 1, 1)$

As an example $C_1 = (1, 0, 1)$ means that K_1 is faultless, K_2 is faulty and K_3 is faultless.

The **system function probability** is calculated as follows

$$\varphi(S) = \varphi(K_1 \wedge (K_2 \vee K_3)) = \gamma(C_1) + \gamma(C_2) + \gamma(C_3) =$$

$$= \varphi(K_1) \cdot (1 - \varphi(K_2)) \cdot \varphi(K_3) + \varphi(K_1) \cdot \varphi(K_2) \cdot (1 - \varphi(K_3)) + \varphi(K_1) \cdot \varphi(K_2) \cdot \varphi(K_3)$$

$$= 0{,}9 \cdot 0{,}2 \cdot 0{,}7 + 0{,}9 \cdot 0{,}8 \cdot 0{,}3 + 0{,}9 \cdot 0{,}8 \cdot 0{,}7 = 0{,}846$$

Recursive Binary Distinction of Fault Cases

The second method to calculate the system function probability is based on the rule of total probability:

By arbitrarily selected components K the system is recursively split into the subsystems "K faultless" and "K faulty". For each step holds:

$$\forall\ K \in S:\quad \varphi(S) = \varphi(K) \cdot \varphi(S \mid K) + \varphi(\overline{K}) \cdot \varphi(S \mid \overline{K})$$

Here, $\varphi(S \mid X)$ denotes the conditional probability $\frac{\varphi(S \wedge X)}{\varphi(X)}$, which can be easily determined:
In the system function S we substituts X by 1 and $\overline{X}$ by 0, respectively. This simplifies the system function because the variable X is eliminated.

- advantage: usually low computation overhead
- disadvatage: the optimal selection of appropriate components requires human intuition.

Basic equations

- Theorem of the total likelihood

$$Pr\left(X\right)=\sum_{j=1}^{k}Pr\left(\Theta_j\right)\cdot Pr\left(X\middle|\Theta_j\right).$$

- Multiplication theorem for probabilities

$$Pr\left(\Theta_j\cap X\right)=Pr\left(\Theta_j\right)\cdot Pr\left(X\middle|\Theta_j\right)=Pr\left(X\right)\cdot Pr\left(\Theta_j\middle|X\right).$$

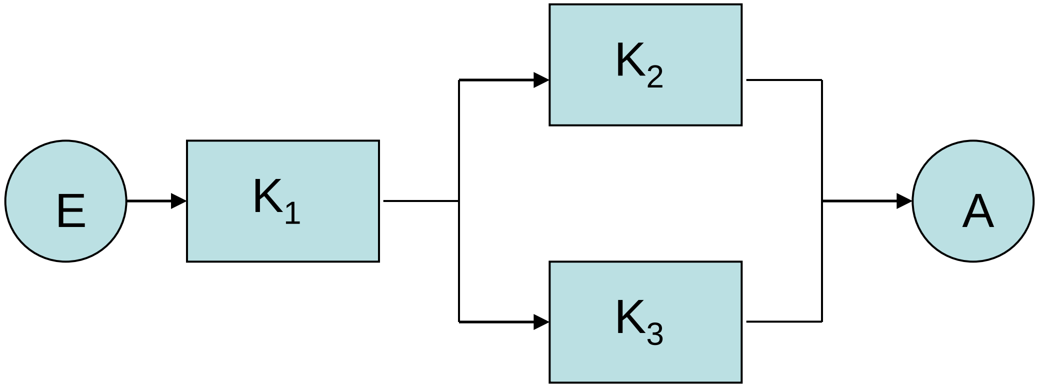
Pr(X): Probability of the event X („impact“)

$Pr(\Theta_j)$: Probability of the „cause“ j

$Pr(X|\Theta_j)$: probability of the impact X assuming cause Θ_j.

$Pr(\Theta_j|X)$: according to $Pr(X|\Theta_j)$

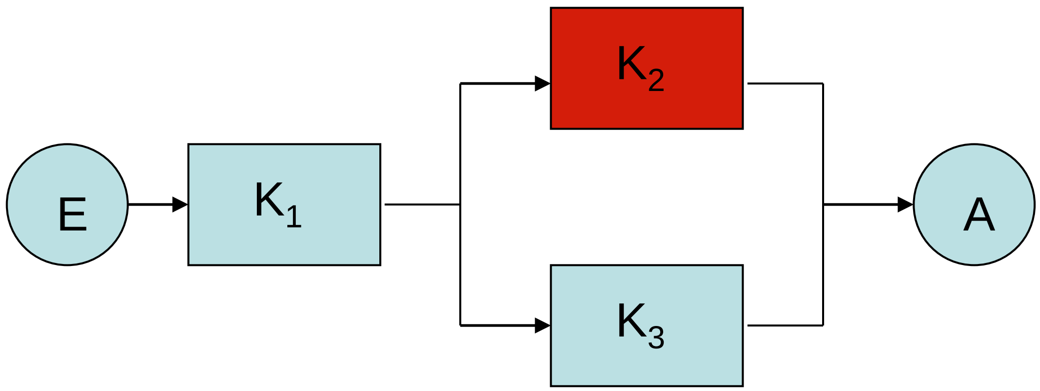
Recursive Binary Distinction of Fault Cases: Example



As in the previous example let be $S = K_1 \wedge (K_2 \vee K_3)$

$\varphi(K_1) = 0.9 \quad \varphi(K_2) = 0.8 \quad \varphi(K_3) = 0.7$

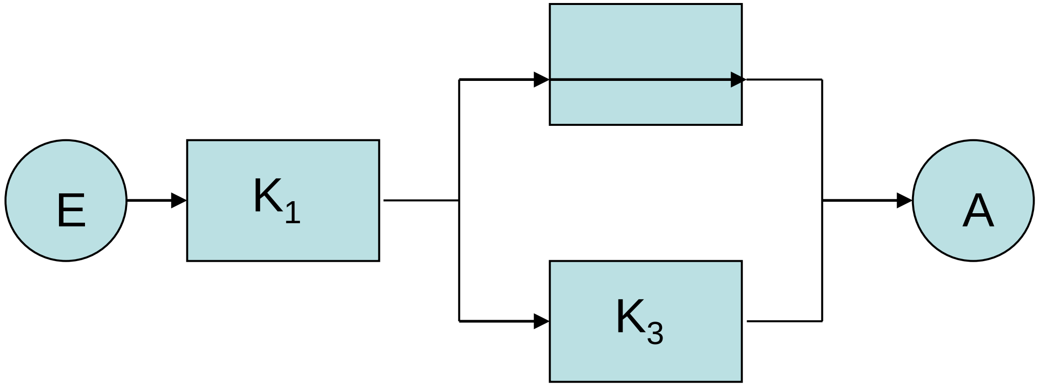
Recursive Binary Distinction of Fault Cases: Split for K_2



$$\varphi(S)=\varphi(K_2)*\varphi(K_1\wedge (K_2\vee K_3)|\ K_2) + \varphi(\overline{K_2})*\ \varphi(K_1\wedge (K_2\vee K_3)|\ \overline{K_2})$$

Recursive Binary Distinction of Fault Cases

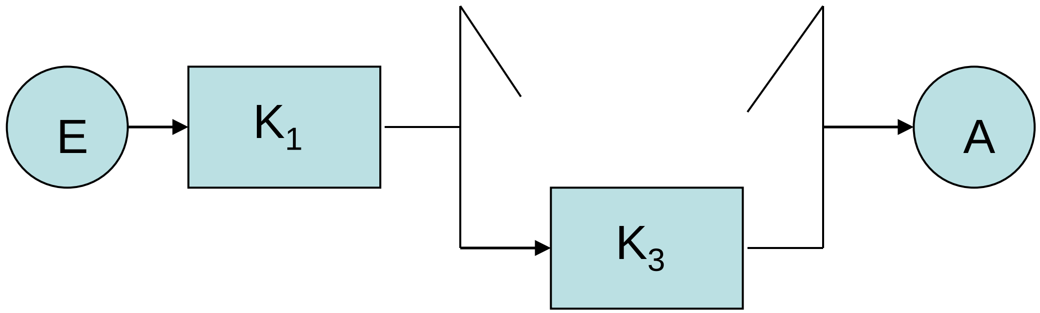
K_2 is faultless



$$\varphi(K_2)*\varphi(K_1 \wedge (K_2 \vee K_3)|\ K_2) = \varphi(K_2)*\ \varphi(K_1 \wedge (1 \vee K_3)) = \varphi(K_2)*\varphi(K_1)$$

Recursive Binary Distinction of Fault Cases

K_2 is faulty



$$\varphi(\overline{K_2})^* \, \varphi(K_1 \wedge (K_2 \vee K_3) | \overline{K_2}) = (1- \varphi(K_2))^* \, \varphi(K_1 \wedge (0 \vee K_3)) = (1- \varphi(K_2))^* \, \varphi(K_1 \wedge K_3)$$

1st Split for K_2:

$$\varphi(S) = \varphi(K_2) \cdot \varphi(K_1 \wedge (K_2 \vee K_3)|K_2) + \varphi(\overline{K_2}) \cdot \varphi(K_1 \wedge (K_2 \vee K_3)|\overline{K_2})$$

$$= \varphi(K_2) \cdot \varphi(K_1 \wedge (1 \vee K_3)) + (1 - \varphi(K_2)) \cdot \varphi(K_1 \wedge (0 \vee K_3))$$

$$= \varphi(K_2) \cdot \varphi(K_1) + (1 - \varphi(K_2)) \cdot \varphi(K_1 \wedge K_3)$$

2nd Split for K_1 in the expression $\varphi(K_1 \wedge K_3)$:

$$\varphi(K_1 \wedge K_3) = \varphi(K_1) \cdot \varphi(K_1 \wedge K_3|K_1) + \varphi(\overline{K_1}) \cdot \varphi(K_1 \wedge K_3|\overline{K_1})$$

$$= \varphi(K_1) \cdot \varphi(K_3) + (1 - \varphi(K_1)) \cdot \varphi(0) = \varphi(K_1) \cdot \varphi(K_3)$$

Substitution of the values: $\varphi(S) = 0{,}8 \cdot 0{,}0 + (1 - 0{,}8) \cdot 0{,}9 \cdot 07 = 0{,}846$

Negative Logic

Question: When is system faulty?

Methodes: Fault tree
Minimal Cut Sets

Variable: $\bar{x}$

Parameters: Failure probabilities

Positive Logic

Question: When is system faultless?

Methodes: CDFC, RBDFC, RD/RBD
Minimal Path Sets

Variable: x

Parameters: Survival probabilities, availabilities

Method of Fault Tree Analysis (FTA)

Starting point of FTA is a **predefined** system state (failed state as "top event"). The sub-sequent task is to find event combinations leading to the "top event". The branches are tracked top-down (top event -> intermediate events -> basic events); the reasoning is **deductive**.

Goals

- Systematic identification of failure modes (causes) and associated unit failures (basic events) leading to a "top event"
- Computation of "top event" probability where appropriate

Working steps of a FTA

(1) Definition of the "top event"
(2) Identification of all basic event combinations which result in the "top event"

If quantitative

(3) Assignment of failure probabilities to basic events
(4) Boolean modelling and calculations of probabilities
(5) Analysis of dominating failure combination and impacts (importance analysis), pro-posals for system improvement/optimisation

(1) Definition of the "top event"

- **In general:** system failure
- **In particular:** loss of specific functions and services meaning the failure of the overall system, (e.g. rupture of a gas storage tank).

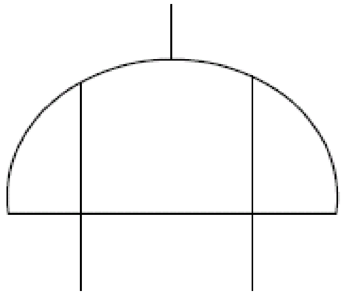
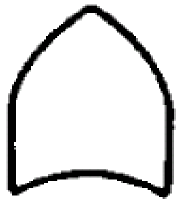
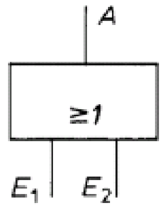
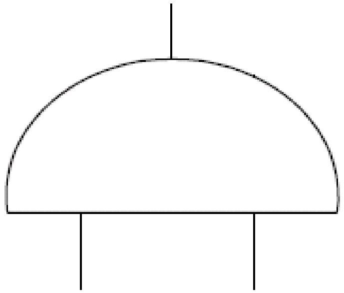
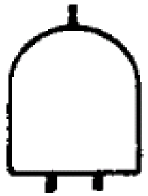
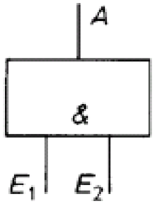
(2) Identification of basic event combinations

The formal combination of events constitutes the logical structure of the system considered or the derived Boolean model (fault tree). The model consists of:

- Input events: lower event ("input" to the gate)
- Gates (logic operation): show the relationship of lower events needed to result in a higher event (logic AND, OR)
- Output events: higher event ("output" of the gate).

The behaviour of the gates is determined by the **Rules of Boolean Algebra**.

Logic Gate Symbols

Symbol	Alternative symbols		Description
			**OR-Gate** Output fault occurs if at least one of the input faults occurs
			**AND-Gate** Output fault occurs if all of the input faults occur

(4) Boolean modelling and calculation of probabilities

Summary of the assumptions/preconditions

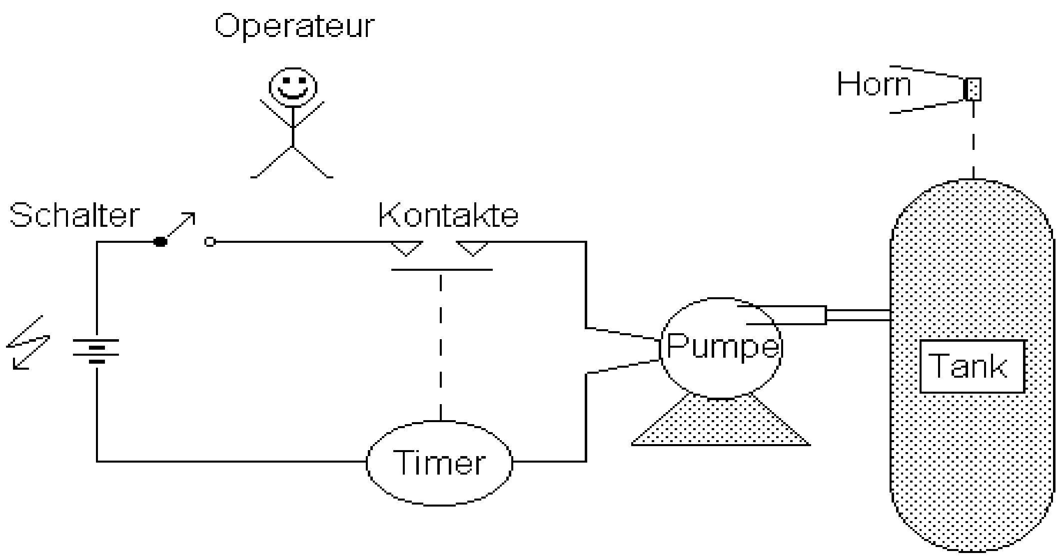
- A technical system consists of units (components)
- The units are both technically and logically connected
- The state of each unit follows a binary logic (TRUE/FALSE, on/off, intact/defect)
- Available logic operators are:
  - conjunction: AND ($\cap$)
  - disjunction: OR ($\cup$)

Labelling of the probabilities:

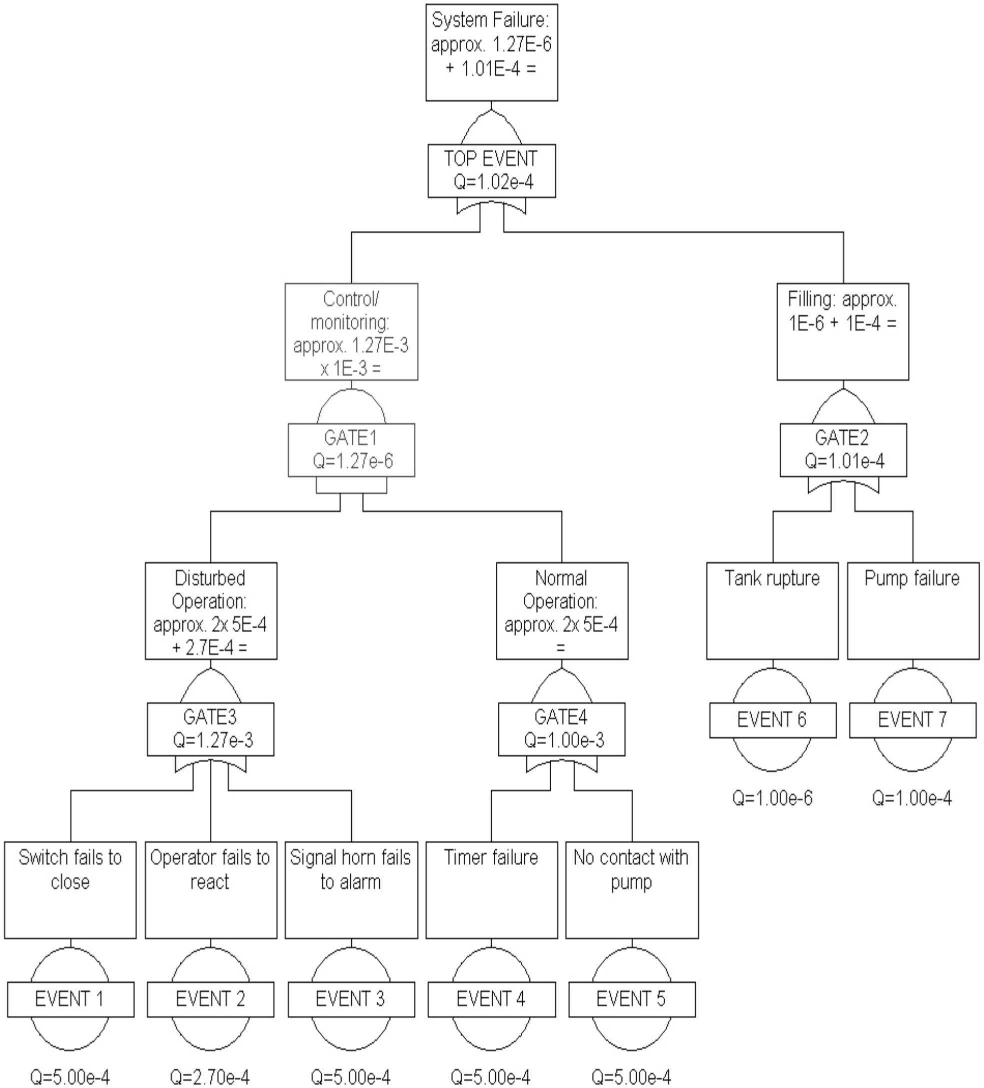
p_i: probability of survival of the *i*-th unit
q_i: probability of failure of the *i*-th unit

Example from industry: pumping system

In a pumping system, a tank is filled in 10 minutes and emptied in 50 minutes; hence, a complete cycle takes 1 hour. The switch is first closed and then the contact will be closed to allow the tank to be filled. After ten minutes (set by a timer), the contact will be opened to allow the tank to be emptied. If this mechanism fails, an alarm goes off and the operator opened the switch to prevent a tank failure due to overfilling.



Example from industry: pumping system



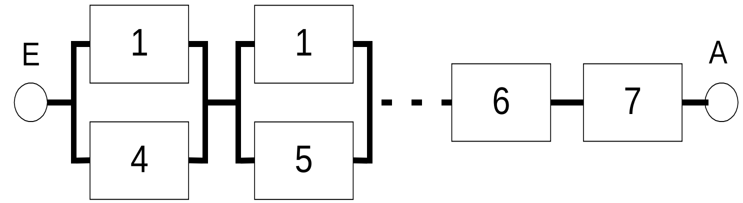
Boolean Function Failure:

$$y=\left[\left(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3\right) \wedge \left(\bar{x}_4 \vee \bar{x}_5\right)\right] \vee \left(\bar{x}_6 \vee \bar{x}_7\right)$$

multiply

$$y=\bar{x}_1\bar{x}_4 \vee \bar{x}_1\bar{x}_5 \vee \bar{x}_2\bar{x}_4 \vee \bar{x}_2\bar{x}_5 \vee \bar{x}_3\bar{x}_4 \vee \bar{x}_3\bar{x}_5 \vee \bar{x}_6 \vee \bar{x}_7$$

This is a serial-parallel and serial system: Reliability Block Diagram



Computation:

$$F_{SP}=1-\prod_{i=1}^{6}\left(1-q_{1i}q_{2i}\right)=1-\left[\left(1-q_1q_4\right)\left(1-q_1q_5\right)....\text{etc}\right]$$

$$F_s=1-\left[\left(1-q_6\right)\left(1-q_7\right)\right]$$

$$F=F_{SP}+F_s=...=1.0227\cdot 10^{-4}$$

Advantages of a FTA

- Well suited for modelling of binary (Boolean) mechanical processes, e.g. valve fails to open/close
- Events occurring on component level due to interaction of multiple failures are easily representable
- Provides reliability figures of a system (if adequate data are available)
- Encourages a methodical way of thinking
- Applicable to a wide field of systems and processes.

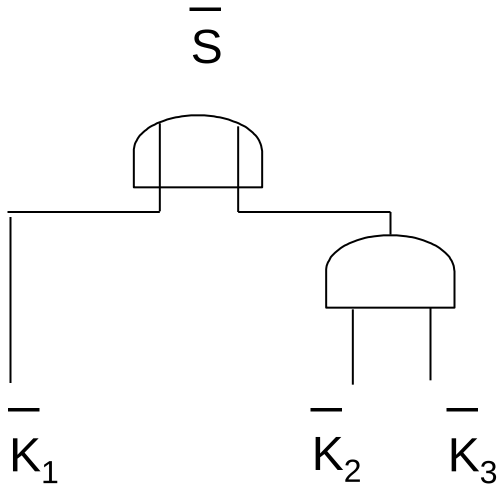
Disadvantages

- Dynamic processes are not representable (a system is considered as "static")
- Complicated systems usually result in an unmanageable amount of basic events and branches
- Reliability figures are often difficult to get.

Negative Logic

Fault tree

Question: When is system faulty?

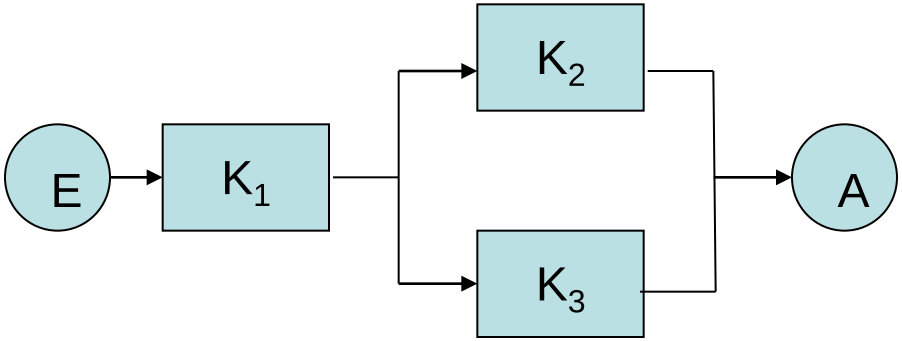


$$\overline{S} = \overline{K}_1 \vee (\overline{K}_2 \wedge \overline{K}_3)$$

Positive Logic

Reliability Block Diagram

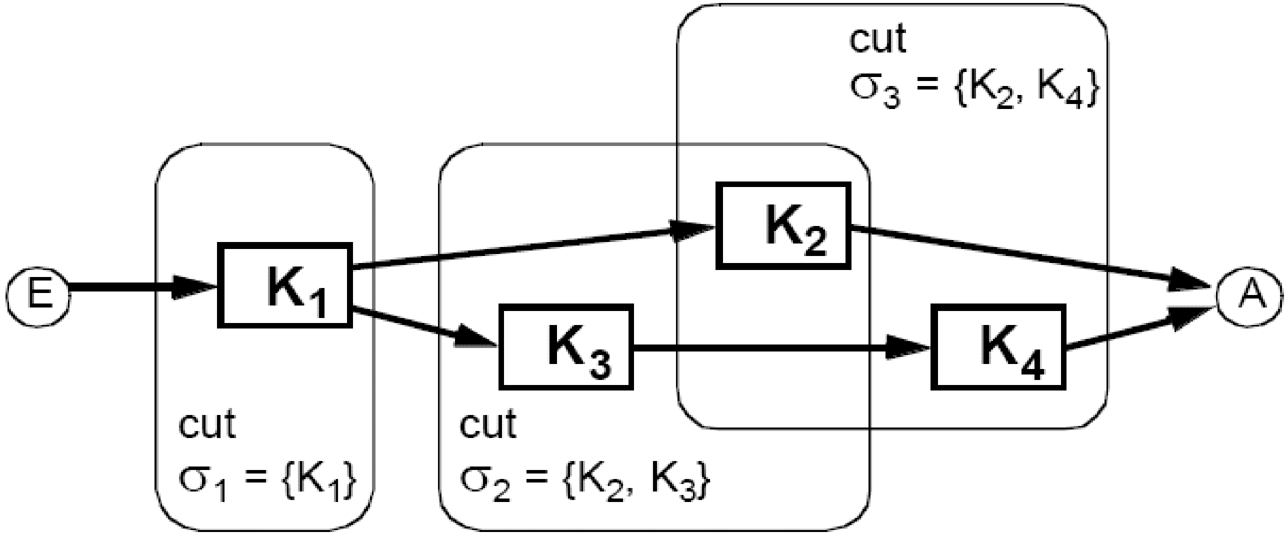
Question: When is system faultless?



$$S = K_1 \wedge (K_2 \vee K_3)$$

Minimal Cut

A set of components cutting all paths from *E* to *A* in the reliability diagram, is called a **cut**. A cut is **minimal**, if it does not contain another cut as a subset.



The cuts σ_1, σ_2 and σ_3 are minimal.

System Function Probability Derived from Minimal Cuts

After all k minimal cuts $\sigma_1, \ldots, \sigma_k$ of a system have been identified we can apply the third method to calculate the system function probability:

Let be cut $\sigma_i = \{K_{i_1}, \ldots, K_{i_{m(i)}}\}$ for $1 \le i \le k$. This cut σ_i is of **order** m(i).

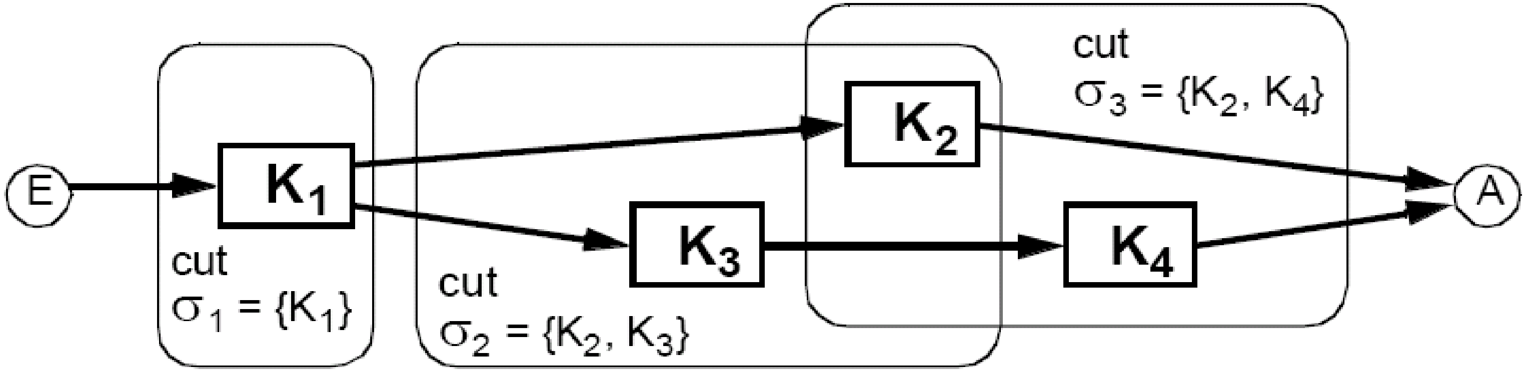
Cut σ_i can be transformed into the corresponding **cut expression** $T_i = \overline{K_{i_1}} \wedge \ldots \wedge \overline{K_{i_{m(i)}}}$.

From all cut expressions the system function probability can be calculated:

$$\varphi(S) = 1 - \varphi(T_1 \vee \ldots \vee T_k)$$

$$= 1 - \varphi((\overline{K_{1_1}} \wedge \ldots \wedge \overline{K_{1_{m(1)}}}) \vee \ldots \vee (\overline{K_{k_1}} \wedge \ldots \wedge \overline{K_{k_{m(k)}}}))$$

3rd Method: Minimal Cuts (Example)



$$\varphi(S) = 1 - \varphi(\overline{K_1} \vee (\overline{K_2} \wedge \overline{K_3}) \vee (\overline{K_2} \wedge \overline{K_4}))$$

$$= 1 - \varphi(\overline{K_1}) - \varphi(\overline{K_2} \wedge \overline{K_3}) - \varphi(\overline{K_2} \wedge \overline{K_4}) + \varphi(\overline{K_1} \wedge \overline{K_2} \wedge \overline{K_3}) + \varphi(\overline{K_1} \wedge \overline{K_2} \wedge \overline{K_4}) \ldots$$

$$\ldots + \varphi(\overline{K_2} \wedge \overline{K_3} \wedge \overline{K_4}) - \varphi(\overline{K_1} \wedge \overline{K_2} \wedge \overline{K_3} \wedge \overline{K_4})$$

$$= 1 - \overline{\varphi}(K_1) - (\overline{\varphi}(K_2) \cdot \overline{\varphi}(K_3)) - (\overline{\varphi}(K_2) \cdot \overline{\varphi}(K_4)) + (\overline{\varphi}(K_1) \cdot \overline{\varphi}(K_2) \cdot \overline{\varphi}(K_3)) + \ldots$$

$$(\overline{\varphi}(K_1) \cdot \overline{\varphi}(K_2) \cdot \overline{\varphi}(K_4)) + (\overline{\varphi}(K_2) \cdot \overline{\varphi}(K_3) \cdot \overline{\varphi}(K_4)) - (\overline{\varphi}(K_1) \cdot \overline{\varphi}(K_2) \cdot \overline{\varphi}(K_3) \cdot \overline{\varphi}(K_4))$$

Accuracy and Approximation

Besides the third method ("minimal cuts") there exist some more, such as "minimal paths" or "disjoint paths".

"Minimal cuts" usually achieves a **higher precision**, because $\bar{\varphi}(X)$ instead of $\varphi(X)$ is closer to 0 (better use of the mantissa in floating point operations).

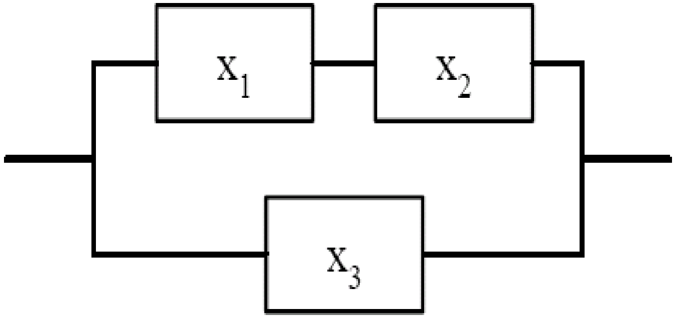
Example with decimal numbers, rounded to 0.1
Calculation with $\varphi(X)$: $0.9 \cdot 10^0 \cdot 0.9 \cdot 10^0 = 0.8 \cdot 10^0$ (deviation 0.01)
Calculation with $\bar{\varphi}(X)$: $0.1 \cdot 10^0 \cdot 0.1 \cdot 10^0 = 0.1 \cdot 10^{-1}$ (no deviation)

Moreover, an **approximation** can be obtained by ignoring the cuts of higher order, because $(\bar{\varphi}(X))^1 >> (\bar{\varphi}(X))^2 >> (\bar{\varphi}(X))^3 >> \ldots$ holds.

By analogy, products with a higher number of factors close to 0 can be omitted in a sum.
Example: Last product of 4 factors in the sum on the previous page.

Negative Logic

Positive Logic

<table>
<tr><th>Minimal Cut Sets</th><th>Minimal Path Sets</th></tr>
<tr><td>Smallest set of failed units, that blocks the path from input to output in a reliability block diagram.</td><td>Smallest set of (operating) units, that leaves open a path from input to output in a reliability block diagram.</td></tr>
<tr><td colspan="2">Example

</td></tr>
<tr><td>Cuts σ_i: $\sigma_1 = \{\bar{x}_1; \bar{x}_3\}$; $\sigma_2 = \{\bar{x}_2; \bar{x}_3\}$</td><td>Paths π_j: $\pi_1 = \{x_1; x_2\}$; $\pi_2 = \{x_3\}$</td></tr>
</table>

Cuts σ_i: $\sigma_1=\{\bar{x}_1;\bar{x}_3\}$; $\sigma_2=\{\bar{x}_2;\bar{x}_3\}$	Paths π_j: $\pi_1=\{x_1;x_2\}$; $\pi_2=\{x_3\}$
Boolean functions	
$\bar{y}=1-\bigcap_{j=1}^{n}(1-\sigma_j)=1-\left[(1-\bar{x}_1\bar{x}_3)(1-\bar{x}_2\bar{x}_3)\right]$	$\bar{y}=\bigcap_{j=1}^{s}(1-\pi_j)=(1-x_1x_2)(1-x_3)$
multiply, Idempotent law ...	
$\bar{y}=1-\left[(1-\bar{x}_1\bar{x}_3)(1-\bar{x}_2\bar{x}_3)\right]$ $=1-(1-\bar{x}_1\bar{x}_3-\bar{x}_2\bar{x}_3+\bar{x}_1\bar{x}_3\bar{x}_2\bar{x}_3)$ $=\bar{x}_1\bar{x}_3+\bar{x}_2\bar{x}_3-\bar{x}_1\bar{x}_2\bar{x}_3$	$\bar{y}=1-x_1x_2-x_3+x_1x_2x_3$
…	*Note:* Calculations in order to get the same formal representation as for cut sets. $\bar{y}=1-(1-\bar{x}_1)(1-\bar{x}_2)-(1-\bar{x}_3)+(1-\bar{x}_1)(1-\bar{x}_2)(1-\bar{x}_3)$ $=$...multiply... $=\bar{x}_1\bar{x}_3+\bar{x}_2\bar{x}_3-\bar{x}_1\bar{x}_2\bar{x}_3$
System failure probability	
$F=q_1q_3+q_2q_3-q_1q_2q_3$	$F=q_1q_3+q_2q_3-q_1q_2q_3$

State Model

Instead of subdividing a system into components it can also be subdivided into **global states** $Z_1, \ldots, Z_m$. Each state Z_i represents a combination of component states: $Z_i = (K_1, \ldots, K_n)$.

The **transition rates** $\alpha_{i,j}$ between states Z_i and Z_j define the mean number of transitions from Z_i to Z_j per time unit provided that the system is in state Z_i $(1 \leq i \leq m$ und $1 \leq j \leq m)$.

Advantages:

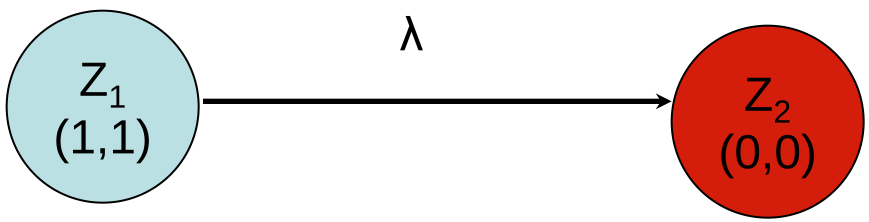
- Modeling in great detail: Any state transitions can be expressed, in particular: fault tolerance techniques (like reconfiguration) and **repair**.
- joint performance and reliability evaluation

Disadvantage: High computation overhead due to the large state space.

p_i is a probability of a state i.

Application of the State Modell:

Unrepairable System



$Z_i(K_i,S)$, where K_i state of component, S- conditional state of system.
Λ - failure rate

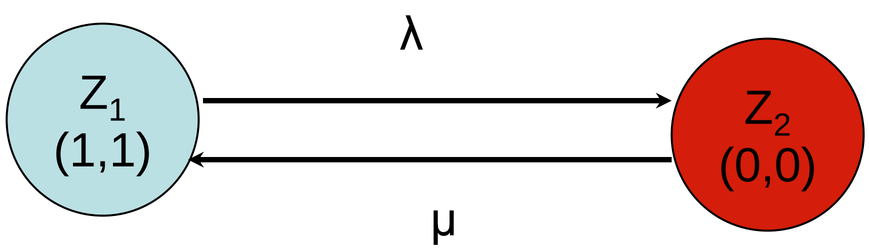
Homogenous system of equalities: $$\left(\dot{p}_1(t), \dot{p}_2(t)\right) = \left(p_1(t), p_2(t)\right) \cdot \begin{pmatrix} -\lambda & \lambda \\ 0 & 0 \end{pmatrix}$$

To be solved unter conditions: $p_1(t)+p_2(t)=1$ and $p_1(0)=1$

Solution (via Laplace transformation if necessary): $$p_1(t) = e^{-\lambda \cdot t} \ \ and \ \ p_2(t) = 1 - e^{-\lambda \cdot t}$$

Application of the State Modell:

Repairable System



$Z_i(K_i,S)$, where K_i state of component, S- conditional state of system.
Λ - failure rate, μ – repair rate

Homogenous system of equalities: $$\left(\dot{p}_1(t), \dot{p}_2(t)\right) = \left(p_1(t), p_2(t)\right) \cdot \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

To be solved unter conditions: $p_1(t)+p_2(t)=1$ and $p_1(0)=1$

Solution: $$p_1(t) = \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} \cdot e^{-(\lambda+\mu)\cdot t}$$

It is the time dependent Availability V(t)!

State models express each combination of component states by a separate global state.

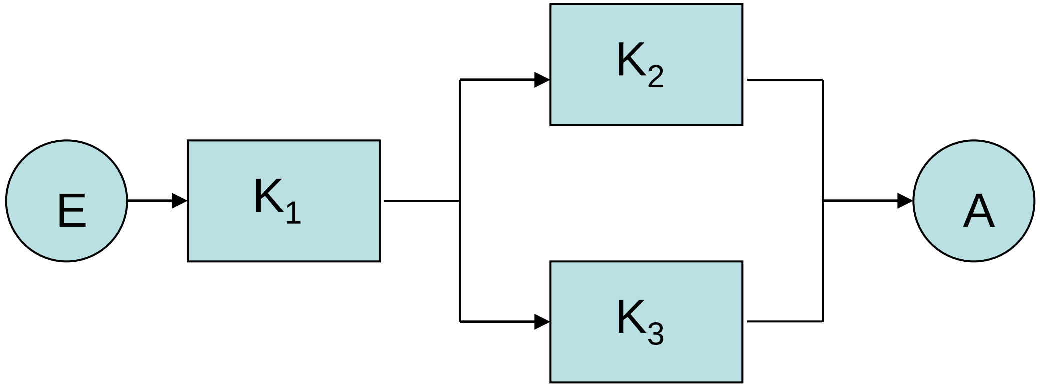
In a system from two components K_1 and K_2 we have the global states:

- Both K_1 and K_2 are faultless, expressed by the global state $Z_1 = (1, 1)$.
- K_1 is faultless and K_2 is faulty, expressed by the global state $Z_2 = (1, 0)$.
- K_1 is faulty and K_2 is faultless, expressed by the global state $Z_3 = (0, 1)$.
- Both K_1 and K_2 are faulty, expressed by the global state $Z_4 = (0, 0)$.

The transition rate $\alpha_{i,j}$ can also be seen as the reciprocal of the mean duration between the entering of state Z_i and the transition from Z_i to Z_j. However, it must be taken into account, that from Z_i also other states could be reached.

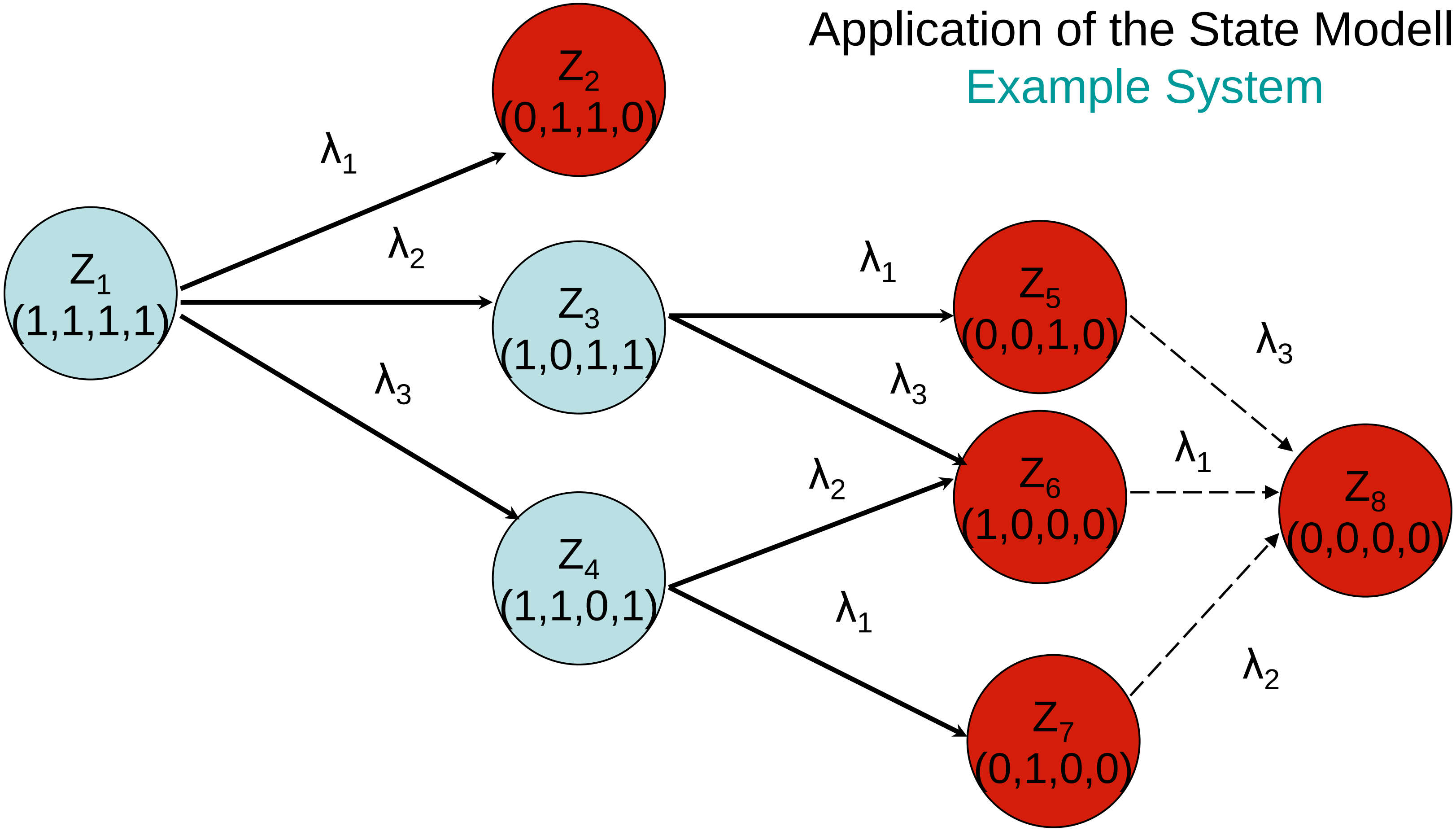
Application of the State Modell:

Example System



Number of states $2^3=8$

Application of the State Modell:
Example System



Stationary availability is calculated by $V=p_1+p_3+p_4$

Calculation of the Steady-State State Probabilities

In the state model with constant transition rates (homogeous Markovian model) the state probabilities p_i approach their final values P_i.

These values express the steady-state behaviour:
For all states the sum of "inputs" is equal to the sum of "outputs".

$$\forall\ j \in \{1, \ldots , m\}: \quad \sum_{i=1}^{m} P_i \cdot \alpha_{i,j} = \sum_{k=1}^{m} P_j \cdot \alpha_{j,k}$$

This formula expresses a **homogenous linear system of equalities** to be solved under the condition

$$\sum_{i=1}^{m} P_i = 1$$

(1st Method to calculate the steady-state state probabilities).

By state probability we mean the probability that the system is in the respective state.

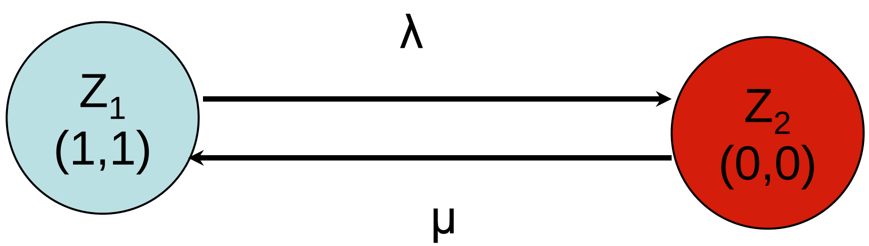
In the steady-state the state probabilities no longer change. Therefor, for each state the sum of "inputs" from other states (increasing the state probability) must be equal to the sum of "outputs" to other states (decreasing the state probability).

In other words, the probability to enter a state is the same as to leave the state in a given time interval.

The homogenous linear system of m equalities with m variables cannot be solved due to the depence of the equalities. The solution is possible with the additional equality stating the the sum of all state probabilities is always one.

Application of the State Modell:

Repairable System



Z_1: $\lambda * p_1 = \mu * p_2$

Z_2: $\lambda * p_1 = \mu * p_2$

$p_1 + p_2 = 1$

Stationary availability (steady-state availability) V is calculated by

$$p_1 = \frac{\mu}{\lambda + \mu}, \text{ the complementis } p_2 = \frac{\lambda}{\lambda + \mu}$$