Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-92-18

Verfahren der automatisierten Diagnose technischer Systeme

Klaus Becker

Juni 1992

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Philips, SEMA Group Systems, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

Verfahren der automatisierten Diagnose technischer Systeme

Klaus Becker

DFKI-D-92-18

Diese Arbeit wurde von Prof. Michael M. Richter und Herrn Dipl.-Inform. Ansgar Bernardi betreut

Diese Arbeit wurde finanziell unterstützt durch das Bundesministerium für Forschung und Technologie (FKZ ITW-8902 C4).

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ein Haupteinsatzbereich von Expertensystemen ist die Diagnose fehlerhafter Systeme. Der Nutzen derartiger *Diagnosesysteme* liegt in der Einsparung von Zeit und Kosten.
Die Verfügbarkeit der Produktionsanlage steigt, der Einsatz von Kundenbetreuern verringert sich, Instandhaltungszeiten und -kosten werden minimiert, Experten und Fachspezialisten erfahren bei einfachen Störungen eine Entlastung und können vermehrt für höherqualifizierte Arbeiten eingesetzt werden.

Störungen bzw. Fehler äußern sich in Form von unerwarteten Symptomen[1] . Im trivialen Fall reicht es aus, die Komponenten, bei denen die unerwarteten Symptome beobachtet wurden, einfach auszutauschen. I. allg. jedoch ist der Ort der Fehlerursache nicht derselbe, wie der Ort, an dem sich der Fehler durch unerwartete Symptome bemerkbar macht. Dies liegt daran, daß sich in einem komplexen System ein Fehler über mehrere Komponenten hinweg fortpflanzt und erst an einer für den Menschen erkennbaren Oberfläche zutage tritt.
Aufgabe eines Diagnosesystems ist es, diese Ursache anhand der aufgetretenen Symptome mit vertretbarem Aufwand an Zeit und Kosten zu ermitteln. Dazu bedient es sich eines *Diagnoseverfahrens.*

Die unterschiedlichen Einsatzgebiete (Prozeßdiagnose, Reparaturdiagnostik, Überwachung, Netzwerkdiagnostik, ...), die Komplexität der zu diagnostizierenden Objekte und die Vielfalt der Formen, in denen Wissen über die zu diagnostizierenden Objekte vorliegen kann, bedingen eine breite Palette von Diagnoseverfahren.
Wissen (bzw. Information) kann z. B. in der Form einer Struktur- und Funktionsbeschreibung des zu diagnostizierenden Objektes vorliegen. Es kann aber auch in Form von auf Erfahrung beruhenden Assoziationen zwischen Symptomen und Fehlerursachen vorhanden sein. Eine weitere Möglichkeit besteht darin, daß das Wissen in Form schon aufgetretener Diagnosefälle vorliegt, die während der Diagnose zum Vergleich mit dem aktuellen Fall herangezogen werden.
Dies sind nur einige der Möglichkeiten, welche Formen Wissen zur Diagnose annehmen kann. Es ist daher ein Diagnoseverfahren — und damit i. allg. auch ein Diagnosesystem — zu wählen, das der zugrundeliegenden Informationsstruktur (Wissensstruktur) und der Komplexität des

[1] Unter einem Symptom ist ein Tupel (M,A) zu verstehen. M steht für ein Merkmal, A für eine Ausprägung, die das Merkmal annehmen kann.

Anwendungsbereiches gerecht wird.

Als prominentes Beispiel für unterschiedliche Anwendungsbereiche bieten sich Medizin und Technik an.
Während in der Medizin über einen langen Zeitraum Aufzeichnungen und Erfahrungen vorliegen, ist dies im technischen Bereich — bedingt durch den technologischen Fortschritt — häufig nicht der Fall. In technischen Domänen liegt dagegen exaktes Wissen über Aufbau und Funktionsweise des Diagnoseobjektes vor, da die Entwurfskonzepte sowie die das Verhalten bestimmenden Gesetze bekannt sind. Aufgrund dessen sind technische Systeme oft exakter modellierbar und es bietet sich z. B. ein Diagnoseverfahren an, das auf einer Funktions- und Strukturbeschreibung des Diagnosebereiches aufsetzt (funktionales Diagnoseverfahren).
Im medizinischen Bereich dagegen sind funktionale Zusammenhänge häufig nur schwer oder überhaupt nicht zu erfassen. Sie liegen daher — wenn überhaupt — in statistischer Form und in Form von Erfahrungen (Heuristiken) vor. Dies liegt daran, daß das zu diagnostizierende Objekt (der menschliche Körper) sehr komplex und seine genaue Funktionsweise nicht bekannt ist. Daher erscheint z.B. die Verwendung eines auf Heuristiken aufgebauten Diagnosesystems im medizinischen Bereich sinnvoll.

Gegenstand dieser Arbeit ist die Vorstellung und Beurteilung allgemeiner und spezieller Diagnoseverfahren für technische Anwendungsbereiche.

Im folgenden wird eine kurze Beschreibung des Inhaltes der Arbeit gegeben.

Die Arbeit ist dreigeteilt. Das erste Kapitel beleuchtet die allgemeinen, ein Diagnoseverfahren betreffenden, Aspekte.
Dazu wird zunächst diskutiert, welche Anforderungen ein potentieller Anwender an ein Diagnoseverfahren stellt bzw. stellen muß. Die Anforderungen ergeben sich teilweise aus dem Wissen über die Anwendungsdomäne. Dieses Wissen kann in verschiedenen Formen vorliegen. Aus diesen Formen ergibt sich eine Einteilung des Diagnoseexpertensystems in verschiedene Kategorien, die im Anschluß an die Anforderungen vorgestellt werden. Da das eigentliche Diagnoseverfahren durch das verwendete Wissen bis zu einem gewissen Grad determiniert ist, wird zu jeder Kategorie von Diagnoseexpertensystemen ein Basisalgorithmus angegeben. Dieser Basisalgorithmus stellt das prinzipielle Gerüst eines Diagnosealgorithmus dar, der auf dem zu jener Kategorie gehörenden Wissen aufbaut.
Das erste Kapitel schließt mit einer Bewertung der Tauglichkeit der einzelnen Diagnoseklassen vor dem Hintergrund der technischen Diagnose.

Das zweite Kapitel zeigt praktische Realisierungen der im ersten Kapitel beschriebenen theoretischen Konzepte auf.
Jedes Diagnoseverfahren läßt sich in verschiedene Phasen bzw. Komponenten zergliedern. Zu Beginn des zweiten Kapitels sollen deshalb die Einzelkomponenten, aus denen sich komplette Verfahren zusammensetzen, vorgestellt werden. Dabei werden zu jeder Komponente mögliche Realisierungen angegeben. Auf der Komponentenbeschreibung aufbauend werden anschließend existierende Diagnosesysteme beschrieben. Dabei wurde darauf geachtet, daß zu jeder Diagnoseklasse zumindest ein System vorhanden ist.
Es wird versucht, jedes Diagnoseverfahren kurz zu bewerten. Dies geschieht u. a. auf der Basis der Anforderungen an ein Diagnoseverfahren aus Kapitel 1.

Der dritte Teil der Arbeit besteht aus der Erstellung eines kleinen technischen Diagnosesystems mittels einer Diagnosesystem–Shell. Der Sinn dieses praktischen Teils besteht weniger darin, ein brauchbares Expertensystem zu kreieren, sondern darin, die theoretischen Konzepte der zugehörigen Diagnosesystem–Kategorie zu verifizieren und die Brauchbarkeit einer allgemeinen Domänentheorie für die Erstellung eines speziellen Diagnosesystems zu prüfen.

Kapitel 2

Allgemeine Konzepte von Diagnoseverfahren

In diesem Kapitel geht es um die Vorstellung der verschiedenen Grundarten von Diagnoseverfahren. Eingeleitet wird es jedoch durch die Angabe von Kriterien, an denen ein Benutzer ein Diagnosesystem mißt. Einige dieser Kriterien sind relativ unabhängig von der Art des Diagnoseverfahrens. Auf sie wird erst in Kapitel 3.2 über konkrete Realisierungen eingegangen. Die anderen Kriterien werden zur Bewertung der Diagnoseart am Ende dieses Kapitels herangezogen.

2.1 Anforderungen an ein Diagnoseverfahren

Anforderungen an ein Diagnoseexpertensystem lassen sich in folgende Teilanforderungsbereiche aufteilen:

- Eingabe
- Benutzerinteraktion
- Wissensakquisition
- Wissensbasis
- Diagnostik
- Temporale Aspekte

Anforderungen an die Wissensakquisition und die Wissensbasis sind für das eigentliche Diagnoseverfahren relativ irrelevant (abgesehen von einer Steigerung der Effizienz des Verfahrens durch geschickte Organisation der Wissensbasis). Deshalb werden sie in der folgenden Diskussion nicht weiter behandelt.

Anforderungen an die Eingabe:

- *Plausibilitätskontrolle*
  Fehlerhafte Eingaben durch den Benutzer oder über eine Prozeßankopplung sollten vom System erkannt und zurückgewiesen werden. Dies sollte dem System dann möglich sein, wenn die Eingabe „den Rahmen des Erwarteten sprengt", oder wenn sie inkonsistent zu anderen dem System bekannten Daten ist.

- *Rücknahmemöglichkeit fehlerhafter Eingaben*
  Wurde eine fehlerhafte Eingabe oder ein fehlerhafter Schluß erst im Nachhinein erkannt, so sollte eine Möglichkeit bestehen, die fehlerhaften Daten und alle Daten, deren Gültigkeit von diesen abhängen, zurückzunehmen. Diese Rücknahme sollte nicht zu einer gänzlichen Wiederholung des Inferenzprozesses führen.

Benutzerinteraktion:

- *Benutzer kann ständig Hypothesen einbringen*
  Diese Möglichkeit kann das Diagnoseverfahren erheblich beschleunigen, da ein fachkundiger Benutzer bei einer Störung häufig einen konkreten Verdacht hat, den das System erst wesentlich später generieren würde. Darüber hinaus erhöht die Interaktionsmöglichkeit durch den Benutzer die Akzeptanz des Expertensystems.

Diagnostik:

- *Ermittlung der korrekten Diagnose*

- *Effiziente Diagnosefindung*

- *Nachvollziehbare Diagnosefindung*
  Wenn die Diagnose schon nicht „beweisbar" ist, so sollte sie zumindest vom Benutzer nachvollziehbar sein, d. h. sie sollte ihm „intuitiv korrekt" erscheinen.

- *Diagnosevorschlag auch bei unvollständigem und unsicherem Wissen*
  In den meisten Fällen ist das zu diagnostizierende Gerät zu komplex, um alle es betreffenden Aspekte repräsentieren zu können. Ferner ist die fallspezifische Information in der Regel unvollständig und kann — wenn überhaupt — nur unter großem Aufwand vervollständigt werden. Gerade unter diesen Bedingungen sollte ein Expertensystem Rat geben können.

- *Erkennen von Mehrfachdiagnosen*
  Das Diagnoseverfahren sollte nicht so erstellt werden, daß es genau eine Diagnose sucht, sondern es sollte auch den Fall behandeln können, daß mehrere Defekte gleichzeitig vorliegen.

- *Notfalldiagnostik*
  Diese aus der Medizin stammende Form der Diagnose ist in zeitkritischen Situationen

anzuwenden. Deuten Symptome auf eine Diagnose hin, die sofortige Abhilfemaßnahmen erfordert, so wird (auch wenn es mehr Evidenz für eine nicht zeitkritische Diagnose gibt) diesem Verdacht sofort nachgegangen. Das Ergebnis der Notfalldiagnostik ist weniger fundiert, als das der normalen Diagnose, da der Diagnosevorgang stark verkürzt wird. Die Dringlichkeit der Situation rechtfertigt jedoch diese Form der Diagnose.

Temporale Aspekte:

- *Repräsentation von Zeitangaben*
  In manchen Anwendungsdomänen ist es für die Fehlerdiagnose unerläßlich, die Zeitpunkte des Auftretens von Symptomen zu kennen. Mittels dieser Zeitpunkte lassen sich die Symptome in eine zeitliche Reihenfolge bringen, und es kann die Zeitspanne zwischen dem Auftreten zweier Symptome berechnet werden.

- *Darstellung zeitlicher Verläufe*
  Vielfach ist zur Fehlerfindung der zeitliche Verlauf eines einzelnen Symptoms, bzw. der zeitliche Verlauf mehrerer Symptome relativ zueinander betrachtet, relevant. Auch hierfür sollten geeignete Mechanismen zur Verfügung stehen.

- *Auswertung von Folgesitzungen*
  Konnten die Fehler nicht durch *eine* Konsultation des Diagnosesystems behoben werden, so müssen weitere „Sitzungen" folgen. Dies erfordert eine Umrechnung von Zeitpunkten, die Speicherung von Historien von Merkmalen und die Rücknahme von Schlußfolgerungen.

2.2 Die verschiedenen Diagnosesystemklassen

Auf oberster Ebene läßt sich die Diagnose als eine Kombination von *Klassifikation* und *Testauswahl* auffassen.
Unter Klassifikation ist die Zuordnung einer Situation zu einem Fehler zu verstehen. Reicht die bisher vorhandene Information nicht aus, um diese Zuordnung durchzuführen, so wird über die Testauswahlkomponente ein möglichst kostengünstiger Test zur Erhebung weiterer Symptome vorgeschlagen, der zugleich einen möglichst großen Informationsgewinn verspricht. Nach Ausführung der zugehörigen Untersuchung wird anhand der neuen Symptome erneut versucht, eine Klassifikation durchzuführen, usw. (siehe Abbildung 2.1).
Spaltet man die Klassifikation in zwei Phasen auf, so erhält man eine weitere gängige Beschreibung des Diagnosevorgangs. Die erste Phase bildet die *Hypothesengenerierung*. Ziel dieser Phase ist die Erstellung einer Menge in Betracht kommender (verdächtiger) Fehlerursachen. Als zweite Phase schließt sich die Phase der *Hypothesenüberprüfung* an, in der diejenigen verdächtigten Fehlerursachen ausgesondert werden, die einer Überprüfung nicht standhalten. Schließlich folgt die Phase der *Hypothesendiskriminierung* (bzw. der Testauswahl), in der nach Mitteln gesucht wird, wie zwischen den übriggebliebenen Hypothesen weiter selektiert werden kann (zum Beispiel durch Erhebung eines Symptoms, dessen Wert bei möglichst vielen der verbliebenen Hypothesen verschieden ausfällt). Diese drei Phasen sind in Abbildung 2.2 dargestellt.

Für die Realisierung von Klassifikation und Testauswahl, bzw. Hypothesengenerierung, Hypothesenüberprüfung und Hypothesendiskriminierung, ergeben sich vielfältige Möglichkeiten. Diese verschiedenen Möglichkeiten führen zu verschiedenen Diagnosesystemklassen. Die Klassen unterscheiden sich hauptsächlich in der Art des verwendeten Wissens, das benutzt wird, um eine Diagnose zu stellen. Da dieses Wissen das Diagnoseverfahren bis zu einem gewissen Grade determiniert, erscheint eine kurze Darstellung der verschiedenen Diagnosesystemklassen sinnvoll. Für genauere Informationen über bei der Diagnose verwendetes Wissen sei auf [Bec92] verwiesen.

Folgende Diagnosesystemklassen werden im weiteren Verlauf des Kapitels vorgestellt:

- Statistische Systeme
- Heuristische Systeme
- Modellbasierte Systeme
  - Funktionale Systeme
  - Pathophysiologische Systeme
- Fallvergleichende Systeme
- Sichere Systeme
  - Entscheidungsbäume
  - Entscheidungstabellen

Natürlich gibt es auch zahlreiche Möglichkeiten, die Diagnosesystemklassen zu kombinieren und so Hybride zu schaffen. Dies geschieht mit der Intention, die Vorteile der einzelnen Klassen zu nutzen und die Nachteile zu vermeiden, d. h. an der Stelle, wo eine Klasse Nachteile hat, wird eine andere Klasse eingesetzt, die dort keine Nachteile oder sogar Vorteile aufweist. Einige Hybride werden daher in Kapitel 3.2 anhand konkreter Realisierungen vorgestellt.

Da die Art des Diagnoseverfahrens innerhalb einer Diagnosesystemklasse festliegt, wird zu jeder Klasse ein sogenannter Basisalgorithmus für die Diagnose angegeben. An diesem Algorithmus werden sich die später vorgestellten konkreten Diagnoseverfahren ausrichten, d. h. die Unterschiede innerhalb der Diagnoseverfahren einer Klasse manifestieren sich in Verfeinerungen und Erweiterungen des durch den Basisalgorithmus repräsentierten Verfahrens.

2.2.1 Heuristische Diagnose

Basis der heuristischen Diagnosesysteme sind Assoziationen zwischen Symptomen und Fehlerursachen (Diagnosen), die dem Erfahrungsschatz von Experten der Anwendungsdomäne entnommen sind[1]. Bei diesen Assoziationen handelt es sich in der Regel um *unsichere* Beziehungen. Die Unsicherheit ist auf eine unvollständige Informationslage oder auf unverstandene Zusammenhänge (innerhalb der Anwendungsdomäme) zurückzuführen. Ihr wird dadurch

[1] In der Literatur finden sich weitere Bezeichnungen für heuristische Diagnosesysteme: *Assoziative* bzw. *symptomorientierte* Diagnosesysteme.

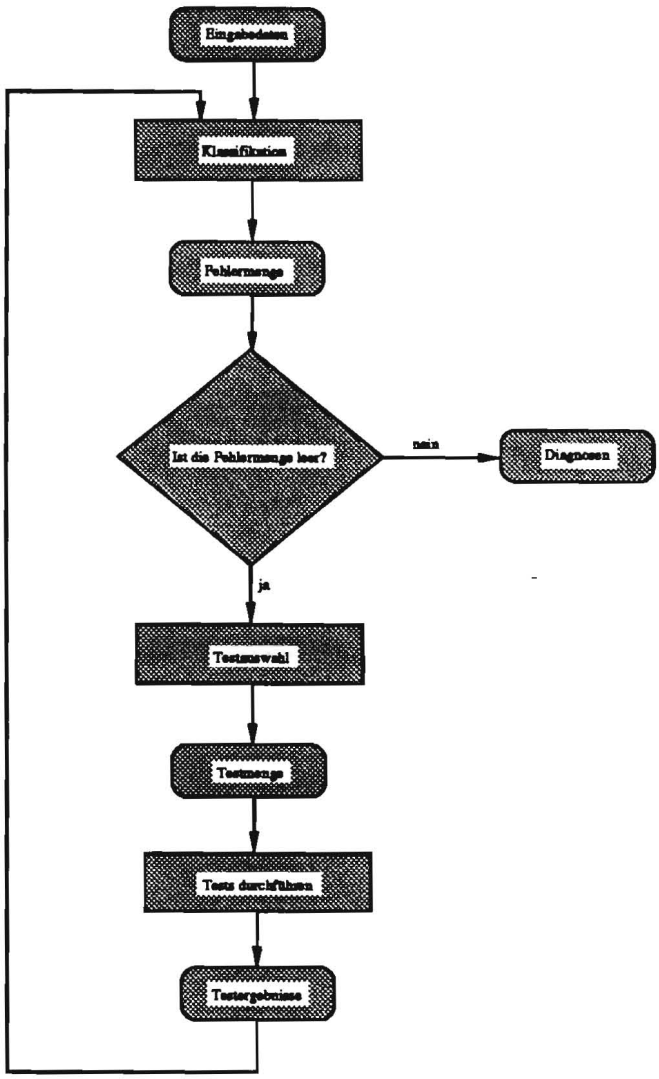

Abbildung 2.1: Klassifikation und Testauswahl

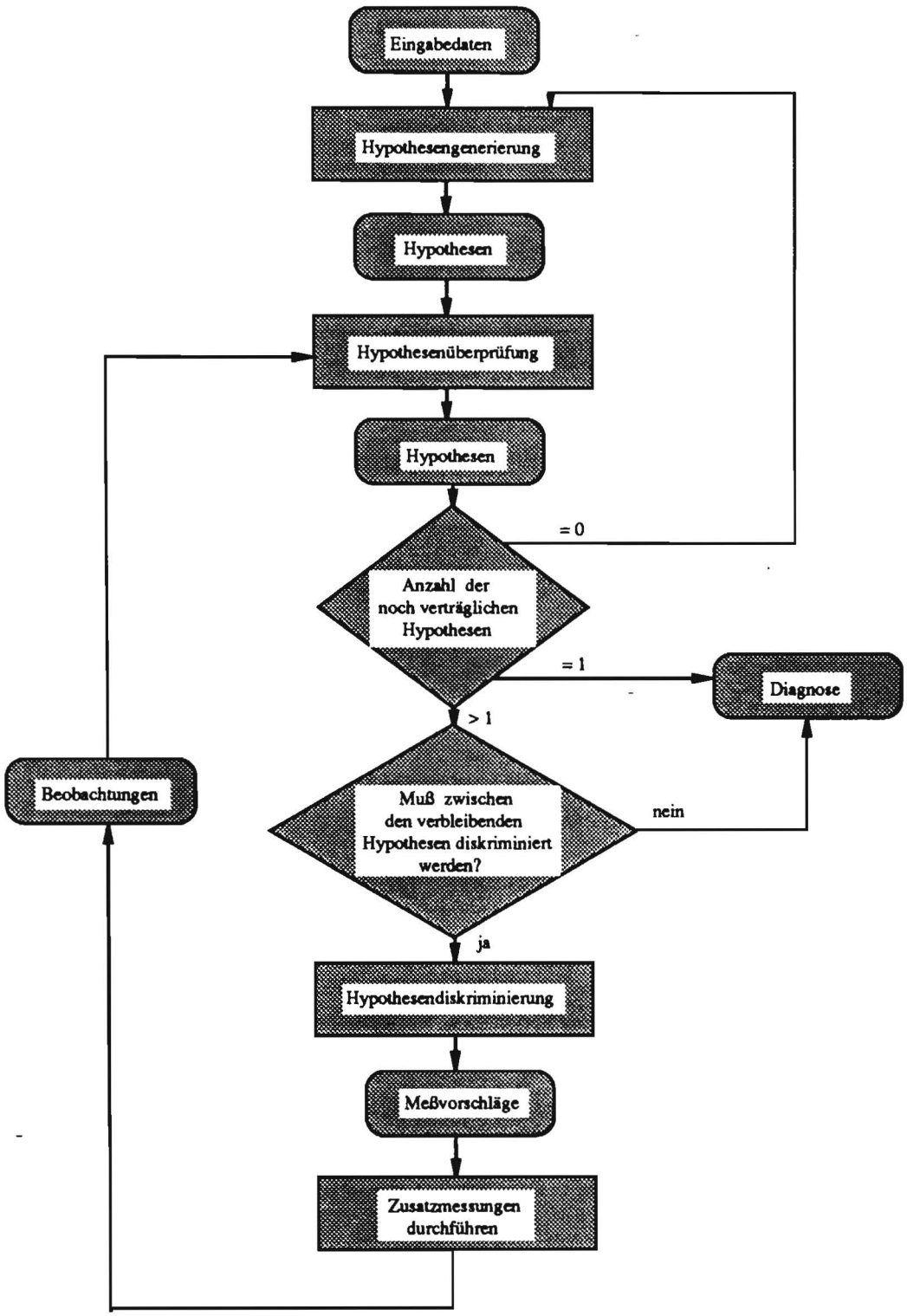

Abbildung 2.2: Hypothesengenerierung, Hypothesenüberprüfung, Hypothesendiskriminierung

Rechnung getragen, daß die Assoziationen (die meist in Form von Regeln vorliegen) mit „Sicherheitsfaktoren" (certainty factors) versehen werden[2]. Die Sicherheitsfaktoren repräsentieren „Wahrscheinlichkeiten[3]", die angeben, wie sicher die zugehörige Assoziation ist. Diese Wahrscheinlichkeiten müssen von entsprechenden Domänenexperten erfragt werden. Die Experten schätzen die Größe der Wahrscheinlichkeit auf der Basis ihrer Erfahrungen.
Das Vorhandensein von Wahrscheinlichkeiten erfordert einen Mechanismus zu ihrer Verknüpfung, ein sogenanntes *heuristisches Verrechnungsschema.* Weisen beispielsweise zwei verschiedene Symptome auf dieselbe Diagnose hin, so müssen die jeweiligen Teilevidenzen[4] zu einer Gesamtevidenz verknüpft werden. Verrechnungsschemata werden in Kapitel 3.1 vorgestellt.

Spricht man von Symptom–Diagnose Assoziationen, so ist nicht notwendigerweise gemeint, daß in einem Schritt von Symptomen auf Enddiagnosen geschlossen wird. Vielfach wird schrittweise von Merkmalen zu Lösungen hin abstrahiert, das heißt es liegen Symptom–Zwischendiagnose–, Zwischendiagnose–Zwischendiagnose– und Zwischendiagnose–Enddiagnose–Assoziationen vor. Die somit nahegelegte schrittweise Verfeinerungsstrategie wird durch die für heuristische Diagnosesysteme typische Repräsentation des *diagnostischen Mittelbaus* eingebaut (siehe Abbildung 2.3). Unter diesem Begriff versteht man sämtliche Zwischenergebnisse, die von einem Diagnoseverfahren auf dem Weg von den Symptomen zur Enddiagnose ermittelt werden. Zwischenergebnisse können gleichzeitig Diagnosen eines Diagnoseproblems und Symptome eines nachgeschalteten Problems sein.
Desweiteren verwendet ein heuristisches Diagnoseverfahren eine der in Kapitel 3.1 erläuterten *Kontrollstrategien.* Die Kontrollstrategie steuert die Anwendung der repräsentierten Assoziationen.

Es erfolgt die Angabe eines Algorithmus, der die prinzipielle Arbeitsweise eines heuristischen Diagnosesystems umreißen soll. Der Algorithmus muß sehr allgemein abgefaßt sein, da die das Diagnoseverfahren bestimmende Kontrollstrategie hier nicht spezifiziert werden soll[5].

Vom Benutzer werden Merkmale eingegeben, die den Problembereich abstecken und unter denen das System wählen kann, welche Merkmalsausprägungen es anfordert. Ausgegeben werden Lösungen mit Empfehlungen.
Benutzt werden Regeln der Art $Symptom_1$ & ... & $Symptom_n \longrightarrow Diagnose_1$ & ... & $Diagnose_n$, eine Agenda, auf die zu erfragende Merkmale geschrieben werden (TESTAGENDA), eine Agenda, auf der potentielle Lösungen vermerkt werden (LÖSUNGSAGENDA) und eine ERGEBNISLISTE, in die etablierte Lösungen aufgenommen werden. Außerdem ist jedem Merkmal, das erfragt werden kann, ein STATUS-Flag zugeordnet, das angibt, ob das Merkmal bisher schon erfragt wurde oder nicht.

---

[2]Neben unsicherem Wissen kann in heuristischen Systemen auch *unscharfes* Wissen repräsentiert sein. Unscharfes Wissen wird bei der Beschreibung des Diagnosesystems Z–11 in Kapitel 3.2.4 vorgestellt.

[3]Mit dem Begriff Wahrscheinlichkeit wird in dieser Arbeit nicht nur die mathematische Definition der Wahrscheinlichkeit gemeint, sondern allgemein Evidenz, die für oder gegen die Anwesenheit eines Objektes spricht.

[4]Mit „Evidenz" ist der Beitrag eines Symptoms oder einer Zwischendiagnose zu einer Gesamtwahrscheinlichkeit gemeint.

[5]Eine Kontrollstrategie würde insbesondere den Programmschritt 2.5 des nachfolgenden Algorithmus formen.

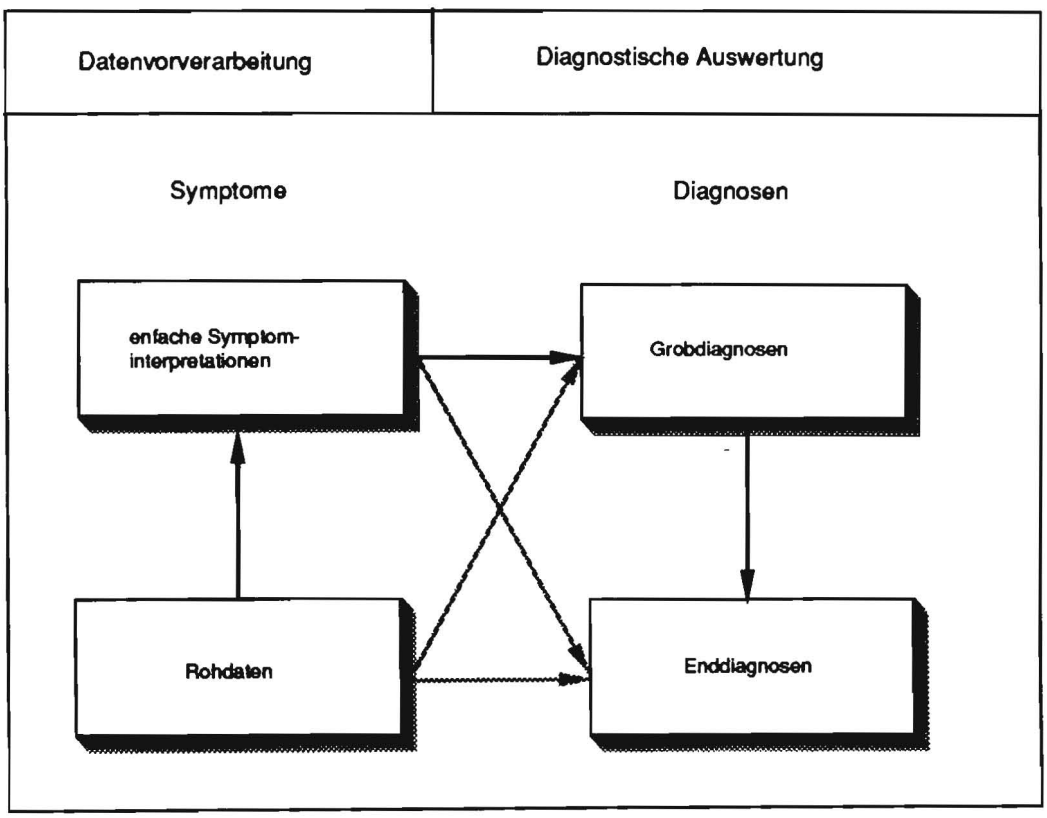

Abbildung 2.3: Diagnostischer Mittelbau

1. Initialisierung:
   TESTAGENDA := Gewichtete Merkmale, die zu Beginn einer Diagnosesitzung erfragt werden sollen.
2. Bearbeitung der TESTAGENDA:
   WIEDERHOLE BIS TESTAGENDA = LEER:

   2.1 TEST := Das maßgeblich einer Bewertungsfunktion bestbewertete Merkmal von TESTAGENDA mit STATUS(Merkmal) = ,,nicht erfaßt''.
   2.2 TESTAGENDA := TESTAGENDA \ TEST.
   2.3 Symptomerfassung: Erfasse die Merkmalsausprägung von TEST.
   2.4 STATUS(TEST) := ,,erfaßt''.
   2.5 WIEDERHOLE BIS keine Regeln mehr anwendbar:

      2.5.1 Führe die Regeln aus, deren Vorbedingungen erfüllt sind.
      2.5.2 Setze Merkmale, deren STATUS auf ,,nicht erfaßt'' steht und deren Erfassung ,,sinnvoll'' erscheint, auf die TESTAGENDA.
      2.5.3 Wurde Evidenz für Diagnosen hergeleitet, so setze die Diagnosen auf die LÖSUNGSAGENDA (bzw. erhöhe oder erniedrige deren bisherige Evidenz).

   2.6 FÜR ALLE Diagnosen D der LÖSUNGSAGENDA:

      2.6.1 FALLS D hinreichend sicher (Vergleich mit Schwellwert), so übertrage D auf die ERGEBNISLISTE.
      SONST, FALLS D hinreichend verdächtig (Vergleich mit Schwellwert), übertrage Merkmale mit STATUS(Merkmal) = ,,nicht erfaßt'', die zur Etablierung von D benötigt werden, auf die TESTAGENDA.
      SONST streiche D von der LÖSUNGSAGENDA.

3 Ausgabe der Diagnosen von ERGEBNISLISTE mit ihren assoziierten Empfehlungen.

2.2.2 Modellbasierte Diagnose

Während man bei heuristischen Systemen reine Symptom–Diagnose Assoziationen verwendet, ohne explizit Wissen über das Diagnoseobjekt einzubeziehen, stützt man sich bei der modellbasierten Diagnose auf ein Modell des zu diagnostizierenden Objektes[6]. Das bedeutet, man verwendet Kausalitäten (Ursache–Wirkung Beziehungen), um Vorgänge bzgl. des Diagnoseobjektes zu beschreiben. Aus diesem Grunde wird die modellbasierte Diagnose auch *kausale Diagnose* genannt.
Ein weiterer wichtiger Unterschied zur heuristischen Diagnose besteht in der Art der Diagnosebewertung. Heuristische Systeme entscheiden sich für eine Diagnose, indem sie die akkumulierten Evidenzen der zur Disposition stehenden Diagnosen vergleichen. Modellbasierte Systeme

[6]Modellwissen wird auch *tiefes Wissen* genannt, während das durch Assoziationen repräsentierte Wissen als *flaches Wissen* bezeichnet wird.

dagegen akzeptieren eine Diagnose dann, wenn möglichst genau die beim Diagnoseobjekt aufgetretenen Symptome aus ihr abgeleitet werden können.

Das Diagnoseverfahren wird durch die Art der Modellierung des zu diagnostizierenden Gerätes geprägt. Aus diesem Grunde soll kurz auf die verschiedenen Modellierungskriterien der modellbasierten Diagnose eingegangen werden.

Zum einen kann man zwischen *quantitativer* und *qualitativer* Modellierung unterscheiden. Häufig kann die Funktionsweise technischer Geräte mathematisch korrekt mittels Differentialgleichungen beschrieben werden. Diese sind jedoch möglicherweise nur unter großen Anstrengungen (Aufwand) lösbar. Vielfach können lediglich Näherungslösungen mit Verfahren der Numerik erreicht werden. Die Genauigkeit dieser Lösungen kann nur unter stark steigendem Aufwand verbessert werden. Eine quantitative Beschreibung mittels Differentialgleichungen kann somit einen erheblichen Berechnungsaufwand verursachen, ohne daß exakte Ausgaben erzeugt werden.
Durch die qualitative Modellierung wird von exakten Daten abstrahiert. Jeder qualitative Wert steht für eine Menge quantitativer Daten. Dadurch wird der Berechnungsaufwand im Modell wesentlich verringert und die Arbeitsweise des Gerätes dem Benutzer plausibler gemacht. Es muß natürlich darauf geachtet werden, daß das anhand der abstrahierten Daten erzeugte Verhalten des qualitativen Modells der Funktionsweise der zu diagnostizierenden Geräte möglichst genau entspricht.
Da das qualitative Modell in vielen Fällen ausreicht, sollte es in diesen Fällen aufgrund seiner Vorteile dem quantitativen Modell vorgezogen werden.

Modellierungen können *flach* oder *hierarchisch* sein. Bei hierarchischer Modellierung wird das Gerät auf verschiedenen Ebenen modelliert, wobei von einer tieferen Ebene zu einer höheren Ebene hin immer mehr von Details abstrahiert wird.
Da das Gerät damit mehrfach modelliert ist (ein Mal auf jeder Ebene) , ist der Modellierungsaufwand um einiges höher, als bei flacher Modellierung. Dafür wird die Diagnose bei hierarchischer Modellierung flexibler und der Diagnoseaufwand kann erheblich sinken (nämlich dann, wenn es ausreicht, das Modell auf einer höheren Abstraktionsebene zu betrachten).

Ferner kann man zwischen *statischen* und *dynamischen* Modellierungen unterscheiden.
Zur Erklärung dieser Begriffe werden die Definitionen aus [Reh91] übernommen.

Definition 2.1 (statisches Verhalten) *Ist zur Modellierung eines Verhaltens keine explizite Betrachtung der Zeit notwendig, so nennen wir es* statisch. *Für derartiges Verhalten läßt sich der Ablauf der Zeit implizit über die Reihenfolge der Ursache–Wirkung–Schritte darstellen. Anders formuliert ist statisches Verhalten dadurch gekennzeichnet, daß für gegebene Eingaben eine Situation entsteht, die Bestand hat und sich nicht mehr ändert, so daß eine Momentaufnahme (Snapshot) ausreicht, um die Auswirkungen der Eingaben zu beschreiben.*

Definition 2.2 (dynamisches Verhalten) *Wir nennen ein Verhalten* dynamisch, *wenn zu seiner Modellierung eine explizite Betrachtung der Zeit notwendig ist, oder einfacher, wenn es nicht statisch ist. Dies ist zum Beispiel immer dann der Fall, wenn sich im Verlauf des modellierten Vorgangs die Werte interessierender Meßpunkte mehr als einmal ändern.*

Die Modellierung dynamischen Verhaltens und die Nutzung des entstandenen dynamischen Modells ist wesentlich aufwendiger, als die statische Modellierung, da die Repräsentation der Zeitdimension und die Repräsentation und Anwendung entsprechender Behandlungsmethoden erforderlich wird. Beispiele dynamisch modellierter Dianosesysteme werden in 3.2 vorgestellt. Deshalb sollte — wenn möglich — eine statisch modellierbare Vereinfachung einer zu untersuchenden Maschine gewählt werden.

Schließlich kann eine Maschine *komponenten-* oder *prozeßorientiert* modelliert werden.
Bei der prozeßorientierten Modellierung werden die in der Maschine ablaufenden Prozesse beschrieben. Die Prozesse treten über gemeinsame Parameter in Verbindung. Modelliert man dagegen komponentenorientiert, so werden die einzelnen Bauteile einer Maschine, sowie deren Verhalten repräsentiert. Bauteile sind über Schnittstellen (Ports) miteinander verbunden.
Da das Ziel der technischen Diagnostik in der Regel die Lokalisierung eines defekten Bauteils ist, eignet sich die komponentenorientierte Modellierung hier besser, als die prozeßorientierte Modellierung.

Zusammenfassend seien noch einmal die diskutierten Modellierungskriterien aufgelistet:

quantitativ	—	qualitativ
hierarchisch	—	flach
dynamisch	—	statisch
prozeßorientiert	—	komponentenorientiert

Es wurden die Möglichkeiten aufgezählt, *wie* modelliert werden kann. Eine noch gravierendere Unterscheidung modellbasierter Diagnosesysteme ergibt sich bei der Frage, *was* modelliert werden soll. Grundsätzlich unterscheidet man zwei Arten modellbasierter Diagnosesysteme: *Pathophysiologische*[7] und *funktionale Systeme.*
Pathophysiologische Systeme basieren auf einem Modell, welches das Fehlverhalten eines Systems darstellt; funktionale Systeme hingegen arbeiten mit einem Modell des intakten Systems.

Pathophysiologische Diagnose

Ein Modell eines fehlerhaften Systems kann man sich im einfachsten Fall als eine Menge von Relationen (Kausalitäten) der Art „Ursache erzeugt Wirkung" vorstellen. Am Anfang einer mit dieser Menge aufgebauten Ursache–Wirkung–Kette stehen Diagnosen und am Ende Symptome. Grob beschrieben versuchen pathophysiologische Systeme alle (oder möglichst viele) der aufgetretenen Symptome durch Ursachen zu „überdecken"[8]. Eine vollständige Erklärung für die aufgetretenen Symptome ist gefunden, wenn die verdächtige Ursache alle diese Symptome im Modell herleiten kann.
Auf den ersten Blick scheint die pathophysiologische Diagnose Ähnlichkeit mit der heuristischen Diagnose zu haben; es bestehen jedoch gravierende Unterschiede: Während man bei der

[7] Der Name dieser Diagnoseart stammt aus der Medizin und wurde von Frank Puppe in [Pup88] vorgeschlagen.
[8] Aus diesem Grunde findet man in der Literatur auch die Bezeichnung *überdeckende Diagnose.* Eine ebenfalls gebräuchliche Bezeichnung ist *Klassifikation mit Fehlermodellen.*

heuristischen Diagnose i. allg. von Symptomen auf Ursachen schließt, erfolgt die Schlußweise bei pathophysiologischen Systemen genau umgekehrt. Ausgehend von einer möglichen Ursache wird über Ursache–Wirkung Kausalitäten versucht, die aufgetretenen Symptome herzuleiten. Beispiel für eine Kausalität: *Eine gebrochene Zuleitung bewirkt einen Druckabfall in der von ihr gespeisten Kammer.*
Im Normalfall ist eine Kausalität im Modell („physikalisch") begründet, und deshalb eine sichere Beziehung, während eine Symptom–Diagnose Assoziation mit Unsicherheit belastet ist (die Quantifizierung dieser Unsicherheit ist wiederum mit Unsicherheit belastet, da ihre Größe in der Regel nur geschätzt ist).
Da die Eingabe für ein Diagnosesystem naturgemäß aus Symptomen besteht, kann bei der heuristischen Diagnose sofort mit der Auswertung der Regeln begonnen werden. Bei der pathophysiologischen Diagnose hingegen ist der Ausgangspunkt für die Auswertung der Regeln (Kausalitäten) eine verdächtige Fehlerursache. Somit ist die Verwendung von zusätzlichem Wissen zur Verdachtsgenerierung erforderlich, um Ausgangspunkte für die Verwendung der Regeln zu schaffen. Auf derartige Techniken zur Verdachtsgenerierung wird in Kapitel 3.1 eingegangen.
Ferner müssen bei pathophysiologischen Diagnosesystemen in manchen Fällen Techniken zur Behandlung von Rückkopplungsschleifen[9]. Rückkopplungsschleifen treten auf, wenn das Modell Zykel aufweist.

Bspl.: *Ein XOR-Gatter habe die Eingänge E1, E2 und den Ausgang A. Der Ausgang sei mit dem Eingang E2 verbunden. Ist der Eingang E1 mit einer logischen Eins belegt, so oszilliert der Ausgang A.*

Für den folgenden Basisalgorithmus sei der Ablauf von Vorgängen im Modell über Zustände beschrieben. Jedem Zustand sind Bedeutungsregeln zugeordnet, über deren Auswertung Folgezustände erzeugt werden. Eine Hypothese besteht aus einer Menge von Anfangszuständen. Der Basisalgorithmus besteht aus Gründen der Übersichtlichkeit aus drei ineinander geschachtelten Komponenten: Einer Simulationskomponente, einer Konsistenztestkomponente und einer Diagnosekomponente.
Die Diagnosekomponente ruft die Konsistenztestkomponente auf, die ihrerseits die Simulationskomponente aufruft. Die Simulationskomponente ermittelt anhand eines Fehlermodells und eingegebener Hypothesen die zu den Hypothesen gehörenden Symptome. Die Konsistenztestkomponente überprüft, ob die simulierten Symptome mit den tatsächlich aufgetretenen übereinstimmen. Die Diagnosekomponente generiert Hypothesen, verifiziert diese über die Konsistenztestkomponente, und wählt die bestbewertete Hypothese aus.

Diagnosekomponente[10]

```
Eingabe:  Endzustände mit Parameterwerten.
Ausgabe:  Anfangszustände mit Parameterwerten.

  1. Hypothesengenerierung.
```

[9] Man spricht von Rückkopplungsschleifen, wenn die Änderung eines Zustandes ggf. über Zwischenzustände den Zustand selbst beeinflußt. Das kann zu Zustandsverstärkung, -abschwächung und Oszillationen führen.

[10] Die Schritte „Verdachtsgenerierung" und „tialdiagnostik" sind nicht weiter erläutert, da Möglichkeiten zu ihrer Realisierung in Kapitel 3.1 vorgestellt werden.

2. Hypothesenüberprüfung.
   Führe für jede in Schritt 1 generierte Menge von Anfangszuständen mit Parameterwerten einen Konsistenztest aus (Aufruf der Konsistenztestkomponente): ERG := ERG ∪ KONSISTENZ(Anfangszustände,Endzustände).
3. Differentialdiagnostik: Wähle aus ERG die Menge von Anfangszuständen aus, die das maßgeblich einer Bewertungsfunktion bestbewertete Modell liefert.

Konsistenztestkomponente

Eingabe: Anfangs- und Endzustände mit Parameterwerten.
Ausgabe: Aussage über die Konsistenz der Eingabe.

1. Führe mit den gegebenen Anfangszuständen eine Simulation aus (Aufruf der Simulationskomponente): ERG := SIMULATION(Anfangszustände).
2. FALLS ERG mit den Endzuständen vergleichbar ist,
   DANN Ausgabe der Anfangszustände.
   SONST Ausgabe der leeren Menge.

Simulationskomponente

Eingabe: Anfangszustände mit Parameterwerten.
Ausgabe: Endzustände mit Parameterwerten.

1. Setze Anfangszustände auf AGENDA.
2. WIEDERHOLE BIS kein Zustand mehr in AGENDA ist, der Folgezustände hat:
   2.1 Wähle einen Zustand aus AGENDA, der Folgezustände hat und dessen Komponenten möglichst vollständig berechnet sind, berechne seine Parameterwerte aus den Komponenten, lösche ihn aus AGENDA, werte seine Regeln aus "Bedeutung" aus, und setze die Folgezustände auf AGENDA.
3. Ausgabe der Endzustände(aktuelle Belegung von AGENDA).

Funktionale Diagnose

Während pathophysiologische Diagnosesysteme sich an einem Modell des Fehlverhaltens des zu diagnostizierenden Objektes orientieren, basieren funktionale Systeme auf Modellen des intakten Diagnoseobjektes[11] . Ein derartiges Modell besteht aus einer Beschreibung der Komponenten des Objektes und ihrer Verbindungen untereinander, sowie aus einer Beschreibung der Funktionsweise der einzelnen Komponenten. Komponenten werden durch ihre Attribute beschrieben; Verbindungen zwischen zwei Komponenten können durch Verwendung gleicher Schnittstellen-Variablen repräsentiert werden. Die Funktion einer Komponente wird durch Beziehungen zwischen Eingangs- und Ausgangsvariablen dargestellt. Diese Beziehungen können

[11][Pup88] spricht auch von *physiologischen* Diagnosesystemen.

Ursache–Wirkung–Regeln sein (z.B.: *Zwei logische Einsen, die an den Eingängen eines UND-Gatters anliegen, bewirken eine logische Eins am Ausgang des Gatters)* oder Differentialgleichungen (z.B.: $f'(x) = 5 * f(x)$). Schließlich gehören zur Beschreibung der korrekten Funktionsweise eines Diagnoseobjektes noch Randbedingungen. Diese werden durch Constraints abgebildet (z.B.: *Die Eingangsspannung ist kleiner 50 Volt*).
Die Probleme der pathophysiologischen Diagnose (Behandlung von Rückkopplungsschleifen, Verdachtsgenerierung) treten bei funktionalen Diagnosesystemen ebenfalls (sogar in verstärktem Maße) auf.

Die Idee eines funktionalen Diagnoseverfahrens ist die folgende: Wenn das beobachtete Verhalten eines technischen Systems (Gerätes) nicht dem bei gleichen Anfangsbedingungen mittels des Modells simulierten Verhalten entspricht, so muß das Gerät defekt sein. Gesucht wird nun ein Modell des fehlerhaften Gerätes, das heißt ein Modell, dessen Verhalten dem beobachteten Verhalten des defekten Gerätes entspricht. Hat man ein solches Modell gefunden, so hofft man, aus den Änderungen gegenüber dem ursprünglichen Modell auf die Fehlerursachen des defekten Gerätes schließen zu können (siehe auch Abbildung 2.4 aus [Reh91]).
Diese Diagnoseskizze wird nun in den folgenden Basisalgorithmus (der starke Ähnlichkeiten zu seinem Pendant bei der pathophysiologischen Diagnose aufweist) umgesetzt. Er besteht aus zwei ineinander geschachtelten Komponenten: Einer Simulationskomponente und einer sie aufrufenden Diagnosekomponente. Die in Form von Modelländerungen generierten Hypothesen werden über die Simulationskomponente geprüft. Die Simulationsergebnisse werden anschließend mit den in der Realität ermittelten Werten verglichen.

Diagnosekomponente[12]

```
Eingabe: Wertebelegungen von Eingangs- und Ausgangs-Variablen, globale
Systemparameter.
Ausgabe: Modelländerungen.

1. Entdeckung von Diskrepanzen:
   Simuliere mit den globalen Systemparametern und den Werten
   der Eingangs-Variablen die Werte der Ausgangs-Variablen
   unter der Annahme, daß alle Komponenten in Ordnung sind, und
   vergleiche sie mit den vorgegebenen Werten. Jede Differenz ist
   eine Diskrepanz. Dieser Schritt ist nicht nötig, wenn die Eingabe-
   daten bereits als Abweichungen vom Normalzustand angegeben sind.
2. Verdachtsgenerierung.
3. Verdachtsüberprüfung:
   Führe für jede Verdachtshypothese (Modelländerung) eine Simulation
   durch.
4. Differentialdiagnostik:
   Vergleiche bei jeder Simulation die vorhergesagten mit den beobachteten
   Zuständen und wähle die Hypothese aus, deren Simulationsergebnisse
   den beobachteten Zuständen am nächsten kommt.
```

[12]Die Schritte „Verdachtsgenerierung“ und „Differentialdiagnostik“ sind nicht weiter erläutert, da Möglichkeiten zu ihrer Realisierung in Kapitel 3.1 vorgestellt werden.

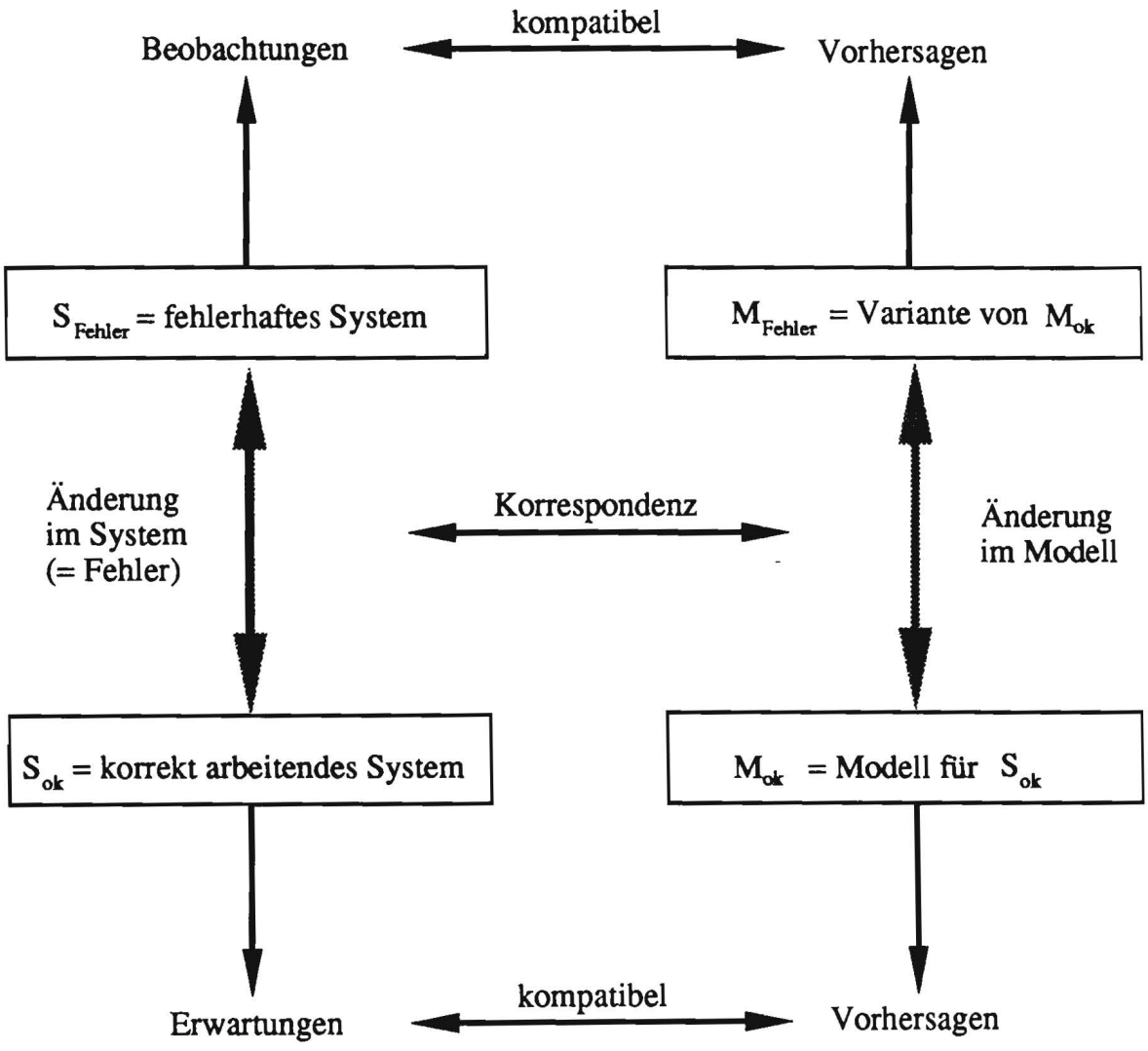

Abbildung 2.4: Funktionale Diagnosestrategie

Simulationskomponente

```
Eingabe:  Wertebelegung der Eingangs-Variablen, globale Systemparameter.
Ausgabe:  Wertebelegung der Ausgangs-Variablen.

1. Setze alle mit den Eingangs-Variablen verbundenen Komponenten auf
   eine globale Liste AGENDA.
2. WIEDERHOLE BIS AGENDA leer ist:
   2.1 Wähle eine Komponente aus AGENDA, die möglichst von keiner anderen
       Komponente von AGENDA abhängt.
   2.2 Berechne mit ihren Eingangs-Variablen (falls welche unbekannt
       sind, gehe davon aus, daß sie ,,normal'' sind) die
       Werte der Ausgangs-Variablen.
   2.3 Lösche die Komponente aus AGENDA.
   2.4 FALLS mit den Ausgangs-Variablen andere Komponenten verbunden sind,
       DANN setze diese Komponenten auf AGENDA,
       SONST setze die Ausgangs-Variablen auf ERGEBNISLISTE.
3. Ausgabe der Werte der Ausgangsvariablen von ERGEBNISLISTE.
```

2.2.3 Fallbasierte Diagnose

Im Unterschied zu anderen Diagnosesystemen repräsentieren fallbasierte Systeme kein abstraktes Wissen über die Anwendungsdomäne, sondern direkt die vorgefallenen Diagnosefälle. Entscheidungen über vorliegende Defekte eines Diagnoseobjektes werden basierend auf Präzedenzfällen getroffen, d. h. es wird die Fehlerursache desjenigen Falles übernommen (bzw. modifiziert übernommen), der die größte Ähnlichkeit zum aktuellen Eingabefall hat (natürlich muß die Ähnlichkeit hinreichend groß sein).
Beschränkt sich das System darauf, den ähnlichsten Fall herauszusuchen, so spricht man von einem *Case-Matching-Diagnosesystem.* Wird hingegen zusätzlich versucht, die Fehlerursache des ähnlichsten Falles für den aktuellen Fall anzupassen, so spricht man von einem *Case-Adaption-Diagnosesystem.*
Normalerweise schließt sich nach erfolgter Diagnose noch eine Lernphase an, um Gewichte von Fällen, die das System als „ähnlich" zum Eingabefall deklariert hat, die aber nicht die richtige Diagnose stellen, zu modifizieren. Außerdem kann natürlich der aktuelle Fall in die Wissensbasis übernommen werden. Auf mögliche Realisierungen dieser Lernphase wird innerhalb dieser Arbeit jedoch nicht eingegangen.

Ein Fall wird als eine Menge von Symptomen (Merkmale und ihre Ausprägungen) und Fehlerursachen repräsentiert. Um jedoch zwei Fälle miteinander vergleichen zu können, benötigt man zusätzliche Informationen über ihre Symptome, weil nicht alle Symptome eines Falles gleichbedeutend sind.
Nach einem Vorschlag von Frank Puppe könnte man jedem Merkmal eines Falles drei Attribute zuweisen:

- Das erste Attribut gibt das Default-Gewicht des Merkmals an.

- Das zweite Attribut gibt die fallspezifische Gewichtung des Merkmals an.
  Man kann sich leicht vorstellen, daß zum Beispiel, je nach zugehöriger Fehlerursache eines Falles, Merkmale, die genau auf diese Fehlerursache hindeuten, in diesem Fall höher gewichtet werden müssen, als es ihrer Bedeutung i. allg. entspricht.
  Beispiel: *Gegeben sei ein Expertensystem zur Motordiagnose. Normalerweise wird dem Merkmal Typ/Baujahr eine relativ geringe Bedeutung beigemessen werden. Ist jedoch bekannt, daß ein bestimmter Wagentyp während eines bestimmten Baujahres mit Rissen im Zylinder ausgeliefert wurde, so ist das Merkmal bei Fällen mit dieser Fehlerursache und der entsprechenden Merkmalsausprägung besonders hoch zu bewerten.*
  Ist also ein Wert für die fallspezifische Gewichtung angegeben, so überschreibt er die allgemeine Merkmalsgewichtung.

- Das dritte Attribut enthält eine Liste von Zahlen, die die Abnormalität von Merkmalsausprägungen darstellen. Je größer die Abnormalität ist, um so mehr Gewicht wird dem Merkmal beigemessen.

Neben der Gewichtung benötigt man für jedes Merkmal Ähnlichkeitsangaben, das heißt im einfachsten Fall ist pro Merkmal eine Formel angegeben, anhand derer die relative Ähnlichkeit zwischen zwei Merkmalen berechnet wird. Bei nominalen Merkmalstypen[13] dagegen müssen explizit Ähnlichkeitsangaben für Ausprägungspaare notiert werden.

Beispiel: *Die Farben „gelb" und „orange" ähneln sich in gewisser Weise, während die Farben „gelb" und „blau" völlig unähnlich sind. Diese Ähnlichkeitsangaben könnten dann folgendermaßen repräsentiert werden:*
(gelb, orange, 0.5)
(gelb, blau, 0.0)

Um bei einem Merkmalsvergleich einen absoluten Ähnlichkeitswert zu erhalten, müssen relative Ähnlichkeit und Gewichtung verrechnet werden.
Hat man sämtliche Einzelmerkmalsvergleiche, so benutzt man sogenannte *Distanz-* oder *Ähnlichkeitsmaße*, um die Einzelvergleiche zu einer Gesamtähnlichkeit der beiden Diagnosefälle zusammenzuziehen.
Bekannte Distanzmaße sind z.B. der *Abstand nach Euklid* und die *Manhattan-Distanz (City-Block-Metrik)*. Als bekanntes Ähnlichkeitsmaß sei der *Simple-Matching-Koeffizient* erwähnt. Eine Erläuterung gebräuchlicher Maße erfolgt in Kapitel 3.1.

Unter Zuhilfenahme all dieser zusätzlichen Informationen kann nun ein Basisalgorithmus formuliert werden:

```
Eingabe:  Merkmale eines neuen Falles.
Ausgabe:  (angepaßte) Lösung des relativ und absolut hinreichend
ähnlichsten Vergleichsfalles.

1. Lokalisierung ähnlicher Fälle:
   Extraktion derjenigen Fälle der Datenbank, die bestimmte
```

[13] Die Ausprägungen eines Merkmals mit nominalem Typ können nicht in eine wertende Reihenfolge gebracht werden.

```
   Bedingungen erfüllen. Das können z.B. Fälle sein, die die
   Merkmale des Eingabefalles in hochgewichteter Form besitzen.
2. Auswahl des besten Falles:
   Für jeden lokalisierten ähnlichen Fall aus 1. tue folgendes:
   Vergleiche die Merkmale und berechne die Gesamtähnlichkeit
   zum Eingabefall (z.B. unter Verwendung eines Änlichkeits-
   maßes).
   Wähle den Fall, der die größte Ähnlichkeit zum Eingabefall
   aufweist.
3. Anpassung der Lösung:
   Transfer der Lösung (Fehlerursache) des ausgewählten Falles auf
   den Eingabefall (falls nötig).
4. Feedback:
   Nach der Überprüfung der Lösung am defekten Objekt teilt der
   Benutzer die Ergebnisse dem System mit. Das Ergebnis wird vom
   System analysiert und in einer Lernphase integriert.
```

2.2.4 Statistische Diagnose

Statistische Diagnosesysteme basieren ähnlich wie heuristische Systeme auf Symptom–Diagnose–Beziehungen. Die Stärken dieser Beziehungen werden jedoch nicht geschätzt, sondern sie werden mittels statistischer Berechnungen aus Falldatenbanken extrahiert. Dem intuitiven Verrechnungsschema für Wahrscheinlichkeiten heuristischer Systeme steht eine *mathematische Evidenztheorie* auf statistischer Seite gegenüber. Bekannte Evidenztheorien sind das *Theorem von Bayes* (siehe [CM85]) und die *Dempster–Shafer–Theorie* (siehe [Sha76]).
Zunächst sei die statistische Diagnose unter Benutzung des Theorems von Bayes erläutert. Mittels des Theorems von Bayes können aus den Apriori–Wahrscheinlichkeiten $P(D_i)$ einer Menge von Diagnosen und den bedingten Wahrscheinlichkeiten $P(S_j/D_i)$ (Wahrscheinlichkeit, daß bei Zutreffen der Diagnose D_i das Symptom S_j vorliegt) die Wahrscheinlichkeiten der Diagnosen D_i unter Annahme der Symptome $S_1 \ldots S_m$ berechnet werden. Hierzu kommt folgende Formel (die eine Form des Theorems von Bayes ist) zur Anwendung:

$$P_r(D_i/S_1 \& \ldots \& S_m) = \frac{P(D_i) \cdot P(S_1/D_i) \cdot \ldots \cdot P(S_m/D_i)}{\sum_{j=1}^{n} P(D_j) \cdot P(S_1/D_j) \cdot \ldots \cdot P(S_m/D_j)}$$

P_r ist die relative Wahrscheinlichkeit einer Diagnose D_i im Vergleich zu allen anderen Diagnosen. Die Apriori–Wahrscheinlichkeiten der rechten Seite des Theorems müssen aus einer Falldatenbank berechnet werden.

Das Theorem von Bayes ist als Evidenztheorie natürlich im mathematischen Sinne korrekt, sofern gewisse Voraussetzungen erfüllt sind. Diese Voraussetzungen seien im folgenden angegeben:

- Statistische Unabhängigkeit der Symptome untereinander.
- Wechselseitiger Ausschluß von Lösungen (single–fault–assumption)

- Repräsentativität der zugrunde liegenden Fallsammlung.
- Vollständigkeit der Diagnosemenge.
- Konstanz der Wahrscheinlichkeiten (d. h. die Apriori–Wahrscheinlichkeiten sollten sich über die Zeit nicht ändern).

Ein Basisalgorithmus für die statistische Diagnose hat nach [Pup88] folgendes Aussehen:

Eingabe: Symptome.
Ausgabe: Wahrscheinlichste Diagnose.

1. Starte mit den Apriori-Wahrscheinlichkeiten aller Diagnosen.
2. Modifiziere für jedes Symptom die Wahrscheinlichkeit aller Diagnosen entsprechend den Symptom-Diagnose-Wahrscheinlichkeiten.
3. Selektiere die wahrscheinlichste Diagnose.

Legen wir das Theorem von Bayes zugrunde, so ergibt sich folgender Algorithmus:

Eingabe: Symptome $S_1 \ldots S_m$.
Ausgabe: Wahrscheinlichste Diagnose.

1. Die Wahrscheinlichkeit einer Diagnose $P(D_i)$ ergibt sich aus dem Quotienten

$$\frac{(Häufigkeit\ der\ Diagnose D_i)}{(Anzahl\ aller\ Fälle)}$$

   Die bedingte Wahrscheinlichkeit $P(S_j/D_i)$ berechnet sich aus dem Quotienten

$$\frac{(Häufigkeit\ des\ Zutreffens\ von\ Symptom\ S_j\ und\ Lösung\ D_i)}{(Häufigkeit\ der\ Lösung\ D_i)}$$

2. Berechne für jede Diagnose D_i die relative Wahrscheinlichkeit $P_r(D_i/S_1 \& \ldots \& S_m)$ gemäß obiger Bayes-Formel.
3. Selektiere die Diagnose mit höchster relativer Wahrscheinlichkeit.

Da die Bedeutung der Einzelsymptome bei der obigen kompakten Bayes–Formel nicht ersichtlich ist, wird auch eine andere Form der Berechnung gewählt, bei der Diagnosen schrittweise mit jedem neuen Symptom aktualisiert werden.
Auch hierzu sei ein Basisalgorithmus angegeben:

Eingabe: Symptome.
Ausgabe: Wahrscheinlichste Diagnose.

1: Setze die aktuelle Wahrscheinlichkeit jeder Diagnose auf ihren *Apriori–Wert*.

2: `Für jedes neue Symptom S: Multipliziere die aktuelle Wahrscheinlichkeit aller Diagnosen D mit einem` *Wahrscheinlichkeitsquotienten*.
3: `Selektiere die Diagnose mit der größten aktuellen Wahrscheinlichkeit.`

Um diesen Algorithmus zu realisieren, sollen zwei Möglichkeiten angegeben werden:
Die Verwendung von Apriori-Wahrscheinlichkeiten der Merkmale und die Verwendung der sogenannten Odds-Likelihood-Form.

- Der *Apriori-Wert* einer Diagnose ist ihre Apriori-Wahrscheinlichkeit und der *Wahrscheinlichkeitsquotient* das Verhältnis von $P(S/D)$ zur Apriori-Wahrscheinlichkeit des Symptoms $P(S)$. Kommt das Symptom S bei gegebener Diagnose D häufiger vor, als $P(S)$ angibt, so ist der Quotient größer 1 und erhöht die aktuelle Wahrscheinlichkeit der Diagnose D. Sind Symptom und Diagnose voneinander unabhängig, so ist der Quotient gleich 1 und die aktuelle Wahrscheinlichkeit der Diagnose bleibt bestehen.
  Wenn man am Schluß des Algorithmus die aktuellen Wahrscheinlichkeiten in relative Wahrscheinlichkeiten umrechnet, erhält man dieselben Ergebnisse, die auch mit dem Theorem von Bayes erhalten werden.

- Der *Apriori-Wert* einer Diagnose wird als ihre *Odds (O(D))* angegeben. O(D) ist als $P(D)/P(\neg D)$ definiert. Der *Wahrscheinlichkeitsquotient* ist das sogenannte *Wahrscheinlichkeitsverhältnis (likelihood-ratio)* $LR(S/D)$. Dieses ergibt sich durch den Quotienten $\frac{P(S/D)}{P(S/\neg D)}$. Ein Wert von 1 bei Odds und Wahrscheinlichkeitsverhältnissen drückt völlige Ungewißheit aus, da das Zutreffen eines Symptoms bzw. einer Diagnose genauso wahrscheinlich ist, wie ihr Nichtzutreffen. Liegen die Werte über 1, so sprechen sie für das Zutreffen, liegen sie unter 1, so sprechen sie gegen das Zutreffen.
  Auch diese Realisierung kann aus dem Theorem von Bayes hergeleitet werden.

Normalerweise wird mit dem Theorem von Bayes von Symptomen direkt auf Fehlerursachen geschlossen. Verwendet man dagegen sogenannte Bayes'sche Netze, so kann auch von Symptomen über Zwischenstufen auf Lösungen (Diagnosen) geschlossen werden. Die Grundlage für die Berechnung von Wahrscheinlichkeiten in Bayes'schen Netzen bildet die oben erwähnte Odds-Likelihood-Form. Dabei kann obiger Algorithmus iterativ für jede Hierarchiestufe angewendet werden. Die Lösungen einer Stufe bilden dann die Merkmale der folgenden Stufe. Da bisher von sicheren Symptomen ausgegangen wurde, muß die errechnete Unsicherheit der Zwischenlösungen in die Berechnung der aktuellen Wahrscheinlichkeit einbezogen werden. Das hat zur Folge, daß für jedes neu auftretende Merkmal die Wahrscheinlichkeitsänderungen durch das gesamte Netz (Heterarchie) propagiert werden müssen.
Für nähere Informationen sei auf [Pea85] und [Coo87] verwiesen. Erfolgreiches Beispiel für die Anwendung Bayesscher Netze ist das Expertensystem PROSPECTOR, das im Abschnitt 3.2 näher beschrieben wird.

Eine weitere bekannte mathematische Evidenztheorie, mittels der ein statistisches Diagnosesystem aufgebaut werden kann, ist die Dempster-Shafer-Theorie.
Der Hauptnachteil dieser Theorie liegt in der Art der Verrechnung der Wahrscheinlichkeiten begründet. Während bei der Verwendung des Theorems von Bayes Wahrscheinlichkeiten für

Einzeldiagnosen berechnet werden, werden bei der Dempster–Shafer–Theorie Wahrscheinlichkeiten von Diagnosemengen miteinander verknüpft. Die Wahrscheinlichkeit einer bestimmten Diagnosemenge muß aus der Verteilung der Wahrscheinlichkeiten über alle Diagnosemengen errechnet werden. Gibt es zu einem Anwendungsgebiet n Diagnosen, so wird theoretisch auf 2^n Diagnosemengen zugegriffen, was einen riesigen Rechenaufwand verursacht. Zur Komplexität der Dempster–Shafer–Theorie siehe auch [Orp90]. In [Bar91] wird ergänzend ausgeführt, daß die Theorie nur unter Einhaltung von gewissen Bedingungen praktikabel sei.
Ein weiterer Nachteil dieser Theorie besteht darin, daß Diagnose–Symptom–Wahrscheinlichkeiten als Apriori–Wahrscheinlichkeiten ($P(D/S)$) vorhanden sein müssen. In der Praxis sind Falldatenbanken jedoch zumeist nach Diagnosen geordnet und nicht nach Symptomen, was die Bildung dieser Wahrscheinlichkeiten erschwert (weil nach Symptomen gesucht werden muß, und jedes Symptom unsystematisch über die Falldatenbank verteilt ist).
Zu den Vorteilen dieser Theorie gehören die Möglichkeit, Diagnoseheterarchien zu repräsentieren und die Darstellung von Unsicherheitsintervallen (je mehr Information zu einer Diagnosemenge vorliegt, um so kleiner wird deren Intervall, in dem die Wahrscheinlichkeit für die Diagnosemenge zu finden ist). Diese Vorteile wiegen jedoch den genannten Hauptnachteil[14] nicht auf, so daß die Dempster–Shafer–Theorie bisher kaum verwendet und deshalb in dieser Arbeit nicht weiter berücksichtigt wird. Für nähere Informationen zu dieser Theorie sei auf [GS85] verwiesen.

2.2.5 Sichere Diagnose

Bei sicheren Diagnosesystemen handelt es sich um Systeme, die Wissen verarbeiten, das nicht mit Unsicherheiten behaftet dargestellt ist und die auch während der Diagnose nicht mit Unsicherheiten operieren [15]. Die *Erstellung* sicherer Diagnosesysteme kann dagegen unter wahrscheinlichkeitstheoretischen Betrachtungen erfolgen. Beispielsweise wird die Reihenfolge, in der Symptome erhoben werden, unter heuristischen Gesichtspunkten festgelegt. Ferner muß entschieden werden, wann eine Diagnose gestellt werden kann, und wann zunächst noch weitere Symptome erhoben werden sollen.

Das Wissen sicherer Diagnosesysteme stellt Assoziationen zwischen Symptomen und Fehlerursachen her.
Man unterscheidet hauptsächlich zwei Typen von sicheren Diagnosesystemen: *Entscheidungsbaum* und *Entscheidungstabelle.*

[14] Das in [Bar91] angegebene „Schlupfloch“ wird nicht einbezogen.

[15] Fallbasierte Systeme werden nicht als sichere Systeme bezeichnet, weil Unsicherheiten während der Diagnose über Merkmalgewichtungen und Ähnlichkeits– bzw. Distanzmaße eingeführt werden.
Quantitativ–modellbasierte Systeme können sicher sein, müssen es aber nicht. Der sichere Charakter hängt davon ab, wie exakt die verwendeten mathematischen Gleichungen das Verhalten des Diagnoseobjektes beschreiben, und ob Wahrscheinlichkeiten (Unsicherheiten) zugelassen sind.

Entscheidungsbäume

Bei Entscheidungsbäumen handelt es sich um Bäume, deren innere Knoten Fragen bzw. „Datenanforderungen" sind, die Äste Antwortalternativen darstellen und Blätter Diagnosen repräsentieren (Beispiel: Abbildung 2.5). Im einfachsten Fall kann man sich einen Entscheidungsbaum als eine große if–then–else–Anweisung vorstellen[16].
Der wichtigsten Unterschiede gegenüber den übrigen Diagnosesystemarten bestehen darin, daß Wissensbasis und Abarbeitungsstrategie nicht mehr getrennt vorliegen (sondern in einem den Baum abarbeitenden Programm verknüpft sind), und daß die Diagnose streng deterministisch verläuft. Obwohl die Abarbeitungsreihenfolge eigentlich keiner weiteren Erläuterung bedarf,

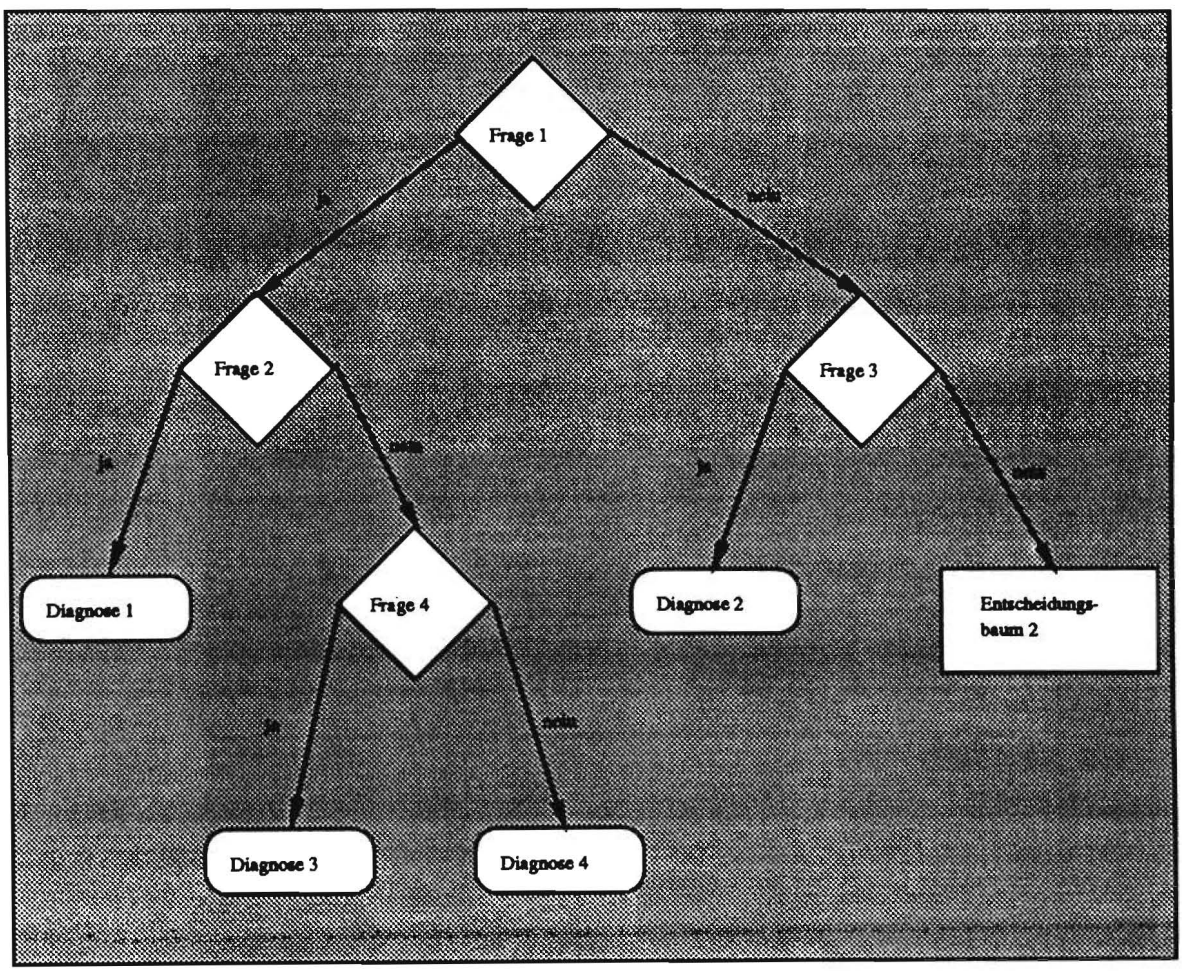

Abbildung 2.5: Entscheidungsbaum

sei der Vollständigkeit halber dennoch ein Basisalgorithmus angegeben:

```
Eingabe:  <wird nicht benötigt>.
Ausgabe:  Fehlerursachen.
```

[16]Deshalb nennt man Entscheidungsbäume auch *Flußdiagramme*.

```
1. AKTUELLER-KNOTEN := Wurzel des Baumes.
2. WIEDERHOLE BIS AKTUELLER-KNOTEN ein Blatt ist:
   2.1 Fordere die mit AKTUELLER-KNOTEN assoziierten Daten an.
   2.2 Verzweige gemäß der erhaltenen Daten zum nächsten Knoten
       (NACHFOLGER).
   2.3 AKTUELLER-KNOTEN := NACHFOLGER.
3 Ausgabe der mit AKTUELLER-KNOTEN assoziierten Diagnose.
```

Entscheidungstabellen

Bei Entscheidungstabellen ist das Wissen in Form von Regeln kodiert. Im Gegensatz zu Entscheidungsbäumen geben sie keine Dialogsteuerung vor; es feuert immer diejenige Regel, deren Vorbedingungen erfüllt sind.
Praktisch ist eine Entscheidungstabelle ein entsequentialisierter Entscheidungsbaum, d. h. ein Pfad von der Wurzel bis zu einem Blatt wird in Form einer Regel kodiert. Dabei stellt jede Kante dieses Pfades eine der Vorbedingungen der Regel dar, während die Konsequenz der Regel der mit dem Blatt assoziierten Diagnose entspricht.
Zunächst sei die zum Entscheidungsbaum aus Abbildung 2.5 adäquate Entscheidungstabelle angegeben. Danach erfolgt zur Erklärung derselben die in Regeln codierte Version der Tabelle.

	Regel1	Regel2	Regel3	Regel4	Regel5
Frage1	+	+	+	-	-
Frage2	+	-	-		
Frage3				+	-
Frage4		+	-		
Diagnose1	+				
Diagnose2				+	
Diagnose3		+			
Diagnose4			+		
Entscheidungstabelle2					+

Regel1: (Antwort1=ja) & (Antwort2=ja) $\longrightarrow$ Diagnose1
Regel2: (Antwort1=ja) & (Antwort2=nein) & (Antwort4=ja) $\longrightarrow$ Diagnose3
Regel3: (Antwort1=ja) & (Antwort2=nein) & (Antwort4=nein) $\longrightarrow$ Diagnose4
Regel4: (Antwort1=nein) & (Antwort3=ja) $\longrightarrow$ Diagnose2
Regel5: (Antwort1=nein) & (Antwort3=nein) $\longrightarrow$ Entscheidungstabelle2

Die Aufführung eines Algorithmus erübrigt sich, da die Diagnosesitzung aus Eingabe von Symptomen und Feuern von Regeln besteht.

Da Entscheidungstabellen im Grunde heuristische Regeln mit einer assoziierten Regelwahrscheinlichkeit von 100% sind, soll in dieser Arbeit unter einem sicheren System ein Entscheidungsbaum verstanden werden, sofern keine genauere Spezifikation erfolgt.

2.3 Bewertung der Diagnosesystemklassen

In diesem Kapitel sollen die Stärken und Schwächen der einzelnen Diagnosesystemklassen beleuchtet werden. Dies soll insbesondere vor dem Hintergrund ihrer Eignung für technische Anwendungsbereiche geschehen, so daß anschließend eine zusammenfassende Wertung für die Eignung versucht werden kann.

2.3.1 Heuristische Diagnose

Stärken

- *Effizienz:* Der Overhead solcher Systeme ist sehr gering, da weder Gerätemodelle noch angeschlossene Datenbanken benutzt werden. Dies hat eine relativ kurze Laufzeit zur Folge.
  Der dem Benutzer zugemutete Aufwand ist ebenfalls sehr gering, da die notwendigen Eingabedaten minimal sind und nur Daten angefordert werden, die während des momentanen Inferenzzustandes benötigt werden.

- *Repräsentation von Unsicherheit:* Der Unvollständigkeit des Wissens, der Unsicherheit der menschlichen Wahrnehmung und der verschiedenartigen Beurteilung gleicher Sachverhalte durch verschiedene Menschen wird durch Zuordnung von Sicherheitsfaktoren zu Fakten und Assoziationen Rechnung getragen. Die Verrechnungsschemata sind häufig intuitiv vernünftig. Eine relativ verständliche Darstellung und Handhabung kann zu erhöhter Akzeptanz des Diagnosesystems durch den Benutzer führen.

- *Erweiterbarkeit:* Die Erweiterung oder Veränderung eines heuristischen Diagnosesystems ist einfach, da Wissensbasis und Inferenzmechanismus getrennt vorliegen. Dadurch ist gewährleistet, daß die Stelle der Veränderung schnell lokalisiert werden kann und keine Folgeänderungen notwendig werden.

- *Einfache Verarbeitung:* Die Abarbeitung heuristischen Wissens kann einfach gestaltet werden. Bei einer simplen Kontrollstrategie werden Regeln vor- oder rückwärts durchlaufen, wobei unbekannte Regelbestandteile erfragt oder durch feuern anderer Regeln geschlossen werden und Evidenzen gemäß dem Verrechnungsschema verrechnet werden.

- *Breite Anwendbarkeit:* Die Anforderungen, die an die Benutzbarkeit eines heuristischen Diagnosesystems gestellt werden, sind geringfügig: Existierende Erfahrungen sind ohne großen Overhead verwendbar.

Schwächen

- *Unvollständigkeit:* Die Diagnose ist auf explizit repräsentierte Fehler beschränkt. Diese Beschränkung rührt daher, daß kein Wissen über Struktur und Funktionsweise der zu diagnostizierenden Objekte einbezogen wird.

- *Schwierigkeiten bei der Diagnose von Mehrfachfehlern:* Es ist Aufgabe der Diagnosestrategie, dafür zu sorgen, daß nicht nur die am stärksten verdächtigte Diagnose etabliert wird, sondern alle Diagnosen, die nach eingehender Überprüfung immer noch hinreichend verdächtig sind. Probleme entstehen, wenn parallel auftretende Fehlerursachen überlappende oder entgegengesetzte Symptommuster haben, d. h. Symptome einer Ursache werden durch Symptome der anderen Ursache unterdrückt oder verfälscht, oder Symptome der einen Fehlerursache liefern Evidenz gegen die andere Fehlerursache.
  Dieses Problem kann theoretisch dadurch umgangen werden, daß jede Kombination von Fehlerursachen durch Regeln repräsentiert wird. Das würde jedoch zu einer kombinatorischen Explosion der Regelanzahl führen und ist deshalb nicht praktikabel.
  Aus diesem Grunde werden häufig behelfsmäßig Regeln mit Ausnahmen repräsentiert oder es wird anderes Zusatzwissen hinzugezogen, um bekannte Überschneidungen auffangen zu können.

- *Verfügbarkeit eines Experten:* Die Erstellung und Wartung des Expertensystems erfordert die (eventuell längere) Mitarbeit eines Experten, der dem System aufgeschlossen gegenübersteht und die notwendige Zeit und das notwendige Engagement aufbringt.

- *Keine Objektivierbarkeit:* Die Korrektheit der Diagnose kann nicht nachgewiesen werden.

Eignung

Heuristische Diagnosesysteme sind mit Abstand die am meisten verwendeten Diagnosesysteme in der industriellen Praxis. Grundsätzlich sind sie in vielen technischen Anwendungsbereichen einsetzbar, da wenig Restriktionen existieren, die ihren Einsatz verbieten. Je komplexer der Anwendungsbereich wird, um so mehr wird es aufgrund der anschwellenden Regelmenge notwendig, eine Kontrollstrategie mit effizienter Verdachtsgenerierung und -überprüfung einzusetzen, die Regelmenge zu strukturieren und Diagnosehierarchien zu repräsentieren.
Bei der Diagnosefähigkeit an sich (unter Vernachlässigung anderer Aspekte, wie z. B. der Effizienz) sind heuristische den modellbasierten (funktionalen) Diagnosesystemen unterlegen.

2.3.2 Modellbasierte Diagnosesysteme

Stärken

- *Vollständigkeit des Wissens:* Kausales Wissen eignet sich insbesondere dafür, technische Geräte (die im Gegensatz zum menschlichen Körper weder so komplex noch teilweise unverstanden sind) explizit zu modellieren. So kann das Problem des Umgangs mit unvollständigem Wissen heuristischer Systeme in manchen Anwendungsgebieten umgangen werden, wenn es möglich ist, das Gerät auf einer für die Diagnose hinreichenden Abstraktionsebene vollständig zu modellieren.

- *Objektivierbarkeit:* Das Ergebnis einer modellbasierten Diagnose ist eher „beweisbar" (bzw. plausibel), da der Fehler (unter Voraussetzung eines korrekten Modells) simuliert und damit Schritt für Schritt nachvollzogen werden kann. Bei heuristischen Systemen

kann man an die Richtigkeit der Diagnose nur auf Basis von Wahrscheinlichkeiten *glauben.*

- *Erkennung von Mehrfachfehlern:* Die Probleme der Symptomüberlappung heuristischer Systeme fallen bei funktionalen Systemen weg. Das liegt daran, daß Fehler und Symptommuster nicht explizit modelliert sind. Werden mehrere Komponenten eines Diagnoseobjektes verdächtigt, so werden die entsprechenden Modelländerungen vorgenommen und das Simulationsergebnis wird mit den aufgetretenen Symptomen verglichen. Können die Symptome erklärt werden, so werden die modellierten Fehler als Diagnose ausgegeben.

- *Vollständigkeit:* Da bei der funktionalen Diagnose Fehler nicht explizit repräsentiert werden, kann jeder Fehler erkannt werden, der durch eine Modelländerung simulierbar ist. Die Vollständigkeit der Fehlererkennung hängt von der Vollständigkeit und Korrektheit des zugrundeliegenden Modells und der gewählten Abstraktionsebene ab.

- *Änderungsfreundlichkeit:* Änderungen der technischen Spezifikationen der Diagnoseobjekte können leicht in die Modelle eingebracht werden.

- *Verständlichkeit der Lösung:* Durch die Objektivierbarkeit der Lösung über die Simulation ist auch die Verständlichkeit der Lösung gut, was sich ebenfalls positiv auf die Akzeptanz des Diagnosesystems auswirkt.

- *Erfahrungswissen nicht erforderlich:* Zur Erstellung der Wissensbasis wird die Erfahrung eines Diagnose–Experten nicht benötigt. Gebraucht werden stattdessen Konstruktionspläne des Diagnoseobjektes und Techniker, die mit der Funktionsweise des Objektes vertraut sind.

- *Anwendungsneutrales Wissen:* Funktionales Wissen ist relativ anwendungsneutral, so daß neben Diagnosesystemen auch andere Systeme[17] aus diesem Wissen Nutzen ziehen können.

Schwächen

- *Ineffizienz:* Wenn die Modelle sehr komplex sind, wird der Simulationsaufwand möglicherweise untragbar. Eine Vereinfachung der Modelle durch Abstraktion geht auf Kosten der Vollständigkeit (nicht jeder Fehler kann mehr gefunden werden).

- *Schwierigkeiten bei der Hypothesengenerierung:* Eine effiziente Hypothesengenerierung ist nur unter Verwendung von Zusatzwissen möglich (siehe dazu auch die Verdachtsgenerierungstechniken aus Kapitel 3.1).

- *Rückkopplungsschleifen:* Die Behandlung von Rückkopplungsschleifen erfordert Zusatzwissen und kann dazu führen, daß die Abarbeitung eines Modells ineffizient wird.

- *Aufwand der Modellerstellung:* Je nach Komplexität der Diagnoseobjekte kann die Modellerstellung sehr aufwendig werden.

[17] Z. B. Planungs– und Konstruktionssysteme.

- *Probleme bei fehlendem Wissen:* Sind Wechselwirkungen nicht bekannt oder aufgrund zu hoher Komplexität des Modells nicht berücksichtigt, so können nicht mehr alle Fehler diagnostiziert werden. Heuristische Systeme können in diesem Fall mutmaßen, modellbasierte nicht.

- *Fehlen von Erfahrungswissen:* Das Auftreten von Fehlern kann durch die Vorgeschichte des Diagnoseobjektes bedingt sein. Z. B. kann ein elektrisches Gerät länger Feuchtigkeit ausgesetzt gewesen sein. Ein modellbasiertes System findet zwar die unmittelbare Ursache für den Defekt, nicht aber die mittelbare länger zurückliegende Ursache, nämlich die Feuchtigkeit. Was also nicht berücksichtigt werden kann, sind Umwelteinflüsse, weil sie außerhalb des Modells liegen.

Eignung

Modellbasierte (besonders funktionale) Systeme eignen sich dann für die Diagnose, wenn der Anwendungsbereich die Erstellung eines Modells erlaubt, das möglichst vollständig ist und dessen Bearbeitung hinreichend effizient durchgeführt werden kann. Die effiziente Bearbeitung eines Modells wird durch die folgenden Aspekte bestimmt:

- Multiple gleichzeitige Ursachen für einen Zustand (erfordert Repräsentation von Schweregraden von Zuständen und die Fähigkeit zur Summation der Einzeleffekte)
- Verschiedene Arten von Beziehungen: z. B. linear, multiplikativ, Schwellwerteffekt, reversibel, irreversibel
- Abhängigkeiten von verstärkenden oder abschwächenden Faktoren (Katalysatoreffekt) und von Randbedingungen
- Rückkopplungsschleifen
- Zeitliche Beziehungen zwischen Ursache und Wirkung (z. B. gleichzeitig, verzögert, überlappend)
- Verschiedene Abstraktionsebenen

Vom Prinzip her eignen sich modellbasierte Systeme auch für komplexe Anwendungsbereiche (insbesondere dann, wenn die Simulation auf Teile des Modells beschränkt werden kann). Leider gibt es bisher hauptsächlich nur Prototypen und nur wenig industriell eingesetzte Systeme.

2.3.3 Fallbasierte Diagnose

Stärken

- *Effizienz zur Laufzeit:* In der Regel sind fallbasierte Systeme effizient, da sequentiell der aktuelle Fall mit der Menge der verdächtigten Fälle verglichen wird. Der Aufwand beträgt damit O(n) mit n = Anzahl der Verdachtshypothesen. Die Effizienz hängt maßgeblich von der Hypothesengenerierungs- und Hypothesendiskriminierungsphase ab.

- *Vermeidung früherer Fehler:* Viele fallbasierte Systeme sind in der Lage, aus Fehldiagnosen zu lernen. Aber auch korrekt diagnostizierte Fälle können in die Falldatenbank

aufgenommen werden. Damit verbessert ein fallbasiertes System seine Diagnosefähigkeit ständig.

- *Wenig Akquisitionsaufwand:* Aufwendige Regelakquirierungs- oder Modellierungsphasen entfallen.
- *Keine Sonderbehandlung von Ausnahmesituationen:* Ausnahmesituationen werden mit den gleichen Methoden behandelt, wie häufig vorkommende Fälle. Deshalb wird kein Zusatzwissen zu ihrer Behandlung benötigt. Auch einen etwaigen Performanzverlust (Verlängerung des Diagnoseprozesses) gibt es nicht.
- *Breite Anwendbarkeit:* Fallbasierte Diagnosesysteme können in jedem Anwendungsbereich eingesetzt werden. Dieser Vorteil macht sich insbesondere bei noch nicht wissenschaftlich durchdrungenen Anwendungsdomänen bemerkbar.

Schwächen

- *Kleine Falldatenbanken:* Da Maschinentypen bedingt durch den technischen Fortschritt häufig modifiziert oder ausgetauscht werden, können größere Fallsammlungen nur schwer entstehen. In medizinischen Anwendungsdomänen dagegen existieren teilweise jahrzehnte- oder gar jahrhundertealte Fallsammlungen.
- *Problemabhängige Anpassungsmethoden:* Die Anpassung der Diagnose des ähnlichsten Falles auf den aktuellen Fall erfordert Methoden, die stark problemabhängig sind. Man kann also nicht auf irgendwelche Universalwerkzeuge zurückgreifen, sondern muß möglicherweise Anpassungsmethoden für jede Anwendung individuell gestalten.
- *Rückständigkeit:* Diagnoseobjekte, die erst seit kurzer Zeit auf dem Markt sind, können nicht mittels fallbasierter Diagnose diagnostiziert werden, da noch keine sie betreffenden ausreichend großen Falldatenbanken existieren.
- *Keine Vollständigkeit:* Wie bei der heuristischen Diagnose können nur explizit repräsentierte Fehler diagnostiziert werden, bzw. Fehler, auf die durch (auf repräsentierte Fehler) angewandte Anpassungsmethoden geschlossen wird.

Eignung

Einzige Bedingung für den Einsatz fallbasierter Diagnosesysteme ist eine hinreichend große und strukturierte Falldatenbank. In technischen Anwendungsgebieten ist diese Bedingung häufig nicht erfüllt. Insbesondere für komplexe Diagnoseobjekte (komplexe Anwendungsbereiche) existieren so viele verschiedene Fehlerfälle, daß an die Größe der Falldatenbank höhere Ansprüche gestellt werden müssen, um den Benutzeranforderungen bezüglich der Güte des Diagnosesystems gerecht zu werden. Deshalb bleibt trotz der Attraktivität der noch jungen Idee der fallbasierten Diagnose die Anwendung auf kleine und gut strukturierte Anwendungsbereiche beschränkt, solange nicht bewußt jeder Fehlerfall auf Datenträgern festgehalten wird. Die Notwendigkeit der Speicherung vorgefallener Fälle in adäquater Form dürfte für Anwender schwer einzusehen sein, solange in der Industrie hauptsächlich heuristische Diagnosesysteme eingesetzt werden, die derartige Informationen nicht benötigen.

2.3.4 Statistische Diagnose

Stärken

- *Objektivierbarkeit:* Die Ergebnisse der statistischen Diagnose zeichnen sich durch eine hohe Objektivierbarkeit aus, sofern die Voraussetzungen für ihre Anwendbarkeit eingehalten werden.
- *Effizienz zur Laufzeit:* Da das repräsentierte Wissen bei rein statistischen Systemen sehr kompakt (Formeln und Apriori–Wahrscheinlichkeiten) und die Abarbeitung einfach ist, ist der Zeitbedarf zur Diagnosefindung relativ gering. Dies gilt nur unter der Voraussetzung, daß die Apriori–Wahrscheinlichkeiten nicht mehr aus einer Falldatenbank berechnet werden müssen, sondern schon in verdichteter Form vorliegen.

Schwächen

- *Beschränkte Anwendbarkeit:* Die Voraussetzungen zur Anwendbarkeit der statistischen Diagnose (Theorem von Bayes) (Unabhängigkeit der Symptome untereinander, Vollständigkeit der Diagnosemenge, wechselseitiger Ausschluß von Diagnosen, Repräsentativität der Fallsammlung, Konstanz der Wahrscheinlichkeiten) sind selten erfüllbar. Wie oben erwähnt, ist technischer Standard schnell veraltet. Deshalb sind auch Falldatenbanken häufig zu klein, um die Vollständigkeit und die Repräsentativität der Diagnosemenge zu gewährleisten. Wird eine kleine Falldatenbank aufgefüllt, so können sich die Apriori–Wahrscheinlichkeiten drastisch ändern, wodurch die Voraussetzung der Konstanz nicht erfüllt ist.
- *Rückständigkeit:* Siehe Schwächen der fallbasierten Diagnose.
- *Keine Parallelen zu menschlicher Vorgehensweise:* Die Auswertung komplexer mathematischer Formeln hat nichts mit der Art und Weise zu tun, wie ein menschlicher Experte in der Regel diagnostiziert. Deshalb sind die Begründungen eines statistischen Systems für seine Diagnose für den Benutzer nicht sehr einsichtig. Dies kann zu Mißtrauen gegenüber dem Diagnosesystem führen, was sich negativ auf die Akzeptanz auswirkt.
- *Einfehler–Annahme (single–fault–assumption):* Dies ist eine Voraussetzung (gegenseitiger Ausschluß der Diagnosen), und deshalb können Mehrfachfehler nicht diagnostiziert werden.
- *Nullwahrscheinlichkeiten:* Bei besonders seltenen Diagnosen, bei denen die Apriori–Wahrscheinlichkeiten nahe bei Null liegen, werden auch die Endwahrscheinlichkeiten trotz auf sie hindeutender Symptome sehr gering sein, da ihre jeweiligen Apriori–Wahrscheinlichkeiten als Faktoren in die Formel eingehen (siehe Bayes–Formel aus Kapitel 2.2). Sehr seltene Diagnosen können also nicht diagnostiziert werden.
- *Keine Vollständigkeit:* Dies ist zwar eine der Voraussetzungen zur Anwendbarkeit der statistischen Diagnose, sie ist jedoch in der Praxis in der Regel nicht erfüllbar, da Fehler explizit repräsentiert werden müssen.

Eignung

Die Voraussetzungen zur Anwendung statistischer Diagnosesysteme sind selten gegeben. Die schon bei der fallbasierten Diagnostik genannten Gründe für die Beschränkung auf kleine, strukturierte Anwendungsgebiete gelten auch hier. Aufgrund der vielfältigen Nachteile ist ein statistisches System als stand–alone Diagnosesystem für die technische Diagnose nicht geeignet.

2.3.5 Sichere Diagnose

Stärken

- *Kompaktheit:* Bei kleinen Systemen ist die Darstellung kompakt und auch übersichtlich.
- *Effizienz zur Laufzeit:* Da das System schon in Form eines Programms vorliegt, entfallen teure Such–, Zugriffs– und Übersetzungszeiten.
- *Einfache Erstellung und Erweiterung durch Erfahrungswissen:* Der Lösungsweg eines Experten kann direkt in Form eines Teilbaumes in das Diagnosesystem übernommen werden. Neue Diagnosen und zugehörige Lösungswege können gleichermaßen einfach integriert werden.

Schwächen

- *Keine Flexibilität:* Symptome müssen in einer vorgeschriebenen Reihenfolge verarbeitet werden. Dies ist insbesondere bei Prozeßankopplungen, bei denen Daten in unvorhersehbaren Reihenfolgen und Zeitabständen anfallen, untragbar.
- *Änderungsfeindlichkeit:* Muß das Diagnosesystem modifiziert werden, so treten bei größeren, sicheren Systemen Probleme auf. Durch die Vermischung von Wissen und Abarbeitungsstrategie ist die Lokalisierung der zu ändernden Stellen erschwert. Zusätzlich kann eine einfache Änderung oder Einfügung eine Änderung der Baumstruktur nach sich ziehen, da die Bedeutung einer Frage nur im Kontext der vorherigen Fragen gegeben ist. Es kann sogar vorkommmen, daß während einer Diagnose dynamisch Pfade geändert werden müssen. Das ist bei Entscheidungsbäumen praktisch nicht machbar.
  Alle diese Probleme treten bei einer Trennung von Wissen und Abarbeitungsstrategie in diesem Maße nicht auf.
- *Keine Mehrfachfehlerbehandlung:* Aus den gleichen Gründen wie bei heuristischen Systemen entfällt eine Repräsentation jeder Fehlerkombination (Repräsentiert würde sie über einen Pfad von der Wurzel bis zu dem der Fehlerkombination entsprechenden Blatt). Heuristische Systeme können dies durch eine flexible Kontrollstrategie bis zu einem gewissen Grade kompensieren. Entscheidungsbäume können das aufgrund ihrer statischen Struktur nicht. Jede weitere Fehlerverfolgung müßte durch Implementierung eines weiteren Teilbaumes realisiert werden.

- *Wenig Benutzerinteraktion möglich:* Die Initiative während einer Diagnosesitzung liegt beim System. Der Benutzer kann weder zu erhebende Symptome, noch zu untersuchende Hypothesen vorschlagen.

- *Probleme bei fehlender Eingabe:* Kann dem System auf eine gestellte Frage keine Antwort gegeben werden, so ist die weitere Vorgehensweise des Systems nicht definiert, da ein Knoten nur in Abhängigkeit der Eingabe verlassen werden kann. Um dieses in den Griff zu bekommen, muß bei jeder Frage der „unknown–Fall" (Antwort unbekannt) mittels eines eigenen Teilbaums vorgesehen werden. Der Aufwand hierfür könnte beträchtlich sein.

- *Falsche Darstellung:* Ist das Wissen über die Anwendungsdomäne unsicher, so kann eine Repräsentation ohne Unsicherheiten (und damit ohne Verrechnungsschema) leicht zu Fehldiagnosen führen.

- *Unterstellung eines idealen Diagnoseverlaufs:* Entscheidungsbäume gehen davon aus, daß während eines Diagnoseverlaufs keine unvorhergesehenen Fehler auftreten. Bedienungsfehler während der Diagnosesitzung, die zu einer Zerstörung eines Bauteils des Diagnoseobjektes führen, können nicht einbezogen werden.

- *Keine Vollständigkeit.*

- *Redundanzen:* Entscheidungsbäume können Knoten haben, die mehrfach (redundant) repräsentiert sind. Das liegt daran, daß an sich gleiche Knoten in verschiedene Kontexte eingebettet sein können, so daß von ihnen aus in unterschiedliche Teilbäume verzweigt wird.

Beispiel: *Ein Knoten stelle das Geräusch eines Wagens nach dem Anlassen dar. Wurde vorher der Kontext Zündanlage festgestellt, so kommen als Nachfolgeknoten nur solche in Frage, die sich auf die Zündanlage beziehen. Wurde dagegen der Kontext Motor etabliert, so werden potentielle Nachfolgerknoten sich inhaltlich auf den Motor und seine Bestandteile beziehen.*

Eignung

In Anwendungsgebieten, in denen die Diagnose einen stereotypen Verlauf hat, ist es sinnlos, Unsicherheiten, Modelle oder Falldatenbanken zu verwenden. In diesem Fall eignen sich Entscheidungsbäume.
Sobald das Anwendungsgebiet komplexer wird und der Verlauf der Diagnose nicht von vornherein deterministisch ist, blähen sich fallbasierte Systeme aufgrund ihrer unstrukturierten Darstellungsform auf, werden unübersichtlich und enthalten Redundanzen.
Damit ist die Anwendbarkeit von Entscheidungsbäumen auf kleine Bereiche, deren zugehöriges Diagnosewissen vollständig ist, beschränkt.

Kapitel 3

Praktische Umsetzungen

Nachdem die theoretischen Konzepte der verschiedenen Diagnosesystemklassen diskutiert worden sind, sollen in diesem Kapitel praktische Umsetzungen vorgestellt werden.
Dazu werden zunächst die möglichen Bestandteile, aus denen sich ein Diagnoseverfahren zusammensetzen kann, jeweils durch Angabe verschiedener Realisierungsmöglichkeiten erläutert. Anschließend werden die Diagnoseverfahren einiger existierender Diagnosesysteme analysiert und bewertet. Dies geschieht unter Zuhilfenahme der zuvor vorgestellten Bestandteile, der Anforderungen an ein Diagnosesystem aus Kapitel 2.1 und der Bewertung der allgemeinen Diagosesystemklassen aus Kapitel 2.2.

3.1 Komponenten eines Diagnoseverfahrens

Jedes Diagnoseverfahren setzt sich aus mehreren Komponenten (Funktionsmodulen, Bausteinen) zusammen. Betrachtet man ein heuristisches Diagnoseverfahren, so besteht dieses typischerweise aus zumindest folgenden Komponenten: Einer Kontrollstrategie, einer Methode zur Testauswahl, einem Verrechnungsschema für Wahrscheinlichkeiten und einem Bewertungsverfahren für die Enddiagnosen.
Ein modellbasiertes Diagnoseverfahren könnte dagegen aus folgenden Komponenten zusammengesetzt sein: Einer Kontrollstrategie, einer Komponente zur Verdachtsgenerierung, einer Methode zur Testauswahl, einer Methode zur Behandlung von Rückkopplungsschleifen und einem Bewertungsverfahren für die Enddiagnosen.
Es gibt Komponenten, die für mehrere Diagnoseklassen geeignet sind (z.B. Kontrollstrategien und Bewertungsverfahren), und Komponenten, die speziell für eine bestimmte Klasse gedacht sind (z. B. Ähnlichkeits– und Distanzmaße für die fallbasierte Diagnose).
In diesem Abschnitt sollen die möglichen Komponenten eines Diagnoseverfahrens und ihre Ausprägungen vorgestellt werden. Im einzelnen handelt es sich um folgende Komponenten:

- *Prädiagnostische Methoden*
  Diese Methoden leisten eine Vorverarbeitung des für die Diagnose relevanten Wissens.

- *Kontrollstrategien*
  Sie bestimmen den Ablauf des Diagnoseverfahrens.

- *Methoden zur Verdachtsgenerierung*
  Sie erzeugen möglichst schnell einen Verdacht, der im weiteren Verlauf des Verfahrens überprüft wird.

- *Verrechnungsschemata für Wahrscheinlichkeiten*
  Hierbei handelt es sich um Vorgaben, nach denen auftretende Teilwahrscheinlichkeiten zu Gesamtwahrscheinlichkeiten verrechnet werden.

- *Methoden zur Behandlung von Rückkopplungsschleifen*
  Dies sind Methoden, die den Wert sich selbst verstärkender, oszillierender oder sich selbst abschwächender Effekte bestimmen.

- *Ähnlichkeits- und Distanzmaße*
  Es handelt sich um Formeln, die zur Bestimmung der Ähnlichkeit zweier Diagnosefälle herangezogen werden.

- *Methoden zur Testauswahl*
  Testauswahlmethoden bestimmen, welches Merkmal als nächstes mit welchem Test erhoben werden soll.

- *Mechanismen zur Rücknahme von Schlußfolgerungen*
  Werden Daten zurückgezogen, so müssen sämtliche von ihnen abhängige Schlußfolgerungen rekursiv revidiert werden.

- *Bewertungskriterien*
  Hierbei handelt es sich um Kriterien, die herangezogen werden, um die Diagnosewahrscheinlichkeit zu bestimmen.

- *Differentialdiagnostik*
  Um eine Entscheidung für eine Diagnose zu finden, werden die in Frage kommenden Diagnosen relativ zueinander bewertet.

Abbildung 3.1 zeigt die logische Abarbeitungsreihenfolge der Bausteine. Jeder Pfad durch den Graphen bildet ein mögliches Diagnoseverfahren. Knoten auf dem Pfad können abgearbeitet werden, müssen es aber nicht. Ein modellbasiertes Verfahren wird nicht unbedingt ein Verrechnungsschema für Wahrscheinlichkeiten benötigen. Prädiagnostische Methoden und Fehlerfeststellungsmethoden müssen auch nicht Bestandteil eines Diagnoseverfahrens sein. Die Kontrollstrategien in der Mitte des Graphen steuern dessen Abarbeitung. Teilweise ist es nicht eindeutig, welcher Phase eines Diagnoseverfahrens welche Bausteine zuzuordnen sind. Der Baustein Bewertungskriterien könnte sowohl der Verdachtsüberprüfung, als auch der Verdachtsdiskriminierung zugesprochen werden. Aus diesem Grund sind die Phasen des Diagnoseverfahrens nicht mit den entsprechenden Bausteinen verbunden, sondern stehen ungefähr in einer Höhe mit ihnen zuzurechnenden Bausteinen.
Die folgende Tabelle beschreibt, welche Bausteine bei welchen Diagnoseverfahren zum Einsatz kommen können.

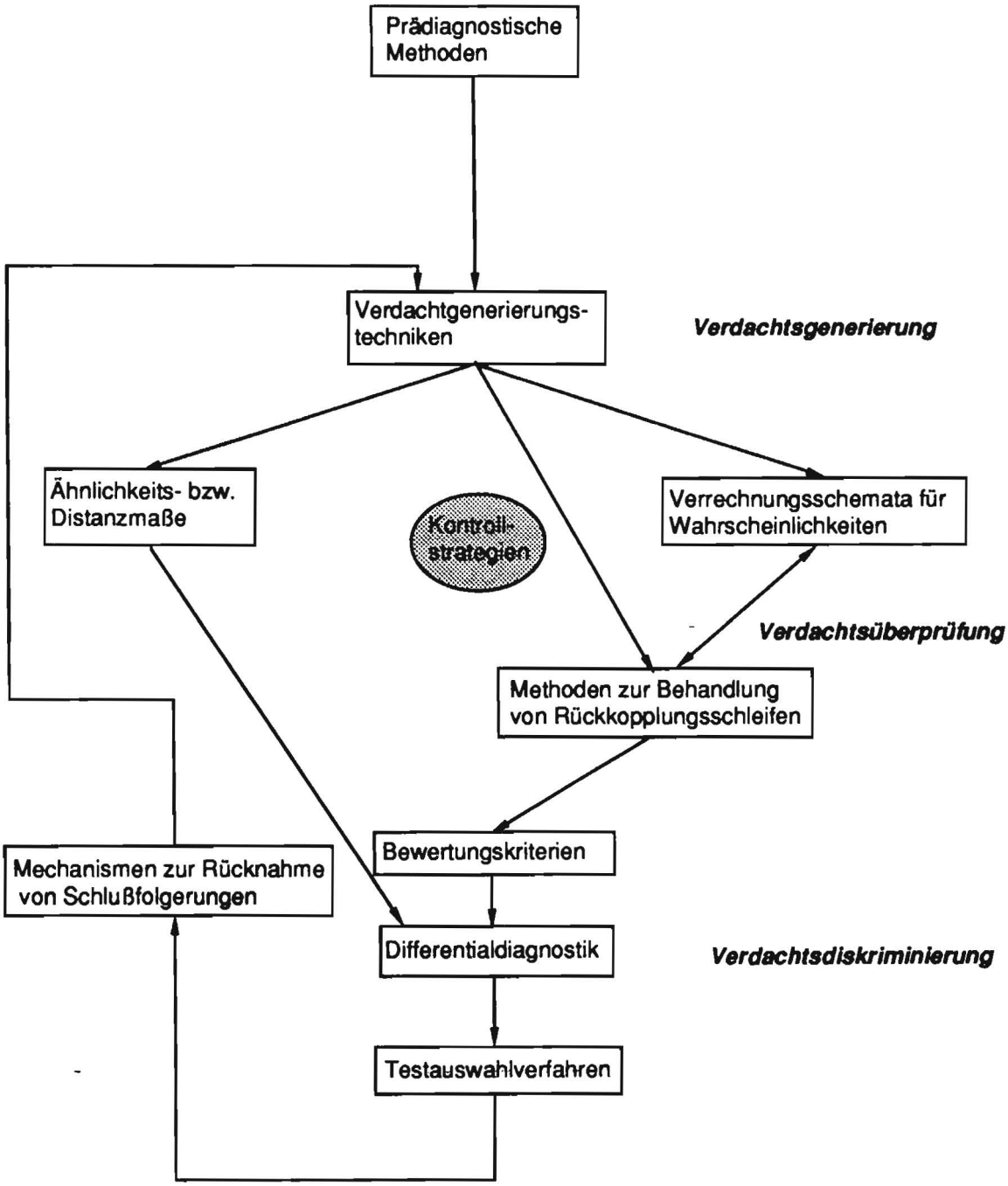

Abbildung 3.1: In Bausteine zerlegte Diagnoseverfahren.

	EB	ET	HD	PD	FD	FB	SD
Prädiagnostische Methoden	x	x	x	x	x	x	x
Kontrollstrategien	(x)	x	x	x	x	(x)	(x)
Methoden zur Verdachtsgenerierung			x	x	x	x	x
Verrechnungsschemata für Wahrscheinlichkeiten			x	(x)			x
Behandlung von Rückkopplungsschleifen				x	x		
Ähnlichkeits- oder Distanzmaße						x	
Methoden zur Testauswahl		(x)	x	x	x	x	x
Rücknahme von Schlußfolgerungen		*x*	*x*	*(x)*	*(x)*	*(x)*	*(x)*
Bewertungskriterien	(x)	(x)	x	x	x	x	x
Differentialdiagnostik	(x)		x	x	x	x	x

EB = Entscheidungsbaum, ET = Entscheidungstabelle, HD = Heuristische Diagnose, PD = Pathophysiologische Diagnose, FD = Funktionale Diagnose, FB = Fallbasierte Diagnose, SD = Statistische Diagnose. Die in Klammern gesetzten Kreuze deuten an, daß der entsprechende Baustein bei der entsprechenden Diagnose weniger deutlich oder seltener in Erscheinung tritt.

3.1.1 Prädiagnostische Methoden

Unter prädiagnostischen Methoden sind der Diagnose vorgeschaltete und symptomvorverarbeitende Methoden zu verstehen.
Die der Diagnose vorgeschalteten Methoden verarbeiten fallunspezifisches Diagnosewissen und bringen es in eine für die jeweilige Diagnose adäquate Form. Als Beispiele werden die Methode des *causal ordering* und die Anwendung von *Diagnosekennzahlen* kurz erläutert.
Die symptomvorverarbeitenden Methoden verarbeiten fallspezifisches Wissen, d. h. sie kommen zur Laufzeit des Diagnosesystems zur Anwendung. Beispiele hierfür sind die *Umwandlung quantitativer in qualitative Werte* und die *Propagierung von Zeitrelationen.*

Die Methode des causal ordering: Um ein modellbasiertes Diagnosesystem zu erstellen, werden Kausalbeziehungen der Art „Ursache erzeugt Wirkung" benötigt. Liegt nun ein Modell des zu diagnostizierenden Objektes vor, in dem die funktionalen Beziehungen in Form von Gleichungen angegeben sind, so ist es erforderlich, diese zu richten, um Kausalitäten zu erhalten. Dazu müssen die Gleichungen entsprechend aufbereitet werden. Aus Effizienzgründen sollte ebenfalls darauf geachtet werden, Redundanzen aus dem Gleichungssystem zu entfernen. Das Verfahren wird nur kurz und vereinfachend skizziert. Für nähere Informationen und die Voraussetzungen seiner Anwendbarkeit sei auf [Iwa88] verwiesen.

Das Einrichten einer kausalen Ordnung aus einem vorgegebenen Gleichungssystem beginnt mit dem Auffinden von Untermengen von Variablen, deren Werte unabhängig von anderen Variablen berechnet werden können. Diese Variablen werden dazu verwendet, die Gleichungsstruktur auf eine äquivalente kleinere Menge von Gleichungen zu reduzieren, die nur noch die übrigen Variablen enthalten. Aus jeder dieser neuen Gleichungen wird die Variablenmenge bestimmt, die von den übrigen Variablen dieser Gleichung direkt kausal abhängt. Dem entsprechend wird jede Gleichung abschließend gerichtet.

Diagnosekennzahlen: Um den Informationsgehalt von Merkmalen, die für sich alleine be-

trachtet nicht sehr aussagekräftig sind, zu erhöhen, werden mehrere Merkmale zu sogenannten Diagnosekennzahlen kombiniert. A. Sturm und R. Förster zeigen in [SF87] eine Methode auf, Diagnosekennzahlen systematisch zu erstellen.

Umwandlung quantitativer in qualitative Werte: Häufig reicht es aus, bzw. ist anschaulicher und/oder effizienter, statt kontinuierlicher Wertebereiche diskrete Wertebereiche für Merkmale zu verwenden. In diesem Fall muß eine Zuordnungsfunktion realisiert werden, die einer Merkmalsausprägung aus einem kontinuierlichen Wertebereich den entsprechenden diskreten Wert zuordnet.

Beispiel: *Einteilung des kontinuierlichen Wertebereiches „Temperatur" in den diskreten Wertebereich „zu niedrig (kleiner 10°), ausreichend (10° - 20°), ideal (20° - 30°), zu hoch (über 30°)".*

Propagierung von Zeitrelationen: Oft ist es der Fall, daß sich eine Fehlerursache nicht aus den Ausprägungen einzelner Merkmale zu einem bestimmten Zeitpunkt bestimmmen läßt, sondern über die zeitliche Reihenfolge des Auftretens der Symptome.
Durch die Eingabe der Symptome zusammen mit dem Zeitpunkt ihres Auftretens oder den Zeitintervallen ihrer Dauer erhält man sogenannte *primitive* Relationen zwischen den Symptomen.

Beispiel: *Symptom1 liegt zeitlich vor Symptom2*
Symptom2 liegt zeitlich vor Symptom3

Aufgabe der Zeitpropagierung ist es nun, aus den primitiven Relationen weitere Relationen zu schließen. Für dieses Beispiel läßt sich folgende Relation ableiten: *Symptom1 liegt zeitlich vor Symptom3.*
Die bekanntesten Methoden zur Verarbeitung von Zeitrelationen liefert die Allen'sche Zeitlogik (siehe [All83]).

3.1.2 Kontrollstrategien

Die Vorgehensweise bei der Fehlerursachenfindung innerhalb des durch die Diagnosesystemklasse gesteckten Rahmens wird durch die Kontrollstrategie bestimmt. Sie zeichnet zu einem großen Teil für die Effizienz des Diagnoseverfahrens verantwortlich. Insbesondere für heuristische Systeme ist eine der folgenden vier hauptsächlich benutzten Kontrollstrategien typisch.
Für die Darstellung der Strategien sei angenommen, daß das Wissen in Regelform vorliegt.

- *Vorwärtsverkettung (Forward-Reasoning)*

  Geschlossen wird von vorgegebenen Symptomen ggfs. über Zwischenstufen auf Fehlerursachen. Es feuern alle Regeln, deren Vorbedingung erfüllt ist. Die Ausführung der Konklusion ermöglicht das Feuern weiterer Regeln. Der Schlußfolgerungsprozeß endet, wenn keine Regeln mehr feuerbereit sind.
  Aufgrund ihrer Ineffizienz durch die erschöpfende Suche eignet sich diese Strategie nur für kleinere Probleme, bei denen die Symptome möglichst vollständig angegeben sind.

- *Rückwärtsverkettung (Backward-Reasoning)*

  Bei der Rückwärtsverkettung werden von der Fehlerursache ausgehend alle Regeln untersucht, die in ihrem Konklusionsteil auf eine Fehlerursache schließen. Sind Vorbedingungen dieser Regeln nicht bekannt, so wird versucht, diese Vorbedingungen vom Benutzer zu erfragen oder über erneute Rückwärtsverkettung herzuleiten.
  Da alle denkbaren Fehlerursachen untersucht werden müssen, eignet sich die Rückwärtsverkettung ebenfalls nur für kleinere Probleme. Die Angabe der Symptome kann jedoch unvollständig sein, da die Strategie gezielt nach unbekannten Symptomen forscht.

- *Establish-Refine*

  Die Establish–Refine–Strategie vermeidet die ungesteuerte Suche der beiden vorherigen Strategien, indem eine hierarchische Struktur von Fehlerursachen (bzw. Fehlerursachenbereichen) durchlaufen wird.
  Diese Hierarchie ist so aufgebaut, daß von der Wurzel zu den Blättern hin die Fehlerursachen immer weiter spezifiziert werden.
  Auf einer Ebene der Hierarchie wird versucht, eine Ursache (z. B. mittels Rückwärtsverkettung) zu etablieren (establish). Ist dies gelungen, so werden auf der nächsttieferen Ebene die Nachfolger untersucht (refine), von denen wieder einer etabliert wird, usw. .
  Voraussetzungen für die Anwendung dieser Strategie sind die Hierarchisierbarkeit der Fehlerursachen(bereiche) und die Erfüllung der Forderung, daß auf jeder Ebene eine Ursache etabliert werden kann.
  Da die Suche nach der Fehlerursache relativ gezielt abläuft, eignet sich diese Strategie auch für größere Probleme.

- *Hypothesize-and-Test*

  Die Hypothesize–and–Test–Strategie ermittelt über Vorwärtsverkettung mögliche verdächtige Fehlerursachen. Daraufhin wird versucht, die am stärksten verdächtigte Fehlerursache über Rückwärtsverkettung zu etablieren. Gelingt dies nicht, so wird die am zweitstärksten verdächtigte Fehlerursache untersucht, usw.
  Diese Strategie ist flexibler als die Establish–Refine–Strategie, da deren relativ enge Voraussetzungen hier nicht gegeben sein müssen und mit ihr alle übrigen Strategien simuliert werden können. Da sie zudem sehr zielgerichtet sein kann, ist sie insbesondere bei heuristischen Diagnosesystemen am häufigsten anzutreffen. Sie kann jedoch ebenso in anderen Diagnosesystemklassen realisiert werden.

Welche Kontrollstrategien sich für welche Diagnosesystemklassen eignen, kann der folgenden Tabelle ([Pūp90]) entnommen werden.

	Vorwärts	Rückwärts	Establish–Refine	Hypothesize–and–Test
Entscheidungsbäume			x	
Entscheidungstabellen	x	(x)		
Heuristische Diagnose	(x)	(x)	(x)	x
Pathophysiologische Diagnose			(x)	x
Funktionale Diagnose			(x)	x
Statistische Diagnose	x			
Fallbasierte Diagnose			(x)	x

Die Klammern deuten geringere Präferenz an.

Neben den hier aufgeführten Kontrollstrategien seien noch die *Divide-and-Conquer-* und die *Branch-and-Bound*-Strategie erwähnt.
Die Divide-and-Conquer-Strategie versucht, den Problembereich in mehrere Unterbereiche einzuteilen, und diese dann unabhängig voneinander auf Fehlerursachen zu testen. Voraussetzung für die Anwendung dieser Strategie ist die Teilbarkeit des Problembereiches in Unterbereiche, die nicht voneinander abhängen.
Wird die Branch-and-Bound-Strategie verwendet, so wird immer der Fehlerursachenbereich expandiert, der momentan am stärksten verdächtigt wird. Die übrigen Bereiche werden jedoch nicht verworfen, sondern bei der nächsten Entscheidung, welcher Bereich expandiert werden soll, erneut in Betracht gezogen. Dies ist der entscheidende Unterschied zur Establish-Refine-Strategie, bei der immer nur lokal einer der Söhne des aktuellen Fehlerursachenbereiches expandiert wird. Da sich der Fehlerursachenbereich bei der technischen Diagnose in der Regel relativ schnell herauskristallisiert, ist eine lokale Suche effizienter und für den Benutzer leichter nachvollziehbar. Voraussetzung zur sinnvollen Anwendung ist also ein Anwendungsbereich, dessen Struktur hierarchisierbar ist, und bei dem sich die Fehlerursachenbereiche erst relativ spät herauskristallisieren.

Da die Voraussetzungen zur Anwendung dieser beiden Strategien selten gegeben sind, und man sie von existierenden Diagnosesystemen selten benutzt sieht, sei auf eine detailliertere Darstellung hier verzichtet.

3.1.3 Methoden zur Verdachtgenerierung

Zu Beginn des vorherigen Kapitels wurde der Diagnosevorgang u. a. allgemein in drei Phasen eingeteilt: Hypothesengenerierung, Hypothesenüberprüfung und Hypothesendiskriminierung. Verdachtgenerierungstechniken dienen dazu, die erste Phase zu realisieren.
Im Gegensatz zu Kontrollstrategien sind sie relativ gezielt auf eine Diagnoseklasse zugeschnitten und in der Regel nicht für andere Diagnoseklassen verwendbar (natürlich nur unter der Voraussetzung, daß kein Hybrid erzeugt werden soll). Insbesondere modellbasierte Systeme benötigen eine effiziente Verdachtgenerierung. Aber auch heuristische (je nach verwendeter Kontrollstrategie und Komplexität) und fallbasierte Diagnosesysteme (bei großer Falldatenbank) verwenden zum Teil Zusatzwissen, um den Suchraum entsprechend einzuengen.

- *Vorwärtsregeln*
  Mittels Vorwärtsregeln werden Assoziationen zwischen Symptomen und Diagnosen repräsentiert.

- *Indexing*
  Dieses Verfahren kann bei der fallbasierten Diagnose angewendet werden. Dabei wird aus „relevanten“[1] Fallmerkmalen ein Index aufgebaut. Zugriffe auf potentiell ähnliche Fälle erfolgen dann über diesen Index. Für die Gestaltung des Index und die Anordnung der Falldatenbasis gibt es zahlreiche Möglichkeiten, die hier jedoch nicht weiter erörtert werden sollen. Für detailliertere Informationen sei auf die verschiedenen Techniken aus [Wer89],[Leb87], [BM88], [DM86], [Ham86] verwiesen.

[1] Das können Merkmale des Eingabefalles sein, die für die Diagnose essentiell erscheinen.

- *Verdachtgenerierungstechniken bei modellbasierten Diagnosesystemen*
  - *Bildung minimaler Treffermengen*
    Für jeden beobachteten Endzustand (Symptom) wird eine sogenannte „Konfliktmenge" berechnet. Diese Menge enthält alle seine potentiellen Anfangszustände (Ursachen). Aus diesen Konfliktmengen werden minimale Treffermengen[2] gebildet. Eine Treffermenge enthält aus jeder Konfliktmenge mindestens ein Element. Jede Treffermenge kann also sämtliche Symptome erklären. Somit bildet jede minimale Treffermenge einen Verdacht, der zu überprüfen ist.
    Das Verfahren wird komplexer, wenn man die Parameterwerte der einzelnen Zustände einbezieht.
  - *Rückwärtsschließen mit Berücksichtigung der Parameterwerte*
    Diese Methode wird insbesondere bei großer Verzweigungsrate des Modells ineffizient, da die Menge der Zustände, die als Begründung für einen Zustand fungieren können, bei Einbezug der Parameterwerte sehr groß werden kann. Beispiel: *Über einen Addierer ist die Zahl 5 ermittelt worden. An den Eingängen können folgende Zahlenkombinationen (nur positive ganze Zahlen werden betrachtet) angelegen haben: 0 und 5, 1 und 4, 2 und 3, 3 und 2, 4 und 1, 5 und 0.*
  - *Vereinfachung des Modells*
    Das Modell des defekten Systems wird verändert, indem Zustände eliminiert und betroffene Beziehungen abstrahiert werden. Mit dem so vereinfachten Modell wird eine der beiden vorherigen Techniken der Verdachtgenerierung angewendet und damit deren Effizienz gesteigert.
  - *Repräsentation von Zusatzwissen*
    - *Verwendung heuristischer Regeln*
      Man kann z. B. die anfangs erwähnten Vorwärtsregeln verwenden, um eine auf Erfahrungswerten aufgebaute Verdachtgenerierung durchzuführen. Damit hat man natürlich kein rein modellbasiertes System mehr, sondern schon ein Hybrid aus modellbasierter und heuristischer Diagnose.
    - *Verwendung von Statistiken über die Häufigkeit der Fehlerursachen*
      Die Fehlerursachen werden in der Reihenfolge ihrer Häufigkeit verdächtigt. Diese Methode hat den Nachteil, daß sie nicht sehr zielgerichtet ist, da sie die fallspezifischen Symptome bei der Verdachtgenerierung nicht in Betracht zieht.

    Neben den hier genannten Arten der Verwendung von Zusatzwissen können natürlich auch andere Wissensarten bzw. Diagnosesystemklassen zur Verdachtgenerierung herangezogen werden. Einige der sich dadurch ergebenden Hybride werden in Kapitel 3.2 vorgestellt.

3.1.4 Verrechnungsschemata für Wahrscheinlichkeiten

Sobald ein Diagnosesystem mit unsicherem Wissen umgeht, benötigt es ein Schema, das angibt, wie die auftretenden Wahrscheinlichkeiten behandelt werden müssen. Ein solches Verrechnungs-

[2] Eine Treffermenge ist minimal, wenn sie keine Obermenge einer anderen Treffermenge ist.

schema kommt insbesondere bei heuristischen und statistischen Diagnosesystemen zum Einsatz. Bei heuristischen Diagnosesystemen werden in der Regel Schemata benutzt, die mathematisch nicht korrekt sind (sogenannte *heuristische Verrechnugsschemata*), das heißt die Art, wie sie Wahrscheinlichkeiten verknüpfen, ist streng genommen falsch. Das kann daran liegen, daß unzulässig vereinfacht wird, daß notwendige Vorraussetzungen für die Korrektheit vernachlässigt werden, oder daß einfach „intuitiv" mit Wahrscheinlichkeiten umgegangen wird.
Ein Beispiel für eine Voraussetzung, die von heuristischen Verrechnungsschemata häufig nicht eingehalten wird, ist die Unabhängigkeit von Symptomen untereinander. Wenn zwei Symptome auf eine Diagnose hinweisen, so werden die entsprechenden Teilevidenzen zu einer Gesamtevidenz verrechnet, die größer ist, als das Maximum der Teilevidenzen. Besteht jedoch eine starke Korrelation zwischen den Symptomen, so ist der Wert des zweiten Symptoms bei der Anwesenheit des ersten Symptoms für die Diagnose nicht mehr von großer Bedeutung. Das Schema verrechnet es aber unabhängig vom ersten Symptom, wodurch die Diagnosewahrscheinlichkeit zu stark angehoben wird.
Da jedoch häufig Unkorrektheiten in Kauf genommen werden können (weil man z. B. mehr an qualitativen Aussagen interessiert ist), die Vorraussetzungen zur Anwendung eines korrekten Schemas nicht erfüllbar sind, oder die Art der Verknüpfung intuitiv vernünftig erscheint und seine Verwendung effizient ist, werden solche Schemata dennoch oft verwendet.
Ein mathematisch korrektes Verrechnungsschema nennt man auch *mathematische Evidenztheorie* oder *statistisches Verrechnungsschema.* Diese Schemata werden von statistischen Diagnosesystemen benutzt.

Bevor ein heuristisches Verrechnungsschema vorgestellt wird, soll kurz auf die Objekte der Wissensbasis eingegangen werden, die üblicherweise mit Unsicherheiten belastet sein können.
Unsicherheiten können bei der Eingabe der Symptome bestehen, da verschiedene Menschen gleiche Situationen unterschiedlich beurteilen und sich hin und wieder unsicher in ihren Wahrnehmungen sind. Daher kann einem Symptom bei der Eingabe eine Wahrscheinlichkeit zugeordnet werden, die aussagt, wie sicher seine tatsächliche Existenz ist.
Ferner können Assoziationen zwischen Symptomen und Diagnosen mit Unsicherheit belastet sein, das heißt trotz der Existenz der geforderten Symptome trifft die Diagnose nicht notwendigerweise zu. Diese Unsicherheiten bei Symptomen und Diagnosen werden durch das Regelgeflecht propagiert, so daß auch die Enddiagnosen mit Unsicherheiten belastet sind.

Es gibt ungezählte Möglichkeiten, die auftretenden Wahrscheinlichkeiten zu verrechnen. Daher ist es im Rahmen dieser Arbeit nicht sinnvoll, eine Aufstellung heuristischer und statistischer Verrechnungsschemata zu präsentieren. Deshalb erfolgt an dieser Stelle nur die Angabe eines bekannten heuristischen Schemas. Weitere Beispiele für Verrechnungsschemata erfolgen bei der Vorstellung der Expertensysteme MED2 (Kapitel 3.2.1) und PROSPECTOR (Kapitel 3.2.11). Ein Beispiel für ein statistisches Schema wurde schon im vorherigen Kapitel mit dem Theorem von Bayes aufgeführt.

Beispiel: Das Verrechnungsschema von Mycin

Der Erklärung des Schemas liegen Regeln der Art $S_1 \wedge \ldots \wedge S_n \longrightarrow D$ oder $S_1 \vee \ldots \vee S_n \longrightarrow D$

zugrunde. Jedes Symptom S_i ist mit einer Wahrscheinlichkeit PS_i assoziiert, jede Diagnose D_k mit einer Wahrscheinlichkeit PD_k und jede Regel R_j mit einer Regelwahrscheinlichkeit PR_j.

Sind Symptome innerhalb der Vorbedingung mit „$\wedge$" verknüpft, so wird die Gesamtwahrscheinlichkeit des Zutreffens der Vorbedingung als das Minimum der Einzelwahrscheinlichkeiten angenommen. Sind die Symptome mit „$\vee$" verknüpft, so wird die Gesamtwahrscheinlichkeit des Zutreffens der Vorbedingung als das Maximum der Einzelwahrscheinlichkeiten angenommen.

Die Wahrscheinlichkeit für eine Diagnose D_k innerhalb einer Regel R_j errechnet sich durch Multiplikation der Gesamtwahrscheinlichkeit der Vorbedingung mit der Regelwahrscheinlichkeit PR_j.

Liegt eine Diagnose D_k mit einer Wahrscheinlichkeit $PD_{k_{alt}}$ nach bisherigem Inferenzverlauf vor, und schließt eine weitere Regel auf dieselbe Diagnose mit der Wahrscheinlichkeit $PD_{k_{neu}}$, so berechnet sich die neue Wahrscheinlichkeit der Diagnose gemäß folgender Formel:

$$PD_k = \begin{cases} PD_{k_{alt}} & : \; falls\, PD_{k_{neu}} < 0.2 \\ PD_{k_{alt}} + PD_{k_{neu}} * (1 - PD_{k_{alt}}) & : \; sonst \end{cases}$$

Liegen für eine Diagnose sowohl positive Regeln (Regeln, die auf das Vorhandensein der Diagnose schließen), als auch negative Regeln (Regeln, die auf das Nicht–Vorhandensein der Diagnose schließen) vor, so wird die Gesamtwahrscheinlichkeit der Diagnose berechnet, indem die Wahrscheinlichkeit, die von den negativen Regeln errechnet wurde, von der Wahrscheinlichkeit, die von den positiven Regeln errechnet wurde, abgezogen wird.

3.1.5 Behandlung von Rückkopplungsschleifen

Man spricht von Rückkopplungsschleifen, wenn ein Zustand über Zwischenzustände auf sich selbst wirkt. Dabei unterscheidet man 3 Typen:

- *Positive Rückkopplungsschleifen*
  Die Änderungen eines Zustandes verstärken sich selbst.
- *Negative Rückkopplungsschleifen*
  Die Änderungen eines Zustandes schwächen sich selbst ab.
- *Oszillierende Rückkopplungsschleifen*
  Positive und negative Änderungen wechseln sich ab.

Grundsätzlich kommen Rückkopplungsschleifen in technischen Anwendungsbereichen weniger oft vor als bei biologischen Systemen. Wenn sie aber im Zusammenhang mit kausalen Modellen[3] vorkommen, und keine Mechanismen zu ihrer Behandlung vorgesehen sind, so würde der diagnostizierende Algorithmus bei der Simulation in eine Endlosschleife geraten.
Rückkopplungsschleifen, die nicht konvergieren, enthalten Fehler. Sie können durch Setzen

[3]Das Problem der Behandlung von Rückkopplungsschleifen stellt sich natürlich nur bei der modellbasierten Diagnose.

willkürlicher Ober- bzw. Untergrenzen für die betroffenen Parameter festgestellt werden. Negative, nicht konvergierende Rückkopplungsschleifen kommen in der Realität nicht vor. Im Modell jedoch kann es zu solch unangenehmen Phänomenen kommen, so daß das Setzen einer Untergrenze tatsächlich erforderlich wird.

Die wichtigsten beiden Mechanismen zur Behandlung von Rückkopplungsschleifen sind:

Approximation: Bei jedem Schleifendurchgang werden die neuen Werte der entsprechenden Parameter mit den alten Werten verglichen. Unterschreitet die Differenz einen bestimmten Schwellwert, wird der weitere Durchlauf der Schleife gestoppt und der so approximierte Wert weitergegeben. Durch Veränderung des Schwellwertes kann ein sinnvoller Kompromiß zwischen Effizienz und Genauigkeit der Berechnung eingestellt werden. Um nicht konvergierende Rückkopplungsschleifen zu erkennen, können die oben schon erwähnten Grenzen aufgestellt werden. Damit ist die Termination der Schleife sichergestellt.

Vorberechnung: Unter der Voraussetzung negativer Rückkopplungsschleifen und linearer Beziehungen zwischen den Parametern kann man die Rückkopplungsschleifen vorberechnen und die resultierende Abschwächung vorwegnehmen. Da sich nicht-lineare Beziehungen oft durch abschnittsweise lineare Beziehungen approximieren lassen, ist es häufig möglich, mit diesem Verfahren das Approximationsverfahren, das (relativ zu diesem Verfahren) ineffizient ist, zu ersetzen.

3.1.6 Distanz- und Ähnlichkeitsmaße

Die Aufgabe der fallbasierten Diagnose besteht darin, einen zum vorliegenden Fall möglichst ähnlichen Vergleichsfall zu finden.
Zur Ermittlung der Ähnlichkeit zweier Fälle werden Distanz- oder Ähnlichkeitsmaße verwendet.

Distanzmaße

Mittels eines Distanzmaßes läßt sich der Abstand zwischen zwei Objekten berechnen. Je geringer dieser Abstand ist, um so ähnlicher sind sich die beiden Objekte (bzw. Fälle) maßgeblich des Distanzmaßes.
Es folgen ein paar Beispiele bekannter Distanzmaße.

Gegeben seien die Fälle X und Y mit den Merkmalsvektoren $x = (x_1, \ldots, x_n)$ und $y = (y_1, \ldots, y_n)$ mit $x_i, y_i \in \mathcal{R}$:

Der Abstand nach Euklid:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Dieses Maß ist durch die Quadrierung der Differenzen sensitiv gegenüber großen Unterschieden zwischen zwei Merkmalsausprägungen.

Die Manhattan-Distanz (City-Block Metrik):

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i|$$

Die Manhattan–Distanz ist ein neutrales Maß, da alle Differenzen gleich behandelt werden.

Die Maximum–Metrik:

$$d(x,y) = \max_{i=1...n}\{|x_i - y_i|\}$$

Dieses Maß berücksichtigt nur genau ein Merkmal, nämlich das, bei dem die Differenz der beiden Ausprägungen am höchsten ist.

Ähnlichkeitsmaße

Ähnlichkeitsmaße haben im Gegensatz zu Distanzmaßen keinen uneingeschränkten Wertebereich. Ihr Wertebereich erstreckt sich über das Intervall [0,1]. Je näher der Wert des Ähnlichkeitsvergleiches bei eins liegt, um so ähnlicher sind die Vergleichsfälle durch das Maß bewertet.

Um ein paar bekannte Ähnlichkeitsmaße vorzustellen, wird eine abkürzende Schreibweise eingeführt. Wir gehen im folgenden davon aus, daß es sich bei den Merkmalen der zu vergleichenden Fälle X und Y um binäre Merkmale handelt (hierfür werden sogenannte *symmetrische*[4] Maße angegeben), oder um „pseudo-binäre Merkmale". Mit „pseudo–binären" Merkmalen seien Merkmale gemeint, die beliebig viele Ausprägungen haben, bei denen aber nur zwischen den Zuständen „Merkmal x_i hat Ausprägung a_i" (positive Ausprägung) und „Merkmal x_i hat diese Ausprägung nicht" (negative Ausprägung) unterschieden wird. Wenn zwei „pseudo–binäre" Merkmale positive Ausprägungen haben, so ist dies für die Ähnlichkeit ausschlagkräftiger, als wenn beide Merkmale negative Ausprägungen haben[5] . Aufgrund des asymmetrischen Charakters „pseudo–binärer" Merkmale werden zu ihrer Verarbeitung asymmetrische Maße (Definition analog zu symmetrischen Maßen) angegeben.

Die in den folgenden Formeln verwendeten Buchstaben bedeuten:

a: Anzahl der übereinstimmenden positiven Merkmalsausprägungen.

d: Anzahl der übereinstimmenden negativen Merkmalsausprägungen.

b: Anzahl der nicht übereinstimmenden Merkmalsausprägungen, bei denen das Merkmal des ersten Falles eine positive Ausprägung hat, das entsprechende Merkmal des zweiten Falles eine negative Ausprägung.

c: Anzahl der nicht übereinstimmenden Merkmalsausprägungen, bei denen das Merkmal des ersten Falles eine negative Ausprägung hat, das entsprechende Merkmal des zweiten Falles eine positive Ausprägung.

Der Zusammenhang wird in der folgenden Kontingenztafel noch einmal zusammengefaßt:

x/y	1	0
1	a	b
0	c	d

Es folgt die Angabe einiger Ähnlichkeitsmaße:

[4] Positive und negative Übereinstimmungen von Merkmalsausprägungen werden gleichgewichtet.

[5] Beispiel: *„Pseudo–binäres Merkmal: Autofarbe ist blau.*
Haben zwei Autos eine blaue Farbe, so ist diese positive Ausprägung relevanter für ihre Ähnlichkeit, als wenn beide nicht blau wären, da die negative Übereinstimmung nicht ausdrückt, daß beide Autos die gleiche Farbe haben.

Der Simple-Matching-Koeffizient (symmetrisch):

$$sim(x,y) = \frac{a+d}{a+b+c+d}$$

Hierbei handelt es sich um ein neutrales Maß, da alle Gruppen gleichstark in den Koeffizienten eingehen.

Der Rogers/Tanimoto-Koeffizient (symmetrisch):

$$sim(x,y) = \frac{a+d}{a+d+2(b+c)}$$

Die nicht übereinstimmenden Merkmalsausprägungen werden stärker gewichtet, als die übereinstimmenden Merkmalsausprägungen.

Der S-Koeffizient (asymmetrisch):

$$sim(x,y) = \frac{a}{a+b+c}$$

Es fallen nur die Merkmalspaare ins Gewicht, bei denen mindestens eine der beiden Ausprägungen positiv ist.

Der Ochial-Koeffizient (asymmetrisch):

$$sim(x,y) = \frac{a}{\sqrt{(a+b)(a+c)}}$$

Es gilt das Gleiche wie beim vorherigen Koeffizienten. Zusätzlich spielt der Größenunterschied zwischen b und c eine Rolle. Der Koeffizient wird um so kleiner, je größer b und c sind und je geringer ihre Differenz ist.

3.1.7 Testauswahlmethoden

In Kapitel 2.2 wurde ein Diagnoseverfahren allgemein als eine Kombination von Klassifikation und Testauswahl bezeichnet. Ist die Information, die benötigt wird, um eine Diagnose stellen zu können, noch zu unvollständig, so müssen mittels Testauswahlkomponente Symptome vorgeschlagen werden, die zusätzlich zu erheben sind, und Tests, mit denen diese Symptome erhoben werden können.
Die Vorgehensweise bei der Testauswahl sollte möglichst ökonomisch sein, das heißt es sollten die Symptome erhoben werden, mit deren Kenntnis das Diagnoseverfahren schnell terminiert, und bei denen die Tests zur Symptomerhebung den geringsten Aufwand erfordern.
Die Begriffe „Testauswahl", „Symptomauswahl" und „Merkmalsauswahl" werden im folgenden synonym verwendet, da die Unterschiede für dieses Kapitel nicht relevant sind.

Folgende Methoden der Testauswahl sollen im weiteren Verlauf des Kapitels vorgestellt werden:

- Implizite Methoden

- Regelbasierte Methoden
- Wissensbasierte ad hoc Methoden
- Strategische Testplanung
- Informationstheoretische Methoden
- Fallbasierte und konnektionistische Testauswahl

Implizite Methoden: Wissen zur Testauswahl ist nicht explizit repräsentiert. Unbekannte Symptomwerte werden im Verlauf der Klassifikation automatisch erhoben, wenn der entsprechende Symptomwert benötigt wird. Die zuvor behandelten Kontrollstrategien Vorwärtsverkettung, Rückwärtsverkettung, Hypothesize–and-Test und Establish–Refine beinhalten in ihrer Grundversion eine implizite Testauswahl: Wird eine Regel behandelt, die bisher unbekannte Symptome enthält, so werden diese Symptome automatisch erhoben. Natürlich können an diese Kontrollstrategien auch andere Testauswahlmethoden adaptiert werden.

Regelbasierte Methoden: Die Strategie eines Experten bei der Testerhebung wird über Regeln formalisiert. Diese Regeln geben an, in welcher Situation welche Merkmalsausprägungen angefordert werden sollen.
Die Faktoren, die ein Experte bei der Testauswahl berücksichtigt, wurden somit schon bei der Erstellung des Systems eingearbeitet und werden nicht mehr zur Laufzeit verrechnet. Das später erläuterte MOLTKE 3–System (siehe [Ric91]) verwendet sogenannte Reihenfolgeregeln, mit denen es den hier erläuterten Ansatz zur Testauswahl realisiert.

Wissensbasierte ad hoc Methoden: Diese Methoden setzen die explizite Repräsentierung von Kosten und Nutzen von Symptomerhebungen voraus. Die Verrechnung dieser Faktoren erfolgt über ein ad hoc Verrechnungsschema, das intuitiv vernünftig erscheint, jedoch nicht wissenschaftlich motiviert ist.

Der Nutzen einer Symptomerhebung ergibt sich daraus, wie stark der Suchraum für die Diagnosefindung durch das Symptom eingeschränkt wird, bzw. wie stark sie zwischen den verbliebenen Fehlerhypothesen diskriminiert.
Im einfachsten Fall erhalten die Symptome den größten Nutzenwert, die bei den statistisch am häufigsten vorkommenden Fehlerursachen auftreten. Eine weitere Möglichkeit der Symptomauswahl bestünde darin, zuerst solche Merkmale zu untersuchen, die zur Bestimmung möglichst vieler Fehlerursachen notwendig sind.

Die Kosten einer Symptomerhebung zergliedern sich in folgende mögliche Aufwandsposten:

- *Zeitaufwand*
  Hierzu zählt die unmittelbare Zeit für die Durchführung des Testes, sowie die Vor– und Nachbereitung desselben (z.B. Heranschaffen notwendiger Geräte, Testaufbau, Testabbau).
- *Materialaufwand*
  Hierzu zählt Verbrauch von Material und Energie, sowie Kosten zur Beschaffung von Material und Geräten (Leih– und Kaufgebühr).

- *Gefährlichkeit des Tests*
  Hierbei handelt es sich um eine Beurteilung des Risikos, das dieser Test in sich birgt (Gefährdung für Mensch, Maschine und Meßgerät). Als Kosten werden die möglichen Folgekosten des Tests betrachtet.

- *Zeitkritische Fehler*
  Bei Fehlerursachen, deren Nichtbehebung innerhalb einer kritischen Zeitspanne Folgekosten verursachen würde, bietet sich die schnelle Untersuchung der zugehörigen Symptome zur Vermeidung dieser Kosten an.

- *Qualifikation (Ausbildungsstand) der testdurchführenden Person*
  Unter dem Aspekt, daß die Zeit einer hoch qualifizierten Fachkraft mehr kostet, als die Zeit einer wenig qualifizierten Kraft, ist der Test zur Symptomerhebung umso kostengünstiger, je weniger qualifiziert ein Mensch zu seiner Durchführung sein muß.

Strategische Testplanung: Die Auswahl des nächsten Tests bzw. der gesamten Testfolge übernimmt ein Planungssystem.

Informationstheoretische Methoden: Diese Methoden versuchen, ausgehend von der aktuellen Situation, denjenigen Test vorzuschlagen, der den größten Informationsgewinn verspricht (siehe [dKW87]). Basis dieser Methoden ist die Informationstheorie nach Shannon [Sha48, SW49][6].
Eine Verrechnung von Informationsgewinn und anfallenden Testkosten erfolgt jedoch nicht.

Fallbasierte und konnektionistische Ansätze: Voraussetzung für die fallbasierte Testauswahl ist das Vorhandensein einer fallbasierten Klassifikationskomponente. Mittels dieses Diagnosesystems wird ein zur aktuellen Situation ähnlicher Fall aus einer Falldatenbasis herausgesucht. Der Fall muß so repräsentiert sein, daß die Reihenfolge, in der die Symptome erhoben wurden, ersichtlich ist. Vorgeschlagen wird das Symptom, welches gemäß dieses Vergleichsfalles „an der Reihe“ ist.
Fallbasierte Ansätze behandeln somit die Testauswahl als Klassifikationsaufgabe.

Konnektionistische Ansätze verhalten sich ähnlich den fallbasierten Ansätzen. Der Unterschied besteht hauptsächlich darin, daß konnektionistische Ansätze bei einer Entscheidung, welches Symptom als nächstes zu erheben ist, nicht auf real existierende Fälle zurückgreifen, sondern auf erlernte Assoziationen von Situationen mit einem jeweils in der Situation erhobenen Symptom.

3.1.8 Mechanismen zur Rücknahme von Schlußfolgerungen

Ein Diagnosesystem muß in der Lage sein, Schlußfolgerungen oder Fakten (Symptome) zurückziehen zu können.
Die wichtigsten Gründe für eine solche Rücknahme sind:

1. Der Benutzer hat sich bei der Erhebung eines Symptoms geirrt, und muß dessen Vorkommen revidieren.

[6] Shannon beschäftigt sich unter vielem anderem damit, wie hoch der Informationsgehalt einer gegebenen Situation ist.

2. Es wurden gewichtige Gründe (z.B. negative Evidenz) bekannt, die gegen eine schon etablierte Diagnose sprechen.

3. Es wird eine Ausnahme einer Regel bekannt. Der fälschlicherweise vorher gezogene Schluß muß aufgrund dieser Ausnahme zurückgezogen werden.

Es reicht natürlich nicht aus, das betroffene Datum einfach aus dem Arbeitsspeicher zu löschen, da das System basierend auf den falschen Werten andere Objekte gefolgert haben kann, die ihrerseits zur Folgerung weiterer Objekte beigetragen haben können, etc. .
Es ist demnach erforderlich, sämtliche im Zusammenhang mit den revidierten Daten getätigten Inferenzen auf ihre Gültigkeit unter den neuen Bedingungen hin zu überprüfen.

Aus der Literatur sind einige Mechanismen zur Rücknahme von Schlußfolgerungen bekannt. Die bekanntesten sollen hier kurz vorgestellt werden:

Chronologisches Backtracking: Die gezogenen Schlüsse werden in umgekehrter Reihenfolge bis zu der Stelle zurückverfolgt und ungültig gemacht, an der das zurückgezogene Objekt das erste Mal verwendet wurde.
Dieses Verfahren ist natürlich denkbar ineffizient, da alle Schlüsse zurückgenommen werden; unabhängig davon, ob sie von dem ursprünglich zurückgezogenen Objekt abhängig waren, oder nicht.

TMS (Truth Maintenance System) [Doy79], [McA80], [Goo82]: Für jede Schlußfolgerung werden ihre direkten Begründungen abgespeichert. Eine dieser Begründungen, die nicht zirkulär[7] ist, wird als sogenannter *Current Support* für eine Schlußfolgerung ausgezeichnet.

Die Rücknahme eines Datums durch ein TMS läßt sich in zwei Schritte gliedern:

1. Zunächst werden alle Begründungen daraufhin überprüft, ob das zurückgezogene Datum in ihnen enthalten ist. Für jede betroffene Begründung wird festgestellt, ob sie der Current Support einer Schlußfolgerung ist. Derartige Schlußfolgerungen werden zeitweise als „unsicher" markiert, und dieser Schritt wird mit ihnen als zurückgezogenem Datum wiederholt.

   In diesem Schritt werden also sämtliche von der Änderung direkt oder indirekt betroffene Schlußfolgerungen markiert.

2. Jede markierte Schlußfolgerung wird daraufhin überprüft, ob andere Begründungen für sie gefunden werden können. Wenn ja, dann wird die Schlußfolgerung demarkiert, wenn nicht, so wird sie aus der Wissensbasis entfernt.

Die Effizienz dieses Algorithmus hängt von der Menge der Schlußfolgerungen ab, die markiert wurden. Diese Menge wiederum hängt vom Vernetzungsgrad des Symptom–Diagnose–Netzwerkes ab. Damit wird der TMS–Mechanismus für große Diagnoseanwendungen zu ineffizient.

[7] Beispiel für eine zirkuläre Ableitungskette: $A \longrightarrow B, B \longrightarrow C, C \longrightarrow A$

Ein weiterer Nachteil eines TMS macht sich bei der Verwendung unsicherer Regeln bemerkbar. Jede mögliche Begründung für die Existenz eines Datums bedeutet eine Verstärkung der Evidenz für dieses Datum. Damit bilden alle Begründungen einer Schlußfolgerung ihren Current Support, was dazu führt, daß wesentlich mehr Schlußfolgerungen als unsicher markiert werden, als im Falle sicherer Regeln.

ATMS (Assumption–Based TMS) [dK84], [dK86] : Der wesentliche Unterschied zwischen einem ATMS und einem TMS besteht in der Art der Begründung, die für jedes gefolgerte Objekt nachgehalten wird. Während das TMS direkte Begründungen von Schlußfolgerungen abspeichert, führt ein ATMS Begründungen für ein gefolgertes Datum immer auf Basisannahmen (Fakten bzw. Symptome) zurück. Jedem Datum wird eine (möglichst redundanzfreie) Menge von Basisannahmen, die als Grundlage für die Gültigkeit des Datums fungieren, zugeordnet.
Werden Basisannahmen ungültig, so müssen die Begründungsmengen der einzelnen Schlußfolgerungen mit der Menge der momentan gültigen Basisannahmen verglichen werden.
Die bloße Rücknahme eines Datums ist somit wesentlich unkomplizierter als beim TMS. Dafür steigt der Verwaltungsaufwand für das Abhängigkeitsnetz mit der Anzahl der Basisannahmen (De Kleer stellt fest, daß das ATMS bei 1000 Basisannahmen noch praktikabel ist).
Schwierig ist die Behandlung von unsicherem Wissen, da die Rückführung einer Begründung auf Basisannahmen Propagierungsmechanismen für Wahrscheinlichkeiten erfordert, und die Zahl der Basisannahmen pro Begründung ansteigt.
Eine weitere Schwierigkeit entsteht, wenn statt einer Basisannahme der Schluß einer Regel zurückgezogen werden muß (eine Ausnahme von der Regel wird bekannt). Damit können die Schlußfolgerungen, die auf diesem zurückgezogenen Schluß beruhen, nicht mehr auf Basisannahmen zurückgeführt werden, da sie möglicherweise ungültig geworden sind, obwohl keine Basisannahme zurückgezogen wurde. Für solche Fälle muß eine Sonderbehandlung vorgesehen sein.

ITMS (Immediate Check TMS): Die Idee dieses Mechanismus ist die Verbesserung des TMS, indem die Gültigkeit einer Schlußfolgerung direkt überprüft wird, anstatt daß sie erst einmal vorläufig markiert wird. Erst wenn festgestellt wird, daß die Schlußfolgerung tatsächlich nicht mehr begründet ist, wird die Änderung weiter propagiert.
Für diese direkte Überprüfung auf Gültigkeit muß statt des Current Support ein besserer Indikator benutzt werden. Der Indikator einer Schlußfolgerung könnte z. B. die Summe ihrer nicht zirkulären Begründungen sein. (Zirkuläre Begründungen können schon in der Entstehungsphase der Wissensbasis erkannt und entsprechend gekennzeichnet werden[8]).

Der folgende Algorithmus ist (leicht modifiziert) [Pup87] entnommen.

```
Eingabe:  Zurückzuziehendes Objekt.
Ausgabe:  Korrigierte Wissensbasis.

  1. Revision bzw.  Ausführung aller unmittelbar betroffenen Regeln (wenn sich
     ein Faktum oder eine Schlußfolgerung geändert hat, werden alle Regeln,
     die den alten Wert benutzen, zurückgezogen, und alle Regeln, die auf den
```

[8]Dies ist nur dann möglich, wenn die betroffenen Regeln der Wissensbasis keine Variablen enthalten (was in der Regel der Fall ist).

```
   neuen Wert zugreifen, aktiviert).
   Die Rücknahme einer Regel wird als die Ausführung der Regel mit inverser
   Aktion durchgeführt (hierbei erfolgt eine Modifikation des Indikators der
   von der Rücknahme betroffenen Objekte).
2. Alle von diesen Regeländerungen betroffenen Objekte werden auf eine
   AGENDA gesetzt.
3. FALLS AGENDA leer,
   DANN STOP
   SONST Selektion eines Objektes X von AGENDA, von dem kein anderes Objekt
   von AGENDA direkt abhängt.
4. Lösche X von AGENDA.
5. FALLS die Änderungen von Schritt 1 die Gültigkeit von X veraendert haben
   (dies kann anhand des Indikators überprüft werden),
   DANN GOTO 1
   SONST GOTO 3.
```

Durch die sofortige Überprüfung mittels eines Indikators wird der ITMS-Mechanismus auch für unsichere Regeln anwendbar. Der Wertebereich des Indikators erstreckt sich in diesem Fall über das Intervall [0..1].
Die Effizienz des Algorithmus ist durch die Sofortentscheidung über die Gültigkeit eines Objektes und die Vorberechnung der Zirkularitäten so hoch, daß er sich auch für große Anwendungsbereiche mit vernetzten Wissensbasen eignet.
Durch diese Vorteile erscheint er den anderen vorgestellten Mechanismen zumindest in Bezug auf das Anwendungsgebiet Diagnostik überlegen.

3.1.9 Diagnosebewertung

Betrachtet man das allgemeine 3-Phasen-Diagnoseverfahren (Hypothesengenerierung, Hypothesenüberprüfung, Hypothesendiskriminierung), so muß jede generierte Hypothese während oder nach der Phase der Hypothesenüberprüfung hinsichtlich ihrer Brauchbarkeit (ihres Wahrheitsgehaltes) bewertet werden. Hierzu gibt es verschiedene Bewertungskriterien, die zu einer Gesamtbewertung miteinander kombiniert werden können.

Ein Kriterium, das zur Diagnosebewertung benutzt wird, wird durch die Menge der Symptome ausgemacht, die zu der Hypothese „gehören" (also üblicherweise bei dieser Fehlerursache auftreten) und gleichzeitig bei dem zu diagnostizierenden fehlerhaften Gerät auftreten. Diese Symptome nennt man auch die *erwarteten und beobachteten* Symptome.
Die zweite Symptommenge, die zur Diagnosebewertung hinzugezogen wird, ist die Menge der Symptome, die von der Hypothese erwartet werden, beim fehlerhaften Gerät aber nicht auftreten; also die Menge der Symptome, die *erwartet, aber nicht beobachtet* werden.
Die dritte relevante Symptommenge schließlich besteht aus den Symptomen, die am fehlerhaften Gerät auftreten, von der Hypothese jedoch nicht vorgesehen sind. Dies ist die Menge der Symptome, die *nicht erwartet, aber beobachtet* werden.
Zusammenfassend seien die drei Mengen noch einmal aufgeführt:

- Erwartete und beobachtete Symptome.
- Erwartete, aber nicht beobachtete Symptome.
- Nicht erwartete, aber beobachtete Symptome.

Je größer die Menge der erwarteten und beobachteten Symptome ist, um so mehr Evidenz spricht für die entsprechende Diagnosehypothese.
Je größer die Mengen der erwarteten, aber nicht beobachteten Symptome und der nicht erwarteten, aber beobachteten Symptome sind, um so mehr Evidenz spricht gegen die entsprechende Diagnosehypothese.
Heuristische Diagnosesysteme beziehen insbesondere die ersten beiden Mengen in ihre Bewertung ein, da hauptsächlich auf Symptome zugegriffen wird, die über Regelketten mit Diagnosehypothesen verknüpft sind. Diese Symptome werden im allgemeinen von der Hypothese erwartet.
Statistische Systeme, die auf dem Theorem von Bayes beruhen, betrachten nur Symptome, die beobachtet wurden. Damit können sie nur die erste und dritte Symptommenge in ihre Berechnung einbeziehen (siehe Formel in Kapitel 2.2).
Fallbasierte Diagnosesysteme arbeiten mit allen drei Symptommengen; modellbasierte Diagnosesysteme hingegen bewerten besonders stark die dritte Symptommenge, da es ihnen auf den Erklärungswert einer Hypothese ankommt, d. h. die beobachteten Symptome sollen durch die Hypothese erklärt (hergeleitet) werden können. Gelingt diese Herleitung für manche Symptome nicht, so sind diese nicht erwartet, aber beobachtet, und fallen für die Bewertung der Hypothese stark negativ ins Gewicht.
Sichere Diagnosesysteme repräsentieren keine expliziten Bewertungsverfahren. Wie welche Symptome zu beurteilen sind, wird während ihrer Erstellung festgelegt, und ist von Fall zu Fall verschieden.

Neben den drei erwähnten Symptommengen kann noch ein weiteres Bewertungskriterium in die Bewertung der Diagnosehypothese einfließen: Die sogenannte *Prädisposition*. Darunter versteht man die von der statistischen Diagnose her bekannten Apriori–Wahrscheinlichkeiten der Diagnosen.
Apriori–Wahrscheinlichkeiten spielen außer bei statistischen Diagnosesystemen keine so große Rolle für die Diagnosebewertung. Ihr hauptsächliches Einsatzgebiet ist die Verdachtgenerierung.

3.1.10 Differentialdiagnostik

Nachdem die verschiedenen Hypothesen bewertet sind, müssen unter ihnen diejenigen herausgesucht werden, die etabliert werden sollen. Um diese Entscheidung zu treffen, wird häufig die sogenannte *Differentialdiagnostik* angewandt.
Allgemein bezeichnet man mit Differentialdiagnostik den sorgfältigen Vergleich der gemäß den Bewertungskriterien wahrscheinlichsten Diagnosehypothesen und den Versuch, die beste Hypothese auszuwählen. Man kann die Differentialdiagnostik auch als Bewertungsmethode betrachten, die eine schon absolut bewertete Diagnose relativ zu anderen Diagnosen — den sogenannten Differentialdiagnosen — neu bewertet.

In der Literatur findet man eine weitere Definition der Differentialdiagnostik. Danach besteht die Idee der Differentialdiagnostik darin, leicht verwechselbare Diagnosen mit ähnlicher Symptomatik relativ zueinander zu bewerten, und unter ihnen die beste Diagnose zu etablieren, wenn sie erheblich besser als die anderen bewertet wird.

Eine Aufgliederung des Begriffs Differentialdiagnostik in drei verschiedene Strategien wird von Peter Jackson in seiner Darstellung des INTERNIST-Diagnosesystems beschrieben ([Jac87]). Das System bewertet die alternativen Strategien in Abhängigkeit von der Zahl der Hypothesen, die es momentan verdächtigt.

- Wenn es mehr als vier Hypothesen gibt, wendet es eine *Refutationsstrategie* an, und versucht, so viel wie möglich zu eliminieren, indem es Fragen über Symptome stellt, die deutlich auf die Korrektheit einer Hypothese hinweisen.
- Gibt es zwischen zwei und vier Hypothesen, so wird eine *Differenzialstrategie* angewendet, die Fragen stellt, die helfen, eine Entscheidung zwischen den Hypothesen zu treffen.
- Wenn es nur einen Kandidaten gibt, wird eine *Verifikationsstrategie* angewandt. Es werden Fragen gestellt, die das Zutreffen dieser Hypothese bestätigen sollen.

Man erkennt an den verschiedenen Definitionen, daß die Differentialdiagnostik eng mit der Testauswahlkomponente verzahnt ist. Beides zusammengenommen ergibt die Phase der Hypothesendiskriminierung.

Da in den folgenden Kapiteln der Begriff Differentialdiagnostik nicht jedesmal näher spezifiziert werden soll, wollen wir darunter die erste allgemeine Definition verstehen, wenn keine weiteren Erläuterungen vorhanden sind.

3.2 Existierende Diagnose–Expertensysteme

Nachdem eine Klassifizierung und Wertung der möglichen Diagnosearten vorgenommen wurde, und eine Untergliederung des Verfahrens in seine Einzelteile erfolgte, sollen nun existierende Systeme hinsichtlich ihres Verfahrens vorgestellt werden. Dabei werden viele der theoretischen Konzepte aus 2.2 und 3.1 in einer praktischen Realisierung gezeigt. Zusätzlich tauchen interessante neue Konzepte auf, die bisher nicht behandelt worden sind. U. a. werden ein paar Hybride vorgestellt, die versuchen, die Vorteile mehrerer Wissensarten zu kombinieren, und dadurch die Nachteile der einzelnen Wissensart zu vermeiden.
Die folgende Tabelle gibt einen Überblick über die Systeme, die in diesem Abschnitt behandelt werden.

System	Jahr	Kategorie	Merkmale
MED2	1986	heuristisch	Hypothesize–and–Test, Notfalldiagnostik, ITMS, viele Bewertungskriterien, Zeitbehandlung, Questionsets
MegaFilex	1989	heuristisch	Hypothesize–and–Test, Testhierarchien, Tool S.1
Xfrac	1988	heuristisch	Divide–and–Conquer, Hypothesize–and–Test, Tool S.1, zwei Konsultationsmodi, MYCIN–Verrechnungsschema
Z–11	1989	heuristisch	Backward–Reasoning, Verarbeitung von Unschärfen
QUASIMODIS	1990	modellbasiert	funktional, dynamisch, qualitativ, keine Hypothesengenerierung, Testauswahl unter Einbezug von Nutzen und Aufwand
MIMIC	1989	modellbasiert	funktional, dynamisch, qualitativ, online–Überwachung, Regeln zur Verdachts–generierung, Simulation mit QSIM
System von Davis	1984	modellbasiert	funktional, statisch, quantitativ, hierarchisch, Zusatzwissen zur Hypothesengenerierung
ARTEX	1989	modellbasiert	statisch, heuristische Aspekte, selbständige Durchführung von Tests
ESCORT	1985	modellbasiert	pathophysiologisch, prozeßorientiert, dynamisch, online–Überwachung
PATDEX	1989	fallbasiert	case–matching–System, Erfahrungsgraph, Ähnlichkeitsmaß
PATDEX/2	1990	fallbasiert	case–matching–System, fallbasierte Testauswahl, Ähnlichkeitsmaß, Gewichtung von Symptomen
PROSPECTOR	1979	statistisch	Bayes'sche Netze, heuristische Aspekte, drei Aktionsmodi
DIWA	1991	sicher	Entscheidungsbaum, zuladbare Module, Reihenfolgeknoten, Rücksprungknoten, Wartungsausgänge
System zur Autodiagnose -	1988	Hybrid	heuristisch, funktional–modellbasiert, objektorientiert, Branch–and–Bound, Kontexthierarchie, MYCIN–Verrechnungs–schema
MOLTKE 3	1991	Hybrid	heuristisch, modellbasiert, fallbasiert (PATDEX/2), Establish–Refine, Reihenfolgeregeln zur Testauswahl, Kontextheterarchie
SECQ	1991	Hybrid	heuristisch (zur Makrodiagnose), neuronales Netz (zur Mikrodiagnose)

Die Vorstellung der Systeme kann nicht unter einheitlichen Gesichtspunkten erfolgen, da die verschiedenen Anwendungsdomänen verschiedene Voraussetzungen haben und unterschiedliche Erwartungen an ein Diagnoseverfahren stellen. Werden bei Systemen Aspekte weggelassen, die bei anderen Systemen diskutiert wurden, so liegt dies in der Regel daran, daß sie entweder in diesem Rahmen als nicht so wichtig erachtet werden, oder die uns zur Verfügung stehende Information über das System in dieser Hinsicht keine Aussage zuläßt.
Teilweise werden die Verfahren auch bewußt etwas vereinfacht dargestellt, um sich nicht in für diese Betrachtung unnötigen Details zu verlieren.

3.2.1 MED2

MED2 ist eine Diagnoseshell, die in vielfältigen, sowohl technischen als auch medizinischen Anwendungsbereichen eingesetzt wird. Sie gehört in die Klasse der heuristischen Diagnosesysteme (bzw. Diagnoseshells). Den Kern des realisierten Verfahrens bilden die Hypothesize–and–Test–Strategie und die Bewertungskomponente mit anschließender Differentialdiagnostik. Weitere wichtige Komponenten des Verfahrens sind die Belief–Revisionskomponente und die Notfalldiagnostik.

Das Verfahren

Der in Abbildung 3.2 dargestellte Algorithmus soll kurz erläutert werden. MED2 beginnt mit der Abarbeitung der vom Benutzer angegebenen Questionsets (s. u.). Die Questionsets werden erfragt, es erfolgt eine Plausibilitätskontrolle der erfragten Daten, und anschließend werden die Eingabedaten aufbereitet. Jetzt greift die Hypothesize–and–test–Strategie: Verdächtige Diagnosen werden ermittelt und anschließend überprüft. Mittels Differentialdiagnostik werden die immer noch verdächtigen Diagnosen relativ zueinander bewertet. Werden dabei weitere Questionsets indiziert, so wird unter ihnen die Menge ausgesucht, die zuerst vom Benutzer erfragt werden soll. Danach beginnt der Zyklus von vorn.
Müssen Daten aufgrund zusätzlicher Informationen zurückgezogen werden, so greift die Belief–Revisions–Komponente (ein ITMS).
Sind keine Questionsets mehr indiziert, so erfolgt eine Plausibilitätskontrolle der etablierten Diagnosen. Außerdem werden Therapievorschläge unterbreitet. War die Therapie nicht erfolgreich, so können Folgesitzungen durchgeführt werden, wobei die alte Diagnose verworfen wird, und möglicherweise Änderungen in der Symptomatik verarbeitet werden müssen (Belief–Revisionskomponente).

Einige Aspekte von MED2 sollen näher erläutert werden:

Das Questionset–Konzept: Bei einer Untersuchung eines technischen Gerätes oder eines Menschen fallen pro Test gewöhnlich mehrere Daten an. MED2 vereinigt alle Fragen, die sich auf Daten beziehen, die durch einen Test erhoben werden können, in einem Questionset.
Wird im Verlaufe der Diagnose dieser Questionset ausgewählt, so werden sämtliche Fragen des Sets gestellt.

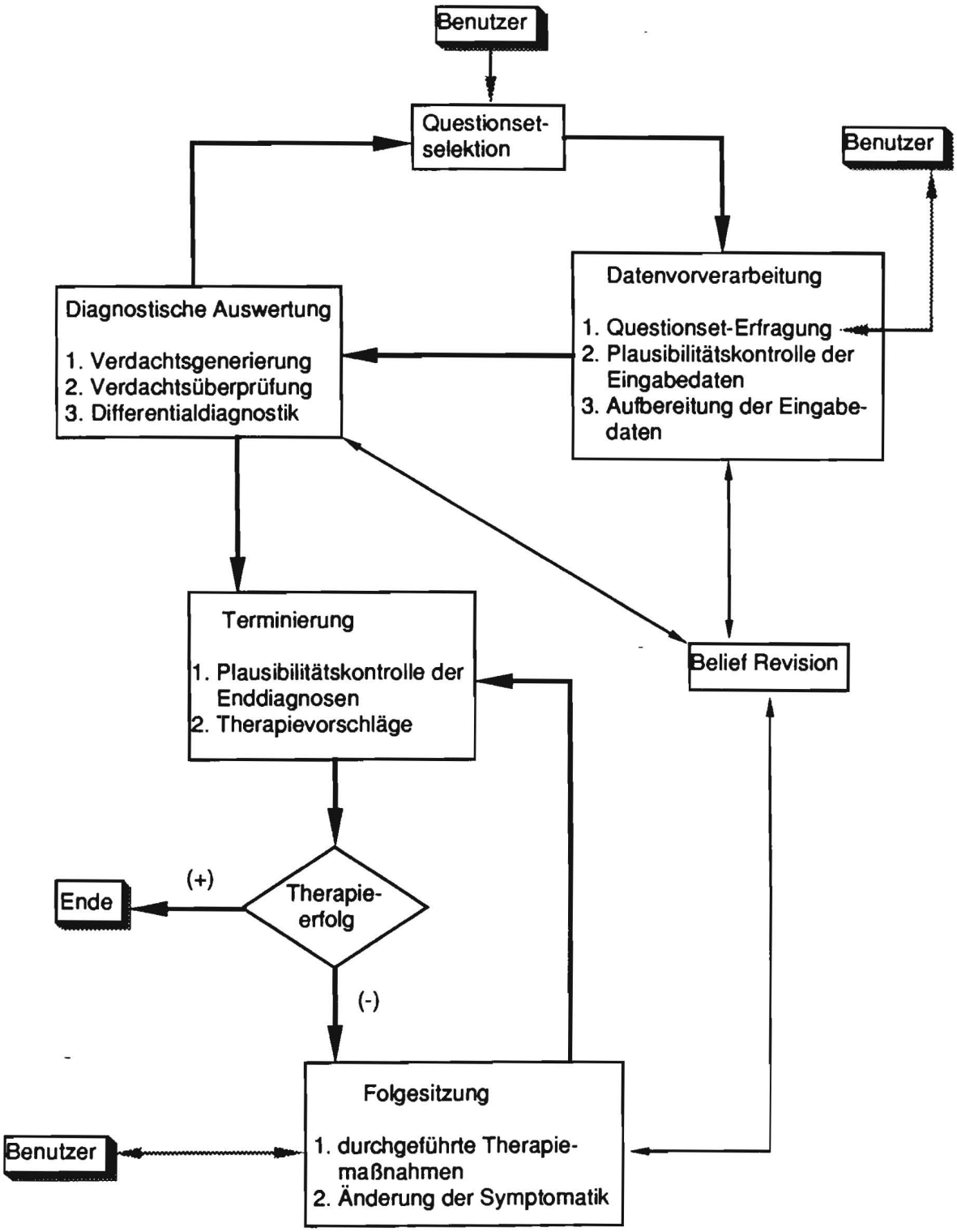

Abbildung 3.2: Das Diagnoseverfahren von MED2

Sind mehrere Questionsets indiziert, so wird nach folgendem Schema ausgewählt:

1. Zuerst werden Questionsets betrachtet, die von kategorischen (sicheren) Regeln indiziert worden sind.
2. Sind keine derartigen Questionsets vorhanden, wird der Questionset ausgewählt, der zur Überprüfung der Diagnosen im Working–Memory (hierin befinden sich verdächtige, aber bisher nicht etablierte Diagnosen) am nützlichsten ist.
3. Befinden sich keine Diagnosen im Working–Memory, so werden Questionsets zur systematischen Erfassung der Symptomatik ausgewählt.

Die Dialogsteuerung muß nicht von MED2 ausgehen. Wird ein Questionset vom Benutzer ausgewählt, so wird er vor den vom System indizierten Questionsets behandelt.

Plausibilitätskontrolle der Eingabedaten: Zur Überprüfung der Konsistenz der Benutzerdaten gibt es in MED2 folgende Mechanismen:

- Einschränkung des Wertebereichs bei Fragen.
- Kennzeichnung von inkonsistenten Kombinationen verschiedener Fragen mit Regeln vom Typ „contra“ (wenn eine solche Regel feuert, wird der Benutzer aufgefordert, einen der Fakten, die zu dieser Inkonsistenz geführt haben, zurückzuziehen).

Verdachtsgenerierung: Eine Diagnose wird im Working–Memory verankert, wenn über Forward–Regeln eine bestimmte Punktsumme überschritten wurde, und dieser Schwellwert nach Aktivierung entsprechender Rückwärtsregeln überschritten bleibt.

Diagnosebewertung: Zunächst unterscheidet MED2 zwischen kategorischer und probabilistischer Bewertung. Kann mittels einer kategorischen Regel auf eine Diagnose geschlossen werden, so wird diese etabliert.
Bei der probabilistischen Diagnosebewertung werden die Kriterien Pro und Kontra verrechnet. Zusätzlich wird eine Diagnoseprädisposition einbezogen. Diese errechnet sich aus verfügbaren Grunddaten und speziellen Parametern (wie diagnosespezifische Risikofaktoren).
Etablierte Diagnosen werden einer Plausibilitätskontrolle unterworfen. Hierzu werden kausale Regeln benutzt, um den Erklärungswert der Diagnose zu überprüfen, d. h. mittels der Diagnose sollen über die Kausalbeziehungen alle aufgetretenen Symptome hergeleitet werden[9].

[9]Da diese kausale Vorgehensweise nur einen sehr geringen Teil der Diagnose ausmacht, kann man MED2 nicht als Hybrid bezeichnen.

Zündkerzen	Strength: 40 (wahrscheinlich)
Bewertung von Zündkerzen	etabliert (44)
notwendige Bedingung	:-
hinreichende Bedingung	:-
Ausschluß	:-
PRO	:wahrscheinlich (40)
KONTRA	:neutral (0)
Erklärungswert	:alle Symptome erklärt (98%)
Prädisposition	:relativ häufig
Differentialdiagnostik	:-
Begründung für PRO	:wahrscheinlich (40)
p4 (20)	weil Kaltlauf = Motor schüttelt Kontext: Zündanlage_Benzin ist etabliert
p4 (20)	weil Kraftstoffverbrauch = erheblich zu hoch Kontext: Zündanlage_Benzin ist etabliert
Begründung der Prädisposition	:relativ häufig
weil Apriori-Wahrscheinlichkeit der Diagnose relativ häufig.	
Begründung für Erklärungswert	:alle Symptome erklärt (98%)
40% der Explanationsets werden durch Zündkerzen erklärt.	
58% der Explanationsets werden durch andere Diagnosen erklärt.	
Zündkerzen erklären:	→ Zündanlage_Benzin ⟶ X-Fahreigenschaften ⟶ X-Kraftstoffverbrauch

Beispiel einer Bewertung durch MED2

Verrechnungsschema: Bei MED2 werden Regelwahrscheinlichkeiten nicht wie bei MYCIN mit Prozentzahlen repräsentiert. Stattdessen wird jede Symptom-Diagnose-Regel in eine Kategorie eingeteilt, die angibt, wie wahrscheinlich die durch die Regel ausgedrückte Assoziation zwischen Symptom und Diagnose ist. Die Kategorien reichen von „selten" bis „fast immer". Jeder Kategorie ist eine stellvertretende Punktzahl zugeordnet. Die Punktzahl einer Kategorie ist etwa doppelt so hoch, wie die Punktzahl der nächsttieferen Kategorie. Für jede Diagnose werden die Punkte, die Regeln mit zutreffender Vorbedingung laut ihrer Kategorieeinteilung haben, addiert (Pro - und Kontra-Regeln werden getrennt behandelt). Damit erhält man eine Grundbewertung.
Danach wird die Prädisposition einer Diagnose errechnet. Dazu werden die Apriori-Wahrscheinlichkeit der Diagnose (die in Form einer Punktkategorie angegeben wird) und die Punktkategorien spezifischer Prädispositions-Regeln summiert.
Die Prädisposition wird mit der Grundbewertung zu einer Gesamtbewertung verrechnet.
Gemäß dieses Wertes wird entschieden, ob eine Diagnose ins Working-Memory aufgenommen, etabliert oder ausgeschlossen wird.
Zum Schluß einer Sitzung überprüft MED2 mittels kausaler Regeln, ob alle Symptome durch die hergeleiteten Enddiagnosen erklärt werden können. Falls das nicht der Fall ist und es Diagnosen gibt, die relativ gut bewertet, aber nicht etabliert sind, und die die noch unerklärten Symptome erklären können, dann werden diese Diagnosen etabliert.

Das genaue Verrechnungsschema kann in [Pup87] auf Seite 140 ff. nachgelesen werden.

Differentialdiagnostik: MED2 betrachtet nur die Diagnosen als Differentialdiagnosen, die bei der Erstellung des Diagnosesystems als solche gekennzeichnet wurden. Dadurch bleibt die Möglichkeit, mehrere Diagnosen zu stellen, erhalten, da innerhalb jeder Menge von Differentialdiagnosen die beste Diagnose ermittelt wird.
Eine Differentialdiagnose wird dann etabliert, wenn sie erheblich besser bewertet wird, als die mit ihr konkurrierenden Differentialdiagnosen, und einen bestimmten Punkte–Schwellwert überschreitet.

Diagnostischer Mittelbau: Er wird in MED2 durch die einfachen Symptomvorverarbeitungen direkt nach der Datenerfassung und durch die Verfeinerung von Grobdiagnosen zu Feindiagnosen (Enddiagnosen) realisiert. Zu diesem Zweck sind neben Symptom–Diagnose–Regeln auch Diagnose–Diagnose–Regeln realisiert.

Nichtmonotonie: Neben den probabilistischen Regeln, die Punkte für oder gegen eine Diagnose verteilen, sind auch nicht–monotone Ableitungen möglich. Zu diesem Zweck sind Regeln mit Ausnahmen repräsentiert. Diese Regeln feuern, wenn ihre Vorbedingungen erfüllt sind und keine der Ausnahmen bekannt ist.
Wird eine Ausnahme erst im Nachhinein bekannt, so muß der falsche Schluß mit allen seinen Konsequenzen revidiert werden. Dazu wird die Belief–Revision–Komponente benutzt.

Einbezug der Zeit: MED2 rechnet sämtliche Zeitbezüge von der vorhergehenden Sitzung zur aktuellen Sitzung hin um. Bei Änderungen der Symptomwerte werden die alten Werte gerettet und Regeln aktiviert, die die Geschichte des Symptoms auswerten.

Notfalldiagnostik: Eine kritische Situation kann in MED2 durch eine hohe Priorität ausgezeichnet sein, die auch bei relativ geringem Verdacht vorrangig untersucht wird. Falls diese Diagnose etabliert wird, werden die entsprechenden Therapiemaßnahmen sofort ausgedruckt. Um die Bearbeitung zusätzlich zu beschleunigen, gibt es spezielle Notfall–Questionsets, die bei Erkennen des Notfalls aktiviert werden, und nur die jeweils wichtigsten Symptome erfassen.

Wertung und Einsatz

Was MED2 besonders auszeichnet, ist die Vielfalt der realisierten Konzepte (Bausteine), die in Kapitel 3.1 vorgestellt worden sind. Auch die allgemeinen Anforderungen aus Kapitel 2.1 wurden alle in Betracht gezogen. Die Fähigkeit, mehrere Fehler zu erkennen, wurde durch die Beschränkung der Differentialdiagnostik auf Gruppen von Diagnosen realisiert (natürlich unterliegt diese Fähigkeit den Einschränkungen, die von der heuristischen Diagnostik allgemein auferlegt sind (siehe Kapitel 2.2)).

Diese Vielfalt gewährleistet eine gewisse Variabilität, die MED2 für unterschiedliche Anwendungsbereiche geeignet erscheinen läßt. Beispiele für die Verwendung von MED2 sind die Expertensysteme TADIS [BW88] (dient zur Schadensfrüherkennung und Fehlerdiagnose bei schnellaufenden Industriegetrieben (Turbogetriebe), die z. B. in Kraftwerken und Schiffen eingesetzt werden), EFFEKT [NP88] (macht Vorschläge zur Behebung von Fehlern bei der Formgebung von Elastomeren (z. B. Gummidichtungen), vor allem beim Spritzgießen) und DAX

[MLE88] (dient als Qualitätssicherungskomponente für die Automatikgetriebefertigung).

3.2.2 MegaFilEx

Das heuristische Expertensystem MegaFilEx (siehe [Kar89]) wird in der Endprüfung der Plattenspeicher des Typs MegaFile[10] der Firma Siemens eingesetzt.
MegaFilEx wurde mit der Expertensystemshell S.1 ([s1]) erstellt.
Den Kernpunkt des Diagnoseverfahrens bildet die Hypothesize–and–Test–Strategie.

Das Verfahren

Den Ablauf der Diagnose beschreibt das Flußdiagramm aus Abbildung 3.3.
Am Anfang erfolgt eine Eingabe durch den Benutzer, die den aufgetretenen Fehler beschreibt. Aufgrund dieser Beschreibung erfolgt eine Hypothesengenerierung mittels sogenannter Auswahlregeln. Diese feuern, wenn die Fehlerbeschreibung sich mit ihrem Bedingungsteil deckt. Die feuernde Regel liefert eine Liste der mit dem Fehler assoziierten Verdachtshypothesen. Den Verdachtshypothesen sind Wahrscheinlichkeiten zugeordnet.

```
IF Fehlersituation[aktuelle_Diagnose] is fine_track_fehler
  THEN
      Liste_der_Verdachtshypothesen[aktuelle_Diagnose] =
         positionssignal_schwingt < 0.95 >,
         lageabhaengiges_einfahrverhalten < 0.90 >,
         zweiter_ueberschwinger_beim_einfahren_auf_zielzylinder < 0.85 >,
         bremsstromreserve_nicht_ausreichend < 0.80 >,
         nachschwingen_positionssignal < 0.75 >,
         unsymmetrisches_einfahrverhalten < 0.70 >
```

Beispiel einer Auswahlregel

Die Hypothesen werden in der Reihenfolge ihrer Wahrscheinlichkeit überprüft. Die Überprüfung besteht darin, daß eine der Hypothese zugeordnete Testhierarchie durchlaufen wird. Muß die Hypothese aufgrund der Testergebnisse verworfen werden, so wird die nächstwahrscheinliche Hypothese überprüft. Konnte sie bestätigt werden, so wird die Liste der Fehlerursachen um die mit der Hypothese assoziierte Fehlerursache erweitert. Die Liste der Verdachtshypothesen wird gegebenenfalls modifiziert (z. B. kann eine Verfeinerung der Fehlerursache zugefügt werden). Neben den Ergebnissen „Hypothese bestätigt“ und „Hypothese verworfen“ kann auch das Ergebnis „Hypothese unterbrochen“ erzeugt werden. Dies geschieht, wenn bei einem Test eine unerwartete Fehlersituation auftritt (was durch fehlerhafte Testdurchführung in der Anwendungsdomäne Festplatten leicht passieren kann). In diesem Fall fordert das System eine Beschreibung der neuen Fehlersituation an. Wenn es möglich ist, dann wird die Diagnose nach Wiederholung des Tests normal fortgesetzt. Im anderen Fall müssen neue Verdachtshypothesen generiert werden, und dem neuen Fehler muß nachgegangen werden. Sind alle Hypothesen

[10] Das MegaFile ist ein Festplattenspeicher.

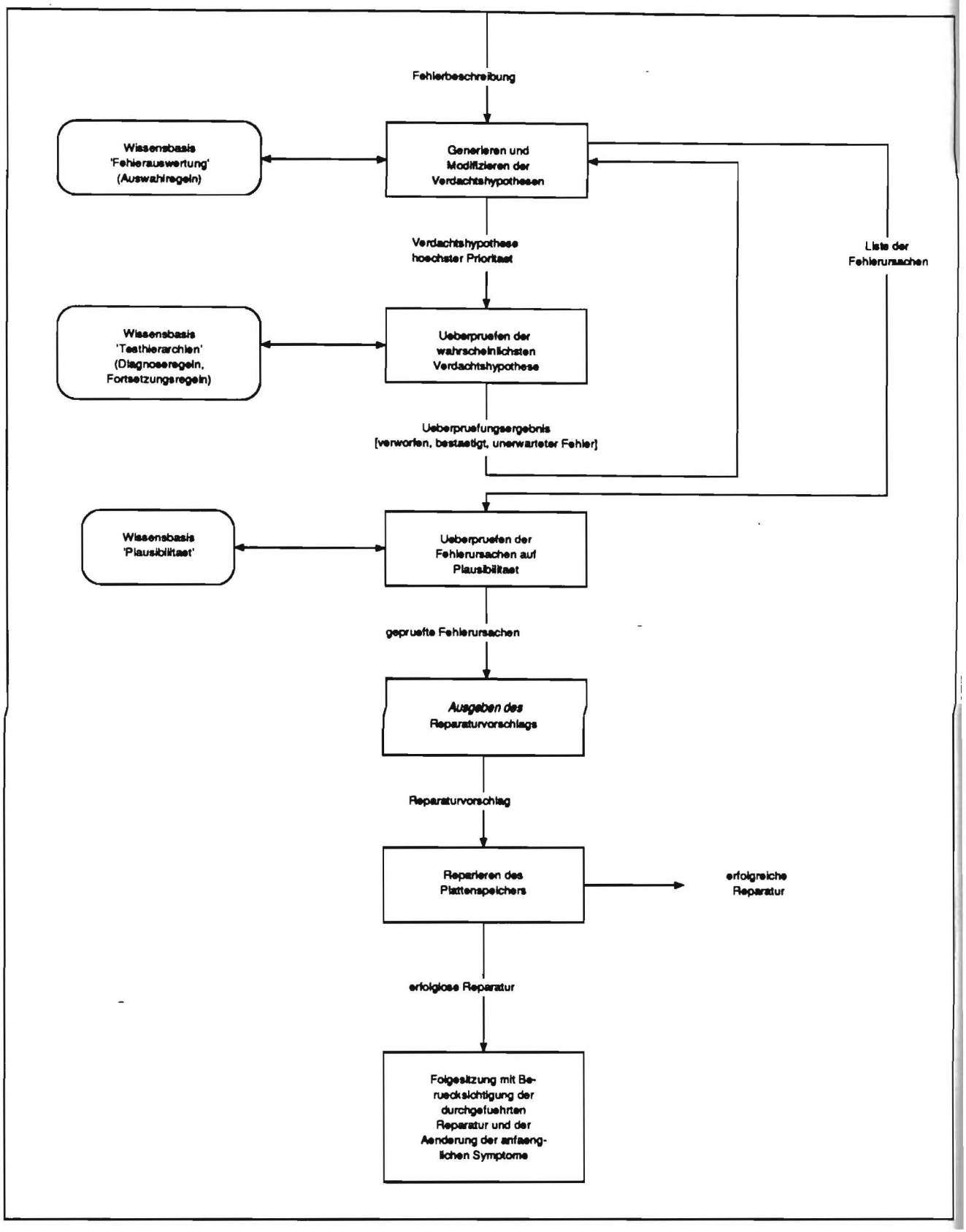

Abbildung 3.3: Das Diagnoseverfahren von MegaFilEx

genaue Aufbau des Systems ist Abbildung 3.13 zu entnehmen.
Der Wissensbasis liegen Diagnosefälle und ein Modell des zu diagnostizierenden Objektes zu-

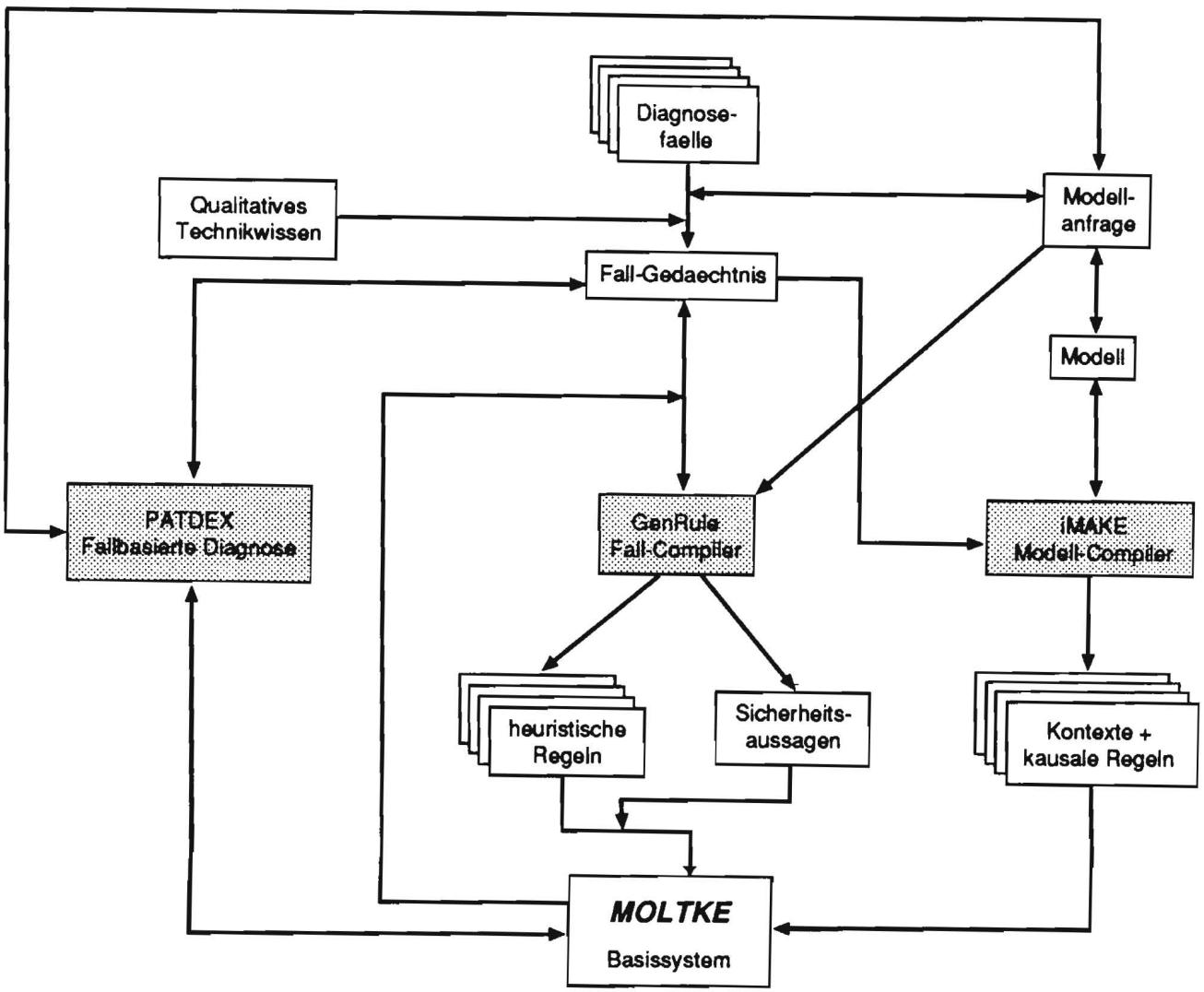

Abbildung 3.13: MOLTKE 3

grunde. Der Modell–Compiler iMAKE erzeugt aus dem Diagnosemodell Kontexte und kausale Regeln. Kontexte repräsentieren Zwischen– bzw. Enddiagnosen zusammen mit dem dazugehörigen Strategiewissen (Das Konzept des Kontextes wird weiter unten erläutert). Die Menge aller Kontexte bilden einen gerichteten, azyklischen Graphen: die Kontextheterarchie.
Der Fall–Compiler GenRule erzeugt aus der Falldatenbasis heuristische Regeln, sogenannte Abkürzungsregeln. Sie dienen dazu, den Dialog mit dem Benutzer auf das Nötigste zu beschränken und Symptome, die (zwar mit Unsicherheit) hergeleitet werden können, nicht zu erfragen.
Die Kontextheterarchie, die kausalen und die heuristischen Regeln bilden das *MOLTKE–Basissystem.*

Der fallbasierte Teil von MOLTKE 3 besteht aus dem PATDEX/2-Diagnosesystem (wurde zuvor in diesem Kapitel vorgestellt) und einer Falldatenbasis.
Es kann entweder nur mit PATDEX/2, nur mit dem MOLTKE-Basissystem oder mit beiden gleichzeitig diagnostiziert werden. Eine Interaktion der Systeme ist im Normalfall[23] nicht vorgesehen (im Gegensatz zu dem zuvor vorgestellten Hybrid).
Wir beschränken uns bei der Darstellung des Diagnoseverfahrens auf das MOLTKE-Basissystem.

Das Diagnoseverfahren des MOLTKE-Basissystems

Das Diagnoseverfahren realisiert eine Establish-Refine-Strategie, die auf der Kontextheterarchie operiert. Vor der Präsentation des Algorithmus soll das Konzept des Kontextes erläutert werden. Jeder Kontext besitzt eine Vorbedingung, die beschreibt, wann der Kontext gültig ist. Desweiteren umfaßt er eine Menge von Reihenfolgeregeln. Reihenfolgeregeln ordnen einer Situation das nächste zu testende Symptom zu. Der interessanteste Teil eines Kontextes ist sein lokaler Strategieinterpreter. Dieser Interpreter liefert das nächste zu erhebende Symptom. Dazu verwendet er per default die zum Kontext gehörenden Reihenfolgeregeln. Es kann jedoch auch spezifiziert werden, daß ein neuronales Netz oder die Testauswahlkomponente von PATDEX/2 benutzt wird. Nach der Symptomerhebung trägt er den Wert in die Situation ein und arbeitet die erfüllten Abkürzungsregeln ab.

1. *Initialisierung.*
2. *Verarbeitung aller feuerbereiten Abkürzungsregeln.*
3. *Bestimmung des aktuellen Kontext*
   `Der aktuelle Kontext hat eine erfüllte Vorbedingung und liegt möglichst tief im Graphen. Alle seine Vorgängerkontexte sind ebenfalls nachgewiesen.`
   `Ist dieser Kontext ein Blatt des Graphen, dann wird die Enddiagnose gestellt, ein Therapievorschlag gemacht und terminiert.`
4. *Aktivierung des Strategieinterpreters*
   `Es wird der Strategieinterpreter des aktuellen Kontextes gestartet. Dieser liefert als Ergebnis das aus Systemsicht zu testende Symptom.`
5. *Symptomerhebung*
   `Es wird entweder das vom System bestimmte Symptom oder ein vom Benutzer vorgeschlagenes Symptom erhoben.`
   `GOTO 2.`

Wertung und Einsatz

Die Vielfalt des verwendeten Wissens und die mit der Erweiterung der Falldatenbasis erzielbaren Lerneffekte (sowohl für PATDEX/2 als auch für das Basissystem) bieten starke Vorteile hin-

[23] Eine Interaktion ist insoweit möglich, als daß jeweils eine Klassifikationskomponente die jeweils andere Testauswahlkomponente benutzt.

sichtlich der Diagnosefähigkeit und der Anwendbarkeit der MOLTKE 3–Expertensystemshell. Die regelbasierte Darstellung des Basissystems führt zu einer guten Laufzeiteffizienz. Insbesondere der Modell–Compiler iMAKE ermöglicht eine relativ bequeme Erstellung des Basisexpertensystems, so daß der Nachteil eines langwierigen Modellaufbaus bei MOLTKE 3 entfällt.
Eine Beschränkung der Anwendbarkeit liegt darin, daß Systeme mit Rückkopplungen nicht modelliert werden können, da der Modell–Compiler iMAKE statisch ist.
Die bei funktionalen Diagnosesystemen hervorzuhebende Fähigkeit der Diagnose von Mehrfachfehlern ist bei MOLTKE 3 eingeschränkt, da mittels der von iMAKE erzeugten kausalen Regeln keine Simulation von Fehlermodellen betrieben werden kann. Eine Situation, in der ein Mehrfachfehler vorliegt, muß explizit modelliert werden, um erkannt werden zu können.
Schließlich fällt auf, daß die heuristischen Regeln, die von GenRule erzeugt werden, zwar einen Determinationsfaktor (eine Art Regelwahrscheinlichkeit) haben, ein Verrechnungsschema für diese Wahrscheinlichkeiten jedoch fehlt. Dies schränkt ihre Verwendung über mehrere Folgerungsstufen ein und erfordert die Markierung und gegebenenfalls die Verifikation der hergeleiteten Symptome.

Das MOLTKE 3–System wird zur Zeit noch erweitert und verbessert. Bisher wurde nur im Rahmen einer Diplomarbeit das MOLTKE–Basissystem zur Diagnose von Rechnernetzen industriell getestet. Es wies, abgesehen von inzwischen abgestellten Handhabungsschwächen, keine bemerkenswerten Mängel auf.

3.2.15 SECQ

Das Diagnosexpertensystem SECQ (siehe [CGJ91]) dient der Analyse der Prüfergebnisse in der Qualitätskontrolle für Leiterplatten. Es bietet eine im Vergleich mit den bisher vorgestellten Diagnosesystemen grundlegende Neuerung: Die bisherigen Diagnosesysteme sind alle der symbolischen künstlichen Intelligenz zuzurechnen. Dieses System dagegen ist ein Hybrid aus einem symbolischen Expertensystem und einem neuronalen Netz. Das symbolische Expertensystem ist regelbasiert und wurde mit der Shell S.1 erstellt ([s1]). Es dient zur sogenannten Makrodiagnose, d. h. es bestimmt den betroffenen Fehlerursachenbereich. Dazu verwendet es Regeln, durch die generisches und konstantes Wissen repräsentiert werden. Die Assoziationen zwischen Fehlern und Ursachen wurden in neuronale Netze verlagert. Für jeden Fehlerursachenbereich existiert ein neuronales Netz, mit dem die sogenannte Mikrodiagnose durchgeführt wird.
Die Begründung für den Einsatz neuronaler Netze liegt darin, daß der Problemlösungsprozeß für diesen Anwendungsbereich nicht gut formulierbar ist. Dies führt dazu, daß ständig neue Regeln und Fehler–Ursache Gewichtungen auftauchen. Der damit verursachte Wartungsaufwand erfordert den häufigen Einsatz eines Wissensingenieurs. Die Verwendung neuronaler Netze (die eine erfolgversprechende Technologie für das automatische Lernen darzustellen scheinen) versetzt den Domänen–Experten in die Lage, die Änderungen selbständig einzubringen, in dem er die Netze mit neuen Mustern für die Assoziationen von Symptomen und ihren Ursachen trainiert.
Der Grund für die Verwendung des regelbasierten Teils liegt darin, daß die Verwendung neuronaler Netze in anwendungstechnischer Hinsicht noch nicht ausgereift ist.

Das Diagnoseverfahren

Die dem Algorithmus zugrundeliegenden Grundinformationseinheiten sind Datensätze für Fe tigungslose. Zu jedem Los gibt es einen Datensatz, der folgende Informationen enthält: Erfa sungsdatum, eine Paarliste (Fehlercode, Fehlerhäufigkeit), Arbeitsbereich.

1. *Makrodiagnose*
   Durch Auslesen aus Datenbanken und Befragen des Benutzers (Erhebung von Symptomen) wird regelbasiert das Problemsegment (Fehlerursachenbereich) ausgesondert, das zur Bestimmung der Fehlerursache näher untersucht werden muß. Dazu wird Wissen über die Kritikalität und die Fehlergeschichte jedes Fehlerursachenbereiches benutzt.
   Für den ausgewählten Bereich wird die Mikrodiagnose gestartet.
2. *Mikrodiagnose*
   Jedem Eingabeneuron eines neuronalen Netzes ist ein Symptom zugeordnet, jedem Ausgabeneuron eine bestimmte Ursache. Die Anregungsniveaus der Eingabeneuronen sind proportional zu der Beobachtungshäufigkeit der entsprechenden Symptome in der gesamten geprüften Stichprobe. Die nach der Propagierung durch das Netz an den Ausgabeneuronen auftretenden Anregungsniveaus bestimmen die Fehlerursache: Die Ursachen mit den höchsten Anregungsniveaus werden als wahrscheinlichste Diagnose betrachtet.

Wertung und Einsatz

Neuronale Netze zeichnen sich insbesondere durch ihre Fähigkeit des automatischen Lerne aus, die sie für Anwendungsbereiche, bei denen das Wissen um Fehler–Ursache–Assoziatione häufig modifiziert werden muß, geeignet erscheinen läßt. Das hier vorliegende Diagnosesyste erlaubt es deshalb, daß der Experte selbständig Anpassungen vornimmt (das Netz mit neue Mustern trainiert). Er kann aber nicht die Netzarchitektur ändern; eine neue Art Sympto oder Fehlerursache kann nur vom entsprechenden Entwickler eingebracht werden. Im vorliege den Anwendungsbereich fällt diese Einschränkung jedoch nicht so stark ins Gewicht, weil da Wissen sich hauptsächlich bezüglich der Unsicherheiten der Beziehungen zwischen Ursache und Symptomen ändert.

Das Diagnosesystem SECQ war zum Zeitpunkt des Erscheinens von [CGJ91] erst als Prototy vorhanden und wurde noch von Experten geprüft. Deshalb und weil der verfolgte Ansatz noc zu neu und unerforscht ist, soll an dieser Stelle keine weitergehende Wertung erfolgen.

Anhang A

Erzeugung eines Diagnosesystems mit TMYCIN unter Verwendung einer Domänentheorie

Nachdem die theoretischen Aspekte und praktischen Realisierungen von Diagnoseverfahren behandelt worden sind, wird anhand einer vorliegenden Expertensystem–Shell eine technische Beispielanwendung entwickelt. Mittels dieser Anwendung sollen einige der aus der Literatur bekannten Konzepte in der Praxis untersucht werden. Anhand dieser Untersuchung werden die Mängel und Vorzüge des Systems erläutert.

Als zweite Aufgabenstellung wird die Nutzbarkeit einer Domänentheorie zur Erstellung der Wissensbasis für die gewählte Expertensystem–Shell geprüft. Diese Aufgabe wurde unter Mitwirkung eines Experten angegangen.

Als Expertensystem–Shell stand das System TMYCIN zur Verfügung. Die Anwendung sollte ein kleines Diagnosesystem für Drehfehler werden.

A.1 Die Expertensystemshell TMYCIN

TMYCIN ist eine verkleinerte Version der heuristischen Expertensystem–Shell EMYCIN. EMYCIN selbst ist eine Verallgemeinerung des Diagnosesystems MYCIN zu einer Shell, und wurde an der Universität Stanford entwickelt (siehe auch [VMBP81]).
Vor der Darstellung des eigentlichen Diagnoseverfahrens soll kurz auf die Wissensrepräsentation eingegangen werden.

Daten über einen konkreten Fall werden in Form eines Kontextes gespeichert.

(defcontext *<context–name>*
<parameters>
<initial–data>
<goals> *)*

Die zur Diagnose verwendeten Regeln bestehen aus Vorbedingung (Prämisse) und Konklusion. Eine Vorbedingung besteht aus mehreren Teilbedingungen, die durch die Junktoren „AND" und „OR" verknüpft sind. Die meisten Teilbedingungen testen Merkmale (Parameter) auf spezielle Ausprägungen. Dabei sind u. a. die Testprädikate „SAME" (testet auf Übereinstimmung mit Certainty–Factor > 0.2) und „THOUGHTNOT" (testet auf Differenz mit Certainty–Factor < -0.2) möglich. Neben diesen Tests können auch numerische Vergleichsprädikate, wie z. B. „GREATERP" („größer") oder „LESSEQ" („kleiner gleich") verwendet werden.
Der Konklusionsteil kann aus einer oder mehreren Konklusionen bestehen. Bei mehr als einer Konklusion muß das Funktionssymbol „DO-ALL" vorangestellt werden. In der Regel schließt eine Konklusion (mit einer durch einen Certainty–Factor ausgedrückten Unsicherheit) auf die Ausprägung eines Merkmals. Es kann jedoch stattdessen auch eine Funktion aufgerufen werden (derartige Funktionen könnten beispielsweise einen Alarm auslösen).

Beispiel 1:

```
(rule43  ($AND (SAME cntxt fehler rillen)
               ($OR (SAME cntxt werkstoff klebend)
                    (SAME cntxt werkstoff kaltverfestigend)))
         (CONCLUDE cntxt fu zu_geringe_schnittgeschwindigkeit tally 500))
```

Diese Regel bedeutet: „Wenn innerhalb des aktuellen Kontextes der Fehler „Rillen" aufgetreten ist, und der verwendete Werkstoff entweder klebend oder kaltverfestigend ist, dann ist die Fehlerursache mit einer Wahrscheinlichkeit von 50% eine zu geringe Schnittgeschwindigkeit."

Beispiel 2:

```
(rule25  ($AND      (SAME cntxt fehler rillen)
                    (THOUGHTNOT cntxt vzg yes)
                    (THOUGHTNOT cntxt ezg yes)
         (DO-ALL    (CONCLUDE cntxt fu zu_grosser_einstellwinkel tally 500)
                    (CONCLUDE cntxt fu zu_grosse_schnittiefe tally 250)
                    (CONCLUDE cntxt fu zu_geringe_schnittgeschwindigkeit tally 250))
```

Die Regel bedeutet: „Wenn innerhalb des aktuellen Kontextes der Fehler „Rillen" aufgetreten ist, und weder Vorschub noch Eckenradius zu groß sind, dann ist die Fehlerursache mit einer Wahrscheinlichkeit von 50% ein zu großer Einstellwinkel, mit einer Wahrscheinlichkeit von 25% eine zu große Schnittiefe und mit einer Wahrscheinlichkeit von ebenfalls 25% eine zu geringe Schnittgeschwindigkeit."

Das im folgenden beschriebene Verfahren verwendet eine Backward–Reasoning–Strategie mit vollständiger[1] Tiefensuche. Das verwendete heuristische Verrechnungsschema ist das MYCIN–Schema (siehe Kapitel 3.1.4).

[1] Es werden sämtliche Belegungsmöglichkeiten einer Variablen gesucht, d. h. auch wenn eine sichere Belegung gefunden wurde, werden weitere Regeln abgearbeitet, die zu Belegungen der Variablen führen können.

A.1.1 Das Verfahren

Hauptalgorithmus

1. Initialisierung des aktuellen Kontextes.
2. Erfragung der initialen Parameter (INITIAL-DATA).
3. Für jeden GOAL (aus GOALS):

   3.1 Aufruf von Bc-Goal.

4. Ausgabe der Werte von GOALS.

Bc-Goal

Eingabe: GOAL.

1. Bestimmung der Regeln (RULES), bei denen GOAL Bestandteil des Konklusionsteils ist.
2. Ist das ASKFIRST-Flag gesetzt oder ist RULES leer, so wird entweder der Wert von GOAL erfragt oder über eine zugeordnete Funktion ermittelt. Anschließend wird Bc-Goal beendet (RETURN).
3. Für jede Regel RULE aus RULES:

   3.1 Die Regelprämisse wird ausgewertet (führt eventuell zu rekursiven Aufrufen von Bc-Goal mit Prämissenteilen als Argumenten).

   3.2 Ist der ermittelte Certainty-Factor größer 0.2, so wird die Konklusion ausgewertet.

4. Konnte durch die Regeln kein Wert für GOAL ermittelt werden, so wird der Wert vom Benutzer erfragt.
5. RETURN.

A.1.2 Bewertung von TMYCIN anhand der Beispielanwendung

Die Beispielanwendung besteht aus ca. 50 Regeln. Diese Regeln sind zum Teil kategorisch (sicher), meistens jedoch heuristisch. Ein Teil der kategorischen Regeln dient zur Feststellung des Fehlerursachenbereiches, die übrigen kategorischen und heuristischen Regeln versuchen, auf die konkrete Ursache der aufgetretenen Drehfehler zu schließen.

Stärken von TMYCIN

- *Kompaktheit:* Das Expertensystem ist sehr klein (12 Seiten Code). Seine Arbeitsweise ist deshalb leicht erfaßbar, und es ist verhältnismäßig einfach, Änderungen und Erweiterungen einzubringen.
- *Erkennung von Mehrfachfehlern:* Durch die vollständige Tiefensuche wird auch nach sicherer Feststellung einer Fehlerursache weiter gesucht. Das inhärente Problem der sich entgegenwirkenden Symptome mehrerer Fehlerursachen wird natürlich nicht gelöst.

- *Geschwindigkeit:* Das Expertensystem ist sehr schnell. Dies ist bei nur 50 Regeln natürlich nicht verwunderlich. Aber der sehr geringe Overhead, die einfache Strategie und die extrem primitive Benutzerschnittstelle werden auch bei größeren Anwendungen zu annehmbaren Geschwindigkeiten führen. Ein „Prescan" sorgt außerdem dafür, daß bei Prämissen, die von dem Junktor „and" angeführt werden, zunächst geschaut wird, ob irgendeine Teilbedingung sicher nicht erfüllt ist. Ist dies der Fall, so kann die gesamte Prämisse verworfen werden.

- *Eingaben können unsicher sein:* Werden Eingaben getätigt, so kann der Benutzer mehrere mit Wahrscheinlichkeiten (Certainty-Factors) versehene Alternativen eingeben.

Schwächen von TMYCIN

- *Vollständige Tiefensuche:* Auch wenn ein Goal schon gefunden wurde, wird dennoch weitergesucht. Dies führt zu einer unnötig langen Suche, bei der vom Benutzer überflüssige Merkmalsausprägungen gefordert werden.

- *Kategorisches In-Erfahrung-bringen von Merkmalsausprägungen:* Kann ein Goal nicht durch Anwendung von Regeln erfüllt werden, so wird der Benutzer gefragt. Die Tatsache, daß eine Wertebelegung nicht zu ermitteln war, kann aber auch bedeuten, daß eine *bestimmte* Wertebelegung nicht zutrifft. Außerdem ist es möglich, daß der Wert auch durch den Benutzer nicht ermittelbar ist. In beiden Fällen ist es zwecklos, den Benutzer danach zu fragen[2].

- *Keine zielgerichtete Suche:* Schon bei dieser kleinen Anwendung fiel unangenehm auf, daß keine Hypothesen generiert werden. Sämtliche Fehlerursachen werden „abgeklappert".

- *Keine Kontexthierarchien möglich:* Das Fehlen der Hierarchisierbarkeit[3] führt dazu, daß globale Kontexte (wie z. B. „Fehlerart = Rillen") immer lokal in jeder Regel als Teilprämisse mitgeführt werden müssen, da die meisten Regeln nur im Zusammenhang dieser Fehlerkontexte gültig sind. Das Vorhandensein einer Kontexthierarchie würde zudem zu einer zielgerichteteren Suche führen, als dies bisher der Fall ist.

- *Symptomerhebung implizit:* Es fehlt eine Testauswahlkomponente. Die Reihenfolge zu erhebender Symptome ist durch ihre syntaktische Reihenfolge in den Vorbedingungen gegeben. Bei der Regelerstellung muß also die Symptomerhebungsreihenfolge bedacht werden.

- *Lisp-Durchgriffe bei der Regelerstellung:* Um das ASKFIRST-Flag zu setzen, ist die Eingabe eines Lisp-Ausdrucks erforderlich. Die Kenntnis der Programmiersprache sollte zur Füllung der Wissensbasis natürlich nicht erforderlich sein.

- *Keine explizite Zeitbehandlung:* Es werden keine Symptomhistorien gespeichert und in den Inferenzprozeß einbezogen. Dadurch wird (in der konkreten Anwendung) das Aufspüren

[2] Diese Schwäche kann durch einen kleinen Zusatz in der Inferenzkomponente beseitigt werden.

[3] Eine Erweiterung des Systems um die Fähigkeit, Hierarchien darstellen und bearbeiten zu können, ist ohne großen Aufwand durchführbar.

von Verschleißerscheinungen erschwert. Um diese Tendenz festzustellen, müßten Werte über einen längeren Zeitraum verteilt vorliegen.

- *Fehlen von Belief-Revision- und Plausibilitätstestkomponenten.*
- *Keine Möglichkeit der Repräsentierung und Behandlung von Unschärfen.*

A.1.3 Beschreibung des Zugriffs auf die Anwendung

Das kleine Diagnosesystem zur Erkennung von Drehfehlern befand sich zum Zeitpunkt der Fertigstellung dieser Arbeit auf dem SUN–Rechner arctecserv-1 des Projektes ARC-TEC in dem Verzeichnis /home/kbecker/tmycin. Im selben Verzeichnis befindet sich auch eine Beschreibung von TMYCIN.
Der Aufruf der Anwendung erfolgt nach laden eines COMMON–Lisp–Systems durch folgende Anweisungen:

```
(load "tmycin")
(load "drehfehler")
(doconsult)
```

A.2 Die Verwendbarkeit der allgemeinen Domänentheorie

Innerhalb des Projektes ARC-TEC wurde versucht, Wissen zu akquirieren, das Zusammenhänge des Drehens von Werkstücken beschreibt. Die akquirierten Zusammenhänge liegen in Form von Fakten, Regeln und Constraints vor. Dieses Wissen bildet eine sogenannte *Domänentheorie.*
Die Akquirierung wurde anwendungsneutral durchgeführt, d. h. insbesondere von einer späteren Nutzung des Wissens zur Erstellung eines Diagnosesystems war nicht ausgegangen worden. Es stellt sich nun die interessante Frage, inwieweit sich die allgemeine Domänentheorie für die Erstellung eines speziellen (auf TMYCIN basierenden) Diagnosesystems einsetzen läßt. Im Einzelnen interessieren folgende Fragestellungen:

- Welche Objektarten der Domänentheorie (Fakten, Regeln, Constraints, ...) sind zur Erstellung der Wissensbasis verwendbar?
- Sind die verwendbaren Objekte der Domänentheorie direkt (ohne Modifizierung) einsetzbar?
- Können nicht direkt einsetzbare Objekte entsprechend modifiziert werden, so daß sie anschließend verwendbar sind?
- Ist ein Experte notwendig, um die Domänentheorie zum Aufbau der Wissensbasis nutzen zu können? Wenn ja, welche Rolle spielt der Experte?

Zunächst muß festgestellt werden, welchen Charakter die Fakten, Regeln und Constraints der Domänentheorie haben.

Durch Fakten werden Fachbegriffe, die in Regeln auftauchen, beschrieben. Die Beschreibung erfolgt entweder umgangssprachlich oder durch Angabe einer Formel.

Beispiel 1: *Schnittkraft:* *Kraft auf den Drehmeißel in Richtung der Umfangsgeschwindigkeit des Werkstücks.*

Beispiel 2: *Spanungshöhe:* $h = f * sin\kappa$

Diese Fakten können in TMYCIN dazu verwendet werden, bei Symptomanforderungen an den Benutzer das entsprechende Merkmal zu erklären. Sie sind damit uneingeschränkt für die Erstellung der Beispielanwendung nutzbar.

Die meisten Regeln der Domänentheorie beziehen sich auf allgemeine Zusammenhänge, die die Anwendungsdomäne des Drehens betreffen. Diese Regeln beinhalten unscharfes Wissen und sind, obschon sie Kausalitäten ausdrücken, weder eindeutig funktional noch eindeutig pathophysiologisch. Sie sollen deshalb im weiteren „Mischregeln" genannt werden.

Beispiel 1: *Passivkraft klein* $\longrightarrow$ *Stabilität des Zerspanprozesses Tendenz groß*

Beispiel 2: *Vorschub groß* $\longrightarrow$ *Oberflächengüte Tendenz klein*

Beispiel 3: *Widerstandsmoment gegen Biegung an der Schneide Tendenz klein* $\longrightarrow$ *Ausbruchgefahr der Schneide Tendenz groß*

Neben diesen Mischregeln gibt es aber auch kausale Regeln, die eindeutig funktionalen bzw. eindeutig pathophysiologischen Charakter haben. Das erste der beiden folgenden Beispiele stellt eine funktionale Regel dar, das zweite eine pathophysiologische.

Beispiel 4: λ *Tendenz klein* $\longrightarrow$ γ *Tendenz groß*

Beispiel 5: *Span von Oberfläche abgerissen* $\wedge$ *Span von Oberfläche nicht getrennt* $\longrightarrow$ *Ausbrüche auf Werkstückoberfläche*

Schließlich gibt es noch einige wenige Constraints.

Beispiel 1: *Eckenverschleiß* > *Hauptschneidenverschleiß*

Nach der Beschreibung der vorhandenen Regeltypen muß spezifiziert werden, welchen Regeltyp TMYCIN erwartet. Da mit MYCIN-Verrechnungsschema heuristisch rückwärts geschlossen wird, muß mit Symptom-Diagnose-Assoziationen gearbeitet werden. Der Prämissenteil einer Regel enthält also Symptome (oder Zwischendiagnosen), der Konklusionsteil Diagnosen (oder Zwischendiagnosen). Es müssen sowohl die Certainty-Faktoren der Prämissen als auch die Regelwahrscheinlichkeit repräsentiert sein.
Damit läßt sich folgendes feststellen: Keiner der angegebenen Regeltypen kann direkt in TMYCIN verwendet werden.
Man benötigt also an dieser Stelle den Experten, um die vorgegebenen Regeln der Domänentheorie umzuschreiben. Dabei eignen sich die Regeln je nach Art mehr oder weniger:

- Pathophysiologische Regeln müssen umgedreht werden, d. h. der Prämissenteil wird der Konklusionsteil und vice versa. Dabei muß der Experte beurteilen, ob die umgedrehte Beziehung (zuzüglich entsprechender Unsicherheit) gilt. Ist dies der Fall, so muß er die Regelunsicherheit schätzen. Wenn die umgedrehte Beziehung nicht gilt, so ist (vom Experten) zu überlegen, ob durch Hinzunahme weiterer Bedingungen und Konklusionen in die Regel doch noch eine brauchbare Beziehung modelliert werden kann.

- Für Mischregeln gilt im Grunde das Gleiche. Überwiegt bei der Regel der funktionale Aspekt, so ist sie (ohne grundlegende Veränderungen) kaum zu gebrauchen.

- Für funktionale Regeln gilt dasselbe, wie für Mischregeln mit überwiegend funktionalem Anteil. Der Experte kann bestenfalls neue Regeln unter dem Aspekt der gestörten Funktionalität entwerfen.

- Constraints sind für den Regelentwurf unter TMYCIN unbrauchbar.

Ein Problem bei der Regelkonvertierung ist die Verarbeitung der Unschärfen der Regeln der Domänentheorie (z.B.: „Vorschub groß", „Tendenz steigend"). Das Problem wurde so angegangen, daß entweder Unschärfen weggelassen wurden, oder daß sie in Unsicherheiten umgewandelt wurden, oder daß sie in Fehlerursachen umformuliert wurden (z. B. wird „Vorschub groß" zu „Vorschub zu groß"). Dadurch wird natürlich die Intention der Unschärfe verfälscht. Der Experte muß abklären, ob und in welchem Maße solche Modifikationen zulässig sind.

Im Endeffekt konnte nur ein kleiner Teil der Regeln der Domänentheorie in TMYCIN-Regeln umgewandelt werden. Die meisten Regeln waren für die Diagnoseaufgabe unbrauchbar, zu allgemein oder sie betrafen einen längeren Zeitraum und waren deshalb nur schwer umwandelbar. Dennoch konnte der Experte vielen Regeln Anregungen entnehmen, selbst neue Beziehungen aufzustellen. Wäre bei der Erstellung der Domänentheorie ein späterer Einsatz für die Diagnose mit einem bestimmten Expertensystem berücksichtigt worden, und hätte der Theorie eine gewisse Strukturierung zugrunde gelegen, so wäre ihr Nutzen erheblich größer gewesen. Auf die Mitarbeit eines Experten konnte auf keinen Fall verzichtet werden.

Literaturverzeichnis

[All83] J. Allen. Maintaining Knowledge about Temporal Intervals. *CACM*, 26(11):832-843, 1983.

[AS88] G. Armano und M. Solimano. Empirical And Functional Knowledge In An Expert System For Fault Diagnosis. In *International Workshop on Artificial Intelligence for Industrial Applications*, 1988.

[Bar91] Jeffrey A. Barnett. Calculating Dempster-Shafer Plausibility. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6), 1991.

[Bec92] Kerstin Becker. Möglichkeiten der Wissensmodellierung für technische Expertensysteme. Diplomarbeit, Universität Kaiserslautern, 1992.

[BM88] R. Bartletta und W. Mark. Explanation-based indexing of cases. In *Proceedings of 7.th National Conference on Artificial Intelligence*, 1988.

[BW88] H. P. Borrmann und F. Winklhofer. Expertensystem zur Schadensfrüherkennung bei Turbogetrieben. *Qualität und Zuverlässigkeit*, 33(3):139–141, 1988.

[CGJ91] G. Calabrese, E. Gnerre, und M. Jula. Ein Expertensystem mit neuronalen Netzen zur Qualitätssicherung. In K. Winkelmann, editor, *Wissensbasierte Systeme in der Praxis*. SIEMENS AG Verlag, 1991.

[CM85] E. Charniak und D. McDermott. *Introduction to Artificial Intelligence*. Addison-Wesley, 1985.

[Coo87] G. Cooper. Expert Systems Based on Belief Networks — Current Research Directions. Memo KSL-87-51, Stanford University, 1987.

[Dav84] Randall Davis. Diagnostic Reasoning Based on Structure and Behaviour. *Artificial Intelligence*, 24:347–410, 1984.

[DH88] Randall Davis und Walter C. Hamscher. Model-Based Reasoning: Troubleshooting. AI Memo 1059, MIT, 1988.

[DHN76] Richard O. Duda, P. E. Hart, und N. J. Nilsson. Subjective bayesian methods for rule-based inference systems. In *Proceedings of the National Computer Conference*, volume 45, pages 1075–1082, 1976.

[dK84] J. de Kleer. Choices Without Backtracking. In *AAAI-84*, pages 79–85, 1984.

[dK86] J. de Kleer. An Assumption Based TMS. *AI-Journal*, 28:127–162, 1986.

[DK89] Daniel Dvorak und Benjamin Kuipers. Model-Based Monitoring of Dynamic Systems. In 12^{th} *IJCAI*, pages 1238–1243, 1989.

[dKW87] J. de Kleer und B. C. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 24, 1987.

[dKW89] Johan de Kleer und Brian Williams. Diagnosis with Behavioral Modes. In 12^{th} *IJCAI*, pages 1324–1330, 1989.

[DM86] G. DeJong und R. Mooney. Explanation Based Learning: An Alternative View. *The Journal of Machine Learning*, 1, 1986.

[Doy79] J. Doyle. A Truth Maintenance System. *AI-Journal*, 12:231–272, 1979.

[DR83] Richard O. Duda und René Reboh. AI and Decision Making: The PROSPECTOR Experience. In Walter Reitman, editor, *Artificial Intelligence Applications for Business*. Alex Publishing Corporation, 1983.

[FCJ88] R. A. Fjellheim, G. Coll, und B. Johanson. Modularity and User Initiative in an Expert System for Fracture Analysis. In D. T. Pham, editor, *Expert Systems in Engineering*. IFS Publications/Springer-Verlag, 1988.

[FRH91] G. Fleischanderl, J. Retti, und W. Höllinger. ARTEX — Ojektorientierte Fehlerdiagnose eines Tondurchschaltesystems. In K. Winkelmann, editor, *Wissensbasierte Systeme in der Praxis*. SIEMENS AG Verlag, 1991.

[Goo82] J. Goodwin. An Improved Algorithm for Non-monotonic Dependency Net Update. Technical Report LITH-MAT-R-82-23, Linjoeping University, 1982.

[GS85] J. Gordon und E. Shortliffe. A Method for Managing Evidential Reasoning in a Hierarchical Hypotheses Space. *AI-Journal*, 26:323–357, 1985.

[Ham86] Kristian Hammond. Chef: A model of case-based planning. In *Proceedings of AAAI*, 1986.

[Ham88] Walter C. Hamscher. *Model-Based Troubleshooting of Digital Systems*. PhD thesis, MIT AI-Lab, 1988. Technical Report 1074.

[Iwa88] Yumi Iwasaki. Qualitative Causal Reasoning About Device Behaviour. In *International Workshop on Artificial Intelligence for Industrial Applications*, 1988.

[Jac87] Peter Jackson. *Expertensysteme - Eine Einführung*. Springer-Verlag, 1987.

[Kar89] P. Karsten. MEGAFILEX, ein Expertensystem zur Diagnose des Megafile — der Entwicklungsprozeß. In *Expertensysteme in Entwicklung und Konstruktion*. VDI Berichte 775, 1989.

[Kui86] Benjamin Kuipers. Qualitative Simulation. *Artificial Intelligence*, 29:289–338, 1986.

[Leb87] M. Lebowitz. Experiments with Incremental Concept Formation: UNIMEM. *Machine Learning*, 2:103–138, 1987.

[LWL89] K. S. Leung, W. S. Felix Wong, und W. Lam. Applications of a novel fuzzy expert system shell. *Expert Systems*, 6(1), 1989.

[McA80] D. McAllister. An Outlook on Truth Maintenance. AI–Memo 551, MIT, 1980.

[MLE88] P. Mertens, T. Legleitner, und G. Ernst. DAX — ein Echtzeit Diagnosesystem als Integrationskomponente in einem Prüfprozeß. *VDI–Z*, 130(5), 1988.

[NP88] C. Nedeß und J. Plog. *EFFEKT — Diagnose–Expertensystem für die Spritzgießfertigung von Elastomeren.* Carl–Hanser–Verlag, 1988.

[Orp90] P. Orponen. Dempster's rule of combination is #P–complete. *Artificial Intelligence*, 44:245–253, 1990.

[Pea85] J. Pearl. How to Do with Probabilities What People Say You Can't. Technical Report CSD-85003, University of California and Proceedings of the 2. Conference on Artificial Intelligence Applications, 1985.

[PST85] Andy Paterson, Paul Sachs, und Michael Turner. The Application Of Causal Knowledge To Real–Time Process Control. In *Expert Systems*, 1985.

[Pup87] Frank Puppe. *Diagnostisches Problemlösen mit Expertensystemen.* Springer–Verlag, 1987.

[Pup88] Frank Puppe. *Einführung in Expertensysteme.* Springer–Verlag, 1988.

[Pup90] Frank Puppe. *Problemlösungsmethoden in Expertensystemen.* Springer–Verlag, 1990.

[Reh91] Robert Rehbold. *Integration modellbasierten Wissens in technische Diagnostik–Expertensysteme.* Dissertation, Universität Kaiserslautern, 1991.

[Ric89] Michael M. Richter. *Prinzipien der Künstlichen Intelligenz.* B. G. Teubner, 1989.

[Ric91] Michael M. Richter. Das MOLTKE–Buch. (in Vorbereitung), 1991.

[s1] S.1 Referenc Manual. Framentec, Paris, 1986.

[Sch91] Gabriele Schmiedel. DIWA — Diagnosewerkzeug mit graphischer Wissensakquisition. Siemens AG, ZFE IS INF 31, 1991.

[SF87] A. Sturm und R. Förster. Application of diagnostic numbers in damage diagnostics. *VDI Berichte*, (644), 1987.

[Sha48] C. Shannon. A Mathematical Theory of Communications. *Bell. Syst. Tech.*, 27, 1948.

[Sha76] G. Shafer. *A Mathematical Theorie of Evidence.* Princeton University Press, 1976.

[SW49] C. Shannon und W. Weaver. *The Mathematical Theorie of Communication.* University of Illinois Press, 1949.

[VMBP81] W. Van Melle, A. C. Bennett, und M. Peairs. The Emycin Manual. Technical Report STAN-CS-81-885, Computer Science Dept., Stanford University, 1981.

[Wer89] Wolfgang Wernicke. Ein System zur Verarbeitung von Erfahrungswissen in MOLTKE 2.0. Diplomarbeit, Universität Kaiserslautern, 1989.

[Weß90] Stefan Weß. PATDEX/2 – ein System zum adaptiven, fallfokussierenden Lernen in technischen Diagnosesituationen. Diplomarbeit, Universität Kaiserslautern, 1990.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
D-6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all publisched papers so far can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-91-17
Andreas Dengel, Nelson M. Mattos:
The Use of Abstraction Concepts for Representing and Structuring Documents
17 pages

RR-91-18
John Nerbonne, Klaus Netter, Abdel Kader Diagne, Ludwig Dickmann, Judith Klein:
A Diagnostic Tool for German Syntax
20 pages

RR-91-19
Munindar P. Singh: On the Commitments and Precommitments of Limited Agents
15 pages

RR-91-20
Christoph Klauck, Ansgar Bernardi, Ralf Legleitner
FEAT-Rep: Representing Features in CAD/CAM
48 pages

RR-91-21
Klaus Netter: Clause Union and Verb Raising Phenomena in German
38 pages

RR-91-22
Andreas Dengel: Self-Adapting Structuring and Representation of Space
27 pages

RR-91-23
Michael Richter, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Akquisition und Repräsentation von technischem Wissen für Planungsaufgaben im Bereich der Fertigungstechnik
24 Seiten

RR-91-24
Jochen Heinsohn: A Hybrid Approach for Modeling Uncertainty in Terminological Logics
22 pages

RR-91-25
Karin Harbusch, Wolfgang Finkler, Anne Schauder:
Incremental Syntax Generation with Tree Adjoining Grammars
16 pages

RR-91-26
M. Bauer, S. Biundo, D. Dengler, M. Hecking, J. Koehler, G. Merziger:
Integrated Plan Generation and Recognition
- A Logic-Based Approach -
17 pages

RR-91-27
A. Bernardi, H. Boley, Ph. Hanschke, K. Hinkelmann, Ch. Klauck, O. Kühn, R. Legleitner, M. Meyer, M. M. Richter, F. Schmalhofer, G. Schmidt, W. Sommer:
ARC-TEC: Acquisition, Representation and Compilation of Technical Knowledge
18 pages

RR-91-28
Rolf Backofen, Harald Trost, Hans Uszkoreit:
Linking Typed Feature Formalisms and Terminological Knowledge Representation Languages in Natural Language Front-Ends
11 pages

RR-91-29
Hans Uszkoreit: Strategies for Adding Control Information to Declarative Grammars
17 pages

RR-91-30
Dan Flickinger, John Nerbonne:
Inheritance and Complementation: A Case Study of *Easy* Adjectives and Related Nouns
39 pages

RR-91-31
H.-U. Krieger, J. Nerbonne:
Feature-Based Inheritance Networks for Computational Lexicons
11 pages

RR-91-32
Rolf Backofen, Lutz Euler, Günther Görz:
Towards the Integration of Functions, Relations and Types in an AI Programming Language
14 pages

RR-91-33
Franz Baader, Klaus Schulz:
Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures
33 pages

RR-91-34
Bernhard Nebel, Christer Bäckström:
On the Computational Complexity of Temporal Projection and some related Problems
35 pages

RR-91-35
Winfried Graf, Wolfgang Maaß: Constraint-basierte Verarbeitung graphischen Wissens
14 Seiten

RR-92-01
Werner Nutt: Unification in Monoidal Theories is Solving Linear Equations over Semirings
57 pages

RR-92-02
Andreas Dengel, Rainer Bleisinger, Rainer Hoch, Frank Hönes, Frank Fein, Michael Malburg:
Π_{ODA}: The Paper Interface to ODA
53 pages

RR-92-03
Harold Boley:
Extended Logic-plus-Functional Programming
28 pages

RR-92-04
John Nerbonne: Feature-Based Lexicons: An Example and a Comparison to DATR
15 pages

RR-92-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Michael Schulte, Rainer Stark:
Feature based Integration of CAD and CAPP
19 pages

RR-92-06
Achim Schupetea: Main Topics of DAI: A Review
38 pages

RR-92-07
Michael Beetz:
Decision-theoretic Transformational Planning
22 pages

RR-92-08
Gabriele Merziger: Approaches to Abductive Reasoning - An Overview -
46 pages

RR-92-09
Winfried Graf, Markus A. Thies:
Perspektiven zur Kombination von automatischem Animationsdesign und planbasierter Hilfe
15 Seiten

RR-92-10
M. Bauer: An Interval-based Temporal Logic in a Multivalued Setting
17 pages

RR-92-11
Susane Biundo, Dietmar Dengler, Jana Koehler:
Deductive Planning and Plan Reuse in a Command Language Environment
13 pages

RR-92-13
Markus A. Thies, Frank Berger:
Planbasierte graphische Hilfe in objektorientierten Benutzungsoberflächen
13 Seiten

RR-92-14
Intelligent User Support in Graphical User Interfaces:
1. InCome: A System to Navigate through Interactions and Plans
   Thomas Fehrle, Markus A. Thies
2. Plan-Based Graphical Help in Object-Oriented User Interfaces
   Markus A. Thies, Frank Berger

22 pages

RR-92-15
Winfried Graf: Constraint-Based Graphical Layout of Multimodal Presentations
23 pages

RR-92-16
Jochen Heinsohn, Daniel Kudenko, Berhard Nebel, Hans-Jürgen Profitlich: An Empirical Analysis of Terminological Representation Systems
38 pages

RR-92-17
Hassan Aït-Kaci, Andreas Podelski, Gert Smolka:
A Feature-based Constraint System for Logic Programming with Entailment
23 pages

RR-92-18
John Nerbonne: Constraint-Based Semantics
21 pages

RR-92-19
Ralf Legleitner, Ansgar Bernardi, Christoph Klauck
PIM: Planning In Manufacturing using Skeletal Plans and Features
17 pages

RR-92-20
John Nerbonne: Representing Grammar, Meaning and Knowledge
18 pages

RR-92-21
Jörg-Peter Mohren, Jürgen Müller
Representing Spatial Relations (Part II) -The Geometrical Approach
25 pages

RR-92-22
Jörg Würtz: Unifying Cycles
24 pages

RR-92-24
Gabriele Schmidt: Knowledge Acquisition from Text in a Complex Domain
20 pages

RR-92-25
Franz Schmalhofer, Ralf Bergmann, Otto Kühn, Gabriele Schmidt: Using integrated knowledge acquisition to prepare sophisticated expert plans for their re-use in novel situations
12 pages

RR-92-26
Franz Schmalhofer, Thomas Reinartz, Bidjan Tschaitschian: Intelligent documentation as a catalyst for developing cooperative knowledge-based systems
16 pages

RR-92-27
Franz Schmalhofer, Jörg Thoben: The model-based construction of a case-oriented expert system
18 pages

RR-92-29
Zhaohur Wu, Ansgar Bernardi, Christoph Klauck:
Skeletel Plans Reuse: A Restricted Conceptual Graph Classification Approach
13 pages

DFKI Technical Memos

TM-91-11
Peter Wazinski: Generating Spatial Descriptions for Cross-modal References
21 pages

TM-91-12
Klaus Becker, Christoph Klauck, Johannes Schwagereit: FEAT-PATR: Eine Erweiterung des D-PATR zur Feature-Erkennung in CAD/CAM
33 Seiten

TM-91-13
Knut Hinkelmann:
Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

TM-91-14
Rainer Bleisinger, Rainer Hoch, Andreas Dengel:
ODA-based modeling for document analysis
14 pages

TM-91-15
Stefan Bussmann: Prototypical Concept Formation
An Alternative Approach to Knowledge Representation
28 pages

TM-92-01
Lijuan Zhang:
Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

TM-92-02
Achim Schupeta: Organizing Communication and Introspection in a Multi-Agent Blocksworld
32 pages

TM-92-03
Mona Singh
A Cognitiv Analysis of Event Structure
21 pages

TM-92-04
Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer:
On the Representation of Temporal Knowledge
61 pages

TM-92-05
Franz Schmalhofer, Christoph Globig, Jörg Thoben
The refitting of plans by a human expert
10 pages

TM-92-06
Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

DFKI Documents

D-91-16
Jörg Thoben, Franz Schmalhofer, Thomas Reinartz:
Wiederholungs-, Varianten- und Neuplanung bei der Fertigung rotationssymmetrischer Drehteile
134 Seiten

D-91-17
Andreas Becker:
Analyse der Planungsverfahren der KI im Hinblick auf ihre Eignung für die Abeitsplanung
86 Seiten

D-91-18
Thomas Reinartz: Definition von Problemklassen im Maschinenbau als eine Begriffsbildungsaufgabe
107 Seiten

D-91-19
Peter Wazinski: Objektlokalisation in graphischen Darstellungen
110 Seiten

D-92-01
Stefan Bussmann: Simulation Environment for Multi-Agent Worlds - Benutzeranleitung
50 Seiten

D-92-02
Wolfgang Maaß: Constraint-basierte Plazierung in multimodalen Dokumenten am Beispiel des Layout-Managers in WIP
111 Seiten

D-92-03
Wolfgan Maaß, Thomas Schiffmann, Dudung Soetopo, Winfried Graf: LAYLAB: Ein System zur automatischen Plazierung von Text-Bild-Kombinationen in multimodalen Dokumenten
41 Seiten

D-92-06
Hans Werner Höper: Systematik zur Beschreibung von Werkstücken in der Terminologie der Featuresprache
392 Seiten

D-92-07
Susanne Biundo, Franz Schmalhofer (Eds.):
Proceedings of the DFKI Workshop on Planning
65 pages

D-92-08
Jochen Heinsohn, Bernhard Hollunder (Eds.):
DFKI Workshop on Taxonomic Reasoning Proceedings
56 pages

D-92-09
Gernod P. Laufkötter: Implementierungsmöglichkeiten der integrativen Wissensakquisitionsmethode des ARC-TEC-Projektes
86 Seiten

D-92-10
Jakob Mauss: Ein heuristisch gesteuerter Chart-Parser für attributierte Graph-Grammatiken
87 Seiten

D-92-12
Otto Kühn, Franz Schmalhofer, Gabriele Schmidt:
Integrated Knowledge Acquisition for Lathe Production Planning: a Picture Gallery (Integrierte Wissensakquisition zur Fertigungsplanung für Drehteile: eine Bildergalerie)
27 pages

D-92-13
Holger Peine: An Investigation of the Applicability of Terminological Reasoning to Application-Independent Software-Analysis
55 pages

D-92-15
DFKI Wissenschaftlich-Technischer Jahresbericht 1991
130 Seiten

D-92-16
Judith Engelkamp (Hrsg.)
Verzeichnis von Softwarekomponenten für natürlichsprachliche Systeme
189 Seiten

D-92-17
Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.):
UM92: Third International Workshop on User Modeling, Proceedings
254 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-92-18
Klaus Becker: Verfahren der automatisierten Diagnose technischer Systeme
109 Seiten

D-92-19
Stefan Dittrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokumentanalyse zur Fokussierung und Klassifizierung von Geschäftsbriefen
107 Seiten

D-92-21
Anne Schauder: Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars
57 pages

Verfahren der automatisierten Diagnose technischer Systeme

Klaus Becker

D-92-18
Document