Grundlagen der Künstlichen Intelligenz

Sebastian Iwanowski
FH Wedel

Kap. 5:
Verschiedene Wissensverarbeitungstechniken im Vergleich

5.3: Modellbasierte Verarbeitung

3. Modellbasierte Diagnose

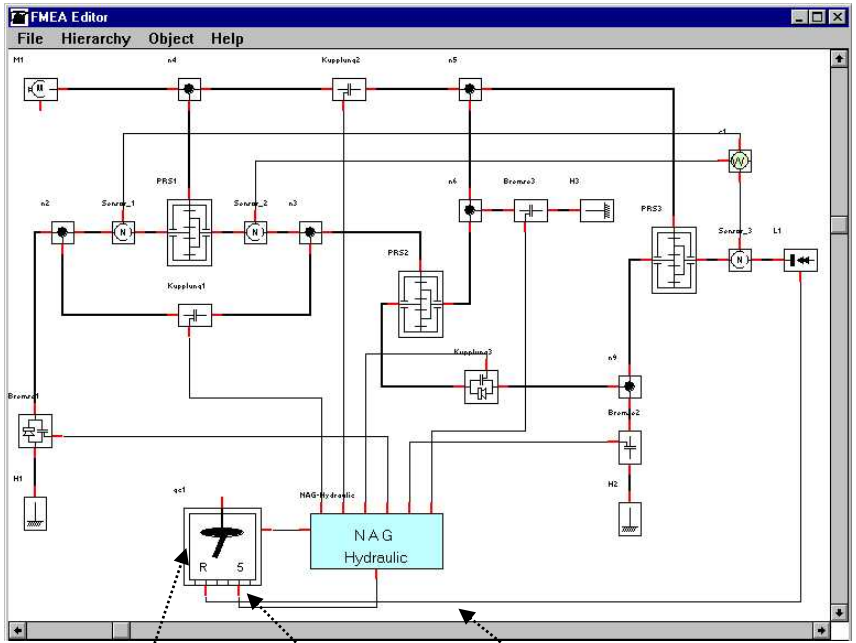
Ziel:

- schneller Wissenserwerb
- exaktes und nachvollziehbares Ergebnis der Problemlösungsmaschine

Schwierigkeit:

- schnelle Antwort der Problemlösungsmaschine zur Laufzeit

3. Modellbasierte Diagnose



<u>Systemstruktur</u>:

Welche Komponenten von welchem Typ sind wie miteinander verbunden ?

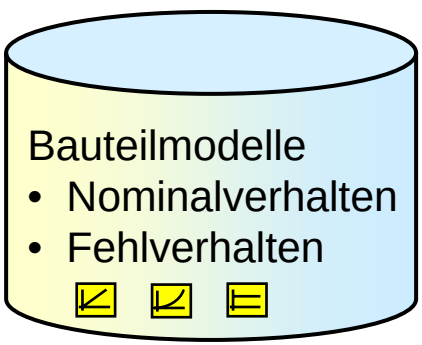
➔ ***erhältlich aus CAD-Daten***

<u>Komponentenmodelle</u>:

Wie ist die Abhängigkeit zwischen den Werten, die an den Verbindungs-punkten einer Komponente anliegen ?

➔ ***pro Komponententyp einmal zu modellieren***

➔***Modell ist wiederverwendbar für alle Systeme, in denen Komponenten dieses Typs enthalten sind***



3. Modellbasierte Diagnose

Eingabe in die Wissensbasis:

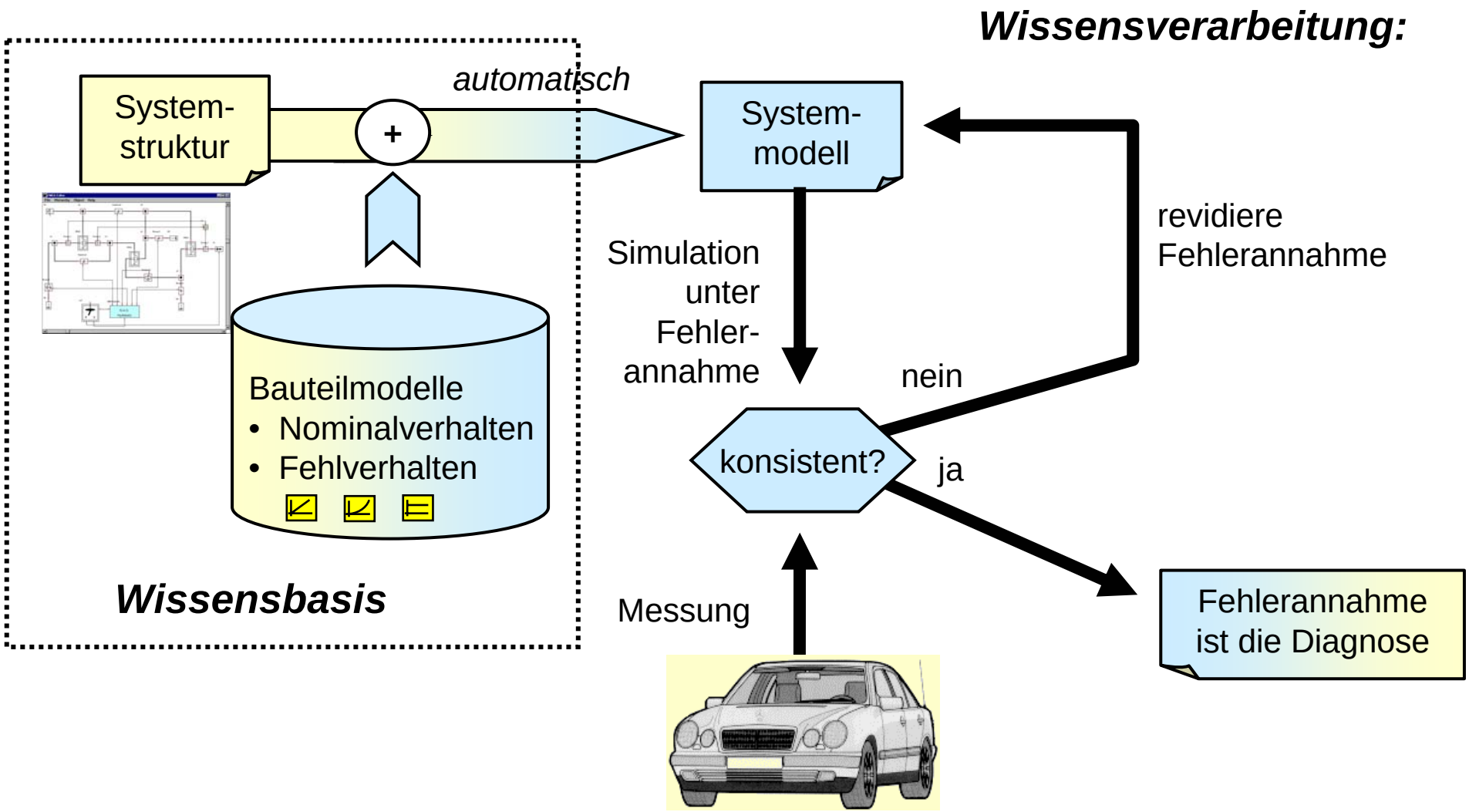
- Hierarchische Struktur des Systems (Aufbau aus Komponenten)
- Komponentenmodelle

Struktur der Wissensbasis:

- Constraint-Netzwerk (automatisch zusammengebaut)
- Gliederung des Constraint-Netzwerks durch:
  - Zuordnung der Constraints zu Komponenten bzw. Ports
  - Zuordnung der Variablen zu Komponenten bzw. Ports

Anwendungsgebiet: Technische Diagnose

Basisfunktionalität der modellbasierten Diagnose



Anwendungsgebiet: Technische Diagnose

Basisfunktionalität der modellbasierten Diagnose

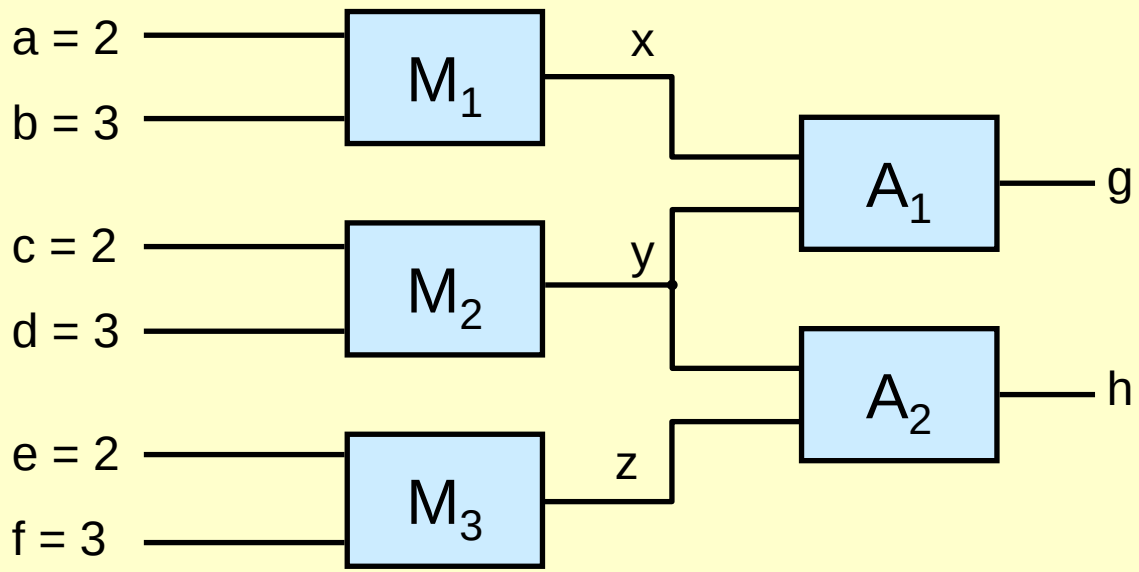
Die GDE 1987: *Der* Prototyp für die modellbasierte Diagnose

Problem:
- ‚brute-force' Simulation ***aller*** Fehlerannahmen kombinatorisch nicht realisierbar

Idee: General Diagnostic Engine GDE, deKleer & Williams 1987
- intelligente Suche im Raum der möglichen Fehlerannahmen
- nutzt inkonsistente Annahmen zum Verkleinern des Suchraums
- Prinzip: konfliktgesteuerte Suche

GDE - Beispiel

Systemstruktur

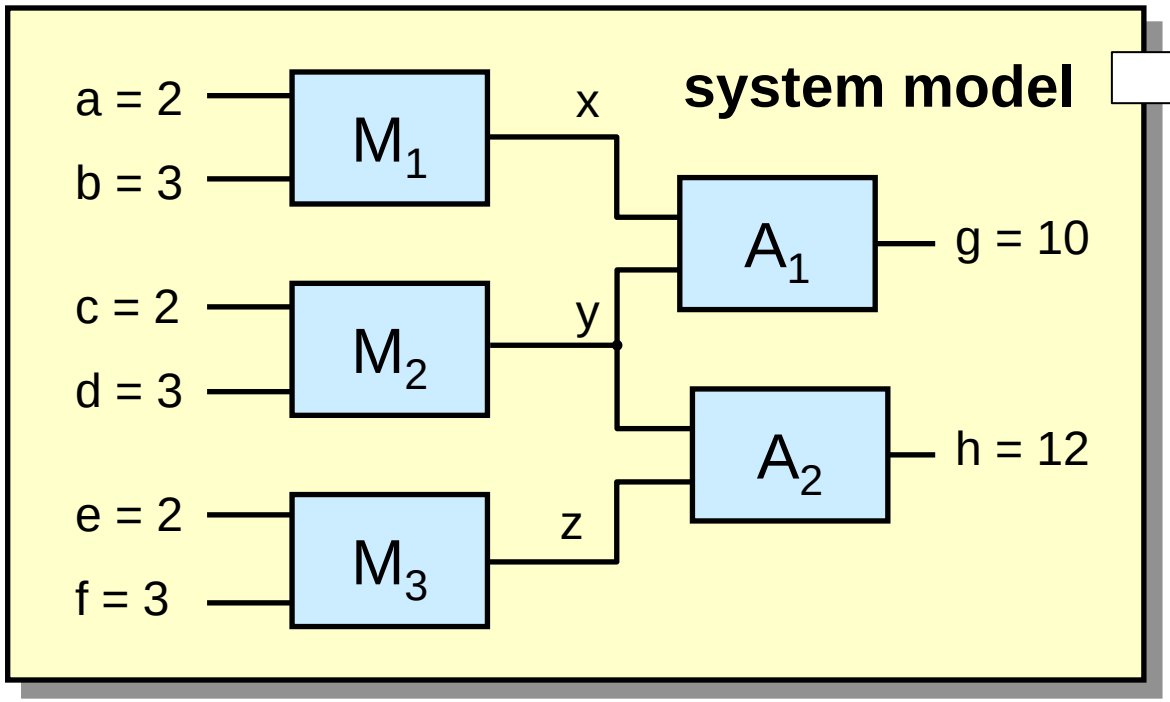


Komponentenmodelle

- Multiplizierer: mode=ok $\Rightarrow$ out = $in_1 * in_2$
- Addierer: mode=ok $\Rightarrow$ out = $in_1 + in_2$

Messungen $g = 10 \wedge h = 12$

GDE - Beispiel



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}

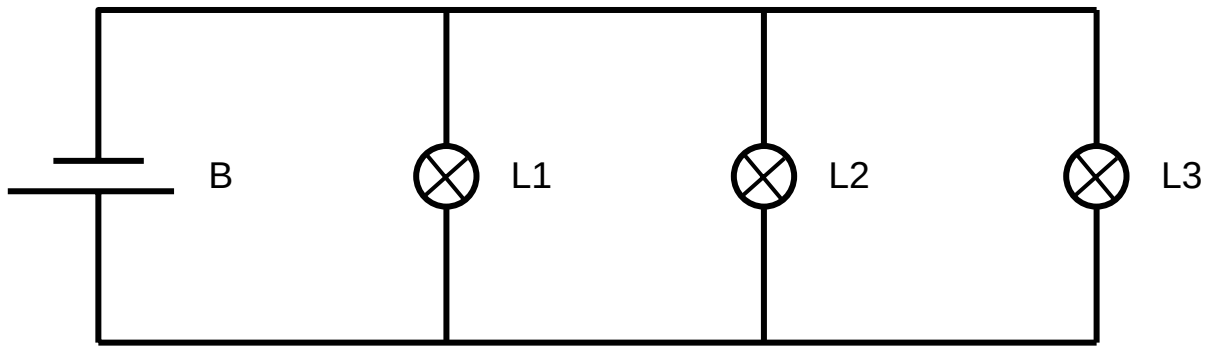
two conflicts

M1	M2	M3	A1	A2
X	X		X	
X		X	X	X

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

Modellbasierte Diagnose: Basisfunktionalität

Beispiel, warum Addierer-/Multiplizierer-Beispiel nicht alle Schwierigkeiten des GDE-Ansatzes aufzeigt:



Beobachtung:

L1, L2 leuchten nicht, L3 leuchtet

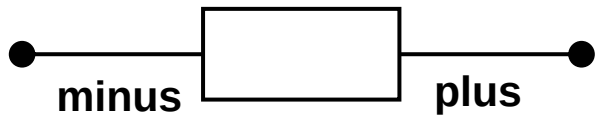
GDE-Diagnosen:

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

Modellbasierte Diagnose: Basisfunktionalität

Modellierung der elektrischen Komponenten:

Batterie:



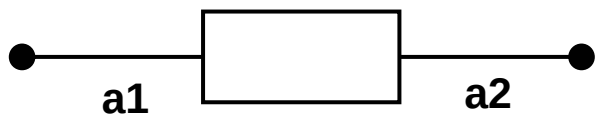
Wertebereiche: minus, plus ∈ { `Masse`, `Versorgungsspannung` }

Regeln:

ok ⇒ (minus = `Masse`)

ok ⇒ (plus = `Versorgungsspannung`)

Kabel:



Wertebereiche: a1, a2 ∈ { `Masse`, `Versorgungsspannung` }

Regeln:

ok ∧ (a1 = `Masse`) ⇒ (a2 = `Masse`)

ok ∧ (a1 = `Versorgungsspannung`) ⇒ (a2 = `Versorgungsspannung`)

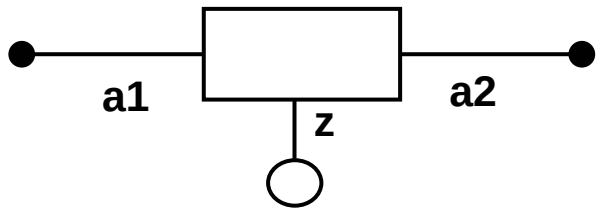
ok ∧ (a2 = `Masse`) ⇒ (a1 = `Masse`)

ok ∧ (a2 = `Versorgungsspannung`) ⇒ (a1 = `Versorgungsspannung`)

Modellbasierte Diagnose: Basisfunktionalität

Modellierung der elektrischen Komponenten:

Lampe:



Wertebereiche:

$a1, a2 \in \{ \texttt{Masse}, \texttt{Versorgungsspannung} \}$

$z \in \{ \texttt{hell}, \texttt{dunkel} \}$

Regeln:

$ok \wedge (a1 = \texttt{Versorgungsspannung}) \wedge (a2 = \texttt{Masse}) \Rightarrow (z = \texttt{hell})$

$ok \wedge (a2 = \texttt{Versorgungsspannung}) \wedge (a1 = \texttt{Masse}) \Rightarrow (z = \texttt{hell})$

$ok \wedge (a1 = \texttt{Versorgungsspannung}) \wedge (a2 = \texttt{Versorgungsspannung}) \Rightarrow (z = \texttt{dunkel})$

$ok \wedge (a1 = \texttt{Masse}) \wedge (a2 = \texttt{Masse}) \Rightarrow (z = \texttt{dunkel})$

$ok \wedge (a1 = \texttt{Masse}) \wedge (z = \texttt{hell}) \Rightarrow (a2 = \texttt{Versorgungsspannung})$

$ok \wedge (a1 = \texttt{Versorgungsspannung}) \wedge (z = \texttt{hell}) \Rightarrow (a2 = \texttt{Masse})$

$ok \wedge (a1 = \texttt{Masse}) \wedge (z = \texttt{dunkel}) \Rightarrow (a2 = \texttt{Masse})$

$ok \wedge (a1 = \texttt{Versorgungsspannung}) \wedge (z = \texttt{dunkel}) \Rightarrow (a2 = \texttt{Versorgungsspannung})$

$ok \wedge (a2 = \texttt{Masse}) \wedge (z = \texttt{hell}) \Rightarrow (a1 = \texttt{Versorgungsspannung})$

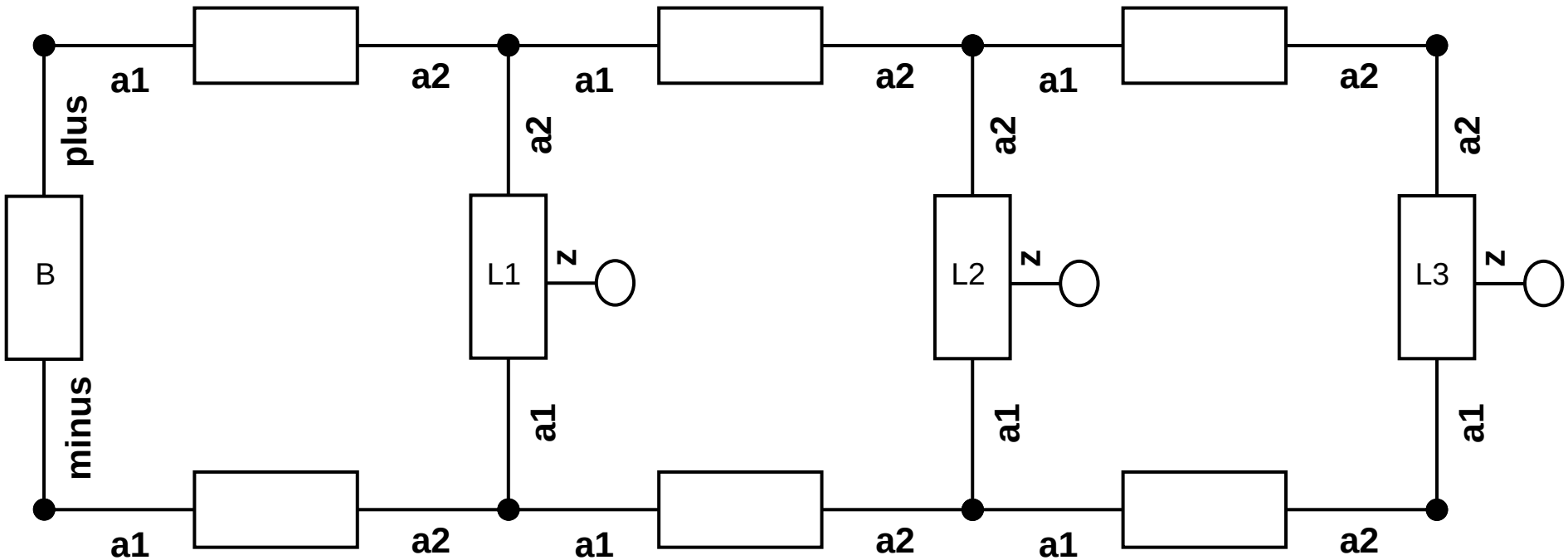
$ok \wedge (a2 = \texttt{Versorgungsspannung}) \wedge (z = \texttt{hell}) \Rightarrow (a1 = \texttt{Masse})$

$ok \wedge (a2 = \texttt{Masse}) \wedge (z = \texttt{dunkel}) \Rightarrow (a1 = \texttt{Masse})$

$ok \wedge (a2 = \texttt{Versorgungsspannung}) \wedge (z = \texttt{dunkel}) \Rightarrow (a1 = \texttt{Versorgungsspannung})$

Modellbasierte Diagnose: Basisfunktionalität

Zusammenbau des Systemmodells aus den Komponentenmodellen:



Werte an den Verbindungsknoten müssen gleich sein

Bei Widerspruch: Konflikt der den Werten zugrunde liegenden Verhaltensmodelle

Diagnosen sind Mengen von Verhaltensmodellen, die keinen Konflikt enthalten

Modellbasierte Diagnose: Basisfunktionalität

Fazit aus der bisher vorgenommenen Modellierung:

Es besteht kein logischer Widerspruch zu folgender Diagnose:

2. (B defekt, L1 ok, L2 ok, L3 defekt)

Grund:

L3 darf im Fehlerfall auch leuchten, wenn keine Spannungsdifferenz besteht

Unvollständigkeit der Wissensbasis !

Noch schlimmer:

Wenn eine Verhaltensregeln nur ausgewertet wird, wenn für ihre Voraussetzungen konkrete Werte vorliegen, dann kann auch kein Widerspruch zu folgender Diagnose gefunden werden:

3. (B defekt, L1 ok, L2 ok, L3 ok)

Grund:

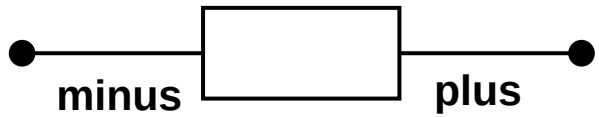
Es werden nirgendwo Spannungwerte berechnet

Mangelnde Beweisfähigkeit der Problemlösungskomponente !

Modellbasierte Diagnose: Basisfunktionalität

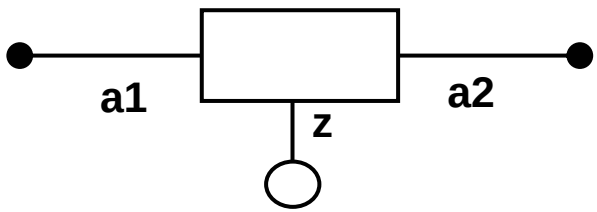
Zusätzliche Regeln für den Ausschluss der Diagnosen 2 / 3:

Batterie:



defekt ⇒ (minus = `Masse`)

Lampe:



defekt ∧ (a1 = `Versorgungsspannung`) ∧ (a2 = `Versorgungsspannung`) ⇒ (z = `dunkel`)

defekt ∧ (a1 = `Masse`) ∧ (a2 = `Masse`) ⇒ (z = `dunkel`)

Es müssen also auch Verhaltensmodelle für Fehler angegeben werden, um physikalisch unmögliches Verhalten auszuschließen.

Modellbasierte Diagnose: Erweiterte Funktionalität

Basisfunktionalität:

Eingabe:

- Einstellung bestimmter Werte im System
- Beobachtung davon abhängiger Werte im System

Ausgabe:

- Mehrere Diagnosen folgender Art:
  - Jede Diagnose weist jeder Komponente einen Verhaltensmodus zu: entweder ok oder ein definierter Fehlermodus
  - Die Regeln aller zugewiesenen Verhaltensmodi sind konsistent (mit allen eingestellten und beobachteten Werten)

Was braucht der Anwender ?

Eingabe: s.o.

Ausgabe:

- Eine eindeutige Anweisung, welche Komponenten wie repariert werden sollen

Modellbasierte Diagnose: Erweiterte Funktionalität

Erweiterte Funktionalität:

1) Vorschlag von Testeinstellungen (control inputs)

- Einstellung bestimmter Werte an bestimmten Stellen im System

  (derart, dass die zu erwartenden Beobachtungen die bisher gültigen Diagnosen bestmöglich unterscheiden)

2) Vorschlag von Beobachtungspunkten

- Auswahl von Messstellen im System

  (derart, dass die zu erwartenden Beobachtungen die bisher gültigen Diagnosen bestmöglich unterscheiden)

} **Test**

Anforderung an die Modellierung:

- Definition von Testpunkten
- Definition von Testwerten, die an den Testpunkten eingestellt werden sollen

} **Control actions**

- Definition von Beobachtungspunkten, die gemessen werden sollen

Observations

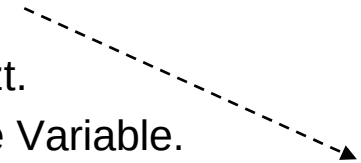
Komponentenmodellierung

Verhaltensmodi *(behavioural modes)*

- Werte der Komponente, nach denen im Diagnosevorgang gesucht wird
- Definitionsbereich immer endlich (in der Regel unter 10 verschiedene Werte möglich)

Variablen

- zum Abspeichern von Werten
- Die Variablenwerte werden in den Constraints benutzt.
- Die Constraints berechnen einen neuen Wert für eine Variable.



Unterscheide interne Variablen von Portvariablen !

Ports

- enthält die Variablen, die in den Verbindungen mit benachbarten Komponenten identifiziert werden sollen.

Constraints

- Menge von Verhaltensregeln, welche Variablen der Komponente miteinander verknüpfen
- Ein Constraint gilt normalerweise nur unter Voraussetzung, dass der Verhaltensmodus einen bestimmten Wert hat.

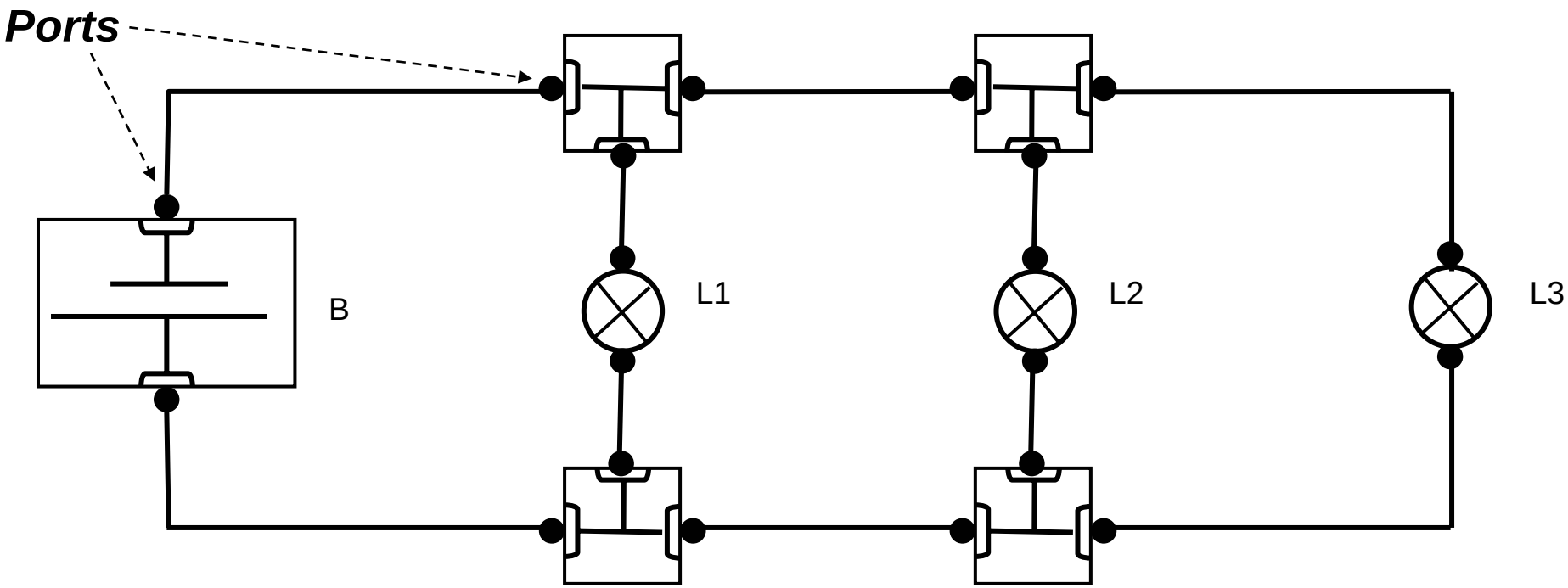
Maßnahmen *(control actions)*

- Variablen und einzustellende Werte dafür
- Maß für Zugänglichkeit und Schwierigkeit, bestimmte Werte einzustellen

Beobachtungen *(observations)*

- Variablen
- Maß für Zugänglichkeit

Modellierung eines einfachen elektrischen Systems

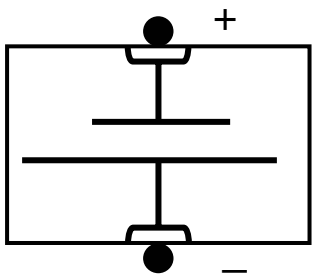


Komponententypen:

- **Batterie**
- **Lampe**
- **Kabel**
- **Steckverbindung (3)**

Modellierung eines einfachen elektrischen Systems

Batterie



Fehlermodi:

- entladen
- Kontaktlücke bei +
- Kontaktlücke bei -
- Wackelkontakt bei +
- Wackelkontakt bei -
- korrodiert

Maßnahmen:

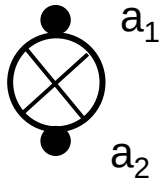
- Klemme bei + lösen
- Klemme bei - lösen
- Klemme bei + befestigen
- Klemme bei - befestigen

Beobachtungen:

- Klemmen ansehen
- Spannung bei + messen
- Spannung bei - messen

Modellierung eines einfachen elektrischen Systems

Lampe



Fehlermodi:
- durchgebrannt
- Lampe nicht eingesetzt
- Wackelkontakt
- korrodiert

Maßnahmen:
- Lampe ausschrauben
- Lampe einschrauben

Beobachtungen:
- Lampe ansehen

Kabel

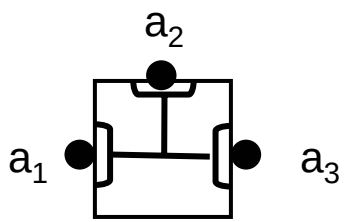
Fehlermodi:
- unterbrochen
- Kurzschluss gegen Masse
- Kurzschluss gegen Spannung
- korrodiert

Maßnahmen:

Beobachtungen:
- Spannung an a_1 messen
- Spannung an a_2 messen
- Kabel ansehen

Modellierung eines einfachen elektrischen Systems

Steckverbindung (3)



Fehlermodi:

- Kontaktlücke bei a_1
- Kontaktlücke bei a_2
- Kontaktlücke bei a_3
- Wackelkontakt bei a_1
- Wackelkontakt bei a_2
- Wackelkontakt bei a_3

Maßnahmen:

- Kontakt bei a_1 schließen
- Kontakt bei a_2 schließen
- Kontakt bei a_3 schließen
- Kontakt bei a_1 lösen
- Kontakt bei a_2 lösen
- Kontakt bei a_3 lösen

Beobachtungen:

- Kontakte ansehen