# ***Applications of Artificial Intelligence***

Sebastian Iwanowski
FH Wedel

**Chapter 4:**
Knowledge-Based Systems

4.4: Model-Based Reasoning

# Application from practice: Technical diagnosis

## Run time system:

(knowledge-based systems call this **problem solver / inference engine**)

**Input:**

- Setting certain control inputs
- Observing values depending on this setting

**Output:**

- A unique instruction how to repair which component

***This is where diagnostic systems do not differ !***

# Application from practice: Technical diagnosis

## Knowledge-based diagnosis:

### 1) Knowledge acquisition: Input into knowledge base

- rule-based (symptom-based)
- case-based
- model-based

} as alternatives

### 2) Knowledge structure

- depends on knowledge acquisition

### 3) Knowledge processing be the problem solver

- depends on knowledge structure

***This is where diagnostic systems may differ !***

# 3. Model-Based Diagnosis

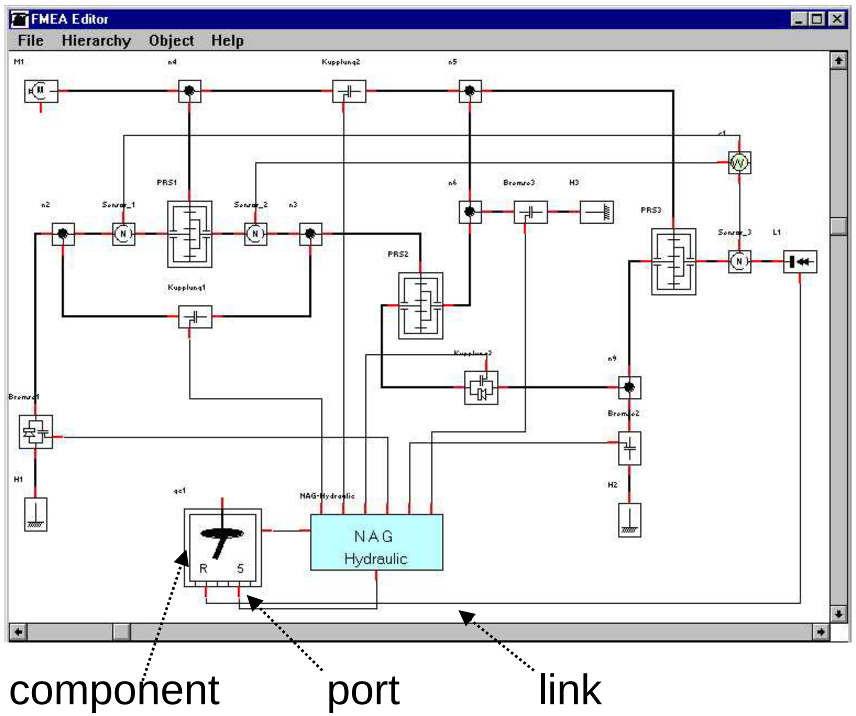
## Goal:

- fast knowledge acquisition
- exact and provable solution of problem solver

## Challenge:

- reasonable response time of problem solver at run time

# 3. Model-Based Diagnosis



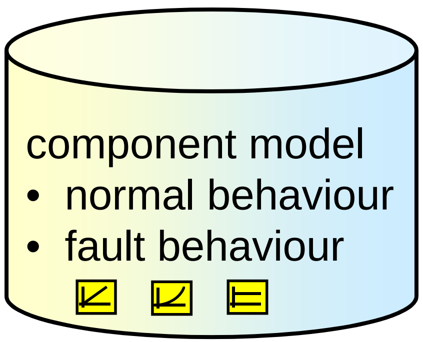

**<u>System model</u>:**

**Which components of which type are connected in which way?**

➔ ***available from CAD data***

**<u>Component models</u>:**

**How do values depend on each other lying at ports of the component?**




➔ ***to be modeled once per component <u>type</u>***

➔***Model is reusable for all systems containing components of this type.***

# 3. Model-Based Diagnosis

## Input to knowledge base:

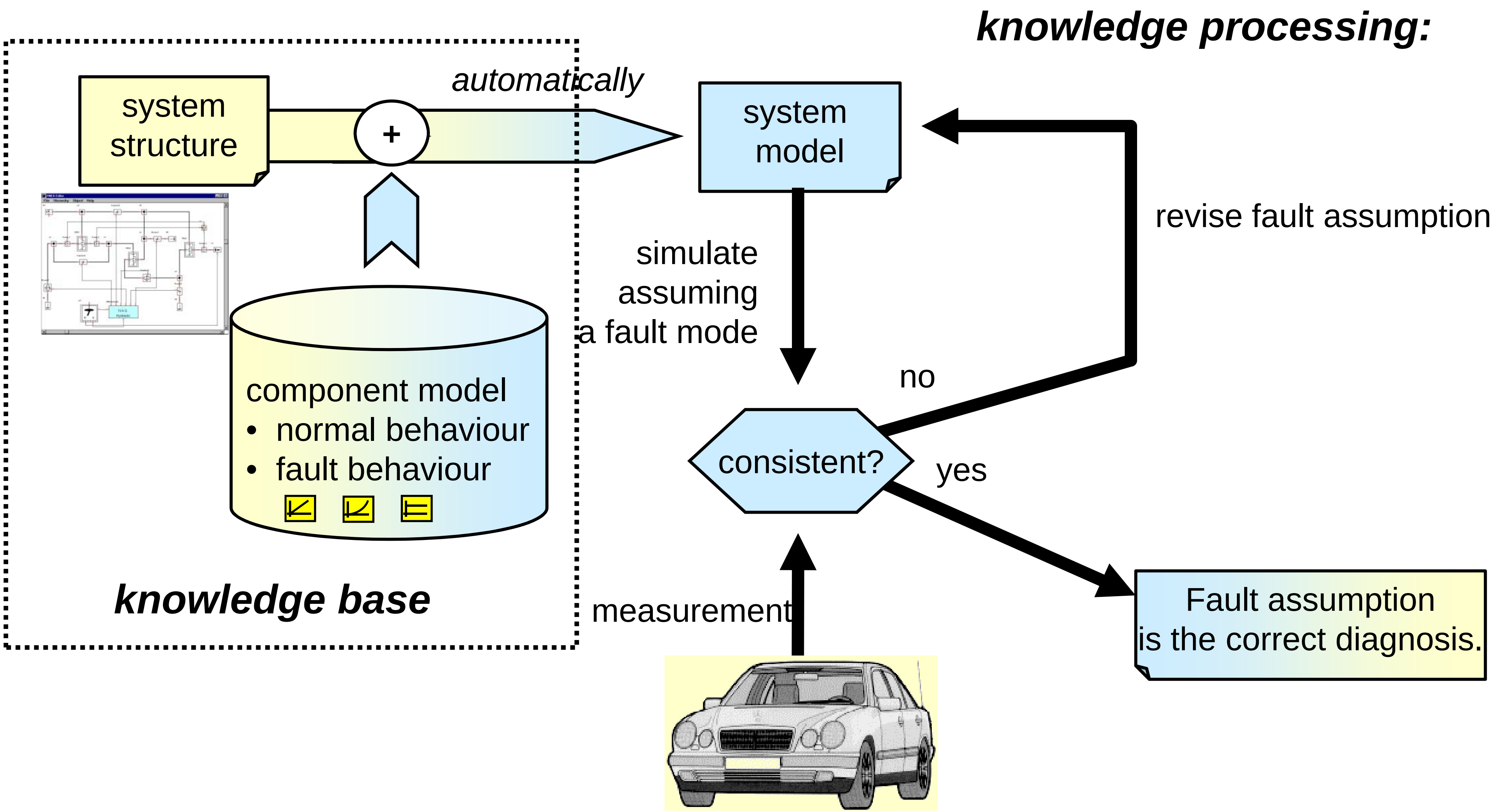
- system model: hierarchical structure of the system (+ how the components are connected)
- component models

## Structure of knowledge base:

- constraint network (assembled automatically)
- structured by:
  - assigning constraints to components and ports
  - assigning variables to components and ports

# 3. Model-Based Diagnosis

## Base functionality: Conflict driven search



# 3. Model-Based Diagnosis

## Base functionality: Conflict driven search

## GDE 1987: *The* prototype for model-based diagnosis

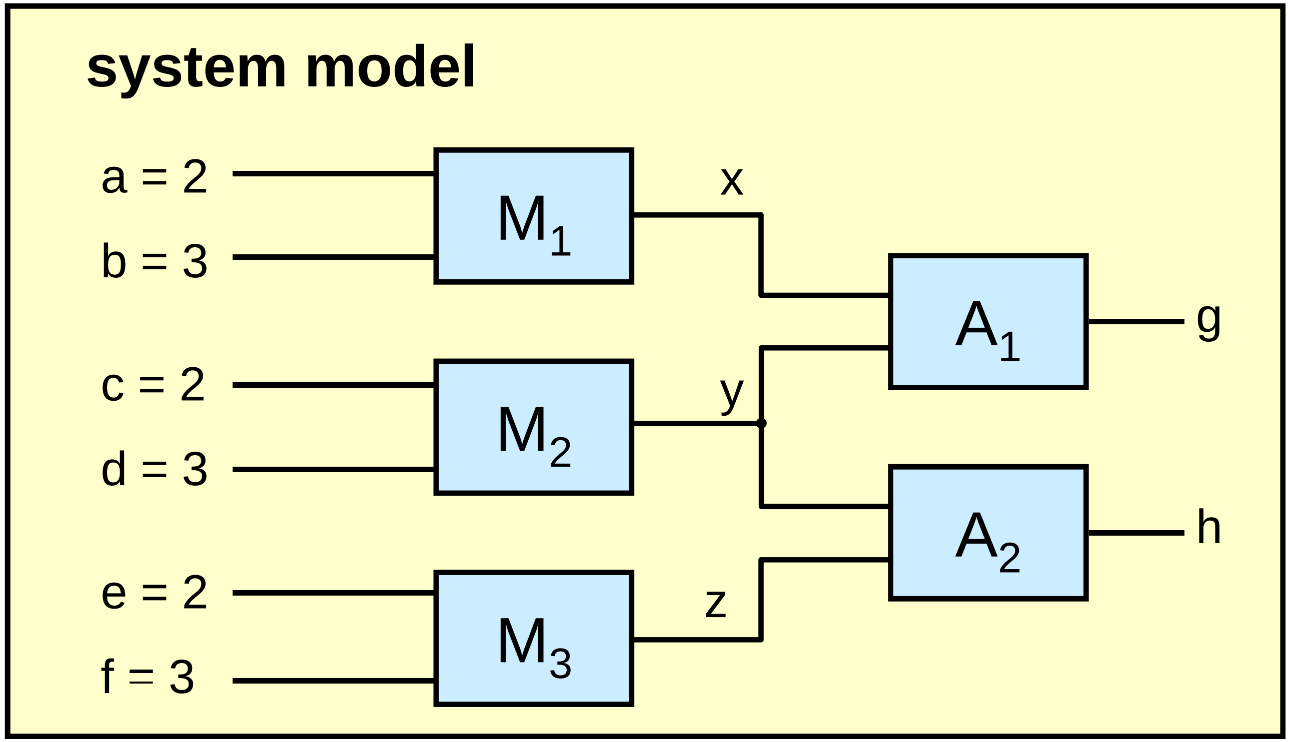
**Problem:**

- ‚brute-force‘ Simulation of ***all*** fault assumptions combinatorically not feasible

**Idea**: General Diagnostic Engine GDE, deKleer & Williams 1987

- intelligent  search in the space of all fault assumptions
- uses inconsistent assumptions for pruning the search space

# GDE - Example

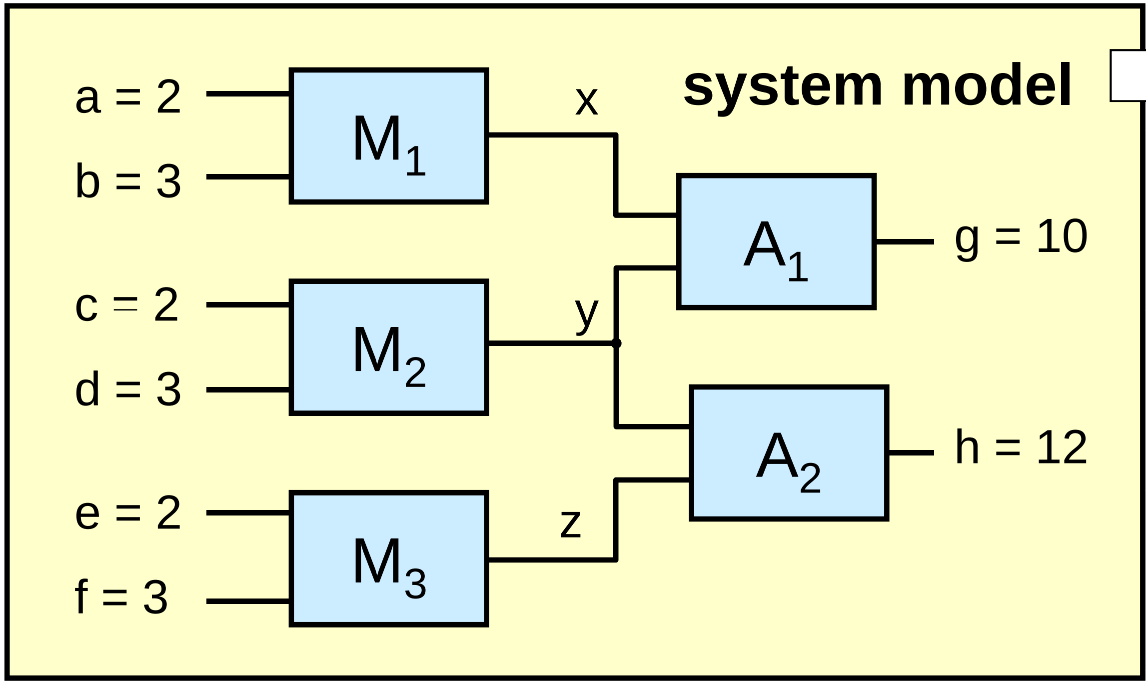



**component models**

- multiplier: mode=ok $\Rightarrow$ out = $in_1 * in_2$
- adder: mode=ok $\Rightarrow$ out = $in_1 + in_2$

**measurements:** $g = 10 \wedge h = 12$

# GDE - Example




**simulation**

**x = 6 {M1}**
**y = 6 {M2}**
**z = 6 {M3}**
**g = 12 {M1 M2 A1}, g =10**
**y = 4 {M1 A1}**
**h = 10 {M1 A1 A2 M3}, h = 12**
**y = 6 {A2 M3}**

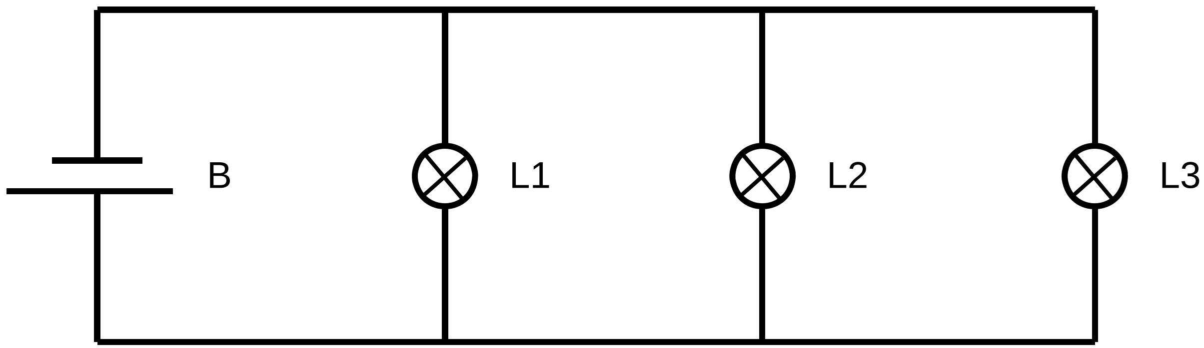
**two conflicts**

| M1 | M2 | M3 | A1 | A2 |
|---|---|---|---|---|
| X | X |  | X |  |
| X |  | X | X | X |

**diagnoses:**
**single-fault M1**
**single-fault A1**
**double fault M2 M3**
**:**

# Model-Based Diagnosis: Base functionality

**Example why adder/multiplier example does not reveal all difficulties for practice:**




**Observation:**

**L1, L2 are not lit, L3 is lit**

**GDE diagnoses:**

1. **(B ok, L1 faulty, L2 faulty, L3 ok)**
2. **(B faulty, L1 ok, L2 ok, L3 faulty)** ???
3. **(B faulty, L1 ok, L2 ok, L3 ok)** ???

# Model-Based Diagnosis: Base functionality

## Models of electric components:

**Battery:**


value ranges: $\text{minus, plus} \in \{ \texttt{ground}, \texttt{supply voltage} \}$

rules:

$\text{ok} \Rightarrow (\text{minus} = \texttt{ground})$

$\text{ok} \Rightarrow (\text{plus} = \texttt{supply voltage})$

**Wire:**


value ranges: $\text{a1, a2} \in \{ \texttt{ground}, \texttt{supply voltage} \}$

rules:

$\text{ok} \wedge (\text{a1} = \texttt{ground}) \Rightarrow (\text{a2} = \texttt{ground})$

$\text{ok} \wedge (\text{a1} = \texttt{supply voltage}) \Rightarrow (\text{a2} = \texttt{supply voltage})$

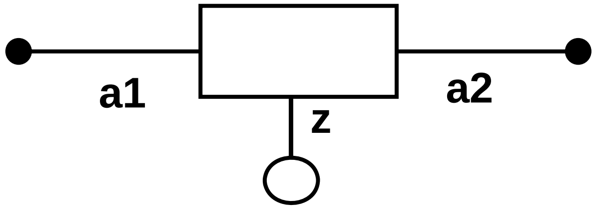
$\text{ok} \wedge (\text{a2} = \texttt{ground}) \Rightarrow (\text{a1} = \texttt{ground})$

$\text{ok} \wedge (\text{a2} = \texttt{supply voltage}) \Rightarrow (\text{a1} = \texttt{supply voltage})$

# Model-Based Diagnosis: Base functionality

## Models of electric components:

**Lamp**:




**value ranges:**

a1, a2 ∈ { `ground`, `supply voltage` }

z ∈ { `lit`, `dark` }

**rules:**

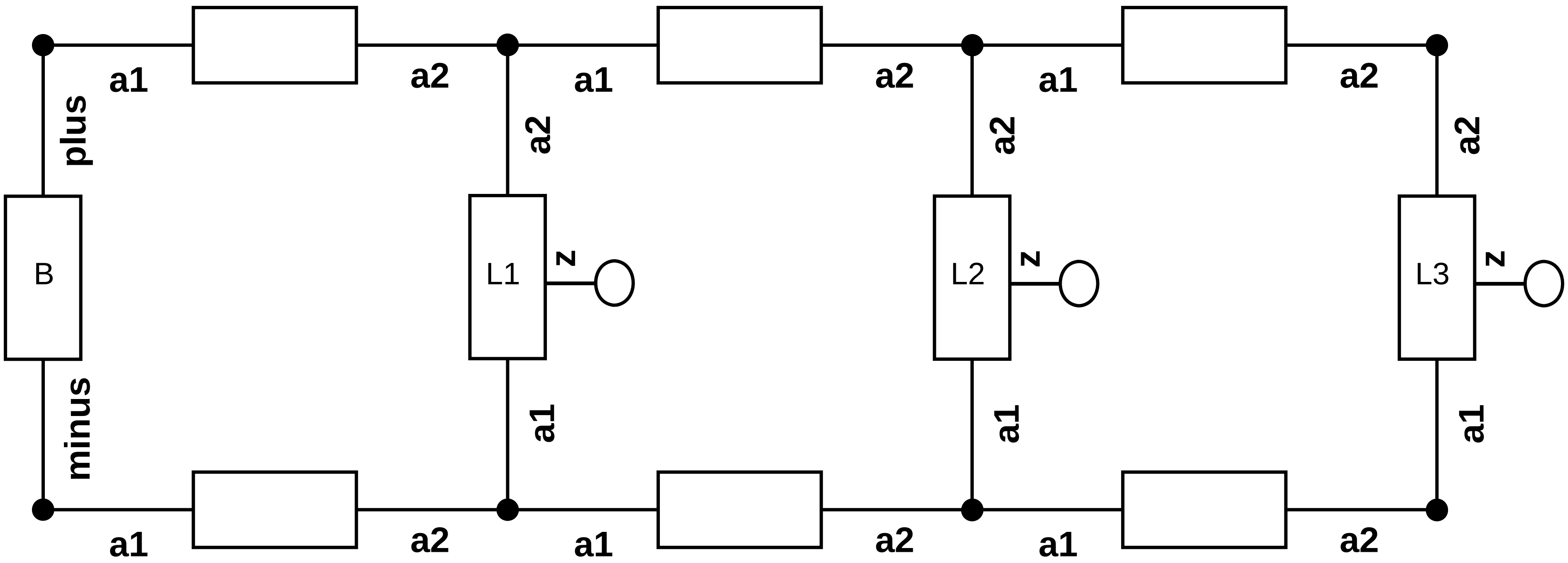
ok ∧ (a1 = `supply voltage`) ∧ (a2 = `ground`) ⇒ (z = `lit`)
ok ∧ (a2 = `supply voltage`) ∧ (a1 = `ground`) ⇒ (z = `lit`)
ok ∧ (a1 = `supply voltage`) ∧ (a2 = `supply voltage`) ⇒ (z = `dark`)
ok ∧ (a1 = `ground`) ∧ (a2 = `ground`) ⇒ (z = `dark`)

ok ∧ (a1 = `ground`) ∧ (z = `lit`) ⇒ (a2 = `supply voltage`)
ok ∧ (a1 = `supply voltage`) ∧ (z = `lit`) ⇒ (a2 = `ground`)
ok ∧ (a1 = `ground`) ∧ (z = `dark`) ⇒ (a2 = `ground`)
ok ∧ (a1 = `supply voltage`) ∧ (z = `dark`) ⇒ (a2 = `supply voltage`)

ok ∧ (a2 = `ground`) ∧ (z = `lit`) ⇒ (a1 = `supply voltage`)
ok ∧ (a2 = `supply voltage`) ∧ (z = `lit`) ⇒ (a1 = `ground`)
ok ∧ (a2 = `ground`) ∧ (z = `dark`) ⇒ (a1 = `ground`)
ok ∧ (a2 = `supply voltage`) ∧ (z = `dark`) ⇒ (a1 = `supply voltage`)

# Model-Based Diagnosis: Base functionality

## Composing the system model from the component models:




**Values at connecting ports must be the same from both sides.**

**In case of contradiction: Conflict between the behavioural modes predicting the resp. values**

**Diagnoses are sets of behavioural modes not containing any conflict.**

# Model-Based Diagnosis: Base functionality

## Conclusion from this modeling:

**There is no logic contradiction to the following diagnosis:**

**2. (B faulty, L1 ok, L2 ok, L3 faulty)**

**Reason:**

**L3 may be lit in fault mode even if there is no voltage difference.**

***Incomplete knowledge base !***

**Even worse:**

**If a behavioural rule is only evaluated when its antecedents assume actual values, then no contradiction can be found to the following diagnosis:**

**3. (B faulty, L1 ok, L2 ok, L3 ok)**

**Reason:**

**There is no voltage value computed anywhere in the system.**

***Incomplete proving ability of the problem solver !***

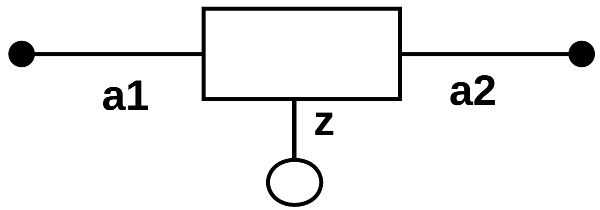
# Model-Based Diagnosis: Base functionality

## Additional rules for the exclusion of diagnoses 2 / 3:

**Battery:**


**faulty ⇒ (minus = ground)**

**Lamp:**




**faulty ∧ (a1 = supply voltage) ∧ (a2 = supply voltage) ⇒ (z = dark)**
**faulty ∧ (a1 = ground) ∧ (a2 = ground) ⇒ (z = dark)**

***There must be models for faulty behaviour, too, in order to exclude diagnoses that are physically impossible.***

# Model-Based Diagnosis: Extended functionality

## Base functionality:

### Input:

- Setting certain control inputs
- Observing values depending on this setting

### Output:

- Several diagnoses of the following kind:
  - Each diagnosis assigns a behavioural mode to each component: ok or a defined fault mode
  - The rules of all behavioural modes assigned agree with all set and observed values.

## What does the user need ?

### Input: see above

### Output:

- A unique instruction how to repair which component

# Model-Based Diagnosis: Extended functionality

## Extended functionality:

### 1) Suggestion of setting certain control inputs

- Setting certain values at certain places in the system

  (such that the observations to be expected differ such that the diagnoses valid so far may be distinguished best)

### 2) Suggestion of observation points

- Selecting observation points

  (such that the observations to be expected differ such that the diagnoses valid so far may be distinguished best)

} **Test** (items 1 and 2)

## Requirement for the modeling:

- Definition of test points
- Definition of test values to be set at the test points
- Definition of observation points to be measured

} **Control actions** (test points and test values)

**Observations** (observation points)

# Modeling the components

## Behavioural modes

- modes of the component to be searched for in the diagnostic process
- Domain of definition must be finite (normall less than 10 values)

## Variables

- containing values
- The variable values are used in the constraints.
- The constraints compute new values for other variables.

*Distinguish internal variables from port variables !*

## Ports

- containing variables to be identified at the connections to adjacent components

## Constraints

- set of behavioural rules connecting the variables of the same component
- Normally, a constraint is only valid under the assumption of a certain behavioural mode.

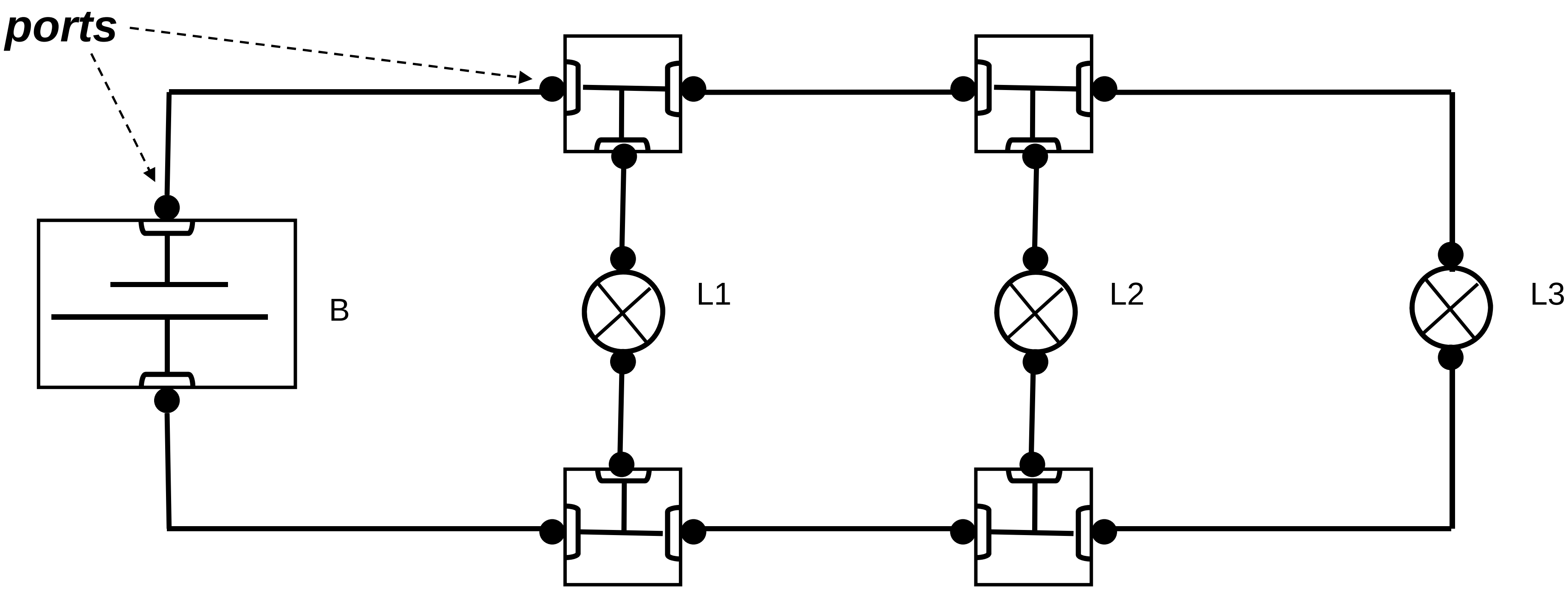
## Control actions

- variables and values to be set
- measure of accessibility and the difficulty to set certain values.

## Observations

- variables
- measure for accessibility

# Modeling a simple electric circuit




**component types:**

**Battery**

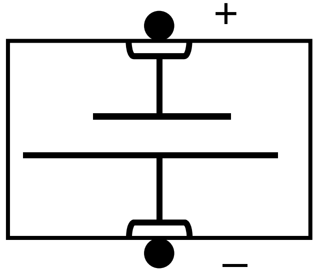
**Lamp**

**Wire**

**Junction (3)**

# Modeling a simple electric circuit

**Battery**




**fault modes:**

discharged
contact gap at +
contact gap at -
loose contact at +
loose contact at -
corroded

**control actions:**

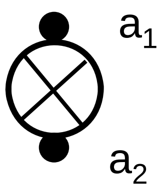
open connector at +
open connector at -
close connector at +
close connector at -

**observations:**

inspect connectors
measure voltage at +
measure voltage at -

# Modeling a simple electric circuit

**Lamp**




**fault modes:**

`blown`
`lamp is not inserted`
`loose contact`
`corroded`

**control actions:**

`remove lamp`
`insert lamp`

**observations:**

`inspect lamp`

**Wire**


**fault modes:**

`broken`
`shorted to ground`
`shorted to voltage`
`corroded`

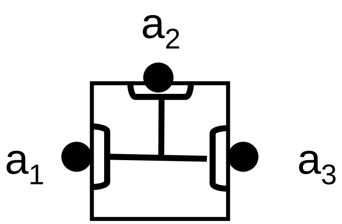
**control actions:**

**observations:**

measure voltage at $a_1$
measure voltage at $a_2$
`inspect wire`

# Modeling a simple electric circuit

**Junction (3)**




**fault modes:**

`contact gap at` $a_1$
`contact gap at` $a_2$
`contact gap at` $a_3$
`loose contact at` $a_1$
`loose contact at` $a_2$
`loose contact at` $a_3$

**control actions:**

`close contact at` $a_1$
`close contact at` $a_2$
`close contact at` $a_3$
`open contact at` $a_1$
`open contact at` $a_2$
`open contact at` $a_3$

**observations:** `inspect contacts`