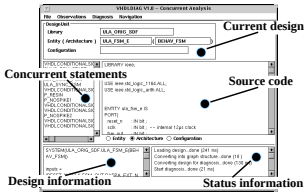
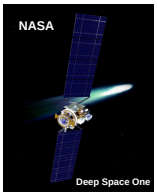
Konzepte der AI – Diagnose

Franz Wotawa

Institut für Informationssysteme,
Database and Artificial Intelligence Group,
Technische Universität Wien
Email: wotawa@dbai.tuwien.ac.at

Anwendungsgebiete

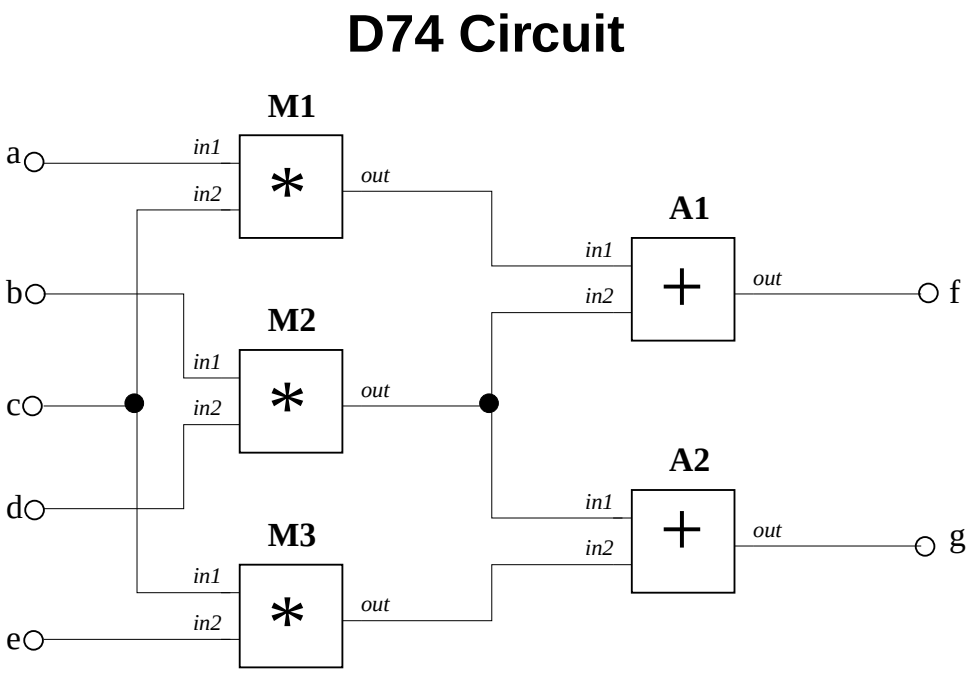




- Software debuggging
- Automobilindustrie (On-board, Off-board Diagnose)
- NASA (Deep Space One)
- Überwachung technischer Systeme (Monitoring)

Ziel der Diagnose

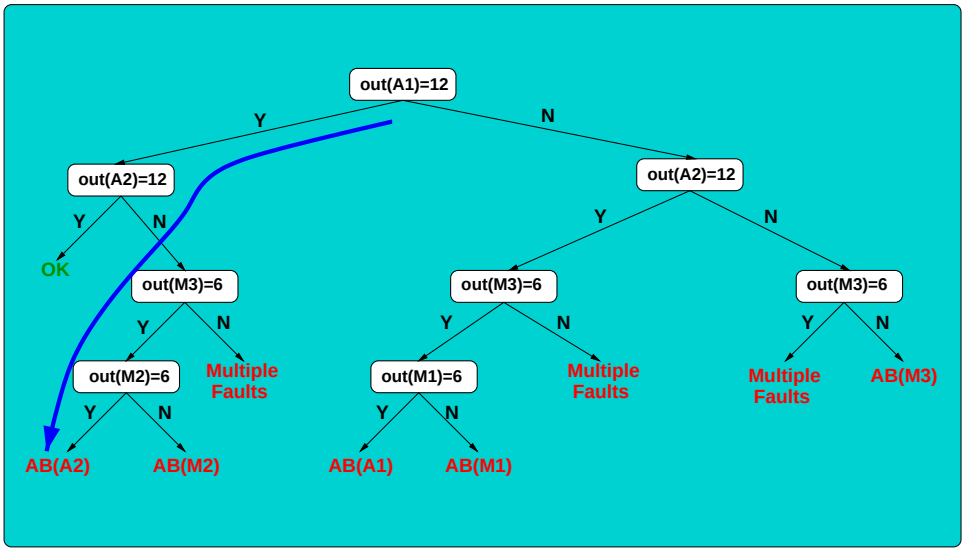
Identifikation von Komponenten eines (physikalischen) Systems, die für ein fehlerhaftes Verhalten verantwortlich sind.



Techniken

- Decision Trees
- Neuronale Netze
- Regelbasierte Systeme
- Modellbasierte Systeme
- Bayes'sche Netze

Decision Tree for D74 Circuit



Probleme..

. . . bei der Verwendung von Decision Trees, Neuronalen Netzen, Regelbasierten Systemen, und Bayes'schen Netzen:

- Kann nur für ein System verwendet werden.
- Schlecht an andere Systeme adaptierbar.
- Es ist i.A. schwer ein entsprechendes Modell (z.B. einen Decision Tree) zu erzeugen.
- Nur eingeschränkte Diagnose (Einfachfehler-Hypothese)
- Wenig direkter Bezug zu physikalischen Modellen

. . . ABER . . .

- Schnelle Diagnoseberechnung
- Begrenzter Speicherplatz notwendig (nur für das Modell)
- Einfachfehler meistens ausreichend

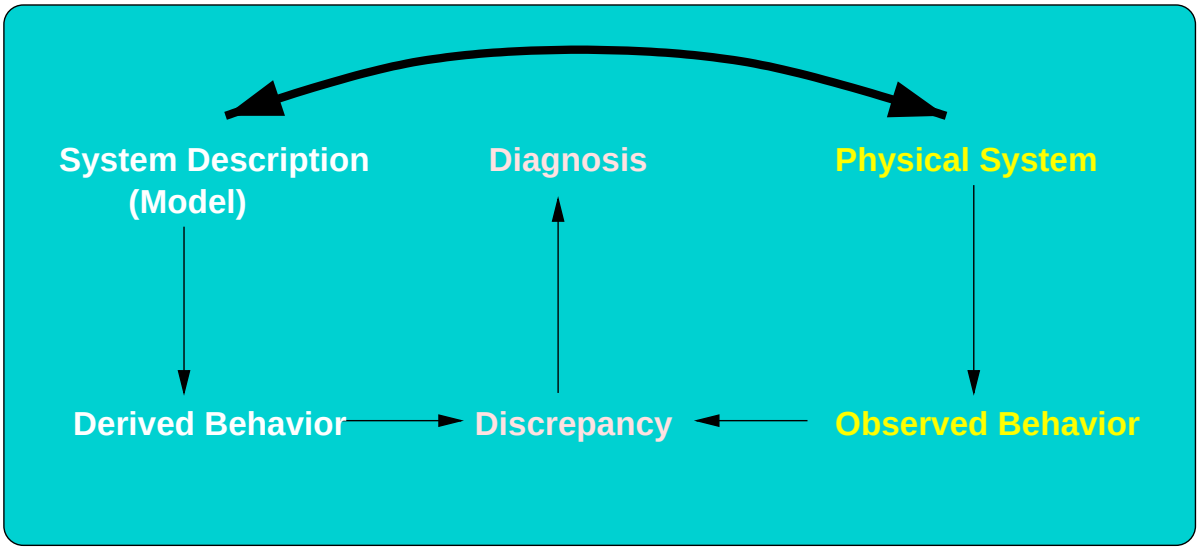
Abhilfe..

. . . durch die Modellbasierte Diagnose.

Reasoning from first principles

- Verwendung von physikalischen Modellen
- Nur das korrekte Verhalten muß modeliert werden.
- Komponentenorientierte Modelle
- Wiederverwendung der Modelle möglich (REUSE)
- Einfache Erstellung Modelle komplexer Systeme
- Änderungen nur lokal (für Komponenten)
- Es existieren auch schnelle Algorithmen
- Berechnungszeit und Speicherplatz begrenzt aber Worst-Case $O(2^{|COMPONENTS|})$.

Idee der Modellbasierten Diagnose



Was ist ein Modell?

Beispiel: D74 Schaltkreis

Multiplizierer

$$MULT(C) \rightarrow (\neg AB(C) \Rightarrow (out(C) = in_1(C) \cdot in_2(C)))$$

Addierer

$$ADD(C) \rightarrow (\neg AB(C) \Rightarrow (out(C) = in_1(C) + in_2(C)))$$

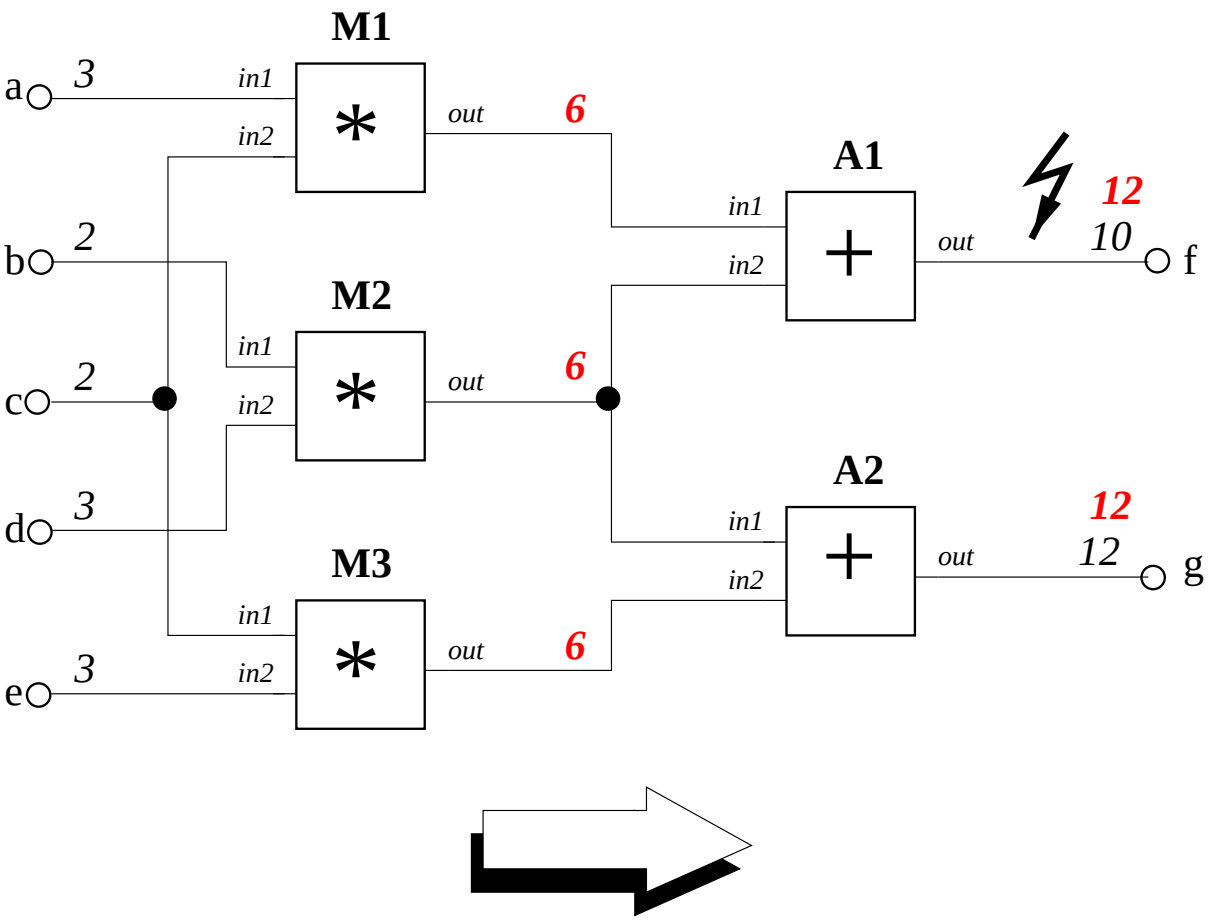
Strukturmodell

$$MULT(M1) \wedge MULT(M2) \wedge MULT(M3)$$
$$ADD(A1) \wedge ADD(A2)$$

$$out(M1) = in_1(A1) \wedge out(M2) = in_2(A1)$$
$$out(M2) = in_1(A2) \wedge \ldots$$

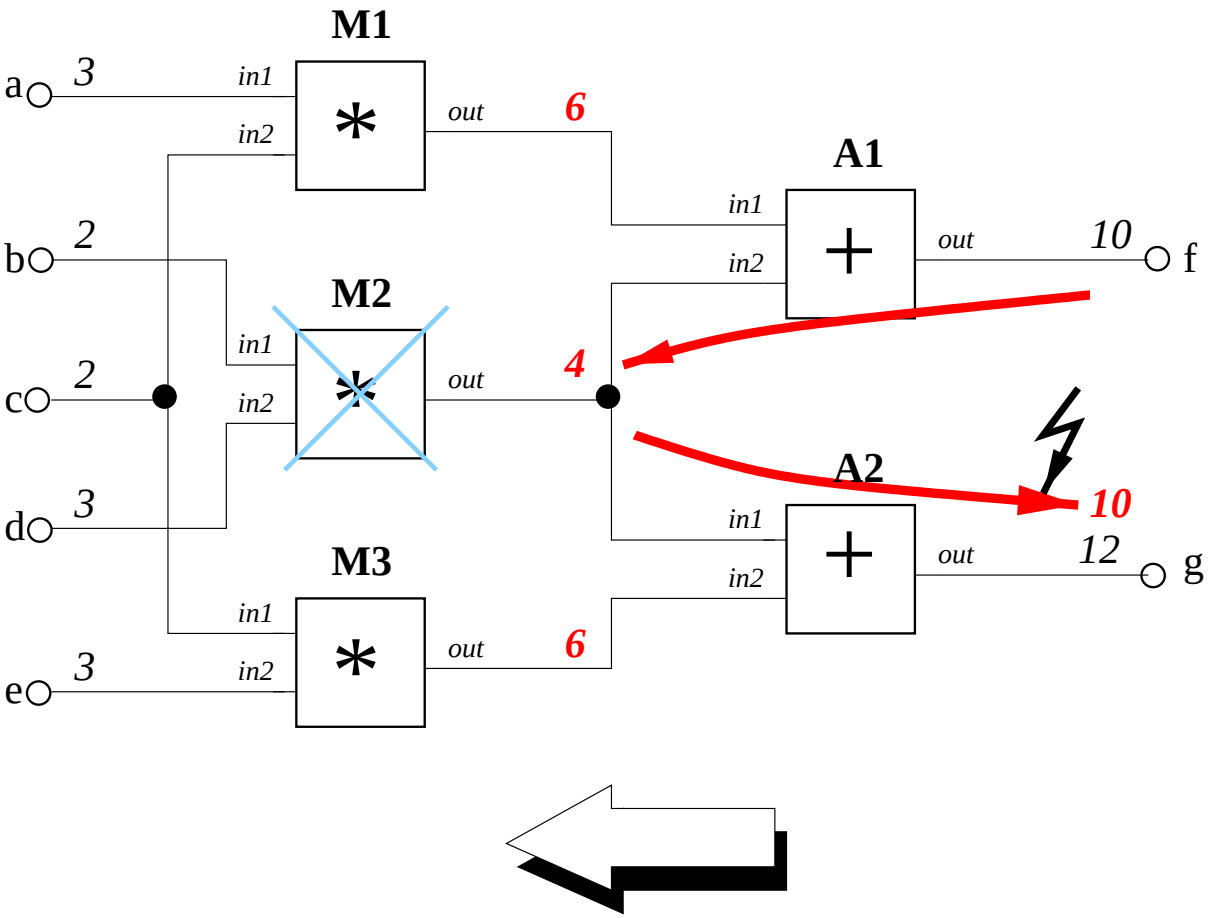
Berechnung von Diagnosen (I)

Forward Propagation



Berechnung von Diagnosen (II)

Backward Propagation



Berechnung von Diagnosen (III)

Resultate

- 2 **Einfachfehler**

$$\{AB(M1)\}, \{AB(A1)\}$$

- 2 **Mehrfachfehler**

$$\{AB(M2), AB(M3)\}, \{AB(M2), AB(A2)\}$$

- Berechnung basiert auf Systemmodell ! (Reasoning from first principles)

Definitionen

Diagnosesystem $(SD, COMP)$ besteht aus einer Systembeschreibung SD (dem Modell) und einer Menge von Komponenten $COMP$.

Diagnoseproblem $(SD, COMP, OBS)$ besteht aus einem Diagnosesystem und einer Menge von Beobachtungen OBS, z.B.: $\{in_1(M1) = 3, \ldots\}$.

Diagnose Eine Diagnose Δ für ein Diagnoseproblem $(SD, COMP, OBS)$ ist eine Teilmenge der Komponentenmenge mit folgender Eigenschaft. Wenn man annimmt, daß alle Komponenten in Δ nicht funktionieren und die anderen Komponenten funktionieren, erhält man keinen Widerspruch zu den Beobachtungen.

$$SD \cup OBS \cup \{AB(C) | C \in \Delta\} \cup \{\neg AB(C) | C \in COMP \setminus \Delta\} \not\models \bot$$

Eigenschaften von Diagnosen

- Eine Diagnose existiert wenn $SD \cup OBS$ konsistent sind.
- $\emptyset$ ist eine Diagnose wenn $SD \cup OBS \cup \{\neg AB(C) | C \in COMP\}$ konsistent ist.
- Jede Übermenge einer Diagnose ist selber eine Diagnose.
- Eine Diagnose heißt minimal, wenn keine Untermenge eine Diagnose ist.

Algorithmen

- **Einfacher Algorithmus**

  Wähle eine Teilmengen Δ der Komponentenmenge aus und prüfe ob das System mit den Beobachtungen und den Korrektheitsannahmen von Komponenten konsistent ist. Ist dies der Fall, dann ist Δ eine Diagnose.

  Algorithmus ineffizient! Konsistenz nicht immer leicht prüfbar!
- Reiter Algorithmus ([Reiter, Artificial Intelligence 1987]) basierend auf Conflicts und Hitting Sets.
- GDE “General Diagnosis Engine” ([DeKleer et al, Artificial Intelligence 1987])
- DRUM ([Fröhlich et al, IJCAI 1997])
- SAB ([Dechter et al, IJCAI 1995])
- TREE ([Stumptner et al, IJCAI 1997])

Runtime (DRUM Algorithmus)

Circuit	Gates	Inputs	Outputs	**DRUM** (Sun Ultra-170 MHz) [Milliseconds]
C499	202	41	32	28
C880	383	60	26	17
C1355	547	41	32	290
C2670	1193	233	140	130
C3540	1669	50	22	2,760
C5315	2307	178	123	60
C6288	2406	32	32	54,940
C7522	3512	207	108	1,230

Berechnung aller Einzeldiagnosen. Die Diagnose-zeit hängt sehr stark von der Struktur ab (C6288 hat sehr viele Verbindungen).

Modellierung

- Quantitative Modelle: infiniter Wertebereich, Differentialgleichungen, Analysis

  Beispiel: *Die Beschleunigung ist die 1. Ableitung der Geschwindigkeit nach der Zeit*

$$a = \frac{\partial v}{\partial t}$$

- Qualitative Modelle: finiter Wertebereich, qualitative Differentialgleichungen (QDE), Algebra

  Beispiel: *Wenn die Geschwindigkeit sich verändert haben wir eine Beschleunigung.*

$$M^{+}(A, V)$$

Qualitative Modelle

- Kausale Modelle, z.B.: *Wenn die Temperatur des Wassers 0 Grad Celsius erreicht bzw. unterschreitet, dann kommt es zur Eisbildung* $Temp \leq 0 \rightarrow Ice$

- Andere Modelle (Algebren, QDE, . . .)

  Beispiel: Widerstand R

Quantitatives M.	Qualitatives M.
$U = R \cdot I$	$current \leftrightarrow voltage\_drop$

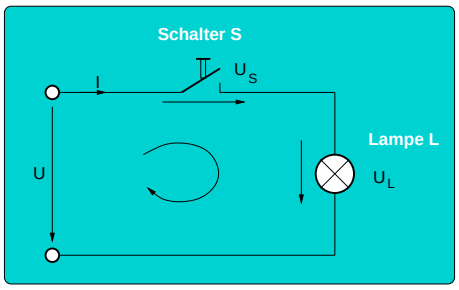
- **Ziel:** Modellierung von (physikalischen) Wissen um Erklärungen geben bzw. um Probleme (z.B. Diagnose) lösen zu können.

Probleme bei der Modellierung

Komponentenmodelle müssen Unabhängig von ihrer Verwendung sein!

Beispiel: Schalter

$$on(S) \rightarrow current \wedge \neg voltage\_drop$$
$$off(S) \rightarrow \neg current$$



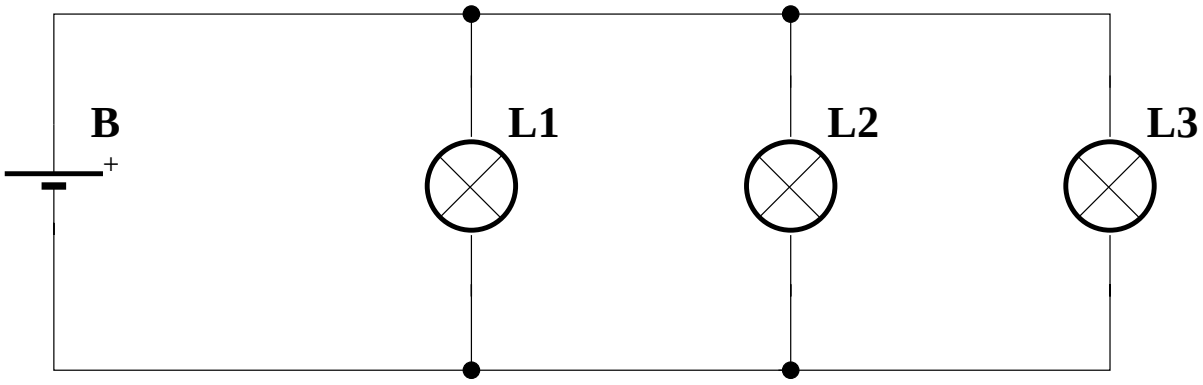
Wenn $U = 0$, aber $on(S)$ folgt $current$ und $\neg voltage\_drop$. Da eine Lampe dann leuchtet wenn Strom durch sie fließt, folgt daß L leuchtet. Im physikalischen System kann dies aber nicht sein (da $U = 0$) $\Rightarrow$ Das Modell ist falsch

Besserers Modell: $on(S) \rightarrow \neg voltage\_drop$, $off(S) \rightarrow \neg current$

Anmerkung: Ob ein Strom durch einen Schalter fließt hängt auch davon ab ob eine (Strom-)Quelle vorhanden ist.

Erweiterungen

Fehlermodelle



Modell des **KORREKTEN** Verhaltens:

$$battery(C) \Rightarrow (\neg ab(C) \rightarrow voltage(C))$$

$$bulb(C) \Rightarrow (\neg ab(C) \rightarrow (voltage(C) \leftrightarrow light(C)))$$

$$light(C) \wedge \neg light(C) \rightarrow \bot$$

$$battery(B) \wedge bulb(L1) \wedge bulb(L2) \wedge bulb(L3)$$

$$voltage(B) \rightarrow (voltage(L1) \wedge voltage(L2) \wedge voltage(L3))$$

Beobachtungen:

$$light(L1) \wedge \neg light(L2) \wedge \neg light(L3)$$

Fehlermodelle (I)

Diagnosen: $\{B\}, \{L2, L3\}$

$\{B\}$ sollte keine Diagnose sein!

Abhilfe: Auch das Fehlverhalten modellieren (Fehlermodelle)

$$battery(C) \Rightarrow (ab(C) \rightarrow \neg voltage(C))$$

$$\neg voltage(B) \rightarrow (\neg voltage(L1) \wedge \neg voltage(L2) \wedge \neg voltage(L3))$$

$$bulb(C) \Rightarrow (ab(C) \rightarrow \neg light(C))$$

Fehlermodelle (II)

- Fehlermodelle nicht immer vorhanden
- Weniger Diagnosen
- Mehr Zeit für die Berechnung der Diagnosen
- Eigenschaften gelten nicht mehr (Jede Übermenge einer Diagnose ist nicht mehr automatisch eine Diagnose)
- Besser **Physical Impossibilities**

  $light(L1) \vee light(L2) \vee light(L3) \wedge \neg voltage(B) \rightarrow \bot$