Möglichkeiten und Grenzen der modellbasierten Diagnose bei hydraulischen Anlagen

Diplomarbeit im Fach Informatik

angefertigt im Fachgebiet Praktische Informatik

Universität-Gesamthochschule Paderborn

Klaus-Ulrich Leweling

Paderborn, im März 1995

Inhaltsverzeichnis

Abbildungsverzeichnis

Zusammenfassung

Diese Arbeit untersucht die Anwendung eines modellbasierten Ansatzes bei der Automatisierung der Fehlerdiagnose in hydraulischen Anlagen. Modellbasierte Verfahren benutzen Wissen über die topologische Struktur und die Funktion der einzelnen Komponenten einer Anlage, um die möglichen Ursachen einer beobachteten Fehlfunktion zu bestimmen. Hier werden solche Anlagen betrachtet, die mit dem System *Art Deco* entworfen worden sind, weil hierbei das benötigte Wissen zum Aufbau der Modelle in einer Wissensbasis zur Verfügung steht. Ein Schema für die modellbasierte Diagnose ist die GDE („General Diagnostic Engine") von de Kleer und Williams, deren Leistungsfähigkeit bislang vorwiegend an diskretwertigen Systemen in der Elektronik gezeigt wurde. Es wird hier untersucht, ob sich dieses Verfahren ebenfalls für die Diagnose von hydraulischen Anlagen eignet, bei denen die zu modellierenden physikalischen Größen kontinuierlich sind. Im Mittelpunkt der Betrachtungen steht die automatische Erzeugung der Modellrepräsentation und die Verhaltenssimulation einer Anlage. Die GDE kann Mehrfachfehler diagnostizieren, wobei jedoch der Zeitaufwand exponentiell mit der Anzahl der Komponenten in der Anlage wächst. Es werden Algorithmen zur näherungsweisen und zur exakten Berechnung der Fehlerhypothesen untersucht. Darüber hinaus wird bewiesen, daß das Verfahren von de Kleer und Williams korrekt und vollständig hinsichtlich der Generierung aller (minimalen) Fehlerhypothesen ist. Für eine bestimmte Klasse von Netzwerken, den Serien-Parallel-Netzwerken, kann durch Einführung von Ersatzwiderständen und durch Fokussierung auf Teilstrukturen der Anlage bei der Fehlersuche die Zeitkomplexität erheblich reduziert werden.

Kapitel 1

Einleitung

Hydraulische Anlagen arbeiten, wie auch andere technische Systeme, nicht fehlerfrei: ein Ventil klemmt, eine Leitung ist verstopft oder undicht, oder der Kolben eines Zylinders bewegt sich nicht wie erwartet. Die Weiterentwicklung von rechnerbasierten Entwurfssystemen ermöglicht es, sehr komplexe technische Anlagen zu bauen, und mit wachsender Komplexität der Anlagen wächst die Schwierigkeit, bei einer eventuellen Fehlfunktion die Fehlerursache(n) zu finden. Besonders in industriellen Anwendungen ist eine extrem hohe Verfügbarkeit der eingesetzten Anlagen erforderlich, und deshalb eine schnelle und genaue Fehlerdiagnose entscheidend. Zum Beispiel können Fehlfunktionen bei hydraulischen Pressen in der Fließfertigung hohe Kosten durch den Stillstand einer ganzen Fertigungslinie verursachen.

Die technische Diagnose bezeichnet nach [Bathelt et al. 1986] die Gesamtheit aller Maßnahmen, die dem Ermitteln und Bewerten des Zustandes eines technischen Systems — d.h. einer Maschine, einer Anlage, eines Prozesses — entsprechend der geforderten Funktionsfähigkeit, einschließlich der Lokalisierung fehlerhafter Komponenten dient. Zu den wichtigsten Problemlösungsmethoden für eine Automatisierung der Diagnose gehören Entscheidungsbäume, Entscheidungstabellen, statistische, heuristische, modellbasierte und fallbasierte Ansätze [Puppe 1990], sowie Diagnostik mit probabilistischen [Heckerman 1991] und neuronalen [Rojas 1993] Netzen. Die meisten bis heute realisierten Diagnosesysteme sind durch den heuristischen Ansatz geprägt, der auf einer regelbasierten Modellierung von Symptom-Diagnose-Assoziationen beruht. Beispiele für solche Diagnosesysteme sind MYCIN [Buchanan & Shortliffe 1984], INTERNIST [Pople 1982], CASNET [Kulikowski & Weiss 1982] im medizinischen Bereich, sowie AUTO-MECH [Tanner & Bylander 1985] im technischen Bereich.

Heuristische Ansätze in der Diagnostik werden in der Literatur häufig als „flach“

oder „empirisch“ bezeichnet [Fink & Lusth 1987], und sind sehr effizient bei der Bestimmung bekannter Fehler, die explizit in die Wissensbasis des Diagnosesystems einkodiert worden sind. Diese Ansätze leiden jedoch unter Problemen bezüglich der Vollständigkeit und Widerspruchsfreiheit ihrer Modellierung.

Die prinzipielle Schwierigkeit bei heuristischen Ansätzen ist, daß die Korrektheit und Vollständigkeit der Wissensbasis schwer zu garantieren ist. Es gibt außerdem keine abgesicherten Verfahren zur systematischen, ingenieursmäßigen Konstruktion der Wissensbasis eines solchen Diagnosesystems [Struß 1989]. Oft werden diese *ad hoc* für die zu untersuchende Anlage erstellt, so daß die Wissensbasis nur für diese eine Anlage einsetzbar ist. Selbst geringfügige Änderungen an der Zielanlage sind mit großem Aufwand bei der Anpassung der Wissensbasis verbunden, und bei der Übertragung des Diagnosesystems auf eine ähnliche Anlage dürfte meistens eine völlige Neuimplementierung sinnvoller sein. Diese Probleme, sowie die schlechte Erklärungsfähigkeit der gefundenen Ergebnisse und die Beschränkung auf Einzelfehler haben die Grenzen dieses Ansatzes zunehmend deutlich gemacht [Struß 1989]. Der modellbasierte Ansatz läßt hoffen, viele dieser Probleme lösen zu können. Bei diesem Ansatz werden die grundlegenden physikalisch-technischen Zusammenhänge der Funktion einer Anlage modelliert. Die modellbasierte Diagnose wird deshalb manchmal auch *„diagnosis from first principles“* genannt [Davis 1984, Reiter 1987, de Kleer & Williams 1987]. Wegen der „tiefen“ Modellierung sind modellbasierte Diagnosesysteme robuster als Heuristik-basierte Systeme, weil sie mit unvorhergesehenen Fällen fertig werden, die nicht durch heuristische Regeln abgedeckt sind.

Die Techniken zur modellbasierten Diagnose befinden sich allerdings noch weitgehend im Forschungsstadium [Kockskämper et al. 1993]. Die große Anzahl von Arbeiten, die zu diesem Thema erschienen sind, deuten auf eine Vielfalt von Problemen hin. Im folgenden sind einige wichtige Aspekte kurz aufgezählt: Bei den modellbasierten Ansätzen sind insbesondere die qualitativen Ansätze von [de Kleer 1984] zu nennen, aber auch die Ansätze von [Struß & Dressler 1989], die das Fehlverhalten von Systemen modellieren. Im Bereich des temporalen Schließens sind die Arbeiten von [Allen 1984] und [Hamscher 1988] relevant. Von [Doyle 1979] und [de Kleer 1986] stammen Ansätze für nicht-monotones Schließen, welche später in dieser Arbeit noch genauer behandelt werden. Probabilistisches Schließen wurde erstmals in dem bereits erwähnten Diagnosesystem MYCIN angewendet. Darüber hinaus wurde eine Reihe von Mechanismen vorgestellt, die den heuristischen und modellbasierten Ansatz kombinieren, um auf diese Weise die Vorteile beider Ansätze nutzen zu können [Fink 1985, Abu-Hanna & Gold 1988].

1.1 Motivation

Die vorliegende Arbeit untersucht die Anwendung des modellbasierten Ansatzes auf die Diagnose von hydraulischen Anlagen. Die Motivation zu dem Thema dieser Arbeit entstand während der Entwicklung von *Art Deco* [Stein & Lemmen 1992], einem System zum Entwurf und zur Konfigurierung von hydraulischen Anlagen, das weiter unten noch genauer beschrieben wird. Das übergeordnete Ziel dieser Arbeit ist die Realisierung eines Diagnosesystems, das eine tatsächlich existierende und mit *Art Deco* entwickelte hydraulische Anlage diagnostizieren könnte. Wie in Abschnitt 2.1.4 gezeigt werden wird, bietet sich der modellbasierte Ansatz besonders für dieses Problem an, weil das zum Aufbau des Modells benötigte Wissen über Struktur und Funktionsweise aus vorhandenen CAD-Daten in *Art Deco* gewonnen werden kann.

Zwei Gesichtspunkte haben die Entscheidung für einen modellbasierten Ansatz wesentlich beeinflußt: zum einen die komponentenorientierte Repräsentation der Anlagen in *Art Deco*, zum anderen die Möglichkeit, das benötigte Modell automatisch generieren zu können. In welcher Weise unterstützt *Art Deco* den modellbasierten Ansatz? Das Prinzip des modellbasierten Ansatzes besteht darin, das Verhalten des Gesamtsystems aus dem Verhalten seiner einzelnen Komponenten und der Art ihrer Verbindung abzuleiten. Die Verhaltensbeschreibungen der Komponenten sind in *Art Deco* in einer Modellbibliothek gespeichert. Diese umfaßt alle Komponenten, aus denen hydraulische Anlagen in *Art Deco* zusammengesetzt sind. Für eine konkrete Anlage können dann die benötigten Komponentenmodelle zur Erzeugung des (Gesamt-)Modells aus der Modellbibliothek entnommen und entsprechend der Anlagenstruktur miteinander verknüpft werden. Es wird in Kapitel 6 gezeigt, daß sich auf diese Weise Basisdiagnosesysteme für spezifische hydraulische Anlagen automatisch generieren lassen.

Der neben Reiters Theorie [Reiter 1987] von mir als Grundlage gewählte Ansatz zur modellbasierten Diagnostik stammt von [de Kleer & Williams 1987]. Sie stellen ein Diagnosesystem namens GDE (General Diagnostic Engine) vor, das Mehrfachfehler zu diagnostizieren vermag. Die GDE und die Theorie von Reiter werden in Kapitel 3 behandelt. Der wesentliche Vorteil der GDE liegt in ihrer inkrementellen Arbeitsweise. Die GDE vermag neue Messungen zur Verbesserung der vorläufigen Diagnosen vorzuschlagen. Die GDE setzt somit den diagnostischen Prozeß fort, der im Verfahren von Reiter bei der Generierung der Kandidaten – von Reiter als Diagnosen bezeichnet – endet.

Leider erweist sich der rein modellbasierte Ansatz in der Praxis als viel zu langsam. Das liegt daran, daß es sich bei der modellbasierten Diagnose um ein inhärent kombinatorisches Problem handelt, welches diesen Ansatz für Anlagen realistischer

Größe (mit mehr als hundert Komponenten) zunächst ungeeignet erscheinen läßt. Wenn zum Beispiel Mehrfachfehler betrachtet werden, dann gibt es fast immer eine exponentielle Anzahl von logischen Möglichkeiten für eine Diagnose [de Kleer & Williams 1989]; jede Komponente kann in dem korrekten oder in einem fehlerhaften Zustand sein. In Abschnitt 4.5 wird jedoch gezeigt, wie für eine bestimmte Klasse von hydraulischen Anlagen – den Serien-Parallel-Netzwerken – das Problem entschärft werden kann, indem durch eine vorgezogene Analyse der Netzwerktopologie Ersatzkomponenten eingeführt werden. Dadurch wird eine Kondensierung der gesamten Netzwerkstruktur auf wenige Komponenten erreicht und außerdem die Grundlage für eine Fokussierung auf einen kleinen Teil der Anlage bei der Diagnose gebildet. In Abschnitt 4.5.1 wird gezeigt, wie die modellbasierte Diagnose auf der Basis von de Kleers GDE für hydraulische Serien-Parallel-Netzwerke prinzipiell funktioniert. Zu diesem Zweck wurde ein rudimentäres Diagnosesystem entwickelt, das für eine konkrete Anlage das Modell generiert, und in dem der Diagnoseablauf nach dem GDE-Schema implementiert ist.

Die meisten Arbeiten in der Literatur zur modellbasierten Diagnostik befassen sich mit der Fehlersuche in diskretwertigen Systemen, insbesondere mit digitalen Schaltkreisen in der Elektronik. Hier sind vor allem die Arbeiten von [Hamscher 1988] und [de Kleer & Williams 1989] zu nennen, aber auch [Dague et al. 1987] und [Chen & Srihari 1989]. In den Systemen HT von [Davis 1984], VMES von [Shapiro et al. 1986] oder DART von [Genesereth 1984] wurde dieser Ansatz erfolgreich zur Fehlersuche (Troubleshooting) in elektronischen Schaltkreisen angewendet. Die Anwendung des modellbasierten Ansatzes auf kontinuierlichwertige Systeme ist bisher wenig untersucht worden.

Bei einem diskretwertigen System nehmen die physikalischen Größen, welche die Signale zwischen den Komponenten transportieren, stets diskrete Werte an. In kontinuierlichwertigen Systemen hingegen müssen kontinuierliche, reelle Variablenwerte modelliert werden. Hydraulische Anlagen sind kontinuierliche Systeme; Druck und Fluß [1] sind kontinuierliche Größen. Mit dieser Arbeit wird gezeigt, wie die Methoden und Erkenntnisse bei der modellbasierten Diagnostik diskretwertiger Systeme auf hydraulische Systeme übertragen werden können, aber auch, welche Schwierigkeiten dabei entstehen, besonders wenn nicht nur einfache Anlagen, sondern Anlagen realistischer Größe untersucht werden sollen. Es werden deshalb in Kapitel 4 die verschiedenen Verfahren zur Bestimmung der Diagnosen genauer untersucht, um eine Grundlage für die Abwägung der Möglichkeiten und Grenzen dieses Ansatzes im Hinblick auf die Diagnose in der Hydraulik zu schaffen.

[1] Mit dem Fluß durch einen hydraulischen Schaltkreis ist in dieser Arbeit die Flußrate gemeint, gemessen in m^3/s.

1.2 Arten der Modellrepräsentation

In der Literatur zur modellbasierten Diagnostik werden i.a. zwei besonders schwerwiegende Aspekte des Problems der Diagnose betrachtet:

1. Es ist Wissen darüber vorhanden, wie die Komponenten eines Gerätes strukturiert sind und wie sie normal funktionieren. Es ist kein Wissen darüber vorhanden, wie Fehlfunktionen auftreten und wie sich diese manifestieren. Dies ist der Bereich, in dem funktionale Modelle [Genesereth 1984, Davis 1984, Reiter 1987, de Kleer & Williams 1987] bei der Diagnose verwendet werden.
2. Wir haben nur Informationen über Fehler (Defekte) und ihre Symptome. In diesem Fall werden pathologische Modelle bzw. Fehlermodelle [Reggia et al. 1983, Peng & Reggia 1990, Poole 1989] verwendet.

Pathologische Modelle werden meistens dort eingesetzt, wo funktionale Modelle nicht bekannt sind oder aufgrund des hohen Vernetzungsgrades und vieler Rückkopplungsschleifen zu komplex wären (z.B. in der Medizin). Ein pathologisches Modell, wie es z.B. in [Reggia et al. 1983] verwendet wird, beschreibt das Fehlverhalten eines Systems. In diesen Modellen sind die kausalen Abhängigkeiten zwischen Diagnosen und Symptomen in Form von Diagnose→Symptom-Beziehungen explizit dargestellt. Es wird beschrieben, wie sich fehlerhafte Veränderungen des Systems kausal auswirken. Nach dem Prinzip des abduktiven Schließens [Peng & Reggia 1990] wird versucht, ausgehend von den Symptomen und den Diagnose→Symptom-Regeln diejenigen Diagnosen zu finden, welche die vorhandenen Symptome erklären können. Der Begriff Diagnose wird hier in der allgemeinen Bedeutung einer Fehlerursache verwendet, aber nicht in der Bedeutung einer Menge von Modellelementen, wie es bei den funktionalen Ansätzen der Fall ist. Ein Beispiel für einen pathologischen Zusammenhang bei dem System „Hebebühne“ (Abbildung 1.1) ist etwa: „Wenn die Pumpleistung nachläßt, verringert sich die Geschwindigkeit des Zylinderkolbens.“

Ein funktionales Modell beschreibt im Gegensatz zu einem pathologischen Modell das Verhalten des korrekten Systems. Es besteht aus einer Struktur- und einer Verhaltensbeschreibung. In der Strukturbeschreibung werden die Komponenten des Systems, ihre technischen Eigenschaften (wie z.B. der Querschnitt bei einer Leitung) und die physikalisch relevanten Größen an ihren Ein- und Ausgängen spezifiziert. Ebenso wird beschrieben, wie die Komponenten miteinander verbunden sind. Die Verhaltensbeschreibung stellt das funktionale Verhalten des Systems dar. Hier werden die Zusammenhänge zwischen den Größen des Systems definiert. Dies geschieht meistens in Form von Constraints (Randbedingungen), die abhängig von den Zuständen einer Komponente Werte für die Anschlüsse dieser Komponente be-

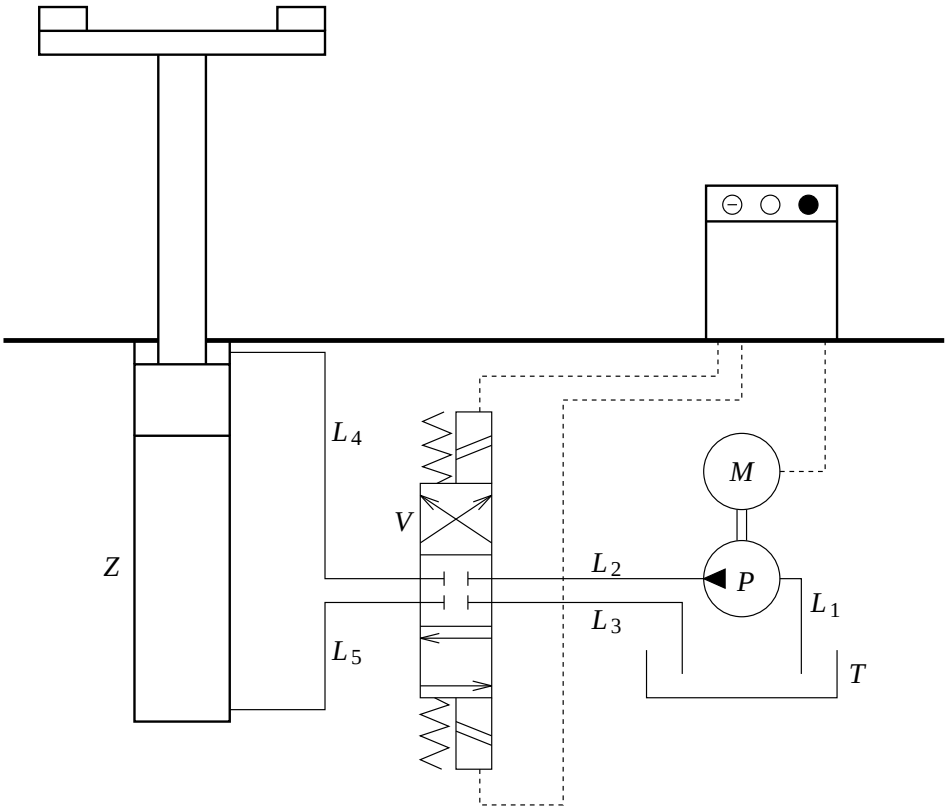
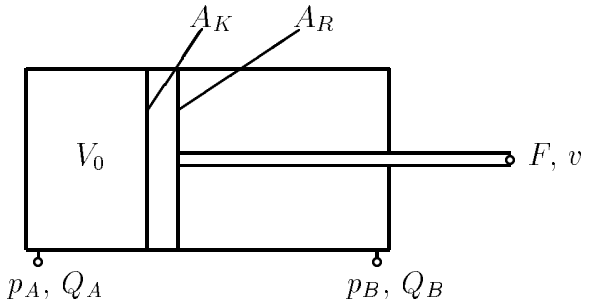


Abbildung 1.1: **Schemadarstellung einer hydraulischen Hebebühne. Die Abkürzungen bedeuten: M = Motor, P = Pumpe, T = Tank, V = Ventil, Z = Zylinder, L_i = Leitungen.**

stimmen. Wir werden uns in Kapitel 6 ausführlich mit Constraints und deren Verarbeitung beschäftigen. Für den Moment reicht es, sich ein Constraint als „Black Box“ vorzustellen, welche versucht, die Gleichungen bzw. Relationen für die Werte an den Anschlüssen einer Komponente aufrechtzuerhalten oder Inkonsistenzen zu entdecken.

Für das obige Beispielsystem „Hebebühne“ (Abbildung 1.1) ist in Abbildung 1.2 das Komponentenmodell für den Zylinder exemplarisch aus dem Gesamtmodell der Hebebühne herausgegriffen und dargestellt. Unter der Abbildung sind die Gleichungen für das Kräftegleichgewicht und die Kontinuitätsbedingung aufgeführt. Diese Gleichungen bilden die Beschreibung der funktionalen Zusammenhänge bei einem Zylinder.



A_K, A_R = Kolbenflächen
p = Druck
V_0 = Zylindervolumen
F = Kraft
Q = Fluß
v = Geschwindigkeit

$$F = p_A \cdot A_K - p_B \cdot A_R \quad \text{(Kräftegleichgewicht)}$$
$$Q_A = A_K \cdot v \quad \text{(Kontinuitätsbedingung)}$$
$$Q_B = -A_R \cdot v$$

Abbildung 1.2: **Struktur eines Differentialzylinders, [Stein 1994]. Zylinder transformieren hydraulische Energie in mechanische Energie. p_A, p_B (bzw. Q_A, Q_B) bezeichnen den Druck (bzw. Fluß) an den Leitungsanschlüssen des Zylinders. F bezeichnet die Kraft am Zylinderkolben und v seine Geschwindigkeit. Das physikalische Verhalten des Zylinders ist durch das Kräftegleichgewicht und die Kontinuitätsbedingung bestimmt.**

1.3 Diagnose mit funktionalen Modellen

Die Diagnose mit funktionalen Modellen beginnt damit, das Verhalten des realen technischen Systems zu beobachten. Die Erwartungen über das Verhalten stützen sich auf ein Modell des Systems, wobei für jede Komponente des Systems angenommen wird, daß diese korrekt funktioniert. Dies setzt voraus, daß anhand des Modells mit Hilfe einer Simulation das korrekte Verhalten der realen Anlage in Form einer (Referenz-)Wertebelegung für die einzelnen Systemgrößen ermittelt werden kann. Bei einem fehlerhaft arbeitenden System wird ein Vergleich des vorhergesagten Verhaltens mit dem tatsächlich beobachteten Verhalten Abweichungen ergeben (s. Abbildung 1.3). Wir nehmen an, daß das System korrekt entworfen wurde und deshalb die Abweichungen in Form von Fehlern (Defekten) im realen System zu suchen sind. Die Abweichungen stellen die Symptome für den Diagnoseprozeß dar. Es gilt nun, die Symptome auf ihre möglichen zugrundeliegenden Ursachen in dem Modell zurückzuführen und dadurch die Fehler zu lokalisieren.

Betrachten wir das Beispiel der Hebebühne aus Abbildung 1.1. Durch eine Simu-

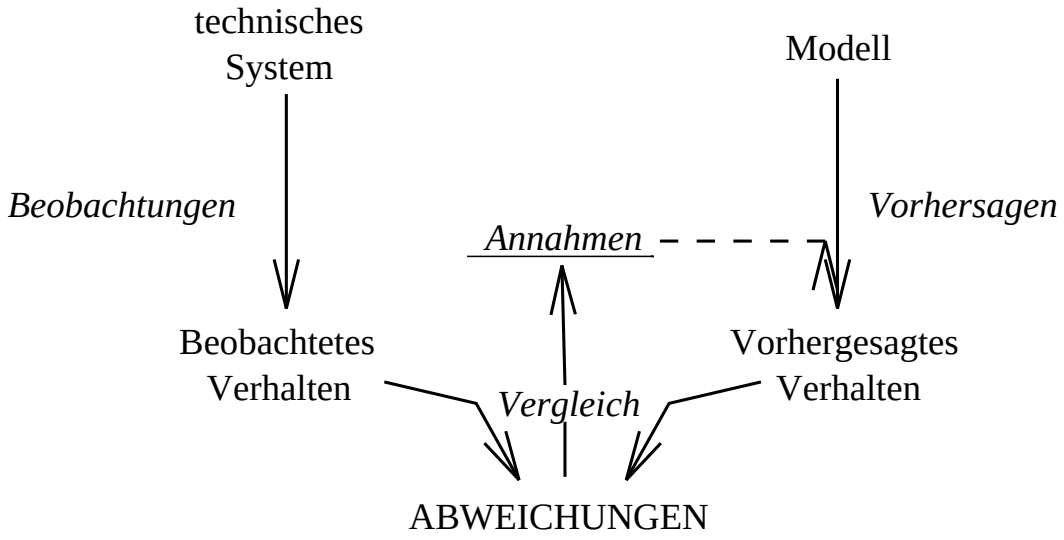


Abbildung 1.3: **Bei der modellbasierten Diagnose produziert ein Modell Vorhersagen über das Verhalten der Anlage, die auf Korrektheitsannahmen der Komponenten beruhen. Beobachtete Abweichungen werden auf mögliche Fehler im Modell zurückgeführt.**

lation anhand des zugehörigen funktionalen Modells ergibt sich mit den aktuellen Betriebsparametern (Schalterstellung des Ventils, Pumpenstrom usw.) eine Hebegeschwindigkeit des Zylinderkolbens von beispielsweise 0.2 m/s. Tatsächlich hebt sich die Bühne jedoch mit einer Geschwindigkeit von 0.1 m/s. Die Abweichung in der Geschwindigkeit ist ein Symptom für die nachfolgende Diagnose. Wenn wie hier das vorhergesagte Verhalten (d.h. die im Modell simulierten Werte für die meßbaren physikalischen Größen) und das beobachtete Verhalten inkonsistent sind, dann muß (wenigstens) ein Fehler vorliegen. Im Rahmen der Diagnose wird nun nach einer Änderung der Struktur oder der Funktion des Modells gesucht, so daß die Vorhersagen des geänderten Modells qualitativ mit den beobachteten Abweichungen am realen Gerät übereinstimmen. Qualitativ bedeutet hierbei, daß die im Modell vorhergesagten Abweichungen dasselbe Vorzeichen haben wie die tatsächlichen Abweichungen. Wir verlangen nicht eine numerisch exakte Übereinstimmung von simulierten und beobachteten Werten. Dies kann auch nicht verlangt werden, wenn nur Wissen über das korrekte Verhalten einer Komponente bekannt ist, nicht aber über das konkrete Fehlverhalten im Defektfall der Komponente.

Bei der Suche nach einer geeigneten Modelländerung wird folgendes Prinzip angewendet: Die Annahmen bezüglich der Korrektheit einiger Komponenten im Modell werden zurückgezogen (d.h. die Korrektheitsannahmen dieser Komponenten werden

negiert), und es wird überprüft, ob dadurch eine qualitative Übereinstimmung zwischen Modell- und Systemverhalten erreicht wird. Anhand des Modells wird also überprüft, ob sich unter der Annahme, daß gewisse Komponenten des Systems nicht korrekt sind, alle Verhaltensabweichungen erklären lassen. Wenn ja, dann können die betreffenden Komponenten auch die im realen System tatsächlich defekten sein. Diese sind deshalb eine mögliche Diagnose.

Im obigen Beispiel könnte eine Änderung des Modells in der Annahme bestehen, daß die Leitung L_5 undicht ist. Aufgrund des Zusammenhangs zwischen Fluß und Geschwindigkeit, der hier durch die Kontinuitätsbedingung im funktionalen Modell des Zylinders (Abbildung 1.2) gegeben ist, führt ein Flußverlust in Leitung L_5 zu einer verminderten Geschwindigkeit des Zylinderkolbens und damit zu einer qualitativen Übereinstimmung mit dem Symptom „Geschwindigkeit = 0.1 m/s“. Die Undichtigkeit der Leitung L_5 kann daher die Beobachtung erklären und ist deshalb eine mögliche Diagnose. In diesem konkreten Fall ist jede der Leitungen zwischen Tank, Pumpe und Zylinder eine mögliche Diagnose, denn ein Leck in jeder dieser Leitungen kann die Beobachtung erklären.

Wie im obigen Beispiel deutlich wird, kann ein und dasselbe Fehlverhalten durch verschiedene Ursachen hervorgerufen werden, so daß es i.a. mehrere Diagnosen gibt. Eine Diagnose im obigen Sinne ist also nur *eine* mögliche Erklärung für das Fehlverhalten. Gesucht wird aber die Diagnose, die die tatsächlich defekten Komponenten des Systems identifiziert. Während mögliche Diagnosen von allen modellbasierten Diagnosesystemen geliefert werden, gibt es für die vergleichende Bewertung der möglichen Diagnosen nur wenige Ansätze.

Eine zusätzliche Forderung an die Diagnostik ist immer, möglichst aussagekräftige Diagnosen zu finden. Die am wenigsten aussagekräftige Diagnose ist, daß alle Komponenten des Systems defekt sind. Diese Diagnose erklärt natürlich jegliches Fehlverhalten des Systems, ist aber für die beabsichtigte Reparatur wertlos. An eine Diagnose wird deshalb immer die Forderung nach einer gewissen Minimalität gestellt. Hier gibt es verschiedene Formen. Das Ziel ist beispielsweise, solche Diagnosen zu suchen, die möglichst wenige Komponenten für defekt erklären.

Kapitel 2

Problembeschreibung

2.1 Charakterisierung der Domäne

Das System *Art Deco* bildet den Ausgangspunkt für diese Arbeit. Für die Entwicklung eines Diagnosesystems ist eine genaue Kenntnis von *Art Deco*, insbesondere des verfügbaren Wissens über hydraulische Anlagen und der Repräsentation des Wissens wesentlich. Deshalb soll zunächst *Art Deco* vorgestellt und die Merkmale kurz beschrieben werden. Für eine ausführliche Beschreibung des Systems wird auf [Stein & Lemmen 1992] verwiesen.

2.1.1 Art Deco

Art Deco ist ein wissensbasiertes System für den Entwurf und die Dimensionierung von hydraulischen Anlagen, das an der Universität-Gesamthochschule Paderborn von D. Curatolo, M. Hoffmann und B. Stein entwickelt worden ist [Stein & Lemmen 1992]. Es erlaubt dem Konstrukteur auf einer Oberfläche, die CAD-Systemen ähnelt, den Schaltplan für ein hydraulisches Netzwerk zu entwerfen und zu testen. *Art Deco* übernimmt die Plausibilitätsprüfung des Entwurfs, indem es beispielsweise prüft, ob sich offene Leitungsenden im Schaltkreis befinden. Darüber hinaus unterstützt es den Konstrukteur bei der Konfigurierung und Bauteildimensionierung der Anlage, indem es zur Entwurfszeit die physikalischen Parameter (z.B. Druck und Fluß) berechnet, die sich bei der jeweils gewählten Netzwerkkonfiguration an den einzelnen Komponenten einstellen. Ein einfacher, in *Art Deco* entworfener hydraulischer Schaltkreis ist in Abbildung 2.1 dargestellt. An den Komponenten sind die verschiedenen Parameterwerte angegeben, die *Art Deco* errechnet.

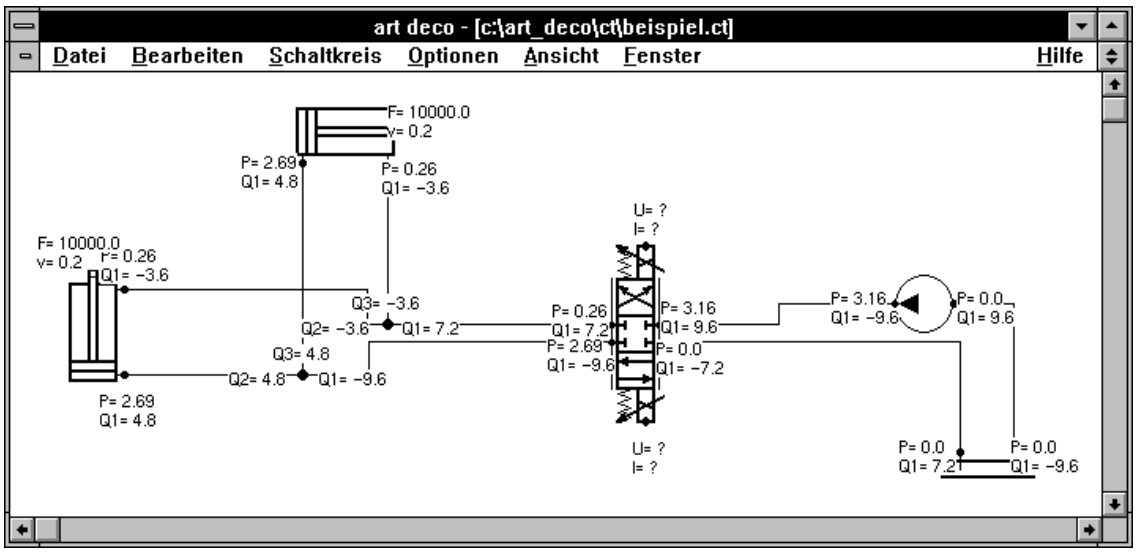


Abbildung 2.1: **Schaltkreise können in *Art Deco* entworfen und konfiguriert werden. *Art Deco* berechnet die global konsistenten Größen an den Komponenten anhand lokaler Vorgaben.**

Wichtig im Hinblick auf die Diagnose und die Wissensmodellierung ist, daß *Art Deco* ein komponentenorientiertes System ist, d.h. der Konstrukteur beschreibt ein hydraulisches System durch die einzelnen Komponenten, die es enthält, und deren Verbindungen (component-connection modeling). Zum Aufbau der Systeme stehen dem Entwickler eine Menge von Standardkomponenten zur Auswahl, d.h. Bauteile, die sich in vielen hydraulischen Anlagen wiederfinden, wie zum Beispiel Zylinder, Ventile, Pumpen, und Tanks. Die Standardkomponenten sind in *Art Deco* in einem erweiterbaren Komponentenkatalog definiert.

Ein wesentliches Prinzip von *Art Deco* ist, das Verhalten des gesamten Netzwerks auf der Basis der Komponentenverhalten zu berechnen (Kompositionalität und Lokalität), und nicht etwa durch ein umfassendes System von Differentialgleichungen. Jede Komponente gehorcht in seinem Verhalten bestimmten physikalischen Gesetzen. In hydraulischen Schaltkreisen sind dies zum Beispiel das Kontinuitätsprinzip und das Kräftegleichgewicht. Für jede Komponente werden in dem Komponentenkatalog diese Gesetze über das Verhalten in Form von Gleichungen oder Relationen für die Werte an den Anschlüssen der Komponente (Ports) abgebildet. Die Relationen werden in *Art Deco* Übergangsgleichungen (transition functions) genannt, weil sie beschreiben, wie die relevanten physikalischen Größen von den Eingängen hin zu den Ausgängen durch die Komponente transformiert werden. Welche physikalischen Größen dabei als relevant bezeichnet werden, hängt von der jeweiligen

Anwendungsdomäne ab. Eine physikalische Größe ist dann relevant, wenn sie in einer Komponentenbeschreibung des *Art Deco*-Komponentenkatalogs als „beobachtbarer Parameter“ definiert ist. Beispiele für relevante physikalische Größen in der Hydraulik sind Druck und Fluß, oder Geschwindigkeit und Kraft bei Zylindern. Nicht relevant ist z.B. die Temperatur des Fluids.

Die Beschreibung einer Komponente umfaßt die folgenden Informationen (hier sind nur Punkte aufgeführt, die für die Modellierung des Verhaltens wichtig sind):

Ein-/Ausgangsports
: Die Ports einer Komponente bilden die Schnittstelle, über die die Komponente mit anderen Komponenten verbunden werden kann. Jeder Port ist eindeutig benannt und hat einen Typ, der angibt, mit welchem Porttyp er verbunden sein kann (z.B. Hydraulik- oder Mechanikverbindung). Ein Port kann, abhängig von der Flußrichtung durch die Komponenten, einmal Eingangs- und ein anderes Mal Ausgangsport sein.

Portvariablen
: Diese sind jeweils einem Port zugeordnet und beschreiben jeweils eine durch diesen Port weitergeleitete physikalische Größe (z.B. Druck und Fluß an einem Port).

Interne Variablen
: Eine Komponente kann interne Variablen besitzen. Z.B. haben Leitungen einen Strömungswiderstand, oder Zylinder eine Querschnittsfläche. Das Verhalten einer Komponente setzt sich aus den konkreten Belegungen der Ein- und Ausgangsvariablen und den internen Variablen zusammen.

Zustandsvariablen
: Einige Komponenten können sich in verschiedenen Zuständen befinden. Z.B. kann ein Ventil je nach Schalterstellung geöffnet oder geschlossen sein. Jeder Zustand erfordert eine eigene Verhaltensbeschreibung.

Verhaltensbeschreibung
: In den Übergangsgleichungen werden die funktionalen Zusammenhänge zwischen den Portvariablen, den internen und den Zustandsvariablen spezifiziert. Ein Beispiel für die Beschreibung eines funktionalen Zusammenhangs ist etwa: „Die Geschwindigkeit des Zylinderkolbens ist das Verhältnis des Flusses durch den Zylinder zum Querschnitt des Kolbens“. Für die Außenwelt geht das Verhalten einer Komponente nur aus den Beziehungen zwischen seinen Ports hervor.

Das konkrete Verhalten einer Komponente ist durch die Belegung der Portvariablen mit einem oder mehreren Werten gekennzeichnet. Dementsprechend ist das Verhalten der Anlage als Ganzes durch die Gesamtheit der Werteausprägungen an den Ports der einzelnen Komponenten ausgedrückt. Mit dem Verhalten einer Komponente bzw. einer Anlage ist in dieser Arbeit immer das statische Verhalten gemeint. Zeitliches Verhalten wird nicht betrachtet.

Grundsätzlich kann eine Komponente verschiedene Verhaltensweisen besitzen. Intakte Komponenten verhalten sich im Sinne der Verhaltensbeschreibung korrekt. Wir sprechen daher vom korrekten Verhalten oder Normalverhalten einer Komponente. Daneben gibt es i.a. mehrere fehlerhafte Verhaltensweisen, die eine Komponente im Falle eines Defektes zeigen kann. Fehlverhalten ist ein vom korrekten Verhalten abweichendes, anderes physikalisches Verhalten. Auch eine defekte hydraulische Komponente verhält sich noch in einer deterministischen, beschreibbaren Weise. Die Verhaltensbeschreibung einer Komponente umfaßt in *Art Deco* zwar das korrekte Verhalten; Fehlverhaltensweisen sind jedoch nicht dargestellt, weil *Art Deco* zum Zwecke der Konfigurierung und Prüfung von hydraulischen Netzwerken, aber nicht für deren Diagnose entwickelt wurde. Wie später noch gezeigt wird, sind die als Fehlermodelle bezeichneten Fehlverhaltensbeschreibungen einer Komponente für die vorgestellte modellbasierte Fehlersuche sehr nützlich, jedoch nicht unbedingt notwendig.

2.1.2 Charakterisierung der hydraulischen Anlagen

Die Diagnoseobjekte in dieser Arbeit sind hydraulische Anlagen, deren Netzwerkrepräsentation in einer *Art Deco*-Wissensbasis vorliegt. Diese Anlagen haben bestimmte Struktur- und Verhaltensmerkmale, die wie folgt charakterisiert werden können:

- Die Anlagen sind Netzwerke, die aus (hydraulischen) Widerständen, Quellen und Senken bestehen. Hydraulische Widerstände sind zum Beispiel Leitungen und Ventile, während Pumpen und Tanks die Quellen bzw. Senken im Netzwerk sind. Ein Zylinder kann sowohl Quelle als auch Widerstand sein, je nachdem, ob er Arbeit an dem Fluid leistet oder das Fluid an ihm.
- Eine Anlage ist aus den Standardkomponenten des Komponentenkatalogs aufgebaut. Diese Komponenten sind elementar in dem Sinn, daß sie nicht weiter in Unterkomponenten zerlegt werden und bei einer Reparatur immer als Ganzes ausgetauscht werden.
- Eine Anlage kann i.a. hierarchisch strukturiert sein (z.B. in Systeme, Subsysteme, Aggregate, Baugruppen, Komponenten). Allerdings wird die hierarchische

Strukturierung von *Art Deco* in der zur Verfügung stehenden Version (2.6) nicht unterstützt. Für das angestrebte Diagnosevorhaben ist deshalb davon auszugehen, daß die gesamte Netzwerkstruktur auf einer einzigen Ebene beschrieben ist, nämlich auf der Ebene der elementaren Komponenten.

- Die Komplexität der untersuchten Anlagen ist groß (d.h. die Anlagen können mehrere hundert Komponenten enthalten).

- Eine Anlage kann in verschiedenen Betriebsarten „gefahren“ werden. Bei einer Hebebühne gibt es z.B. die Betriebsarten „Heben“ und „Senken“. Das Vorliegen einer bestimmten Betriebsart impliziert, daß Wege in der Anlage freigeschaltet und andere gesperrt sind (z.B. über entsprechende Ventilstellungen). Unterschiedliche Betriebsarten implizieren deshalb unterschiedliche Netzwerkstrukturen.

- Es werden nur stationäre Vorgänge beim Verhalten der Anlage betrachtet. Dynamische (d.h. zeitabhängige) Verhalten werden nicht betrachtet. Die zu untersuchende Anlage befindet sich in einem „eingeschwungenen“, d.h. stabilen Zustand.

- Wir haben es bei den hydraulischen Anlagen nicht nur mit linearem, sondern auch mit nichtlinearem Verhalten zu tun. Z.B. ist das Verhalten eines hydraulischen Widerstandes nichtlinear: der Druckabfall (d.h. die Potentialdifferenz) an einem hydraulischen Widerstand ist proportional zum Quadrat des Flusses.

2.1.3 Diagnosemerkmale

Die Fehlersuche in den hydraulischen Anlagen, um die es dieser Arbeit geht, ist durch die folgenden Merkmale gekennzeichnet:

- Die Menge der möglichen Fehlerquellen ist groß. Fehlerquellen können defekte Komponenten sein, oder es kann ein struktureller Fehler vorliegen. Strukturelle Fehler sind Fehler in den Komponentenverbindungen, wie z.B. verstopfte oder undichte Leitungen. Oft enstehen strukturelle Fehler auch durch falsche Ventilstellungen. Strukturelle Fehler sind dadurch charakterisiert, daß die reale Anlage gegenüber seinem Modell zusätzliche oder fehlende Verbindungen aufweist.

- Mehrfachfehler sind möglich. Mit Mehrfachfehlern ist das gleichzeitige Vorliegen von mehreren, voneinander unabhängigen Komponentendefekten oder Strukturdefekten gemeint. Zwei verschiedene Defekte sind dann voneinander

unabhängig, wenn der eine nicht durch den anderen hervorgerufen wird. Die vorhandenen Symptome sind die gemeinsamen Auswirkungen der verschiedenen Defekte.

- Das Wissen über die vorhandenen Symptome der hydraulischen Anlage ist unvollständig. Zu Beginn der Fehlersuche ist oft nur ein Symptom bekannt. In den meisten Fällen reicht dies nicht aus, um den zugrundeliegenden Fehler eindeutig zu identifizieren, so daß im Verlauf des Diagnoseprozesses weitere Symptome durch Beobachtungen und Messungen an der realen Anlage erhoben werden müssen.

- Messungen an der hydraulischen Anlage sind i.a. unvollständig, ungenau und teuer. Jede Messung ist mit ganz bestimmten Kosten verbunden. Die Anzahl der durchzuführenden Messungen ist deshalb zu minimieren.

- Beobachtungen liegen nicht immer in Form eines numerischen Wertes vor. Für den Techniker an der Anlage ist es oft viel einfacher, die Abweichung im Verhalten qualitativ zu erfassen, als diese genau zu quantifizieren (z.B. mit der Beobachtung „der Kolben bewegt sich zu langsam“). An vielen Punkten der Anlage sind genaue Messungen gar nicht möglich, so daß nur eine qualitative Aussage möglich ist.

- Wissen darüber, welches Fehlverhalten einzelne Komponenten bei einem Defekt zeigen, welche verschiedenen Defekte (Fehlermodi) eine konkrete Komponente haben kann und mit welcher Wahrscheinlichkeit ein bestimmter Defekt eintritt, ist nur zum Teil vorhanden. Dieses Wissen existiert jedoch außerhalb der Modellrepräsentation der Anlage. Um solches Wissen für die Diagnose nutzbar zu machen, muß es explizit im Modell dargestellt werden.

2.1.4 Abwägung der verschiedenen Ansätze

Im direkten Vergleich des fall-, regel- und modellbasierten Ansatzes sprechen viele der oben aufgeführten Merkmale für eine modellbasierte Lösung bei der Entwicklung eines Diagnosesystems. Für *Art Deco* bietet sich ganz besonders ein modellbasierter Ansatz an, der auf der funktionalen Modellierung des Anlagenverhaltens beruht. Für eine bestimmte zu untersuchende Anlage ist das zugehörige funktionale Modell in *Art Deco* vorhanden, zumindest was die Modellierung des korrekten Verhaltens anbelangt. Nur muß dieses Modell in die Form gebracht werden, die für die Simulation des korrekten bzw. fehlerhaften Verhaltens im Rahmen des Diagnoseprozesses benötigt wird.

Wegen der Möglichkeit von Mehrfachfehlern ist ein regelbasierter (heuristischer) Ansatz wenig geeignet für das vorliegende Diagnoseproblem. Zwar könnten im Prinzip auch Mehrfachfehler durch Regeln abgedeckt werden, jedoch ist die Anzahl der erforderlichen Regeln, um alle möglichen Kombinationen von Mehrfachfehlern zu erfassen, viel zu groß, denn sie wächst exponentiell mit der Anzahl der Komponenten in der Anlage.

Allerdings würde ein fallbasierter Ansatz hier ebenfalls in Frage kommen, denn für jeden bereits gelösten Diagnosefall könnten die jeweiligen Symptome zusammen mit den bekannten Diagnosen in einer Datenbank gespeichert und bei der Fehlersuche in zukünftigen Fällen darauf zurückgegriffen werden. Wie können die gespeicherten Daten alter Fälle bei der Lösung eines konkreten Diagnoseproblems helfen? Bei dem fallbasierten Ansatz wird zu dem neuen Fall ein möglichst ähnlicher aus der Datenbank der Fälle gesucht, zu denen die Lösung des Diagnoseproblems bekannt ist. Wird ein solcher Fall gefunden, so kann dessen Lösung übernommen werden. Das Wissen besteht aus der Sammlung alter Fälle und aus einem Ähnlichkeitsmaß, das angibt, wie ähnlich und wie wichtig unterschiedliche Ausprägungen eines Symptoms für die zugrundeliegenden Fehler ist. Um die Ähnlichkeit des vorliegenden Falles mit einem Fall aus der Datenbank zu bestimmen, wird für jedes Paar korrespondierender Symptome aus den beiden Fällen die Ähnlichkeit nach dem Ähnlichkeitsmaß berechnet. Aus der Verrechnung aller Einzelvergleiche der Symptome wird auf diese Weise die Gesamtähnlichkeit zweier Fälle ermittelt.

In der vorliegenden Domäne (hydraulische Anlagen in *Art Deco*) wurden jedoch bislang keine Falldaten gesammelt. Daher scheidet auch das fallbasierte Vorgehen zunächst aus. Grundsätzlich ist der modellbasierte (funktionale) Ansatz dem fallbasierten in einem weiteren Punkt überlegen: wenn tiefergehendes Wissen für die zu diagnostizierenden Anlagen vorhanden ist, dann kann mit dem modellbasierten Ansatz die Aufgabe, spezifische Diagnosesysteme automatisch zu generieren, leichter erfüllt werden als bei einem fallbasierten Ansatz.

2.2 Systemarchitektur

Die Abbildung 2.2 zeigt das Architekturkonzept des modellbasierten Diagnosesystems für *Art Deco*. Es wird davon ausgegangen, daß das zu untersuchende hydraulische System mit *Art Deco* entwickelt worden ist und deshalb die Anlagenbeschreibung in einer *Art Deco*-Wissensbasis vorhanden ist. Der Diagnoseprozeß beginnt zunächst damit, das Modell der Anlage aus den Daten der *Art Deco*-Wissensbasis zu erzeugen. In dem Modell ist das korrekte Verhalten der Anlage durch ein Con-

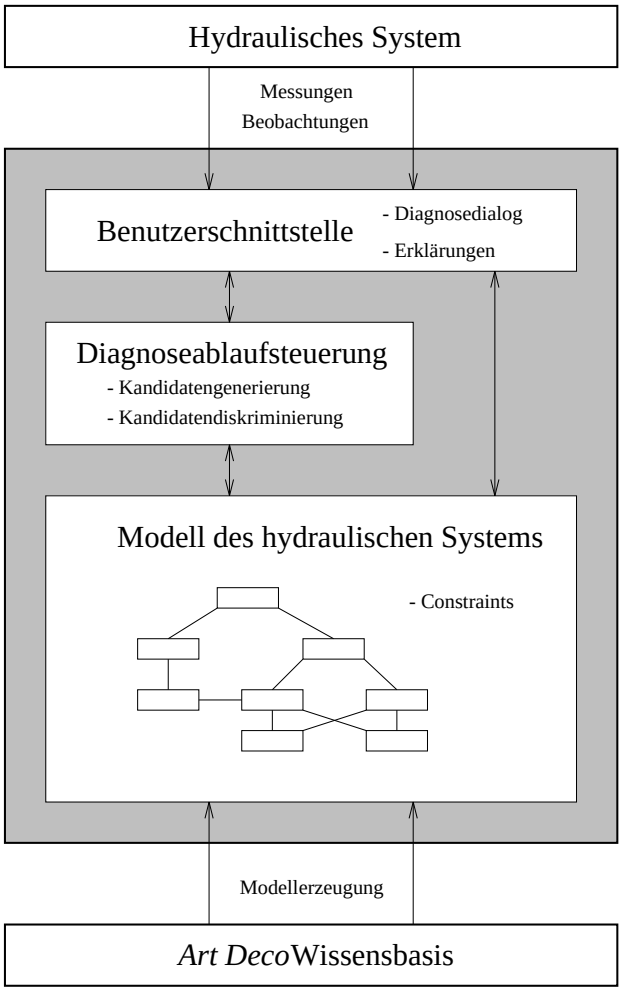


Abbildung 2.2: **Systemarchitektur des Diagnosesystems für hydraulische Anlagen.**

straintnetz definiert. Die Modellerzeugung besteht darin, die einzelnen Verhaltensbeschreibungen der Komponenten, welche in *Art Deco* durch die Übergangsfunktionen gegeben sind, in entsprechende Constraints zu übersetzen und diese entsprechend der Strukturbeschreibung der Anlage zu einem Constraintnetz zu verbinden.

Die Benutzerschnittstelle besteht im einfachsten Fall aus einer Dialogkomponente, die dem Benutzer die Eingabe von Messungen und Beobachtungen, sowie den aktuellen Betriebsparametern erlaubt und als Ergebnis Diagnosen ausgibt und bei Bedarf erklärt. Eine Wissenserwerbskomponente erübrigt sich, weil das Diagnosesystem über keine eigene Wissensbasis verfügt, sondern diejenige von *Art Deco* benutzt. Die Wissensakquisition im Sinne der Beschaffung von Anlagen- und Komponenten-

beschreibungen obliegt *Art Deco*.

Die Diagnoseaufgabe wird begonnen, weil es eine Abweichung gibt zwischen dem vom Modell vorhergesagten Ergebnis einer Messung und dem tatsächlich beobachteten Ergebnis. Da wir das Modell als korrekt unterstellen, kommen als Ursache für alle Modell-Artefakt-Abweichungen nur Komponenten-Fehlfunktionen in Betracht.

Gewöhnlich reichen die vorliegenden Meßergebnisse nicht aus, um die Komponenten eindeutig zu identifizieren, die für die Abweichungen verantwortlich sind, so daß weitere Messungen und Beobachtungen notwendig sind. Deshalb erfordert der Diagnoseablauf zwei Phasen. In der ersten Phase (Kandidatengenerierung) wird anhand des Systemmodells die Menge der möglichen Ursachen für die Abweichungen ermittelt. In der zweiten Phase (Kandidatendiskriminierung) produziert das System einen Vorschlag für die beste nächste Messung oder Beobachtung, um die Menge der Kandidaten zu verkleinern. Messungen sind teuer, deshalb ist die beste nächste Messung diejenige, die – im Mittel – zu der Entdeckung der defekten Komponenten in einer minimalen Anzahl von Messungen führen wird. Das Ergebnis dieser Messung bringt i.a. eine weitere Abweichung hervor. Der Prozeß von Kandidatengenerierung und -diskriminierung wird solange iteriert, bis schließlich (im Idealfall) nur noch eine Ursache übrig bleibt.

2.2.1 Einige grundlegende Voraussetzungen

Für die Fehlersuche in hydraulischen Anlagen werden in dieser Arbeit einige Annahmen getroffen, die sich ebenfalls in anderen Arbeiten zur modellbasierten Diagnostik finden [de Kleer & Williams 1987]. Wir nehmen an, daß die Durchführung einer Messung bzw. Beobachtung keinen Einfluß auf die zu untersuchende Anlage und das gezeigte Fehlverhalten hat. Wir nehmen weiterhin an, daß eine einmal gemessene Größe seinen gemessenen Wert über die gesamte Dauer des Diagnoseprozesses beibehält. Dies ist gleichbedeutend mit der Annahme, daß das Verhalten jeder Komponente — ob korrekt oder fehlerhaft — unabhängig vom Verlauf der Zeit ist. Beobachtungen werden als korrekt angenommen und können im weiteren Verlauf der Diagnose nicht widerlegt werden. Komponentenverhalten, das sich spontan ändern kann, wie z.B. ein Wackelkontakt bei einem elektrisch geschalteten Ventil, wird ausgeschlossen. Andererseits wird angenommen, daß eine Komponente, die defekt ist, bei einer Beobachtung nicht unbedingt Fehlverhalten zeigen muß. Das Fehlverhalten kann sich bei einer anderen Menge von Betriebsparametern zeigen. Daher kann aus einer Beobachtung nicht geschlossen werden, daß eine Komponente korrekt ist, denn bei der nächsten Beobachtung kann sich ein Fehlverhalten der Komponente zeigen.

2.2.2 Eingaben und Ausgaben des Diagnosesystems

Das modellbasierte Diagnosesystem für hydraulische Anlagen erhält als primäre Eingabe ein Modell der zu untersuchenden hydraulischen Anlage. Das Modell ist ein Netzwerk aus Komponenten und Leitungen. In Kapitel 6 wird gezeigt, wie dieses Modell aus der Anlagenrepräsentation in der *Art Deco*-Wissensbasis erzeugt werden kann. Jede Komponente verfügt über eine Beschreibung seines stationären Verhaltens. Die zweite Eingabe des Programms ist eine Beschreibung der Betriebsparameter, mit denen die Anlage „gefahren" wird. Das sind die von außen vorgegebenen Systemgrößen, wie z.B. die Stellung der Ventile, der Tankdruck oder die Pumpendrehzahl. In dem Modell wird dann das Verhalten der Anlage bei den vorgegebenen Betriebsparametern simuliert. Wenn an der realen Anlage Abweichungen von diesem vorhergesagten Verhalten entdeckt werden, produziert das Diagnoseprogramm eine Liste von Komponenten, die für die Abweichungen verantwortlich sein können. Wenn die Komponenten in dieser Liste noch nicht genügend eingegrenzt sind (und dies entscheidet der Benutzer), schlägt das Progamm interaktiv einen weiteren Meßpunkt vor, an dem als nächstes gemessen bzw. beobachtet werden soll, um die Menge der verdächtigen Komponenten weiter einzuschränken. Mit Hilfe des Ergebnisses jeder weiteren Messung kann das Programm schließlich die korrekte Diagnose finden.

2.3 Modellierung hydraulischer Anlagen

Bei der Entwicklung eines Diagnosesystems für hydraulische Anlagen stellt sich die zentrale Frage nach der Detailliertheit der Modellierung. An die Modellierung einer Anlage werden drei Forderungen gleichzeitig gestellt. Ein Modell muß wirklichkeitstreu, präzise und dabei effizient sein [Hamscher 1988]. Ein Modell ist wirklichkeitstreu, wenn es keine inkorrekten Vorhersagen über das Verhalten der Anlage erlaubt. Wirklichkeitstreue ist die oberste Forderung bei der Modellbildung, denn wenn das Modell inkorrekte Vorhersagen macht, dann werden Abweichungen zwischen dem realen Gerät und dem Modell fälschlicherweise Defekten in dem Gerät zugeschoben. Die größtmögliche Detaillierungstiefe der Modellstruktur ist in *Art Deco* durch die Definition der primitiven Komponenten in dem Komponentenkatalog vorgegeben. Wie kann die Wirklichkeitstreue des Modells garantiert werden? Erstens muß gewährleistet sein, daß das Modell nur korrekte Vorhersagen über das Verhalten der primitiven Komponenten zuläßt, wenn diese isoliert betrachtet werden. Zweitens müssen die primitiven Komponenten in einer Weise zusammengesetzt sein, die das Verhalten der primitiven Komponenten nicht beeinflußt. Dies ist die grundlegende Idee des „no function in structure"-Prinzips [de Kleer 1984] und bedeutet, daß die

Beschreibung eines Komponentenverhaltens nicht von der korrekten Funktion des Gesamtsystems abhängen darf.

Um sicherzustellen, daß die Komponentenmodelle der primitiven Komponenten isoliert betrachtet korrekt sind, müssen alle Arten, in denen diese primitiven Komponenten untereinander wechselwirken können, explizit beschrieben werden. Zum Beispiel ist die Aussage „Wenn der Schalter geschlossen ist, wird Strom fließen“ nicht korrekt, denn sie berücksichtigt nicht die Tatsache, daß ein Spannungsabfall erforderlich ist, damit Strom fließt. Jede solche Wechselwirkung, die nicht im Modell beschrieben ist, stellt eine mögliche Quelle von Fehldiagnosen dar.

Die Verhaltensbeschreibungen der einzelnen primitiven Komponenten werden von *Art Deco* in dem Komponentenkatalog vorgegeben. Es wird angenommen, daß die Verhaltensbeschreibungen dem „no function in structure“-Prinzip genügen und deshalb eine wirklichkeitstreue Modellierung des Verhaltens der Gesamtanlage auf der Grundlage der Komponenten-Verhaltensbeschreibungen erfolgen kann.

Präzision ist eine weitere Forderung bei der Modellierung. Ganz allgemein ist ein Modell präzise in dem Maße, wie die Vorhersagen, die es produziert, konkret genug sind, um durch Beobachtungen am realen Gerät widerlegt werden zu können. Ein triviales Gerätemodell macht überhaupt keine Vorhersagen. Es ist wirklichkeitstreu, weil es keine falschen Aussagen macht, aber es ist für die Diagnose nutzlos, weil es ebenso keine Abweichungen produzieren kann. Ein nützliches Modell produziert Vorhersagen, die durch mögliche Beobachtungen bestätigt oder widerlegt werden können.

Wirklichkeitstreue und Präzision des Modells stehen im Zielkonflikt mit seiner Effizienz. Um die Effizienz zu steigern, wird in vielen Arbeiten zur modellbasierten Diagnostik eine qualitative Modellierung der relevanten physikalischen Größen verwendet [de Kleer & Brown 1984, Kuipers 1986, Lambert et al. 1988, Kockskämper et al. 1993, Console et al. 1993]. Die Simulation des Verhaltens in dem Gerätemodell erfolgt in dem Fall auf der Grundlage von qualitativen Werten. Damit wird die Vorgehensweise eines menschlichen Experten nachgebildet, der oft durch eine rein qualitative Betrachtung der Zusammenhänge zwischen den Systemgrößen komplexe hydraulische Systeme korrekt zu diagnostizieren vermag [Struß 1993].

Eine grundsätzliche Frage bei der qualitativen Modellierung ist, in wieviele qualitativ unterschiedliche Regionen der numerische Wertebereich eines Parameters unterteilt werden soll. Das Problem dabei ist, daß bei der Quantisierung einer kontinuierlichen Werteskala zwangsläufig nicht zwischen den Werten desselben Intervalls differenziert werden kann.

Ein Hauptproblem bei der qualitativen Simulation ist, daß diese mehrdeutige Er-

gebnisse liefert. Der Grund dafür liegt darin, daß qualitative Arithmetik [de Kleer & Brown 1984] unterbestimmt ist. Wenn z.B. in einer hydraulischen Anlage bei einer T-Leitungsverbindung an zwei Seiten die Menge einströmenden Flusses qualitativ gegeben ist mit „zu niedrig“ beziehungsweise „zu hoch“, dann gibt es für die Menge ausströmenden Flusses an der dritten Seite der T-Verbindung drei Möglichkeiten der Addition, nämlich „zu hoch“, „normal“ und „zu niedrig“. Bei einer Anlage mit n T-Verbindungen gibt es demnach 3^n Möglichkeiten für den Fluß. Der Verlust an Präzision macht es oft unmöglich — insbesondere lokal — zu bestimmen, welche von zwei Einflüssen auf eine Systemgröße dominiert.

Wegen der Mehrdeutigkeiten bei der Simulation und der Ununterscheidbarkeit von korrektem und fehlerhaften Verhalten wird die Möglichkeit einer qualitativen Modellierung nicht länger betrachtet. Statt dessen werden im Diagnosesystem für *Art Deco* die zu untersuchenden hydraulischen Anlagen quantitativ exakt modelliert. Das bedeutet, es erfolgt eine Verhaltenssimulation mit numerischen Werten.

Kapitel 3

Der modellbasierte Ansatz

Die meisten existierenden Diagnoseprogramme, die auf dem modellbasierten Ansatz aufbauen, sind für die Diagnose von technischen Artefakten gedacht. Die Bezeichnung „Anlage“ wird deshalb synonym für „System“ oder auch „Gerät“ verwendet. Der Schlüssel zum modellbasierten Ansatz mit funktionalen Modellen ist die Darstellung der Struktur und des Verhaltens der korrekt funktionierenden Anlage. Mittels dieser Darstellung werden Vorhersagen über das Verhalten der realen Anlage und über die Ergebnisse von möglichen Messungen getroffen. Abweichungen zwischen dem vorhergesagten Verhalten und den tatsächlichen Beobachtungen werden Symptome genannt, die auf Mengen von möglichen fehlerhaften Komponenten zurückgeführt werden müssen. Wir nehmen an, daß die Komponentenfehler, die den Symptomen zugrundeliegen, nicht direkt beobachtet werden können, sondern statt dessen indirekt über Beobachtungen des Verhaltens der Anlage hergeleitet werden müssen.

Jede Menge von Komponenten, deren Defekt die Beobachtungen erklären kann, wird ein Kandidat genannt. Es kann für eine Menge von Symptomen mehrere Erklärungen und damit mehrere Kandidaten geben. Diese werden entsprechend ihrer relativen Wahrscheinlichkeit bewertet. Um die Menge der möglichen Kandidaten einzugrenzen, werden weitere Beobachtungen gemacht, so daß schließlich (im Idealfall) ein Kandidat die anderen dominieren und als endgültige Diagnose ausgegeben wird.

Der wesentliche Vorteil bei der funktionalen Modellierung ist, daß durch die Verwendung von Wissen über das korrekte Komponentenverhalten die globalen Beziehungen zwischen den zugrundeliegenden Fehlern und dem beobachteten Fehlverhalten der Anlage nicht benötigt werden. Bei diesem Ansatz wird vielmehr gefolgert, daß jede Teilmenge von Komponenten, deren kombiniertes vorhergesagtes Verhalten mit dem tatsächlich beobachteten Verhalten nicht übereinstimmt, wenigstens eine de-

fekte Komponente enthält. Durch die Sammlung weiterer Beobachtungen kann das Diagnoseprogramm diese Menge verkleinern.

Darüber hinaus erfordert dieser Ansatz keinerlei Festlegung bezüglich der Anzahl der vorhandenen Fehler in der Anlage. Das Modell unterstützt Schlußfolgerungen über die Wechselwirkungen zwischen jeder Anzahl von fehlerhaften Komponenten.

Bei dem funktionalen Ansatz besteht die gesamte diagnostische Aufgabe aus fünf Teilaufgaben: Modellbildung, Verhaltensvorhersage, Konflikterkennung, Kandidatenerzeugung und Meßpunktvorschlag (bzw. Kandidatendiskriminierung). Von den fünf Teilaufgaben bilden die letzten vier den Diagnoseprozeß nach der GDE von de Kleer und Williams. Die Modellbildung ist vor dem Beginn der Diagnose abgeschlossen und gehört daher nicht zum eigentlichen Diagnoseprozeß. Im folgenden Abschnitt wird die Arbeitsweise der GDE an einem Beispiel beschrieben und dabei jede der obigen Aktivitäten diskutiert.

3.1 Die GDE

In diesem Abschnitt werden die grundlegenden Prinzipien der GDE („General Diagnostic Engine“) an einem sehr einfachen hydraulischen System beschrieben. Die GDE ist ein domänenunabhängiges Werkzeug [de Kleer & Williams 1987]. Die GDE benötigt ein Modell des Systems, das aus den Komponentenmodellen der im System enthalten Komponenten entsprechend der Systemstruktur zusammengesetzt ist („component connection modelling“). Die Komponentenmodelle sind durch Constraints dargestellt, welche die Relationen zwischen den physikalischen Größen an den Anschlüssen einer Komponente aufrechterhalten, indem die Constraints für bestimmte Größen Werte von bekannten Werten anderer Größen herleiten (Abschnitt 6.2) Im Modell wird für jede Komponente explizit die Annahme getroffen, daß diese Komponente korrekt funktioniert. Diese Annahme geht als zusätzliche Bedingung in die zugehörigen Constraints der Komponente ein und rechtfertigt die Verwendung eines Constraints bei der Bestimmung des korrekten Komponentenverhaltens.

Die GDE führt iterativ die folgenden Schritte durch [de Kleer & Williams 1987]:

- Verhaltensvorhersage: Mit der Annahme, daß jede Komponente korrekt (intakt) ist, berechne Werte für die einzelnen physikalischen Größen im System.

- Konflikterkennung: Bestimme diejenigen Komponenten-Korrektheitsannahmen, die der Berechnung von widersprüchlichen Werten einer bestimmten Größe zugrundeliegen.

- Kandidatenerzeugung: Konstruiere die Mengen derjenigen Komponenten, deren Korrektheitsannahme die vorhandenen Konflikte hervorruft, und welche die Konflikte auflösen, wenn ihre Korrektheitsannahmen gemeinsam (d.h. konjunktiv) zurückgezogen werden.
- Meßpunktvorschlag (Kandidatendiskriminierung): Produziere einen Vorschlag für die nächste Messung an dem System, die am „besten“ zwischen den Kandidaten unterscheiden hilft. Die beste nächste Messung ist die, welche im Mittel zur Ermittlung der korrekten Diagnose in einer minimalen Anzahl von Messungen führt.

In Abschnitt 3.2 wird eine exakte formale Beschreibung der diagnostischen Aufgabe gegeben. Zunächst wollen wir uns jedoch die prinzipielle Arbeitsweise der GDE an einem Beispiel verdeutlichen. Um die wesentlichen Konzepte dabei besser herausstellen zu können, betrachten wir zunächst ein triviales hydraulisches System, welches aus nur zwei Leitungen besteht.

3.1.1 Verhaltensvorhersage

Die Verhaltensvorhersage in dem Modell geschieht durch eine Simulation des Verhaltens. Das Modell der Anlage wird dabei durch ein Netz von Constraints dargestellt. Die Verhaltenssimulation wird mit Hilfe der Constraint-Propagierung erreicht, wobei das korrekte Verhalten der gesamten Anlage auf der Basis der lokalen Propagierung des korrekten Verhaltens einzelner Komponenten ermittelt wird. Wir werden uns in Kapitel 6 noch ausführlich mit der Erzeugung des Constraintnetzes und der Constraint-Propagierung bei hydraulischen Systemen beschäftigen. Für den Moment sei angenommen, wir haben einen Mechanismus zur Verhaltensvorhersage in dem Modell anhand der Beobachtungen und vorgebenen Systemgrößen.

Angenommen, der einströmende Fluß in ein System aus zwei zusammengefügten Leitungen beträgt 6 l/s (Abbildung 3.1). Der zu erwartende Fluß an Punkt C kann dann im Modell durch die lokale Propagierung des Verhaltens der Einzelleitungen berechnet werden und beträgt 6 l/s unter der Annahme, daß jede Leitung korrekt ist. Jeder lokal vorhergesagte Wert hat einen Label, der die Mengen von Komponenten angibt, von deren korrektem Verhalten dieser Wert abhängt. Zum Beispiel hat „C = 6 l/s“ den Label $\{\{L_1, L_2\}\}$, wobei L_1 für die Annahme „Leitung L_1 ist korrekt (nicht defekt)“ steht, und so weiter. Der Label eines Wertes enthält i.a. mehr als eine Menge von Komponenten, denn bei größeren Netzwerken gibt es oft verschiedene Wege von Komponenten und Leitungen, über die der Wert an einer bestimmten Stelle hergeleitet werden kann. Jede Menge im Label enthält die Komponenten auf

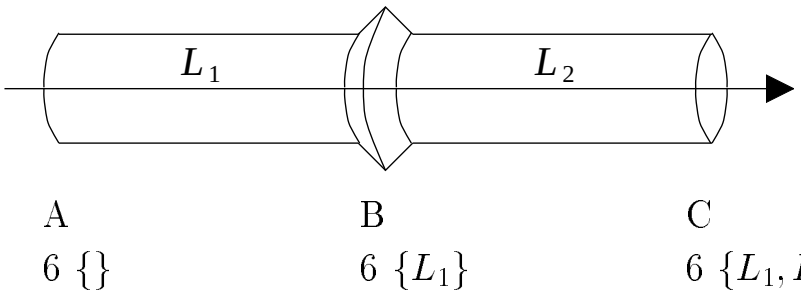
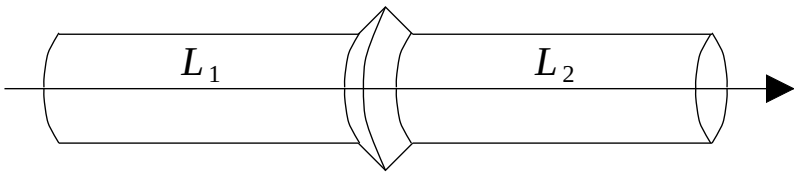


Abbildung 3.1: **Verhaltensvorhersage des Flusses an Punkt B und C bei Vorgabe des Flusses an Punkt A. C ist 6 l/s, wenn Leitung L_1 und L_2 intakt sind.**

einem bestimmten (Inferenz-)Weg. Wozu werden Label benötigt? Die Informationen im Label eines Wertes haben den Zweck, daß im Falle der Beobachtung eines widersprüchlichen Wertes am realen Gerät die verantwortlichen Komponenten leicht gefunden werden können. Wie noch gezeigt wird, ist es dabei wesentlich, für jeden Wert in seinem Label nur die minimalen Mengen von Komponenten zu speichern, auf deren Korrektheit dieser Wert beruht. Die Minimalität einer Menge im Label eines Wertes bedeutet also, daß keine echte Teilmenge von Komponenten dieser Menge ausreicht, um den Wert im Modell an dieser Stelle herleiten zu können.

Eingangsgrößen, die dem System von außen vorgegeben sind — wie im Beispiel der einströmende Fluß von 6 l/s — erfordern keine Korrektheitsannahmen von Komponenten. Deshalb enthält ihr Label nur die leere Menge. Jede der minimalen Mengen im Label eines Wertes wird eine minimale Umgebung genannt. Ein Wert ist in einer bestimmten Umgebung gültig, wenn der Wert aus den Verhaltensbeschreibungen der Komponenten in dieser Umgebung zusammen mit den gegebenen Eingangsgrößen hergeleitet werden kann.

Zu beachten ist, daß die Verhaltensvorhersage im Modell nicht auf die Richtung der Schlußfolgerung von den Eingängen einer Komponente zu den Ausgängen beschränkt ist. Unter der Voraussetzung, daß sich das Verhalten der Komponenten linear beschreiben läßt, kann das Modell jede logische Beziehung zwischen Größen an den Ports einer Komponente herstellen, insbesondere entgegen der physikalischen Werteausbreitungsrichtung. Wenn im Beispiel nur der ausströmende Fluß gemessen wurde, so kann im Modell — unter Verwendung der Korrektheitsannahmen der einzelnen Leitungen — auf die Menge einströmenden Flusses geschlossen werden. Der Grund dafür liegt darin, daß in dem Constraint-Netz, welches das Modell beschreibt, die Wertepropagierung „in alle Richtungen“erfolgt. Auf diesen Punkt wird in Abschnitt 6.2 noch eingegangen.



A	B	C
6 $\{\}$	6 $\{L_1\}$	6 $\{L_1, L_2\}$
	4 $\{L_2\}$	4 $\{\}$

Abbildung 3.2: **Die Verhaltensvorhersage kann im Modell in alle Richtungen erfolgen. Wird an Punkt C der Fluß von 4 l/s beobachtet, läßt sich daraus auf den Fluß 4 l/s an Punkt B schließen — unter Annahme der Korrektheit von Leitung L_2.**

3.1.2 Konflikterkennung

Wir nehmen nun an, daß an dem realen System der drei Leitungen am Ausgang C ein Fluß von 4 l/s beobachtet beziehungsweise gemessen worden ist, und daß diese Werte als Fakten in das Diagnosesystem eingegeben worden sind (siehe Abbildung 3.2). Wird die Beobachtung mit der Vorhersage überlagert, so wird deutlich, daß im Modell an Punkt B der Fluß mit 6 l/s vorhergesagt wird, wenn Leitung L_1 in Ordnung ist, aber mit 4 l/s, wenn Leitung L_2 korrekt funktioniert. Dies ist ein Beispiel für die rückwärtige Wertepropagierung im Modell.

Da für C ein anderer Wert beobachtet wurde als vorhergesagt, ist „C = 4 l/s“ ein Symptom. Eine der Annahmen über die Korrektheit der Komponenten, welche der Berechnung des vorhergesagten Wertes „C = 6 l/s“ zugrundeliegen, muß demnach falsch gewesen sein. Die Menge $\{L_1, L_2\}$ ist also ein Konflikt. Eine dieser Komponenten muß tatsächlich defekt sein, um die Beobachtung erklären zu können.

In komplizierteren Systemen kann ein einzelnes Symptom zu einer Vielzahl von Konflikten führen. Die Menge aller Komponenten des Systems stellt immer einen Konflikt dar, sobald irgendein Symptom beobachtet worden ist. Um die Komplexität der Kandidatenberechnung zu reduzieren, ist es daher entscheidend, die Menge aller Konflikte so knapp wie möglich zu beschreiben. Ein Konflikt ist eine Menge von Komponenten, die einen Widerspruch mit den Beobachtungen bilden, wenn sie alle gleichzeitig als korrekt angenommen werden. Durch das Hinzufügen weiterer Komponenten bleibt der Widerspruch bestehen. Deshalb ist jede Obermenge eines Konflikts ebenfalls ein Konflikt. Zur Beschreibung aller Konflikte reicht es demnach aus, nur die minimalen Konflikte zu identifizieren, wobei ein Konflikt minimal ist,

wenn keine echte Teilmenge ebenfalls ein Konflikt ist. Die Menge aller minimalen Konflikte zu bestimmen ist die Aufgabe der Konflikterkennung bei der GDE.

3.1.3 Kandidatenerzeugung

Ein Kandidat für eine Diagnose ist eine Menge von Komponenten, die alle Symptome erklären können, wenn jede der Komponenten tatsächlich defekt ist. Damit ein Kandidat jedes Symptom erklären kann, muß dieser wenigstens eine Komponente aus jedem Konflikt enthalten, d.h. ein Kandidat muß die aufgetretenen Konflikte überdecken. Bei dem Diagnoseprozeß werden meistens die einfachsten Erklärungen für die Symptome gesucht, woraus die Forderung nach der Minimalität der Kandidaten resultiert. Wenn es zum Beispiel nur einen Konflikt gibt ($\{L_1, L_2\}$), dann gibt es zwei einelementige minimale Kandidaten $\{L_1\}$ und $\{L_2\}$. Sind die beiden Leitungen Teil eines größeren hydraulischen Systems, dann kann die korrekte Diagnose zusätzliche Komponenten enthalten, da es weitere defekte Komponenten im Schaltkreis geben kann. Jedoch können die Mengen $\{L_1\}$ und $\{L_2\}$ nicht verkleinert werden und dennoch den Konflikt erklären. Deshalb sind $\{L_1\}$ und $\{L_2\}$ minimale Kandidaten. Die Kandidatenerzeugung hat das Ziel, alle minimalen Kandidaten zu finden.

Das Schema der Kandidatenerzeugung auf der Grundlage der minimalen Konflikte ermöglicht die Behandlung von Mehrfachfehlern in natürlicher Weise. Nehmen wir für dieses Beispiel einmal an, eine Flußmessung sei an Punkt B möglich, und wir messen den Fluß an dieser Stelle mit 5 l/s. Wir erhalten dann zwei minimale Konflikte $\{L_1\}$ und $\{L_2\}$, und $\{L_1, L_2\}$ ist der einzige minimale Kandidat, der beide minimalen Konflikte erklärt. Der minimale Kandidat $\{L_1, L_2\}$ beschreibt also, daß die Symptome erklärt würden, wenn beide Leitungen im realen System fehlerhaft (leck) wären. Ein Mehrfachfehler-Kandidat für die Diagnose deutet auf das gleichzeitige Vorliegen mehrerer verschiedener Komponentendefekte hin.

Grundsätzlich ist ein Kandidat eine spezielle Hypothese darüber, in welcher Weise sich das reale System von seinem Modell unterscheidet. Bei hydraulischen Systemen, um die es in dieser Arbeit geht, ist ein Kandidat eine Menge von defekten Komponenten, wobei die Komponenten, die nicht in diesem Kandidaten enthalten sind, als korrekt betrachtet werden. Der „leere“ Kandidat $\{\}$ kann interpretiert werden als „alle Komponenten funktionieren korrekt“.

Bei der Berechnung der minimalen Kandidaten sind zwei Probleme zu bewältigen. Erstens muß für die Bestimmung eines minimalen Kandidaten ein minimales Hitting Set (Mengenüberdeckung) über die Menge der bekannten minimalen Konflikte

konstruiert werden (Abschnitt 3.2.2) und die Berechnung eines minimalen Hitting Sets ist NP-schwer [Garey & Johnson 1979]. Zweitens wächst i.a. die Anzahl der verschiedenen minimalen Kandidaten exponentiell mit der Anzahl der minimalen Konflikte. Nehmen wir zum Beispiel ein hypothetisches System mit $2n$ Komponenten und entsprechend $2n$ Korrektheitsannahmen. Sei weiter angenommen, die beobachteten Symptome seien derart, daß es n minimale Konflikte gibt, einen minimalen Konflikt für jedes Paar von Komponenten $2i$ und $2i + 1$. Die Anzahl der daraus resultierenden minimalen Kandidaten ist 2^n. In Kapitel 4 werden Verfahren untersucht, die minimalen Kandidaten zu berechnen, bzw. effiziente Näherungslösungen zu bestimmen.

3.1.4 Meßpunktvorschlag

Das Ziel des Diagnoseprozesses ist, eine möglichst eindeutige Diagnose zu finden. Im Verlauf der Diagnose gibt es gewöhnlich verschiedene minimale Kandidaten, die alle bekannten Symptome erklären können. Um zwischen diesen unterscheiden zu können, werden weitere Beobachtungen benötigt. Grundsätzlich gibt es dafür zwei Möglichkeiten:

1. Es werden weitere Beobachtungen an der Anlage in ihrem derzeitigen Zustand gemacht.

2. Es werden Beobachtungen anhand anderer Betriebsparameter gemacht, d.h. die Anlage wird in einen anderen Betriebsmodus versetzt, durch Veränderung der äußeren Vorgaben (andere Ventilstellung, etc.).

Im obigen Beispiel war nur eine weitere Beobachtung möglich (an Punkt B). Im allgemeinen gibt es jedoch sehr viele mögliche Meßpunkte. Messungen sind teuer, und deshalb wollen wir mit einer möglichst geringen Anzahl von Messungen die korrekte Diagnose finden. Man stelle sich zum Beispiel eine Pipeline bestehend aus tausend Teilstücken vor, an deren Ende weniger Fluß ankommt als am Anfang hineinfließt. Unter der Annahme, daß an jedem Verbindungspunkt zweier Leitungen der Fluß gemessen werden kann — welcher ist der beste Punkt für die Messung? Abhängig davon, wo der jeweils nächste Meßpunkt gewählt wird, gibt es sehr große Unterschiede in der mittleren Anzahl der notwendigen Messungen, um das Leck zu finden. De Kleer und Williams schlagen vor, den Meßpunkt mit dem größten durchschnittlichen Informationsgewinn zu wählen. Eine schlechte Vorgehensweise ist, an der ersten Nahtstelle zu messen, um zu prüfen, ob die erste Leitung undicht ist und dann diesen Prozeß an Nahtstelle zwei, drei, vier, usw. zu wiederholen. Im Mittel

werden auf diese Weise 500 Messungen benötigt. Wird dagegen der nächste Meßpunkt nach dem Prinzip der binären Suche gewählt, kann die defekte Leitung mit maximal 10 Messungen gefunden werden.

Ein grundsätzliches Problem bei der Bestimmung des nächsten Meßpunktes ist, daß der Meßpunkt, der zum gegebenen Zeitpunkt zwischen den Kandidaten am besten unterscheiden hilft, nicht unbedingt derjenige ist, durch den die Anzahl der insgesamt notwendigen Messungen zur Fehlerfindung minimiert wird. Vom Ausgang der nächsten Messung hängt es ab, welche weiteren Messungen erforderlich werden. Es ist möglich, daß die Messung an dem aktuell besten Meßpunkt in der Folge mehr Messungen nach sich zieht als die Messung an einem aktuell weniger guten Meßpunkt. Der Ausgang einer Messung ist jedoch nicht im voraus bekannt, deshalb kann ein optimaler Meßpunkt, der die Gesamtanzahl aller Messungen minimiert, i.a. nicht bestimmt werden. De Kleer und Williams schlagen vor, in einer Ein-Schritt-Vorschau alle vorhergesagten Ausgänge für jeden einzelnen Meßpunkt zu betrachten und dann den Meßpunkt mit dem größten durchschnittlichen Informationsgewinn zu wählen. In [de Kleer 1990] wird die folgende Gleichung angegeben, nach der berechnet werden kann, welcher Informationsgewinn G von einer Messung an Punkt X zu erwarten ist:

$$G(\mathrm{X}) = \sum c_i \ln c_i.$$

Hierbei ist c_i die Anzahl der minimalen Kandidaten, die mit dem i-ten vorhergesagten Ausgang dieser Messung konsistent sind. Die Summe läuft dabei über alle vorhergesagten Ausgänge der Messung. Den größten durchschnittlichen Informationsgewinn liefert die Messung, bei der G minimal ist.

In dem Zwei-Leitungen-Beispiel werden für eine Messung an Punkt B zwei Werte vorhergesagt: entweder ist B = 6 l/s (wenn L_1 in Ordnung ist), oder es ist B = 4 l/s (wenn L_2 in Ordnung ist). Eine Messung von 4 l/s ist mit dem Kandidaten $\{L_1\}$ konsistent, und 6 l/s ist mit dem Kandidaten $\{L_2\}$ konsistent. In diesem Beispiel ist die Berechnung von $G(\mathrm{B})$ trivial: $G(\mathrm{B}) = 1 \ln 1 + 1 \ln 1 = 0$.

3.2 Formale Beschreibung der Diagnose

Wir haben in dem letzten Kapitel die grundlegenden Konzepte der modellbasierten Diagnose intuitiv und an Beispielen definiert. Können diese Konzepte präzise definiert werden? Werden mit dem modellbasierten Ansatz tatsächlich die korrekten Diagnosen berechnet? Um Fragen wie diese beantworten zu können, und um wichtige Zusammenhänge bei der beabsichtigten Implementierung nicht zu übersehen, wollen wir im folgenden die Theorie der modellbasierten Diagnose formal definieren.

Eine ausführlichere Behandlung dieser Formalisierung findet sich in [Reiter 1987] und [de Kleer et al. 1992].

3.2.1 Theorie der Diagnose nach Reiter

Die hier dargestellte Theorie der Diagnose wurde von [Reiter 1987] entworfen. Reiter formalisiert damit den modellbasierten Diagnoseansatz, den er *„Diagnosis from First Principles“* nennt, weil bei der Fehlersuche in einem System auf die Struktur- und Funktionsbeschreibung des Systems zurückgegriffen wird. Entscheidend ist hierbei, daß sich die Funktion des Gesamtsystems auf der Basis der lokalen Verhaltensweisen seiner einzelnen Komponenten beschreiben läßt. Für die Formulierung des Diagnoseproblems sowie der Beschreibung der Komponentenmodelle verwendet Reiter die Prädikatenlogik erster Ordnung.

Die Definition des zu untersuchenden Systems umfaßt die zur Durchführung der Diagnose benötigten Informationen über die Anlage, wie z.B. Beobachtungen, die Anlagenstruktur, Verhaltensbeschreibungen der Komponenten, und so weiter. Die folgende Definition eines Systems stammt aus [Forbus & de Kleer 1993]:

Definition: *Ein System ist ein Tripel* (SD,COMPS,OBS) *mit folgenden Eigenschaften:*

1. SD *ist die Systembeschreibung (System Description) in Form einer Menge prädikatenlogischer Formeln erster Ordnung.*
2. COMPS *bezeichnet die in dem System enthaltenen Komponenten.* COMPS *ist eine endliche Menge von Konstanten.*
3. OBS *ist eine Menge von Beobachtungen, beschrieben durch eine Menge prädikatenlogischer Formeln erster Ordnung.*

In seiner Theorie folgt Reiter der Konvention, eine Diagnose durch eine Menge von fehlerhaften Komponenten zu beschreiben, so daß die Komponenten im Komplement dieser Menge implizit als korrekt betrachtet werden.

Die Korrektheit einer Komponente $c \in$ COMPS wird durch den Term $\neg$AB(c) ausgedrückt, wobei AB(c) ein Prädikat ist, welches gilt, wenn die Komponente c „abnormal“ funktioniert. Eine Diagnose ist deshalb durch eine Menge von AB(c_i) Termen beschrieben, für Komponenten $c_i \in$ COMPS.

Die Systembeschreibung SD ist für eine Anlage relativ umfangreich, weil sie Axiome für Gleichheit und Arithmetik enthalten muß [Forbus & de Kleer 1993]. Die wesentlichen prädikatenlogischen Formeln beschreiben das korrekte Verhalten der

Komponenten. Das korrekte Verhalten eines Addierers in digitalen Schaltkreisen zum Beispiel würde in der Systembeschreibung SD durch die folgende Formel beschrieben werden:

$\text{ADDIERER}(x) \rightarrow [\neg\text{AB}(x) \rightarrow out(x) = in1(x) + in2(x)]$

$\text{MULTIPLIER}(x) \rightarrow [\neg\text{AB}(x) \rightarrow out(x) = in1(x) \times in2(x)]$

Diese Formel kann wie folgt gelesen werden: Wenn x ein Addierer ist, dann gilt: wenn x nicht abnormal funktioniert, dann ist der Output des Addierers die Summe seiner beiden Inputs.

Das korrekte Verhalten eines Systems, bestehend aus den Komponenten COMPS = $\{c_1, ..., c_n\}$, wird durch die folgende Formel beschrieben:

$$\text{SD} \cup \{\neg\text{AB}(c_1), ..., \neg\text{AB}(c_n)\}\,.$$

Die Terme $\neg\text{AB}(c_i)$ entsprechen dabei den Korrektheitsannahmen der Komponenten in der GDE.

Das reale technische System verhält sich fehlerhaft; die Beobachtungen (beschrieben in OBS) stimmen nicht mit dem korrekten Verhalten überein, d.h. die folgende Formel ist inkonsistent:

$$\text{SD} \cup \{\neg\text{AB}(c_1), ..., \neg\text{AB}(c_n)\} \cup \text{OBS}.$$

Wenn für einen bestimmten Parameter im System ein anderer Wert beobachtet wird als sich aufgrund der Systembeschreibung ergeben müßte, dann stellen die Korrektheitsannahmen für die Komponenten, deren korrektes Verhalten in die Berechnung des vorhergesagten Wertes eingeht, einen Konflikt dar. Diese Korrektheitsannahmen können offensichtlich nicht alle gleichzeitig wahr sein; nicht alle diese Komponenten können korrekt funktionieren. Anders ausgedrückt: Wenigstens eine der Korrektheitsannahmen muß falsch gewesen sein, um die Beobachtung erklären zu können.

Eine Konfliktmenge $C = \{c_1, ..., c_k\}$ bezeichnet bei Reiter eine Teilmenge von COMPS, so daß $\text{SD} \cup \{\neg\text{AB}(c_1), ..., \neg\text{AB}(c_k)\} \cup \text{OBS}$ inkonsistent ist. Die Konfliktmengen Reiters entsprechen den Konflikten bei de Kleer. Eine Konfliktmenge ist minimal, wenn sie keine echte Teilmenge enthält, die ebenfalls eine Konfliktmenge ist. Das bedeutet: für eine minimale Konfliktmenge C ist die folgende Formel für jede echte Teilmenge von C konsistent:

$$\text{SD} \cup \{\neg\text{AB}(c) \,|\, c \in C' \text{und} C' \subset C\} \cup \text{OBS}.$$

Andererseits ist jede Obermenge einer Konfliktmenge ebenfalls eine Konfliktmenge.

Im allgemeinen werden in einem fehlerhaften System mehrere Parameterwerte von ihrem korrekten Wert abweichen, so daß es mehr als eine Konfliktmenge geben wird.

Das Ziel des Diagnoseprozesses ist, eine Diagnose zu finden, die alle vorhandenen Abweichungen (bzw. alle vorhandenen Konfliktmengen) erklären kann.

Intuitiv ist eine Diagnose eine Menge von Komponenten, deren Korrektheitsannahme zurückgezogen wird. Die Korrektheitsannahme einer Komponente zurückzuziehen bedeutet, diese als abnormal (d.h. defekt) anzusehen. Die zugrundeliegende Idee ist, daß die Negierung der Korrektheitsannahmen der Komponenten aus der Diagnose alle Inkonsistenzen mit den Beoachtungen auflöst.

Definition: *Eine Diagnose für* (SD,COMPS,OBS) *ist eine Menge* $\Delta \subseteq$ COMPS, *für die folgende Formel konsistent ist:*

$$\mathrm{SD} \cup \{\mathrm{AB}(c) \,|\, c \in \Delta\} \cup \{\neg \mathrm{AB}(c) \,|\, c \in \mathrm{COMPS} - \Delta\} \cup \mathrm{OBS}.$$

Im Unterschied zu Reiter bezeichnet de Kleer nur die endgültige Diagnose als solche. Bei de Kleer ist der Diagnoseprozeß inkrementell: Mit jeder neuen Beobachtung wird die bisher gefundene Diagnose verbessert. Jede Zwischendiagnose wird deshalb bei de Kleer ein Kandidat genannt, der eine vorläufige Hypothese für die Diagnose darstellt.

Für ein Diagnoseproblem kann es $2^{|\mathrm{COMPS}|}$ verschiedene Diagnosen geben. Die Definition einer minimalen Diagnose entspricht der Notwendigkeit, möglichst einfache Erklärungen zu haben.

Definition: *Eine Diagnose* Δ *ist minimal, wenn keine echte Teilmenge* $\Delta' \subset \Delta$ *ebenfalls eine Diagnose ist.*

Enthält eine minimale Diagnose mehrere Elemente, so bedeutet dies, daß mehrere Komponenten gleichzeitig defekt sein müssen, um die Beobachtungen zu erklären. Eine minimale Diagnose repräsentiert dann Mehrfachfehler im System.

3.2.2 Bestimmung der Diagnosen bei Reiter

Wie können die minimalen Diagnosen bestimmt werden? Es kann zunächst festgestellt werden: Wenn in jeder Konfliktmenge eine Komponente tatsächlich defekt wäre, dann würde dies die Inkonsistenz jeder Konfliktmenge auflösen. Demnach muß jede Diagnose aus jeder Konfliktmenge wenigstens ein Element enthalten, um die Inkonsistenzen aufzulösen. Informell sagen wir: Jede Diagnose muß alle vorhandenen Konfliktmengen überdecken.

Reiter hat gezeigt, daß jede minimale Diagnose für (SD,COMPS,OBS) ein minimales Hitting Set für die Menge der minimalen Konfliktmengen ist. Unglücklicherweise ist die Berechnung eines minimalen Hitting Sets ein NP-schweres Problem [Garey &

Johnson 1979]. Nach der Definition von [Garey & Johnson 1979] ist die Menge H ein Hitting Set für eine Menge C von Teilmengen einer Menge S, $C = \{S_1, ..., S_n\}$, wenn gilt:

(i) $H \subseteq S$

(ii) $H \cap S_i \neq \{\} \quad \forall\, S_i, 1 \leq i \leq n.$

Wiederum ist ein Hitting Set H minimal, wenn keine echte Teilmenge von H ebenfalls ein Hitting Set ist.

Reiter schlägt ein Verfahren zur Berechnung der minimalen Diagnosen gemäß der obigen Definition vor. Zuerst werden alle minimalen Konfliktmengen für (SD,COMPS,OBS) berechnet, indem ein Theorembeweiser zur Konsistenzprüfung verwendet wird. Als nächstes werden alle minimalen Hitting Sets für die Menge aller minimalen Konfliktmengen berechnet. Dies geschieht durch die sogenannte H-P-Tree Methode [Reiter 1987], einer Art Breitensuche in einem speziellen, baumförmigen Suchraum.

Um Reiters *„Diagnosis from First Principles“* auf kontinuierliche physikalische Systeme wie hydraulische Anlagen anwenden zu können, sind ein angemessenes Modell und Theorembeweiser für die Beschreibung des Systemverhaltens notwendig. [Ng 1990] verwendet Kuipers qualitativen Simulator QSIM [Kuipers 1986] als einen Theorembeweiser. Jedoch erweist sich QSIM für realistische Systemgrößen als zu aufwendig [Ng 1990] und wird deshalb hier nicht weiter verfolgt.

Set-Covering [Peng & Reggia 1990] ist ein weiterer Ansatz, für eine gegebene Menge von Symptomen die minimalen Hitting Sets (und damit die Diagnosen) zu ermitteln. Dieser Ansatz beruht auf der Verwendung von Fehlermodellen (siehe Einleitung). Die Voraussetzung für diesen Ansatz ist deshalb, daß die kausalen Beziehungen zwischen den zugrundeliegenden Fehlern (Defekten) und den Symptomen klar definiert sind.

3.2.3 Die grundlegende Idee des Set-Covering

Die Grundidee der *Parsimonious Covering Theory* von Peng und Reggia ist (informell) die folgende: Ein bestimmtes Symptom kann i.a. durch verschiedene Fehler verursacht werden. Durch die sogenannte Ursachenfunktion wird jedem Symptom die Menge der möglichen zugrundeliegenden Fehler zugeordnet. Umgekehrt kann ein bestimmter Fehler mehrere verschiedene Symptome hervorrufen. Dies wird durch eine Wirkungsfunktion beschrieben, die jedem Fehler die Menge der resultierenden Auswirkungen (Symptome) zuordnet.

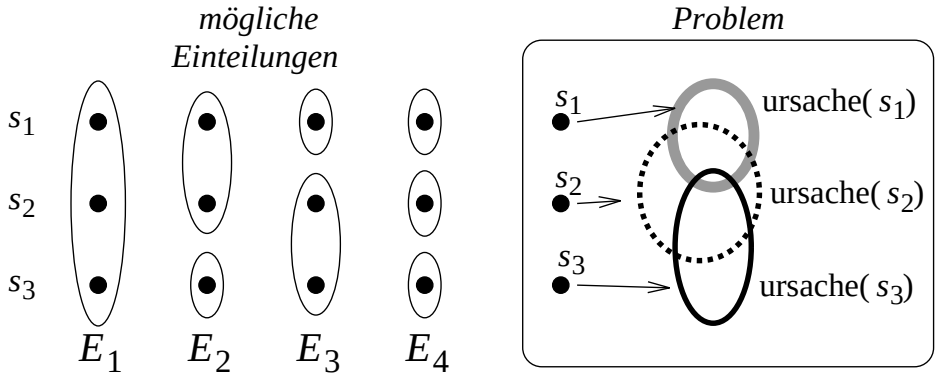


Abbildung 3.3: **Jedes Symptom wird durch eine Menge von Fehlern verursacht. Es gibt vier Möglichkeiten für die Teilmengen, in der die Fehler liegen, welche die Symptome s_1, s_2, s_3 verursachen. Entsprechend lassen sich die Symptome in vier Gruppen einteilen. Bei der Set-Covering-Theorie werden die möglichen Einteilungen berechnet.**

Die Symptome lassen sich nun danach einteilen, durch welche unterschiedlichen Mengen von Fehlern diese verursacht werden können. Ein Beispiel soll dies verdeutlichen. Die Abbildung 3.3 zeigt ein Problem mit drei Symptomen $\{s_1, s_2, s_3\}$. In einem Extrem können sich alle Symptome auf dieselbe Gruppe von Fehlern beziehen, nämlich auf die Fehler im Durchschnitt der drei Mengen ursache(s_1), ursache(s_2), und ursache(s_3). In diesem Fall kann ein einziger Fehler alle Symptome erklären. Dies ist in Abbildung 3.3 durch die Einteilung der Symptome in eine Gruppe (E_1) dargestellt. Im anderen Extrem sind für Erklärung der Symptome drei Fehler aus drei verschiedenen Fehlergruppen erforderlich. Dies entspricht der Symptomeinteilung in drei verschiedene Gruppen (Einteilung E_4). Die Anzahl der Gruppen in E_i entspricht dabei der Anzahl gleichzeitig auftretender Fehler. In diesem Beispiel gibt es insgesamt vier mögliche Einteilungen. Man beachte, daß die Zwei-Gruppen-Einteilung, bei der s_1 und s_3 eine Gruppe bilden und s_2 eine andere, unmöglich ist, da ursache(s_1) $\cap$ ursache(s_3) eine Teilmenge von ursache(s_2) ist.

Bei der Set Covering Theorie werden alle diese möglichen Einteilungen berechnet. Die Einteilungen entsprechen den Generatoren bei Peng und Reggia.

3.2.4 Die Formalisierung durch Peng und Reggia

Im Ansatz von Peng und Reggia wird das Problem der Mehrfachfehler-Diagnose durch ein Vier-Tupel (D, M, C, M^+) charakterisiert, das wie folgt definiert ist:

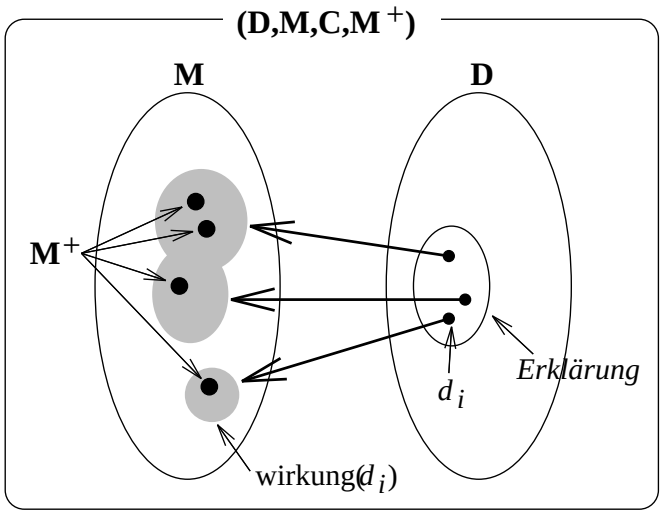


Abbildung 3.4: **Eine Erklärung für das Diagnoseproblem (D, M, C, M^+).**

D ist eine endliche nichtleere Menge von Fehlern („disorders“).

M ist eine endliche nichtleere Menge von Symptomen („manifestations“).

C ist eine Relation, die eine Teilmenge von $D \times M$ ist. Diese Relation verbindet Fehler mit Symptomen. Die Bedeutung von $(d, m) \in C$ ist, daß der Fehler d das Symptom m verursachen kann (aber nicht muß).

M^+ ist eine Teilmenge von M, welche die beobachteten Symptome kennzeichnet. Man beachte, daß Symptome, die nicht in M^+ enthalten sind, als abwesend (d.h. nicht vorhanden) betrachtet werden.

Desweiteren gibt es eine Ursachen- und eine Wirkungsfunktion:

$$\text{ursache}(m) \subseteq D \quad , \quad m \in M$$
$$\text{wirkung}(d) \subseteq M \quad , \quad d \in D.$$

Die *Lösung* für ein diagnostisches Problem (D, M, C, M^+) ist wie folgt definiert:

Definition: *Die* Lösung *für ein diagnostisches Problem* (D, M, C, M^+) *ist die Menge aller* Erklärungen *von* M^+. *Eine* Erklärung *von* M^+ *für* (D, M, C, M^+) *ist eine Menge* $\{d_1, ..., d_n\} \subseteq D$, *wobei*

$$\bigcup_{i=1}^{n} \text{wirkung}(d_i) \subseteq M^+.$$

Eine Erklärung von M^+ ist minimal, wenn diese keine echte Teilmenge enthält, die ebenfalls M^+ erklärt. Wie bei M^+ werden Fehler, die nicht in einer Erklärung enthalten sind, als abwesend betrachtet.

Eine minimale Erklärung korrespondiert mit einem minimalen Hitting Set in Reiters Formalisierung. Die Abbildung 3.4 zeigt eine Erklärung für (D, M, C, M^+).

3.2.5 Diskussion

Die Diagnose wird bei Peng und Reggia beschrieben als das Problem, aus einer vorgegebenen Menge von Fehlern diejenigen Fehler auszuwählen, welche die gegebenen Symptome am besten erklären. Peng und Reggia setzen bei ihrer Definition der Diagnose voraus, daß die möglichen Fehler bereits vor Beginn der Diagnose bekannt sind, und daß die Menge der Fehler vollständig ist. In der Hydraulik, wie auch in der Medizin und in anderen Domänen, ist das Verhalten des zu untersuchenden Systems i.a. sehr komplex und die möglichen Fehler sehr vielfältig, so daß die Vollständigkeit der Fehlermenge schwer zu garantieren ist. Wenn ein Symptom vorhanden ist, welches nicht durch einen bekannten Fehler erklärt werden kann, scheitert dieser Ansatz. Dies ist ein wesentlicher Nachteil gegenüber der Diagnose mit funktionalen Modellen, wie sie in der Theorie von Reiter definiert ist. Reiters Diagnoseansatz ist flexibler, weil bei seiner Definition der Diagnose keinerlei Wissen über die möglichen Fehler benötigt wird. Jede strukturelle Abweichung des realen Systems von seinem Modell, welche die Symptome erklärt, ist als Fehler möglich. Strukturelle Abweichungen werden dabei auf der Grundlage einer Simulation des korrekten Verhaltens in dem Modell des Systems entdeckt.

Ein gravierender Nachteil des Ansatzes von Peng und Reggia ist außerdem, daß dieser das Vorliegen einer geschlossenen Diagnosesituation unterstellt. Damit ist gemeint, daß bei Beginn der Diagnose alle beobachteten Symptome zur Verfügung stehen und alle nicht beobachteten Symptome als nicht vorhanden angesehen werden („*closed world assumption*"). Das Wissen über die Abwesenheit bestimmter Symptome ermöglicht es dann, gewisse Fehler als Ursache auszuschließen. In der Hydraulik-Domäne, wie in vielen Domänen, ist die *closed world assumption* allerdings aus folgendem Grund unrealistisch: Um Symptome zu erkennen, müssen Messungen an der realen Anlage durchgeführt werden. Diese sind in der Regel teuer, so daß am Anfang der Fehlersuche nur ein kleiner Teil der tatsächlich vorhandenen Symptome bekannt ist. Eine solche Situation wird als offen bezeichnet [Forbus & de Kleer 1993]. Diagnostische Schlußfolgerungen können nur anhand der bekannten

Symptome gemacht werden. Über die noch unbekannten Symptome können keine Aussagen getroffen werden. Unter Umständen müssen im Verlauf der Diagnose weitere Symptome durch zusätzliche Messungen an der hydraulischen Anlage erhoben werden. Bei dem für *Art Deco* zu entwickelnden Diagnosesystem ist von einer offenen Diagnosesituation auszugehen. Es wird als eine Aufgabe des Diagnosesystems angesehen, durch geschickte Auswahl der Meßpunkte die Anzahl der insgesamt erforderlichen Messungen zu minimieren.

Ein ganz anderer Ansatz zur Lösung des Mehrfachfehler-Diagnose-Problems ist in [Miller et al. 1993] beschrieben. Sie stellen einen genetischen Algorithmus vor, der in der Lage sei, eine optimale Lösung in allen Fällen bis auf einen winzigen Bruchteil der Fälle zu finden, und mit einer Geschwindigkeit, die um Größenordnungen schneller sei als exakte Algorithmen. Dieser Ansatz basiert jedoch ebenfalls auf der Diagnose mit Fehlermodellen nach Peng und Reggia, so daß auch hierbei das vollständige Wissen und eine geschlossene Diagnosesituation vorausgesetzt wird.

Aufgrund der oben genannten Probleme bei dem Ansatz von Peng und Reggia wird dieser Fehlermodell-basierte Ansatz für die Entwicklung eines Diagnosesystems für hydraulische Anlagen nicht weiter verfolgt.

Kapitel 4

Kandidaten-Berechnungsverfahren

Das Ziel der Kandidatengenerierung ist die Bestimmung der vollständigen Menge der minimalen Kandidaten. Zur Erinnerung: Ein Kandidat ist eine Menge von Komponenten, die alle vorhandenen Konflikte „erklärt“, wenn jede Komponente in dieser Menge tatsächlich defekt ist. Dazu muß ein Kandidat aus jedem minimalen Konflikt wenigstens ein Element enthalten. Zusätzlich wird die Minimalität eines Kandidaten gefordert.

In Abschnitt 3.1.3 wurde an einem Beispiel gezeigt, daß i.a. die Anzahl der möglichen minimalen Kandidaten exponentiell wächst mit der Anzahl der minimalen Konflikte. In den 2^n möglichen minimalen Kandidaten, die es im Beispiel gab, waren allerdings sämtliche Mehrfachfehler-Kombinationen enthalten. Wenn hingegen nur Einfachfehler unterstellt werden („Single Fault Assumption“), bleibt das exponentielle Wachstum aus. Die Einschränkung auf Einfachfehler bedeutet, daß alle gefundenen Symptome auf einen einzelnen Fehler zurückgeführt werden können. Alle minimalen Kandidaten sind dann einelementig und liegen in der Schnittmenge aller minimalen Konflikte.

Bei einer Bitvektor-Darstellung der minimalen Konflikte kann ein minimaler Kandidat in linearer Zeit (in der Anzahl der minimalen Konflikte) bestimmt werden. Zum Beispiel könnte in einem System mit 100 Komponenten jeder minimale Konflikt durch einen 100-Bit-Binärstring dargestellt werden, wobei jeder Komponente eine eindeutige Bit-Position in dem Binärstring zugeordnet ist. Eine Eins an einer bestimmten Bit-Position in dem Binärstring bedeutet, daß die korrespondierende Komponente in dem minimalen Konflikt enthalten ist, den dieser Binärstring darstellt. Hingegen bedeutet eine Null, daß die entsprechende Komponente nicht in dem minimalen Konflikt enthalten ist. Alle einelementigen Kandidaten können nun bestimmt werden, indem alle Binärstrings bitweise mit UND verknüpft werden.

In der Praxis ist jedoch für die Diagnose von komplexen hydraulischen Anlagen die Einzelfehlerannahme nicht sehr realistisch. Bei dem zu entwickelnden Diagnosesystem für *Art Deco* ist mit Mehrfachfehlern zu rechnen.

Unglücklicherweise ist die Berechnung eines minimalen Kandidaten NP-schwer, sobald Mehrfachfehler-Kandidaten mit zwei oder mehr Elementen berücksichtigt werden. Wie bereits in Abschnitt 3.2.2 erwähnt wurde, ist ein minimaler Kandidat ein minimales Hitting Set über die Menge der existierenden minimalen Konflikte, und deshalb ist zu seiner Berechnung die Lösung des Hitting-Set-Problems notwendig.

In dem Spezialfall, daß sämtliche minimalen Konflikte zweielementig sind, gibt es eine Analogie zum Vertex-Cover-Problem. [1] Die zweielementigen Konflikte können durch einen ungerichteten Graphen beschrieben werden. Die Knoten des Graphen sind die Komponenten des zu diagnostizierenden Systems. Zwei verschiedene Knoten sind durch eine Kante verbunden, wenn die zugehörigen Komponenten einen minimalen Konflikt bilden. Ein minimaler Kandidat ist dann ein minimales Vertex Cover in dem „Konfliktgraphen". Zur Erinnerung: Ein Vertex Cover (Kantenüberdeckung) ist ein Menge von Knoten, so daß von jeder Kante $\{u, v\}$ wenigstens einer der Knoten u oder v in diesem Vertex Cover enthalten ist. Ein Vertex Cover S ist minimal, wenn jede echte Teilmenge von S kein Vertex Cover ist. Ein minimales Vertex Cover ist nicht eindeutig: Es gibt i.a. in einem bestimmten Graphen mehrere verschiedene solcher Mengen. Bei der Kandidatengenierung werden alle minimalen Vertex Cover gesucht. Das Problem, für ein gegebenes $k \in \mathbb{N}$ zu bestimmen, ob es in einem ungerichteten Graphen ein Vertex Cover der Kardinalität $\leq k$ gibt, ist NP-vollständig [Garey & Johnson 1979].

Die Analogie zum Vertex-Cover-Problem bei der Berechnung minimaler Kandidaten in dem obigen Spezialfall ist nicht zufällig. In der Tat kann nachgewiesen werden, daß das Hitting-Set-Problem NP-schwer ist, indem das Vertex-Cover-Problem auf das Hitting-Set-Problem reduziert wird [Garey & Johnson 1979].

Wie bei dem Vertex-Cover-Problem, ist ein minimales Hitting Set für die Menge aller minimalen Konflikte i.a. nicht eindeutig. Es sind alle minimalen Hitting Sets zu bestimmen, denn wir sind bei der Diagnose an allen minimalen Kandidaten interessiert. Die Berechnung exakter (minimaler) Kandidaten verbietet sich jedoch in der Praxis für alle größeren Probleminstanzen wegen der astronomischen Rechenzeiten für die Bestimmung der minimalen Hitting Sets. Um Anlagen realistischer Größe mittels

[1] Ganz allgemein korrespondiert das Problem der Kandidatenberechnung mit dem Problem, die minimalen Vertex Cover eines Hypergraphen zu bestimmen, wobei ein Hypergraph wie in [Lengauer 1990] definiert ist. Die Hyperkanten stellen die minimalen Konflikte dar, und die Endpunkte einer Hyperkante korrespondieren zu den Elementen eines minimalen Konflikts. Ein Vertex Cover des Hypergraphen enthält von jeder Hyperkante wenigstens einen Endpunkt.

der GDE erfolgreich diagnostizieren zu können, werden Strategien für die Behandlung der exponentiellen Zeitkomplexität benötigt. Es gibt verschiedene Strategien auf unterschiedlichen Ebenen, mit denen die Komplexität der Kandatengenerierung begrenzt werden kann. Auf der algorithmischen Ebene sind dies Näherungsverfahren. Im nächsten Abschnitt wird ein konkretes Näherungsverfahren vorgestellt und untersucht, ob solche Verfahren sinnvoll für die Kandidatenberechnung sind. Eine andere Strategie auf der Modellebene, die in Abschnitt 4.5.1 verfolgt wird, besteht in der Fokussierung auf Teilbereiche der Anlage. Durch eine Fokussierung soll die Anzahl der Komponenten, die bei der exakten Kandidatenberechnung zu berücksichtigen sind, von vornherein klein gehalten werden.

4.1 Kandidatenapproximierung

Bei dem Ansatz der Kandidatenapproximierung wird nicht länger die Absicht verfolgt, einen minimalen Kandidaten (d.h. eine optimale Lösung) zu finden, sondern statt dessen versucht, eine „gute“ Lösung in akzeptabler Zeit zu berechnen. Möglicherweise ist in der Praxis eine fast optimale Lösung noch gut genug. Sicher werden die Kosten einer erhöhten Rechenzeit für eine exakte Diagnose mit den Kosten abgewogen werden müssen, die entstehen, wenn eine nicht-minimale Diagnose akzeptiert wird, weil dann z.B. mehr Komponenten bei der Reparatur ersetzt werden müssen als nötig. Für die weiteren Untersuchungen in diesem Abschnitt wird angenommen, daß die Kosten einer exakten Lösung weitaus größer sind als die Kosten für eine redundante Diagnose, die zwar nicht minimal, aber auch nicht falsch ist.

Wir beschränken uns zunächst auf das Problem, nur einen minimalen Kandidaten zu berechnen. Das Problem läßt sich formal wie folgt beschreiben:

MINIMALER KANDIDAT

> *Instanz:* Eine Menge $\mathcal{C}$ von minimalen Konflikten C, die Teilmengen einer endlichen Menge K von Komponenten sind.
>
> *Konfigurationen:* Eine Menge $S \subseteq K$, so daß S aus jedem minimalen Konflikt $C \in \mathcal{C}$ wenigstens ein Element enthält.
>
> *Lösungen:* Alle Konfigurationen.
>
> *Minimiere:* $c(S) := |S|$.

Das Problem MINIMALER KANDIDAT ist ein Optimierungsproblem gemäß der Definition in [Lengauer 1990], wobei c die zum Optimierungsproblem gehörende Ko-

stenfunktion ist, die jeder zulässigen Konfiguration S eine natürliche Zahl zuordnet.

Die Approximation einer optimalen Lösung des Problems kann auf zwei grundsätzlich verschiedene Arten geschehen. Die erste Möglichkeit ist, eine optimale Lösung von innerhalb des Lösungsraumes (d.h. der Menge aller zulässigen Lösungen) anzunähern. Das bedeutet, daß jede gefundene Näherung eine zulässige Lösung des Problems ist. Die zweite Möglichkeit besteht darin, eine optimale Lösung aus dem Komplement des Lösungsraumes anzunähern. Eine auf die Weise berechnete Approximation ist zwar keine zulässige Lösung des Problems und damit kein Kandidat; die Idee ist aber, effizient Nicht-Lösungen zu berechnen, die nur noch sehr wenig von einer optimalen zulässigen Lösung abweichen. Nicht-Lösungen sind hierbei Mengen von Komponenten, die nicht alle Konflikte überdecken.

Die zuletzt genannte Vorgehensweise ist problematisch, denn wie soll die Qualität einer Kandidaten-Näherung bestimmt werden, die einige Symptome nicht erklärt? Um ein Maß für die Qualität einer Näherung zu schaffen, müßten die Symptome in unterschiedliche Klassen eingeteilt werden können entsprechend ihrer Wichtigkeit. Wie soll entschieden werden, welche Symptome kritisch sind und deshalb von jeder Näherung überdeckt müssen werden und welche nicht? Es wird deshalb nur die erste Möglichkeit betrachtet. Der folgende Algorithmus liefert Näherungslösungen für einen minimalen Kandidaten:

Algorithmus 4.1: Approximiere einen minimalen Kandidaten

1. `Kandidat` $\leftarrow \emptyset$
2. $\mathcal{C}$ $\leftarrow$ `KONFLIKTE`
3. `solange` $\mathcal{C} \neq \emptyset$
4. `sei` $C = \{a_{c_1}, ..., a_{c_k}\}$ `ein beliebiger Konflikt aus` $\mathcal{C}$
5. `Kandidat` $\leftarrow$ `Kandidat` $\cup \{a_{c_1}, ..., a_{c_k}\}$
6. `entferne aus` $\mathcal{C}$ `jeden Konflikt` C' `mit` $C' \cap C \neq \emptyset$
7. `übergebe Kandidat`

Zeile 1 initialisiert den zu konstruierenden Kandidaten mit der leeren Menge. In Zeile 2 wird $\mathcal{C}$ auf die Menge aller minimalen Konflikte gesetzt. Die Schleife in Zeile 3 – 6 wählt wiederholt einen Konflikt $C \in \mathcal{C}$, fügt die Elemente $a_{c_1}, ..., a_{c_k}$ des Konfliktes zu `Kandidat` hinzu und löscht alle Konflikte aus $\mathcal{C}$, die wenigstens eines dieser Elemente $a_{c_1}, ..., a_{c_k}$ enthalten. Offensichtlich berechnet der Algorithmus, wenn er terminiert, in der Variablen `Kandidat` ein Hitting Set über die Menge der minimalen Konflikte. Der Algorithmus terminiert, denn in jedem Schleifendurchlauf wird wenigstens ein Konflikt aus $\mathcal{C}$ entfernt, d.h., es gibt höchstens $m = |$`KONFLIKTE`$|$ Schleifendurchläufe. Wenn Konflikte durch aufsteigend sortierte Listen dargestellt

werden, dann benötigt die Schnittmengenbildung zweier Konflikte mit n Komponenten in Zeile 6 $O(n^2)$ Zeit. Die Laufzeit beträgt also bei m minimalen Konflikten und n Komponenten $O(mn^2)$.

Der obige Approximierungsalgorithmus ist sehr effizient. Doch wie gut sind die gefundenen Näherungen? Für den praktischen Nutzen ist es entscheidend, zu wissen, wie weit die berechnete Näherung von einer vermeintlich optimalen Lösung entfernt ist. Um eine obere Schranke für die Abweichung von einer optimalen Lösung bestimmen zu können, wird der obige Algorithmus im folgenden analysiert.

Die Differenz zwischen den Kosten der berechneten Näherung und den Kosten einer optimalen Lösung — ausgedrückt durch den korrespondierenden Quotient — wird nach oben beschränkt durch die sogenannte Ratio des Approximationsalgorithmus. Bei Minimierungsproblemen ist die Ratio definiert durch [Lengauer 1990]:

$$\text{Ratio}(A) := \max_{p \in I} \frac{c(A(p))}{c(\text{opt}(p))}.$$

Hierbei ist $A(p)$ die vom Approximationsalgorithmus A berechnete Näherung und $\text{opt}(p)$ eine optimale Lösung der Instanz p aus der Menge aller Probleminstanzen I. c ist die Kostenfunktion aus der Definition des Problems MINIMALER KANDIDAT. Da die Kosten einer Lösung stets positiv sind, ist der Quotient wohldefiniert.

Satz 1: *Der* **Algorithmus 4.1 (Approximiere einen minimalen Kandidaten)** *hat eine Ratio von* $\max\{|C| \mid C \in \texttt{KONFLIKTE}\}$.

Beweis: Die Menge `Kandidat`, die vom Algorithmus zurückgegeben wird, „überdeckt“ `KONFLIKTE` ($\forall$ Konflikt $\in$ `KONFLIKTE`: `Kandidat` $\cap$ Konflikt $\neq \emptyset$), weil die Schleife 3 – 6 so lange wiederholt wird, bis jeder Konflikt aus `KONFLIKTE` überdeckt ist durch ein Element in `Kandidat`.

Sei $\mathcal{C}$ die Menge der in Zeile 4 gewählten Konflikte. Keine zwei Konflikte in $\mathcal{C}$ haben ein Element gemeinsam, da — sobald ein Konflikt C in Zeile 4 ausgewählt worden ist — alle anderen Konflikte, die wenigstens ein Element mit C gemeinsam haben, in Zeile 6 aus $\mathcal{C}$ gelöscht werden. Deshalb werden bei jeder Ausführung von Zeile 5 $k = |\text{Konflikt}|$ Elemente zu `Kandidat` hinzugefügt. In Zeile 5 werden höchstens $\max\{|C| \mid C \in \texttt{KONFLIKTE}\}$ viele Elemente zu `Kandidat` hinzugefügt,

$$|\texttt{Kandidat}| = \sum_{C \in \mathcal{C}} |C| \;\leq\; |\mathcal{C}| \cdot \max\{|C| \mid C \in \mathcal{C}\}. \qquad (1)$$

Um die Konflikte zu überdecken, muß jeder Kandidat — insbesondere ein minimaler Kandidat $\texttt{Kandidat}^*$ — wenigstens ein Element aus jedem Konflikt in $\mathcal{C}$ enthalten. Weil keine zwei Konflikte in $\mathcal{C}$ ein Element gemeinsam haben (alle Konflikte in $\mathcal{C}$

paarweise disjunkt sind), ist kein Element in $\texttt{Kandidat}^*$ in mehr als einem Konflikt in $\mathcal{C}$ enthalten. Deshalb ist

$$|\mathcal{C}| \leq |\texttt{Kandidat}^*| \tag{2}$$

und aus (1) folgt

$$\frac{|\texttt{Kandidat}|}{\max\{|C| \mid C \in \mathcal{C}\}} \leq |\mathcal{C}|. \tag{3}$$

(2) und (3) zusammen ergibt

$$\frac{|\texttt{Kandidat}|}{\max\{|C| \mid C \in \mathcal{C}\}} \leq |\texttt{Kandidat}^*|$$

bzw. $|\texttt{Kandidat}| \leq \max\{|C| \mid C \in \mathcal{C}\} \cdot |\texttt{Kandidat}^*|$. □

Die Ratio des Algorithmus ist also beschränkt durch die maximale Kardinalität der eingehenden minimalen Konflikte. In dem Spezialfall, daß es ausschließlich zweielementige minimale Konflikte gibt, beträgt die Ratio 2. Bei n-elementigen minimalen Konflikten kann der gefundene Kandidat n-mal mehr Elemente enthalten als der minimale Kandidat.

Wie ist dieses Ergebnis für die Diagnose von hydraulischen Anlagen zu bewerten? Der oben angeführte Satz verdeutlicht, daß auf diese Weise berechnete Näherungen i.a. viel zu ungenau sind. Angenommen, es sei eine Anlage mit 100 Komponenten zu untersuchen, bei der zwei Komponenten tatsächlich defekt und alle restlichen intakt sind. Wenn die beobachteten Symptome zu minimalen Konflikten mit bis zu 40 Komponenten führen, dann liefert der obige Approximationsalgorithmus als Erklärung für die Symptome einen Kandidaten, der im schlechtesten Fall 80 Komponenten enthält, im Gegensatz zu zwei Komponenten eines minimalen Kandidaten. Eine solche Diagnose ist in der Praxis unbrauchbar.

Ein ganz anderes Verfahren stammt von de Kleer & Williams. In [de Kleer & Williams 1987] schlagen sie ein inkrementelles Verfahren zur exakten Berechnung aller minimalen Kandidaten vor, das in dem folgenden Abschnitt behandelt wird.

4.2 Kandidatengenierung bei de Kleer und Williams

Ein wesentlicher Bestandteil der inkrementellen Diagnose von de Kleer und Williams ist die inkrementelle Vorgehensweise bei der Kandidatengenerierung. Nachfolgend ist

der Algorithmus zur inkrementellen Diagnose von Mehrfachfehlern aufgeführt, der bereits verbal in Abschnitt 3.1.3 beschrieben wurde und auf [de Kleer & Williams 1987] zurückgeht.

Algorithmus 4.2: inkrementelle Diagnose von Mehrfachfehlern

```
1.   Kandidaten := Ø
2.   Akzeptiere nächste Beobachtung
3.   Erzeuge alle minimalen Konflikte
4.   inkrementelle Kandidatengenerierung
4.1    für jede Menge K  ∈ Kandidaten
4.2      für jeden minimalen Konflikt C
4.3        falls K ∩ C = Ø dann
4.4          streiche K aus Kandidaten
4.5          erweitere K mit jedem Element aus C
4.6          füge die Erweiterungen zu Kandidaten hinzu
4.7          streiche aus Kandidaten die subsummierten Mengen
5.   falls Kandidaten genügend eingeengt, dann stop
6.   schlage nächste Beobachtung vor
7.   zurück zu 2.
```

Wie funktioniert die inkrementelle Kandidatengenerierung bei diesem Algorithmus? Am Anfang der Diagnose gibt es nur den „leeren“ Kandidaten, weil noch keine Konflikte bekannt sind (Zeile 1). In Zeile 2 wird dem Diagnosesystem eine neue Beobachtung übergeben. Sofern es sich bei der Beobachtung um ein Symptom handelt, führt die Symptompropagierung in dem Modell zu den neuen minimalen Konflikten (Zeile 3, siehe hierzu Kapitel 5). In Schritt 4 findet die inkrementelle Kandidatengenerierung statt. Dabei werden die bisherigen Kandidaten bezüglich aller neuen minimalen Konflikte aktualisiert. Im einzelnen wird dazu jeder Kandidat K mit allen minimalen Konflikten verglichen. Wenn der Schnitt von K mit einem minimalen Konflikt C nichtleer ist (Zeile 4.3), dann erklärt der Kandidat K diesen Konflikt schon und muß nicht verändert werden. Wenn allerdings K und C disjunkt sind, dann erklärt K den minimalen Konflikt C nicht und wird deshalb aus der Menge der Kandidaten gestrichen (Zeile 4.4). Angenommen, der minimale Konflikt C enthält k Elemente. Wird nun K jeweils um ein Element aus dem minimalen Konflikt erweitert (Zeile 4.5), so erhalten wir k Erweiterungen $K_1, ..., K_k$, von denen jede den Konflikt C erklärt. Jede Erweiterung K_i $(1 \leq i \leq k)$ wird zunächst in `Kandidaten` neu aufgenommen (Zeile 4.6). Eine neu aufgenommene Menge K_i muß nicht minimal sein. Deshalb wird K_i in Zeile 4.7 wieder entfernt, wenn bereits eine

Teilmenge von K_i in `Kandidaten` vorhanden ist. Wenn die Menge der Kandidaten genügend eingeschränkt ist, dann terminiert der Algorithmus (Zeile 6). Die Entscheidung darüber, ob die Kandidaten genügend genau sind, trifft der Benutzer. Wenn nicht, dann wird eine neue Beobachtung vorgeschlagen (Zeile 7) und das Verfahren iteriert.

4.3 Korrektheit der inkrementellen Kandidatenberechnung

Liefert das Verfahren zur inkrementellen Kandidatengenerierung von de Kleer und Williams wirklich alle minimalen Kandidaten? Betrachten wir die inkrementelle Kandidatenberechnung an einem Beispiel. Angenommen, bei der Diagnose eines Netzwerkes werden zu Beginn aufgrund einer Beobachtung zwei minimale Konflikte $\{A, B, C\}$ und $\{C, D\}$ erzeugt, und dies seien alle minimalen Konflikte zu diesem Zeitpunkt. Der in Abbildung 4.1 dargestellte Kandidatengenerierungsbaum verdeutlicht die Kandidatenberechnung nach dem oben genannten Algorithmus. Das Verfahren berechnet hier die korrekten minimalen Kandidaten $\{A, C\}$, $\{A, D\}$, $\{B, C\}$, $\{B, D\}$ und $\{C\}$. Wir nehmen an, es erfolgt eine weitere Beobachtung, die dazu führt, daß nach dieser Beobachtung die Mengen $\{A, B\}$ und $\{D, E\}$ alle minimalen Konflikte sind. Die vorherigen minimalen Konflikte werden also durch die neuen minimalen Konflikte $\{A, B\}$ und $\{D, E\}$ ersetzt. In dem Algorithmus werden nun die bisherigen Kandidaten bezüglich der neuen minimalen Konflikte aktualisiert.

Die neue Situation ist in der unteren Hälfte des Kandidatengenerierungsbaumes in Abbildung 4.1 gezeigt. Die schattierten Mengen in der Abbildung sind diejenigen, die in Zeile 4.7 des Algorithmus gestrichen werden. Der Algorithmus produziert in diesem Beispiel die Kandidaten $\{A, C, E\}$, $\{A, D\}$, $\{B, C, E\}$ und $\{B, D\}$. Offensichtlich sind jedoch $\{A, C, E\}$ und $\{B, C, E\}$ nicht minimal, denn die Komponente C ist in keinem aktuell minimalen Konflikt enthalten. Ist das Verfahren von de Kleer und Williams also nicht korrekt? Ein formaler Nachweis der Korrektheit der inkrementellen Kandidatengenerierung wird in [de Kleer & Williams 1987] nicht erbracht. Weil beabsichtigt wird, bei der Implementierung des Diagnosesystems für hydraulische Anlagen diesen Algorithmus zu verwenden, lohnt es sich, die Frage nach der Korrektheit des Verfahrens genau zu untersuchen. Dies soll im nachfolgenden Abschnitt geschehen.

Das Beispiel läßt vermuten, daß bei dem Verfahren von de Kleer und Williams die Art der minimalen Konflikte einen wesentlichen Einfluß auf die Minimalität der berechneten Kandidaten hat. Können sich die vorhandenen minimalen Konflikte

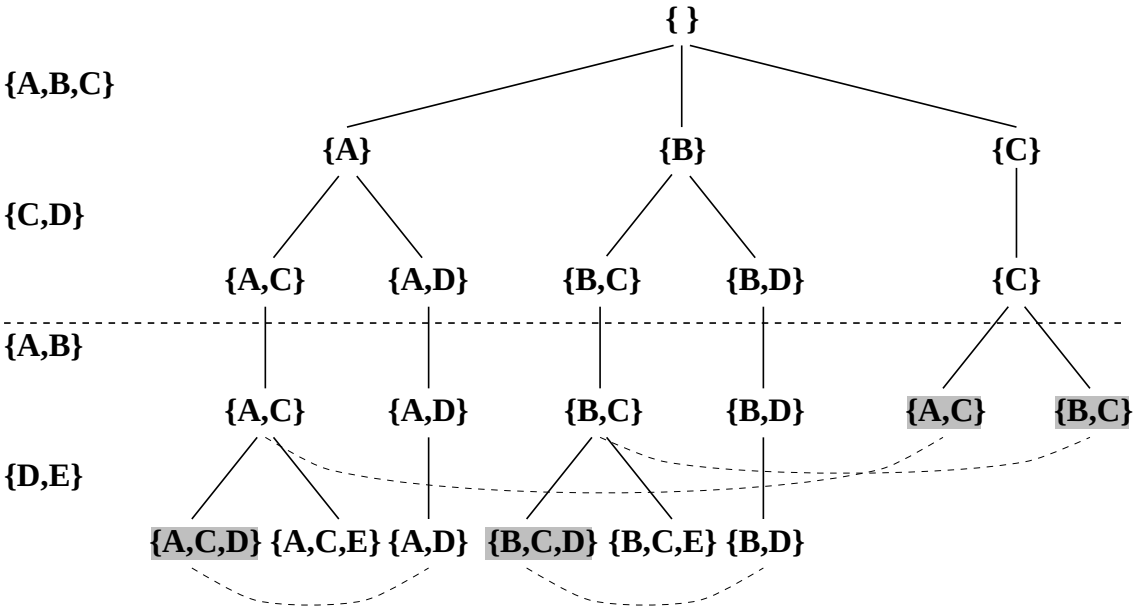


Abbildung 4.1: **Der Kandidatengenerierungsbaum verdeutlicht die inkrementelle Kandidatenberechnung. Die schattierten Mengen sind die subsummierten Kandidaten. Zunächst sind {A,B,C} und {C,D} die einzigen minimalen Konflikte. Ändern sich infolge einer Beobachtung die minimalen Konflikte in {A,B} und {D,E}, liefert das Verfahren die Kandidaten {A,C,E}, {A,D}, {B,C,E} und {B,D}. Nur {A,D} und {B,D} sind minimal.**

aufgrund einer weiteren Beobachtung in beliebiger Weise verändern? Kann ein minimaler Konflikt gänzlich verschwinden? Wie im folgenden gezeigt wird, lautet die Antwort auf beide Fragen „nein“. Durch neue Beobachtungen an der Anlage können neue minimale Konflikte hinzukommen, oder die Anzahl der minimalen Konflikte kann abnehmen. Jedoch können sich die existierenden minimalen Konflikte infolge einer neuen Beobachtung nur auf bestimmte Weise ändern. Aufgrund der Definition von minimalen Konflikten können die neuen minimalen Konflikte nach einer Beobachtung nicht mehr Elemente enthalten als vor der Beobachtung. Neue Konflikte können weniger Komponenten enthalten als die bisherigen minimalen Konflikte. Dies ist z.B. der Fall, wenn eine Komponente, die bis dahin als defekt verdächtigt wurde, durch die neuere Beobachtung „freigesprochen“ wird. Die neuen minimalen Konflikte nach einer weiteren Beobachtung stehen immer in einer bestimmten Beziehung zu den vorherigen minimalen Konflikten, wie das folgende Lemma zeigt. Informell besagt es: Wenn die Anzahl der minimalen Konflikte infolge einer Beobachtung schrumpft, gibt es für jeden vor der Beobachtung minimalen Konflikt C

nach der Beobachtung einen minimalen Konflikt C' mit $C' \subseteq C$.

Lemma 1: (Konflikt-Schrumpfungslemma)
Sei $\mathcal{C}_B = \{ C \mid C$ ist ein minimaler nichtleerer Konflikt vor Beobachtung $B \}$ *und sei* $\mathcal{C}'_B = \{ C' \mid C'$ ist ein minimaler nichtleerer Konflikt nach Beobachtung $B \}$ *mit* $|\mathcal{C}'_B| < |\mathcal{C}_B|$. *Dann gilt für alle* $C \in \mathcal{C}_B$*: Es gibt einen Konflikt* $C' \in \mathcal{C}'_B$ *mit* $C' \subseteq C$.

Beweis: indirekt.

Seien $\mathcal{C}_B$ und $\mathcal{C}'_B$ wie in der Behauptung definiert. Angenommen, es gäbe einen nichtleeren Konflikt $C \in \mathcal{C}_B$, so daß gilt:

$$\forall\, C' \in \mathcal{C}'_B\colon C' \not\subseteq C,$$

Das heißt, es gibt keinen Konflikt aus $\mathcal{C}'_B$, der den „alten“ Konflikt C minimiert. Wenn der Konflikt C durch die Beobachtung B verschwindet, so kann dies nur sein, wenn er durch den leeren Konflikt (die leere Menge) minimiert wird, wenn also durch die Beobachtung B die leere Menge ein miminaler Konflikt wird. Wenn aber die leere Menge ein minimaler Konflikt ist, dann kann es keine weiteren minimalen Konflikte geben, weil jeder weitere minimale Konflikt eine Obermenge der leeren Menge ist und deshalb nicht minimal sein kann. Andererseits kann die leere Menge nur dann ein minimaler Konflikt sein, wenn es kein Symptom gibt, denn nach der Definition stellt jedes Symptom eine Inkonsistenz dar und erzeugt somit einen (nichtleeren) Konflikt.

Vor der Beobachtung B muß es aber wenigstens ein Symptom s gegeben haben, denn C ist nichtleer. Sei B' die Beobachtung gewesen, durch die das Symptom s manifestiert wurde. (Jedes Symptom läßt sich einer Beobachtung zuordnen.) Wenn durch die neuere Beobachtung B alle Symptome verschwinden, dann kann das nur geschehen, indem B „ältere“ Beobachtungen, die diese Symptome manifestierten, widerlegt. Insbesondere widerlegt die Beobachtung B das Symptom s. Das widerspricht der in Abschnitt 2.2.1 für die Diagnose getroffenen Voraussetzung, daß Beobachtungen immer korrekt sind und als unumstößliche Fakten gelten. □

Es ist also nicht möglich, daß durch eine Beobachtung alle bekannten Symptome verschwinden. Mit dem Konflikt-Schrumpfungslemma können wir nun die Korrektheit des Verfahrens von de Kleer und Williams zur inkrementellen Kandidatengenerierung formal beweisen. Der Korrektheitsbeweis ist in vier Teile gegliedert. Der eigentliche Beweis findet in den folgenden drei Lemmata statt. In Satz 2 werden dann die Ergebnisse zusammengefaßt und damit die Korrektheit des Verfahrens von de Kleer und Williams nachgewiesen.

Lemma 2: *Sei* $\mathcal{C} = \{C \mid C$ minimaler Konflikt$\}$. *Nach jeder Ausführung der inkrementellen Kandidatengenerierung (in Schritt 4) in dem Algorithmus „inkrementelle Diagnose" von de Kleer und Williams gilt:*

$$\begin{aligned} \forall\, C \in \mathcal{C}\text{: } &\exists \textit{ Kandidat } K \textit{ mit } a \in K && (1)\\ &\textit{und } \forall\, a' \in C \backslash \{a\}\text{: } a' \notin K && (2)\\ &\textit{und } K \textit{ ist minimal.} && (3) \end{aligned}$$

Beweis: Induktion nach der Anzahl i der Ausführungen von Schritt 4 des Algorithmus (Kandidaten-Aktualisierungsschritt).

Induktionsanfang $i = 1$: $K = \emptyset$ ist der einzige minimale Kandidat. Sei $C = \{a_1, ..., a_n\}$ ein beliebiger minimaler Konflikt. Die Erweiterung von $K = \emptyset$ mit jedem Element aus C (Zeile 4.5) erzeugt die neuen Kandidaten $\{a_1\}, ..., \{a_n\}$ und die Behauptung gilt.

Induktionsschritt $k < i \rightarrow i$: Betrachte den i-ten Kandidaten-Aktualisierungsschritt. Für alle vorherigen Aktualisierungsschritte $k < i$ und für alle vorherigen minimalen Konflikte C gelte bereits die Behauptung, d.h.

$$\begin{aligned} \forall\, a \in C\text{: } &\exists \text{ Kandidat } K \text{ mit } a \in K\\ &\text{und } \forall\, a' \in C \backslash \{a\}\text{: } a' \notin K\\ &\text{und } K \text{ ist minimal.} \end{aligned}$$

Sei nun $C = \{a_1, ..., a_n\}$ der in Zeile 4.3 betrachtete minimale Konflikt, und sei $\mathcal{K} = \{K_1, ..., K_k\}$ die Menge der bis dahin minimalen Kandidaten.

Fall 1: $\forall\, K \in \mathcal{K}$: $K \cap C = \emptyset$
$\Longrightarrow$ K wird in Zeile 4.5 mit jedem Element aus C erweitert.
Seien $K_{a_i} := K \cup \{a_i\}$, $a_i \in C$, $1 \leq i \leq n$ die erweiterten Mengen.

Fall 1.1: Alle K_{a_i}, $1 \leq i \leq n$ sind minimal bezüglich aller weiteren Kandidaten aus $\mathcal{K} \setminus \{K\}$
$\Longrightarrow$ in Zeile 4.7 werden keine der K_{a_i}, $1 \leq i \leq n$ gestrichen, d.h., für $a_i \in C$ existiert mit K_{a_i} ein minimaler Kandidat mit $a_i \in K_{a_i}$, d.h., es gilt (1) und (3) der Behauptung.
Sei andererseits $a' \in C$, $a' \neq a_i$. Dann gilt $a' \notin K_{a_i}$, denn wäre $a' \in K_{a_i}$, dann $a' \in K$ und folglich $K \cap C \neq \emptyset$. Das wäre ein Widerspruch zur Voraussetzung von Fall 1. Also gilt (2) ebenfalls.

Fall 1.2: Nicht alle K_{a_i}, $1 \leq i \leq n$ sind minimal.
Sei K_{a_j} ein nicht minimaler Kandidat
$\Longrightarrow \exists K' \in \mathcal{K} \setminus \{K\}$: $K' \subset K_{a_j}$

$\Longrightarrow$ K_{a_j} wird in Zeile 4.7 gestrichen. Es gilt $a_j \in K'$, denn wäre $a_j \notin K'$, dann $K' \subset K$. D.h., K wäre vor der Betrachtung von Konflikt C nicht minimal gewesen. Wenn für ein $a_j \in C$ der in Zeile 4.5 erzeugte Kandidat K_{a_j} in Zeile 4.7 gestrichen wird, gibt es folglich einen minimalen Kandidaten K' mit $a_j \in K'$. Also gilt (1) und (3) der Behauptung.

Betrachte nun $a' \in C$, $a' \neq a_j$. Nach dem gerade bewiesenen Punkt (1) der Behauptung gibt es einen Kandidaten K' mit $a' \in K'$. Wegen $K' \subset K_{a_j}$ folgt $a' \in K_{a_j}$. Mit $a' \neq a_j$ folgt $a' \in K$. Das hieße aber $K \cap C \neq \emptyset$, was im Widerspruch zur Voraussetzung von Fall 1 steht. Daher gilt auch (2) der Behauptung.

Fall 2: $\forall\ K \in \mathcal{K}$: $K \cap C \neq \emptyset$

$\Longrightarrow$ kein bisheriger Kandidat muß erweitert werden (Zeile 4.5). D.h., die Kandidaten aus dem vorherigen Aktualisierungsschritt sind auch die neuen Kandidaten.

Wenn gezeigt wird, daß der Konflikt C schon früher einmal aufgetreten ist, dann sind die Kandidaten bereits in einem vorherigen Generierungsschritt bezüglich des Konfliktes aktualisiert worden, und die Induktionsvoraussetzung läßt sich auf C anwenden.

Ich zeige nun, daß der Konflikt C schon früher aufgetreten sein muß.

Fall 2.1: C ist paarweise disjunkt mit allen bisher aufgetretenen Konflikten, d.h. kein Element von C ist bereits in einem anderen Konflikt vorgekommen.

$\Longrightarrow$ Kein Element aus C kommt in irgendeinem Kandidaten vor.

$\Longrightarrow$ $K \cap C = \emptyset$ für alle $K \in \mathcal{K}$. Das ist ein Widerspruch zur Voraussetzung von Fall 2.

Fall 2.2: Es hat zu einem früheren Zeitpunkt einen minimalen Konflikt C' gegeben mit $C \cap C' \neq \emptyset$.

Fall 2.2.1: $C' \subset C$.

Dieser Fall kann nach dem Konflikt-Schrumpfungslemma nicht eintreten.

Fall 2.2.2: $C \subset C'$. Sei $a \in C' \setminus C$.

Nach Induktionsvoraussetzung gilt für C': Zu $a \in C' \setminus C$ gibt es einen Kandidaten K mit $a \in K$, und für alle $a' \in C' \setminus \{a\}$ gilt $a' \notin K$

$\Longrightarrow$ Für alle $a' \in C$: $a' \notin K$

$\Longrightarrow$ $K \cap C = \emptyset$. Widerspruch zur Voraussetzung von Fall 2.

Also muß gelten, daß der Konflikt C schon einmal aufgetreten ist. □

Lemma 3: *Sei C ein minimaler Konflikt, $x \in C$, $y \in C$, $x \neq y$ und sei K ein minimaler Kandidat mit $x \in K$. Dann gibt es einen minimalen Kandidaten K' mit $y \in K'$ und $x \notin K'$.*

Beweis: Wegen Lemma 2 gibt es für $y \in C$ einen minimalen Kandidaten K', so daß $y \in K'$ und für alle $a \in C \setminus \{y\}$ gilt $a \notin K'$. □

Lemma 4: *Sei K ein vom Algorithmus „inkrementelle Diagnose" berechneter Kandidat. Dann gibt es für jedes $a \in K$ einen minimalen Konflikt C, so daß*

(i) $a \in C$ *und*
(ii) $(K \setminus \{a\}) \cap C = \emptyset$.

Beweis: Induktion nach der Anzahl t der Aufrufe des Algorithmus.

Induktionsanfang $t = 1$: $K = \emptyset$ ist der einzige Kandidat vor dem ersten Aufruf. Sei $C = \{a_1, ..., a_n\}$ der erste betrachtete minimale Konflikt. Die Erweiterung von K in Zeile 4.5 mit jedem Element aus C erzeugt alle neuen Kandidaten $\{a_1\}, ..., \{a_n\}$, und für jeden einzelnen gilt die Behauptung (i) und (ii).

Induktionsschritt $t - 1 \rightarrow t$: Betrachte die Situation nach dem t-ten Aufruf des Algorithmus, $t \geq 2$, die ich als „aktuelle Situation" bezeichne. Für den vorherigen Aufruf $(t - 1)$ sei die Behauptung (i) und (ii) bereits bewiesen. Sei nun K ein berechneter Kandidat.

Fall 1: K wurde im t-ten Aufruf nicht erweitert (Zeile 4.5). K war demnach bereits nach dem vorherigen Aufruf $(t - 1)$ ein Kandidat bezüglich der aktuell (t-ter Aufruf) minimalen Konflikte ($K \cap C \neq \emptyset$ für alle minimalen Konflikte C).

ad (i): Angenommen, es gäbe in K ein Element, das in keinem aktuell minimalen Konflikt enthalten ist. Sei a dieses Element. Nach Induktionsvoraussetzung (i) gab es zum Zeitpunkt des vorherigen Aufrufes zu $a \in K$ einen minimalen Konflikt C_{t-1} mit $a \in C_{t-1}$. Laut Induktionsvoraussetzung (ii) gilt $(K \setminus \{a\}) \cap C_{t-1} = \emptyset$. Nach dem Konflikt-Schrumpfungslemma gibt es aktuell einen minimalen Konflikt C_t mit $C_t \subseteq C_{t-1}$. Folglich ist $(K \setminus \{a\}) \cap C_t = \emptyset$. Nach obiger Annahme ist jedoch $a \notin C_t$, also $K \cap C_t = \emptyset$, und K wäre bezüglich C_t erweitert worden. Das ist ein Widerspruch zur Voraussetzung von Fall 1. Also gibt es für jedes $a \in K$ einen minimalen Konflikt C mit $a \in C$ ((i) der Behauptung).

ad (ii): Sei $a \in K$ beliebig. Nach Induktionsvoraussetzung gab es beim vorherigen Aufruf $(t - 1)$ einen minimalen Konflikt C_{t-1} mit $a \in C_{t-1}$ und

$(K \setminus \{a\}) \cap C_{t-1} = \emptyset$. C_{t-1} ist entweder aktuell weiterhin ein minimaler Konflikt, oder es gibt nach dem Konflikt-Schrumpfungslemma einen minimalen Konflikt C_t mit $C_t \subseteq C_{t-1}$. Folglich ist $K \setminus \{a\} \cap C_t = \emptyset$ ((ii) der Behauptung).

Fall 2: K wurde im aktuellen Aufruf erweitert (Zeile 4.5). Dann gibt es einen — oder mehrere — minimale Konflikte $C_1, ..., C_k$, $k \geq 1$, und $a_i \in C_i$, $1 \leq i \leq k$, derart, daß $K = K' \cup \{a_i \,|\, a_i \in C_i,\ 1 \leq i \leq k\}$, wobei K' ein Kandidat nach dem vorherigen Aufruf $(t-1)$ gewesen ist. Es sei $\mathcal{C} := \{C_1, ..., C_k\}$ ($\mathcal{C} = \{C \,|\, C$ minimaler Konflikt, $K' \cap C = \emptyset$). $E := \{a_i \,|\, a_i \in C_i,\ 1 \leq i \leq k\}$ bezeichne die Erweiterung von K', d.h.,

$$K = K' \cup E. \tag{1}$$

ad (i): Angenommen, es gäbe in K ein Element $a \in K$, das in keinem aktuell minimalen Konflikt enthalten ist. Dann ist $a \in K'$, denn $a \in E$ stände im Widerspruch zu dieser Annahme. Nach Induktionsvoraussetzung (i) gab es nach dem vorherigen Aufruf $(t-1)$ einen minimalen Konflikt C_{t-1} mit $a \in C_{t-1}$. Es muß laut Konflikt-Schrumpfungslemma aktuell einen minimalen Konflikt C_t mit $C_t \subseteq C_{t-1}$ geben. Nach Induktionsvoraussetzung (ii) gilt:

$$(K' \setminus \{a\}) \cap C_{t-1} = \emptyset, \tag{2}$$

also gilt auch $(K' \setminus \{a\}) \cap C_t = \emptyset$. Weil nach obiger Annahme $a \notin C_t$, ist $K' \cap C_t = \emptyset$. Dann muß allerdings K' bezüglich C_t erweitert worden sein (Zeile 4.5), d.h. $C_t \in \mathcal{C}$, und es gibt ein $b \in K$, $b \neq a$, $b \in E \cap C_t$. Wegen $C_t \subseteq C_{t-1}$ ist $b \in C_{t-1}$. Laut Lemma 3 gab es nach dem vorherigen Aufruf $(t-1)$ einen Kandidaten Q' mit $b \in Q'$ und $b' \notin C_{t-1}$ für alle $b' \in Q' \setminus \{b\}$, bzw. $(Q' \setminus \{b\}) \cap C_{t-1} = \emptyset$. Mit (2) folgt $(K' \setminus \{a\}) \cap C_{t-1} = (Q' \setminus \{b\}) \cap C_{t-1}$

$$\Longrightarrow K' \setminus \{a\} = Q' \setminus \{b\}. \tag{3}$$

Q' und K' sind also gleich bis auf das Element, mit dem sie C_{t-1} überdecken.

Die Menge $Q = Q' \cup E \setminus \{b\}$ ist nach dem t-ten Aufruf ebenfalls ein Kandidat, weil Q' ein Kandidat war und $b \in Q' \cap C_t$. Nun gilt:

$$\begin{aligned} Q = Q' \cup E \setminus \{b\} &= Q' \setminus \{b\} \cup E && (b \in E \cap Q') \\ &= K' \setminus \{a\} \cup E && (\text{mit } (3)) \\ &= K \setminus \{a\} && (\text{mit } (1)) \\ &\subset K. \end{aligned}$$

Das ist ein Widerspruch, denn K ist minimal. Sonst wäre K in Zeile 4.7 gestrichen worden. Also gilt (i) der Behauptung.

ad (ii): Sei $a \in K$ beliebig. Entweder $a \in K'$, oder $a \in E$. Falls $a \in E$, dann existiert ein $C \in \mathcal{C}$. Nach Definition von $\mathcal{C}$ und E ist $(K \setminus \{a\}) \cap C = \emptyset$, sonst wäre $a \notin E$.

Falls $a \in K'$, dann gab es laut Induktionsvoraussetzung bei dem vorherigen Aufruf $(t-1)$ einen minimalen Konflikt C_{t-1} mit $a \in C_{t-1}$ und der Eigenschaft

$$(K' \setminus \{a\}) \cap C_{t-1} = \emptyset. \tag{4}$$

Aus dem Konflikt-Schrumpfungslemma folgt: Es gibt aktuell einen minimalen Konflikt C_t mit $C_t \subseteq C_{t-1}$. Folglich ist

$$(K' \setminus \{a\}) \cap C_t = \emptyset. \tag{5}$$

$$\begin{aligned}
\text{Es ist } (K \setminus \{a\}) \cap C_t &= ((K' \cup E) \setminus \{a\}) \cap C_t && \text{(mit (1))} \\
&= (K' \setminus \{a\} \cup E) \cap C_t \\
&= ((K' \setminus \{a\}) \cap C_t) \cup (E \cap C_t) \\
&= E \cap C_t && \text{(mit (5))}
\end{aligned}$$

Wenn gezeigt wird, daß $E \cap C_t = \emptyset$, dann gilt die Behauptung (ii), also $(K \setminus \{a\}) \cap C_t = \emptyset$. Angenommen, es gäbe ein $a' \in E$ und $a' \in C_t$. Wegen $C_t \subseteq C_{t-1}$ ist $a' \in C_{t-1}$. Mit Lemma 3 hat es nach dem vorherigen Aufruf $(t-1)$ einen Kandidaten Q', $a' \in Q'$, und $(Q' \setminus \{a'\}) \cap C_{t-1} = \emptyset$ gegeben. Zusammen mit Gleichung (4) gilt:

$$\begin{aligned}
(K' \setminus \{a\}) \cap C_{t-1} &= (Q' \setminus \{a'\}) \cap C_{t-1} \\
\Longrightarrow \qquad K' \setminus \{a'\} &= Q' \setminus \{a'\}.
\end{aligned} \tag{6}$$

$Q = Q' \cup E \setminus \{a'\}$ ist nach dem t-ten Aufruf ein Kandidat, denn $a' \in Q' \cap C_t$.

$$\begin{aligned}
\text{Weiterhin ist } Q = Q' \cup E \setminus \{a'\} &= Q' \setminus \{a'\} \cup E && (a' \in E \cap Q') \\
&= K' \setminus \{a\} \cup E && \text{(mit (6))} \\
&= K \setminus \{a\} && \text{(mit (1))} \\
&\subset K.
\end{aligned}$$

Das ist ein Widerspruch zur Minimalität von K. Also mußte $E \cap C_t = \emptyset$ gelten. □

Satz 2: *K ist ein vom Algorithmus 4.2 berechneter minimaler Kandidat mit $c \in K$ genau dann, wenn es einen minimalen Konflikt C mit $c \in C$ gibt.*

Beweis: „$\Longrightarrow$“: Lemma 4.
„$\Longleftarrow$“: Lemma 2. □

Es kann wegen Satz 2 zusammenfassend festgehalten werden, daß die inkrementelle Kandidatengenerierung von de Kleer und Williams korrekt und vollständig ist unter der Voraussetzung, daß das Ergebnis einer einmal durchgeführten Beobachtung nicht durch eine weitere Beobachtung widerlegt werden kann. Das Verfahren ist korrekt, denn jedes produzierte Ergebnis ist ein minimaler Kandidat für die vorhandenen minimalen Konflikte. Das Verfahren ist vollständig, denn es werden alle minimalen Kandidaten berechnet.

4.4 Diskussion

Bei dem Diagnosesystem für hydraulische Anlagen wird der oben beschriebene Algorithmus zur inkrementellen Diagnose von Mehrfachfehlern benutzt. Dieses Verfahren liefert dieselben Kandidaten wie die Suchbaumgenerierung Reiters (siehe Abschnitt 3.2.2), es hat aber praktische Vorteile. Wenn nach einer Generierung aller Diagnosen eine zusätzliche Beobachtung gemacht wird, erfordert Reiters Ansatz eine Wiederholung des Diagnosevorgangs. Das bedeutet, daß neue Konfliktmengen bestimmt und die minimalen Kandidaten von Grund auf neu berechnet werden müssen. Demgegenüber erlaubt die inkrementelle Kandidatengenerierung von de Kleer und Williams eine schrittweise Verbesserung der bisherigen Diagnosen bei jeder weiteren Beobachtung.

Die exakte Kandidatengenerierung nach diesem Verfahren hat zwar eine exponentielle Laufzeit, denn es gibt im schlechtesten Fall eine exponentielle Anzahl von minimalen Kandidaten (siehe Abschnitt 3.1.3). Die Laufzeit kann jedoch oft reduziert werden, wenn nur die minimalen Kandidaten mit maximal k Fehlern (d.h. mit k Elementen) berechnet werden. Wie das folgende Beispiel zeigt, nimmt die Wahrscheinlichkeit von k gleichzeitig vorliegenden Defekten mit zunehmendem k rapide ab. Angenommen, die Komponentendefekte seien unabhängig voneinander, und alle Komponenten versagen mit der gleichen, sehr niedrigen Wahrscheinlichkeit von 0,01. Dann beträgt z.B. in einem System mit 10 Komponenten die Wahrscheinlichkeit eines Dreifachfehler-Kandidaten $0,01^3 \cdot 0,99^7 = 0,000000932$. In der Praxis reicht es deshalb oft aus, die Kandidatenberechnung auf Dreifachfehler-Kandidaten

zu beschränken.

4.5 Begrenzung der Komplexität

4.5.1 Hierarchiebildung und Fokussierung

Wie in Abschnitt 3.2 gezeigt worden ist, wächst die Zeitkomplexität bei dem Problem der Diagnose mit Mehrfachfehlern exponentiell mit der Anzahl der Komponenten in der Anlage. Daher ist es sinnvoll, bei der Fehlersuche die Anzahl der zu berücksichtigenden Komponenten möglichst klein zu halten. Dies kann auf zwei Arten erreicht werden:

1. Zusammenfassen der Komponenten in der hydraulischen Anlage auf einige wenige durch Einführen von Ersatzwiderständen.
2. Fokussierung auf Teilbereiche der Anlage mit relativ wenigen Komponenten.

Eine bestimmte Klasse von Netzwerken — die Serien-Parallel-Netzwerke — lassen sich durch eine Hierachie von Ersatzwiderständen zusammenfassen. Informell sind Serien-Parallel-Netzwerke solche, bei denen je zwei verschiedene Komponenten entweder in Reihe oder parallel zueinander geschaltet sind. Netzwerke mit sogenannten Brückenschaltungen sind keine Serien-Parallel-Netzwerke. Hier wird ein Netzwerk als Serien-Parallel-Netzwerk bezeichnet, wenn es sich durch einen Serien-Parallel-Graphen beschreiben läßt, wobei hier die Definition von Serien-Parallel-Graphen in [Lengauer 1990] zugrundegelegt wird.

Serien-Parallel-Netzwerke können als hierarchische Netzwerke angesehen werden. Die verschiedenen Zerlegungsstufen der Serien-Parallel-Strukturen eines solchen Netzwerkes definieren einen Zerlegungsbaum, dessen Blätter mit den einzelnen Komponenten im Netzwerk korrespondieren. Als Beispiel ist in Abbildung 4.2 der Zerlegungsbaum für ein hydraulisches Netzwerk dargestellt. Jeder interne Knoten des Zerlegungsbaumes repräsentiert die Serien- bzw. Parallelschaltung seiner Kind-Knoten in Form einer Ersatzkomponente E_i. Die Wurzel E_0 des Zerlegungsbaumes ist die Ersatzkomponente für das gesamte Netzwerk. Der Zerlegungsbaum für ein Serien-Parallel-Netzwerk kann anhand des zugehörigen Strukturgraphen des Netzwerkes in linearer Zeit gefunden werden [Lengauer 1990].

Mit der vorgegebenen Zerlegungshierarchie ist die Strategie zur Fokussierung offensichtlich: Wird an Komponente X ein Symptom beobachtet, dann werden in dem Zerlegungsbaum die Ersatzkomponenten expandiert, die auf dem Weg von der Wur-

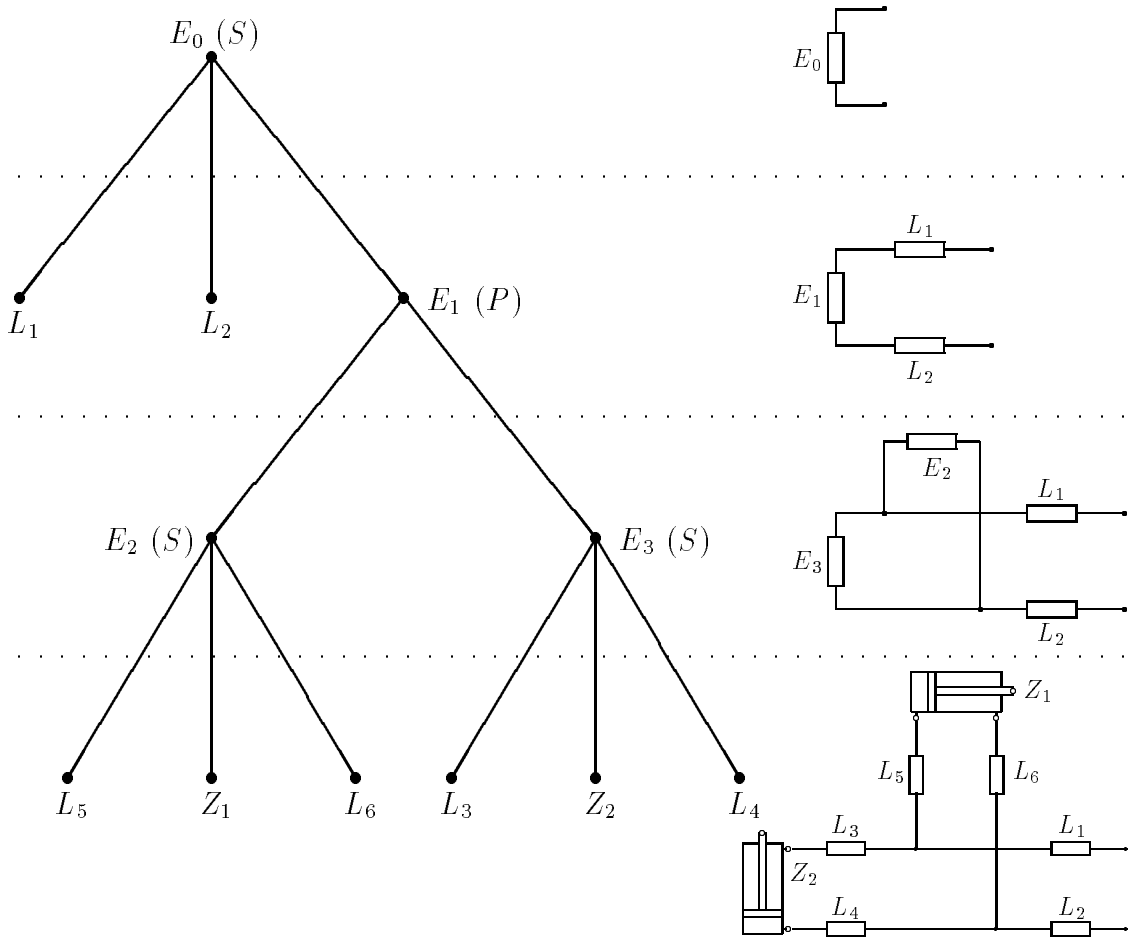


Abbildung 4.2: **Die Komponenten eines Serien-Parallel-Netzwerkes können durch eine Hierarchie von Ersatzwiderständen zusammengefaßt werden. Die verschiedenen Hierarchiestufen eines hydraulischen Netzwerkes definieren einen Zerlegungsbaum, dessen Blätter die Komponenten des Netzwerkes darstellen (hier sind es die Zylinder Z_1 und Z_2 und die Leitungen L_1 bis L_6). Jeder innere Knoten E_i entspricht einem Ersatzwiderstand für die Söhne des Knotens, die entweder parallel (P) oder in Serie (S) geschaltet sind. Auf diese Weise läßt sich das gesamte Netzwerk durch einen Ersatzwiderstand ausdrücken (hier durch die Wurzel des Baumes dargestellt). Auf jedem Weg von der Wurzel zu einem Blatt wechseln sich die (S)- und (P)-Knoten ab, weil in jedem Ersatzwiderstand die größtmögliche Anzahl von Komponenten bzw. Ersatzwiderständen auf der darunter liegenden Ebene zusammengefaßt sind. Dadurch ist der Zerlegungsbaum eindeutig bestimmt.**

zel bis zum Vater von X liegen. Eine Ersatzkomponente expandieren heißt hierbei, daß diese im Modell der Anlage durch die Komponenten auf der darunter liegenden Hierarchiestufe ersetzt wird. Am Anfang der Diagnose ist die Anlage im Modell durch einen einzigen Ersatzwiderstand beschrieben, entsprechend der Wurzel des Zerlegungsbaumes. Wenn im Beispiel an dem Zylinder Z_1 ein Symptom beoachtet wird, dann werden im Modell der Anlage nacheinander E_0, E_1 und E_2 expandiert. Das Beispiel zeigt, daß mit diesem Verfahren die Anzahl der Komponenten reduziert werden kann, die bei der Kandidatenberechnung tatsächlich berücksichtigt werden müssen.

4.5.2 Parallelisierung

Ein weiterer Vorteil des Algorithmus von de Kleer und Williams ist, daß sich die inkrementelle Kandidatengenerierung parallelisieren läßt. Die Aktualisierung der verschiedenen minimalen Kandidaten bezüglich eines bestimmten minimalen Konfliktes kann parallel ausgeführt werden, und zwar nach folgendem Algorithmus:

Algorithmus 4.5: parallele Kandidatenberechnung

```
1. candidates := Kandidaten
2. für jeden minimalen Konflikt C:
3.   für jede Menge K ∈ Kandidaten führe Schritt 4 bis 8 parallel aus:
4.     falls K ∩ C = ∅ dann
5.       streiche K aus candidates
6.       erweitere K mit jedem Element c ∈ C
7.       füge die Erweiterungen zu candidates hinzu
8.       streiche aus candidates die subsummierten Mengen
9. Kandidaten := candidates
```

Kapitel 5

Konflikterkennung mit dem ATMS

Es wurde bereits erwähnt, daß bei der GDE die Modell-Artefakt-Abweichungen auf Verletzungen von Korrektheitsannahmen im Modell zurückgeführt werden. An einem Beispiel wurde gezeigt, daß die beteiligten Komponenten einen Konflikt darstellen, wenn der beobachtete und der vorhergesagte Wert inkonsistent sind. Wie erkennt das Diagnosesystem, ob ein Konflikt vorliegt, und welche Komponenten an einem Konflikt beteiligt sind? Um Konflikte beschreiben zu können, muß das System diejenigen Komponenten identifizieren können, von deren korrektem Funktionieren ein im Modell hergeleiteter Wert abhängt. Das bedeutet: Bei der Herleitung eines bestimmten Wertes im Modell müssen die Abhängigkeiten von den Korrektheitsannahmen bei jedem einzelnen Inferenzschritt protokolliert werden. Ein Werkzeug zur Verwaltung der Abhängigkeiten ist das ATMS von [de Kleer 1986]. Im folgenden werden in einem kurzen Überblick die grundlegenden Konzepte des ATMS vorgestellt. Für Einzelheiten wird auf [de Kleer 1986] verwiesen.

5.1 Das grundlegende ATMS

Das ATMS (*Assumption-Based Truth Maintenance System*, deutsch: ein auf Annahmen basierendes Begründungsverwaltungssystem) ist ein Werkzeug, mit dem die Abhängigkeiten bei Folgerungen der Art $A \wedge B \rightarrow C$ verwaltet werden können. Man beachte, daß das ATMS selbst keinen Inferenzmechanismus darstellt. Es unterstützt die Problemlösungskomponente eines wissensbasierten Systems, indem es für sie die zentrale Aufgabe der Verwaltung des Suchraums übernimmt. Das ATMS ist dabei unabhängig von der Anwendungsdomäne und der genauen Art des Inferenzmechanismus, die in der Problemlösungskomponente verwendet wird.

Im einzelnen konstruiert das ATMS für jedes Datum der Problemlösungskomponente einen Knoten. Die Knoten sind die Einheiten in dem ATMS, die alle Informationen über die Abhängigkeiten aufnehmen, die das ATMS verwaltet. Unabhängig davon, wie die Problemlösungskomponente ihre Daten darstellt und strukturiert, behandelt das ATMS sie als unstrukturierte, aussagenlogische Ausdrücke. Bei dem Diagnosesystem wird jeder einzelne Wert, der für eine bestimmte Systemgröße an einer bestimmten Stelle ermittelt wird, durch einen Knoten im ATMS dargestellt. Eine Teilmenge der Menge der Knoten im ATMS bilden die sogenannten Basisannahmen. Bei der GDE sind dies die Korrektheitsannahmen der einzelnen Komponenten. Eine Menge von Basisannahmen wird Umgebung genannt. Abhängigkeiten zwischen Basisannahmen und Knoten werden im ATMS durch sogenannte Justifications (Rechtfertigungen) beschrieben. Eine Justification kann als eine aussagenlogische definite Klausel (Horn-Klausel) interpretiert werden. Wenn die Knoten $r_1, ..., r_k$ den Knoten s rechtfertigen, so wird dies durch die Horn-Klausel

$$\neg r_1 \vee ... \vee \neg r_k \vee s$$

ausgedrückt, was äquivalent ist zur Implikation:

$$r_1 \wedge ... \wedge r_k \rightarrow s$$

Eine Justification entspricht einem einzelnen Inferenzschritt bei der Verhaltenssimulation in dem Modell der Anlage. Bei der Verhaltenssimulation könnte eine Regel (bzw. ein Constraint) für den Fluß Q in einer Leitung zwischen Punkt A und B besagen: „Wenn $Q_A = 6$ l/s und Leitung = ok, dann $Q_B = 6$ l/s“. Diese Regel wird in dem ATMS durch die in Abbildung 5.1 gezeigte Justification dargestellt.

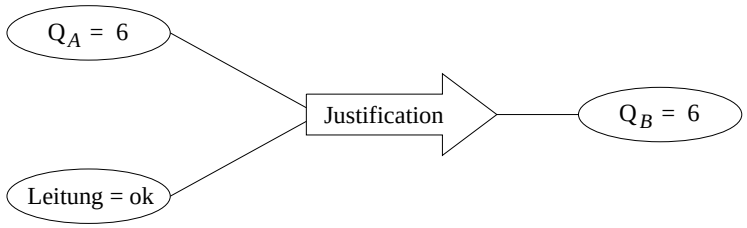


Abbildung 5.1: **Graphische Darstellung einer Justification.**

Ein Knoten gilt in einer Umgebung, wenn der Knoten aus der Umgebung und der Menge der Justifications unter Verwendung der Regeln der Aussagenlogik hergeleitet werden kann. Es ist die Aufgabe des ATMS, die Umgebungen für jeden Knoten zu berechnen. Die Berechnungen, die das ATMS durchführt, sind im wesentlichen Mengenoperationen auf den Umgebungen. Wenn ein hergeleiteter Knoten als Prämisse

in eine andere Justification eingeht, dann werden die Umgebungen dieses Knotens mit denen der anderen Prämissen vereinigt und so die Umgebungen des neu hergeleiteten Knotens gebildet. Dabei ist entscheidend, daß das ATMS nur die minimalen Umgebungen berechnet, in denen ein Knoten gilt. Minimal bedeutet hierbei, daß der Knoten nicht in einer echten Teilmenge dieser Umgebung gilt. Angenommen, ein Knoten sei universell gültig. Dann gibt es bei n Basisannahmen 2^n verschiedene Umgebungen, in denen dieser Knoten gilt. Wenn ein Knoten in einer bestimmten Umgebung gilt, dann gilt er auch in jeder Obermenge dieser Umgebung. Deshalb können durch die minimalen Umgebungen alle Umgebungen charakterisiert werden, in denen ein Knoten gilt. Die leere Menge ist eine besondere Umgebung. Knoten, die in der leeren Umgebung gelten, sind in allen Umgebungen gültig und stellen daher Fakten dar. Ein Fakt wird im ATMS in Form einer Justification mit leerer Prämissenmenge beschrieben.

Eine inkonsistente Umgebung wird als Nogood bezeichnet. Ein Nogood N ist eine Umgebung (also eine Menge von Basisannahmen), so daß die leere Klausel „$\sqcup$“ aus N und der Menge aller Justifications hergeleitet werden kann. Die leere Klausel wird im ATMS durch einen ausgezeichneten Knoten „FALSE“ gekennzeichnet und stellt einen Widerspruch dar. Weil jede Obermenge eines Nogoods ebenfalls widerspruchsvoll ist, berechnet das ATMS die minimalen Nogoods (minimal wiederum im Sinne der Mengeninklusion) für eine vollständige Beschreibung der existierenden Widersprüche.

Die Menge aller minimalen Umgebungen eines Knotens wird als sein Label (Kontext) bezeichnet und vom ATMS zusammen mit dem Datum, das dieser Knoten repräsentiert, gespeichert. Der Label $\{U_1, ...U_k\}$ eines Knotens n besitzt die folgenden vier Eigenschaften [Forbus & de Kleer 1993]:

1. *Korrektheit*: n gilt in jeder Umgebung U_i.
2. *Label-Konsistenz*: $\sqcup$ kann in keinem U_i abgeleitet werden. Alle inkonsistenten Umgebungen werden identifiziert und erscheinen in keinem Label. Knoten, die einen Widerspruch darstellen, haben einen „leeren“ Label. Der Label eines Knotens n ist leer, wenn es a) in dem Abhängigkeitsnetz keinen Weg von den Basisannahmen zu n gibt oder b) alle potentiellen Umgebungen Nogoods sind.
3. *Label-Vollständigkeit*: Jede konsistente Umgebung U, in der n gilt, ist eine Obermenge einer Umgebung U_i des Labels von n.
4. *Label-Minimalität*: Kein U_i ist eine echte Teilmenge einer anderen Umgebung U_j des Labels.

Die Labelinformationen werden durch das Abhängigkeitsnetz, das durch die Justifi-

cations definiert wird, mit einem Algorithmus propagiert, der in [Forbus & de Kleer 1993] beschrieben ist.

Bei der GDE werden die Labelinformationen und die vom ATMS verwalteten Abhängigkeiten zwischen Korrektheitsannahmen und Parameterwerten benutzt, um minimale Konflikte zu bestimmen. Angenommen, an einer bestimmten Stelle der Anlage werde der Fluß $Q_{10} = 4,6$ l/s vorhergesagt. Im ATMS ist dieser Wert durch den Knoten „$Q_{10} = 4,6$“ dargestellt. Der Label dieses Knotens enthält die minimalen Umgebungen, also die minimalen Mengen von Korrektheitsannahmen der Komponenten, die bei der Herleitung des Wertes beteiligt waren. Wird nun an der Anlage der Fluß $Q_{10} = 2,8$ l/s gemessen und dieser Wert dem ATMS als Fakt überreicht (in Form einer Justification mit leerer Prämisse), dann erzeugt das ATMS einen neuen Knoten „$Q_{10} = 2,8$“, dessen Label als einziges Element die leere Menge enthält. Das ATMS hat keinerlei Kenntnis der Semantik eines Knotens. Deshalb existieren diese beiden Knoten im ATMS nebeneinander, ohne einen Widerspruch zu erzeugen. Der Widerspruch muß durch die Problemlösungskomponente erst herbeigeführt werden, indem im ATMS eine der folgenden Regel entsprechende Justification eingeführt wird:

$$Q_{10} = 4,6 \wedge Q_{10} = 2,8 \rightarrow \text{FALSE}.$$

Mit dieser Justification ist aus allen Umgebungen im Label von „$Q_{10} = 4,6$“ FALSE herleitbar, und das ATMS markiert jede der Umgebungen als Nogood. Das ATMS identifiziert weiterhin alle Knoten, die von dem Knoten „$Q_{10} = 4,6$“ abhängen und markiert die nun widerspruchsvollen Umgebungen in ihren Labels als Nogood. Die Nogoods sind minimal, weil die Umgebungen im Label minimal sind, und die minimalen Nogoods entsprechen genau den gesuchten minimalen Konflikten bei der GDE. Dies verdeutlicht, wie das ATMS die Aufgabe der Konflikterkennung erfüllt und wie die minimalen Konflikte automatisch erzeugt werden.

5.2 Komplexität des ATMS

Das ATMS arbeitet inkrementell. Während des Verlaufs der Diagnose führen neue Beobachtungen an der Anlage dazu, daß im ATMS neue Knoten und neue Justifications angelegt werden. Bei jeder neuen Information, die das ATMS erhält, müssen die Knotenlabel entsprechend aktualisiert werden. Ein Hauptproblem bei dem ATMS ist die exponentielle Anzahl möglicher Umgebungen. Insbesondere wächst die Labelgröße exponentiell mit der Anzahl der Basisannahmen [Forbus & de Kleer 1993].

Zur Steigerung der Effizienz konstruiert das ATMS nur diejenigen minimalen Umgebungen, aus denen ein Knoten mit den Justifications hergeleitet werden kann.

Das bedeutet, daß minimale Umgebungen, in denen es keine Inferenz gibt, ignoriert werden (in dem Maße, daß nicht einmal ihr Name erzeugt wird). Wenn es allerdings in jeder minimalen Umgebung eine neue Folgerung gibt, bleibt das Wachstum der Anzahl der im ATMS erzeugten Umgebungen exponentiell. De Kleer und Williams weisen jedoch darauf hin [de Kleer & Williams 1987], daß in der Praxis das durch die Justifications beschriebene Abhängigkeitsnetz schwach zusammenhängend ist, weil die Komponenten in einer Anlage schwach zusammenhängend sind, und deshalb die Anzahl der möglichen Inferenzen relativ klein ist.

5.3 Probleme mit dem ATMS bei ungenauen Messungen

Die Entdeckung von Abweichungen zwischen dem berechneten korrekten Verhalten und dem beobachteten Verhalten ist der wesentliche Punkt für den Diagnoseprozeß. Wie soll jedoch entschieden werden, ob es sich bei einer beobachteten Abweichung tatsächlich um ein Symptom oder um eine Meßungenauigkeit handelt? Um Meßfehlern Rechnung zu tragen, wurde in der Implementierung des Diagnosesystems das Prädikat nearly-equal-p definiert, damit z.B. eine zehnprozentige Abweichung der Werte noch toleriert wird. Wenn z.B. der Druck an einem bestimmten Punkt mit $p_1 = 1,05$ bar gemessen wird, dann wird dieser Wert im ATMS auf den Knoten für den vorausgesagten Wert $p_1 = 1,0$ bar abgebildet, und die Beobachtung erzeugt dann keinen neuen Konflikt.

Der Gebrauch dieses Prädikats bringt jedoch Probleme mit sich. So ist die $n\%$-Abweichung i.a. nicht transitiv. Wenn im Beispiel mit den zwei Leitungen aus Abbildung 3.1 beide Leitungen leck sind, und in jeder einzelnen 10% des Flusses verloren geht, so ist das Diagnoseprogramm nicht in der Lage, die fehlerhaften Leitungen zu identifizieren, weil die Symptome an den Leitungen nicht als solche erkannt werden. Die lokale Natur der Entscheidung, ob es sich um ein Symptom oder um eine Meßungenauigkeit handelt, berücksichtigt nicht die Propagierung von Fehlern durch das Netz.

Hinzu kommt, daß — wenn sich der Benutzer für das Vorliegen eines Symptoms entschieden hat — eine Revision dieses Schrittes zu einem späteren Zeitpunkt im Diagnoseprozeß nicht möglich ist, weil dieser Schritt im ATMS nicht (durch eine entsprechende Basisannahme) explizit dargestellt wird.

Kapitel 6

Das Constraintsystem

6.1 Aufbau des Constraintsystems

Wie bereits erwähnt wurde, geschieht bei dem Diagnosesystem die Modellierung einer Anlage durch Constraints. In diesem Abschnitt wird gezeigt, wie für eine bestimmte Anlage anhand der Wissensrepräsentation der Anlage in *Art Deco* das Constraintsystem erzeugt wird.

Constraints sind ein nützlicher Formalismus, um das Verhalten von Komponenten zu beschreiben. Ganz allgemein besteht ein Constraint aus einer Menge von Variablen und einer Relation über diese Variablen. Im vorliegenden Diagnosesystem werden Constraints für die Simulation des Anlagenverhaltens verwendet. Constraints sind die Objekte, welche die Information darüber enthalten, wie der Wert einer Variablen aus den Werten der anderen Variablen berechnet werden kann. Ebenso werden Constraints für den Test auf Inkonsistenz bei der Verarbeitung von beobachteten Symptomen benutzt.

Die benötigten Relationen zur Beschreibung des Verhaltens einer Komponente sind in *Art Deco* in Form einer Menge von Gleichungen gegeben. Für einen Zylinder, wie in Abbildung 1.2 auf Seite 9 beschrieben, sind dies die folgenden drei Gleichungen für die Kontinuitätsbedingung und das Kräftegleichgewicht:

$$\begin{aligned} F &= p_A \cdot A_K - p_B \cdot A_R && \text{(Kräftegleichgewicht)} \\ Q_A &= A_K \cdot v && \text{(Kontinuitätsbedingung)} \\ Q_B &= -A_R \cdot v \end{aligned}$$

Jede einzelne Verhaltensgleichung einer Komponente wird durch ein Constraint ausgedrückt. Betrachten wir als Beispiel die Gleichung für das Kräftegleichgewicht. Das

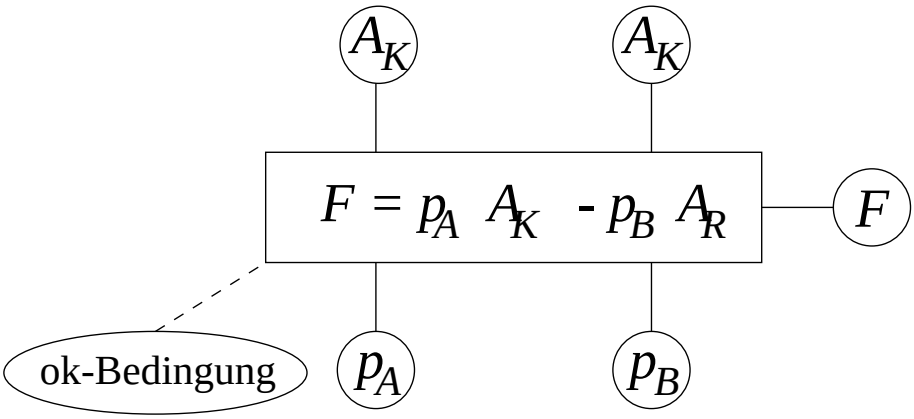


Abbildung 6.1: **Graphische Darstellung des Constraints für das Kräftegleichgewicht.**

zugehörige Constraint ist in der Abbildung 6.1 als „Black Box“ dargestellt. Für jede Variable der Verhaltensgleichung gibt es eine entsprechende Variable in dem Constraint. Darüber hinaus geht die Korrektheitsannahme einer Komponente als Bedingung für die Anwendbarkeit in jedes Constraint ein, so daß es in jedem Constraint eine weitere Variable für die Bedingung gibt.

In unserem Diagnosesystem wird für die Beschreibung von Constraints die Sprache ATCON („A Tiny Constraint Language“) von [Forbus & de Kleer 1993] verwendet. Die Variablen eines Constraints werden in ATCON Zellen genannt. Eine Zelle ist eine Datenstruktur, die neben einem Wert die Information darüber speichert, in welcher Beziehung sie zu den anderen Zellen des Constraints steht und von welchen Werten anderer Zellen ihr Wert abhängt. ATCON basiert auf einem ATMS. Eine Zelle wird durch einen ATMS-Knoten repräsentiert.

Ein Constraint erzwingt die Einhaltung der durch die Constraintrelation vorgebenen Beziehungen zwischen den Werten seiner Zellen. Dies wird in ATCON durch eine Menge von Regeln realisiert. Für jede Zelle gibt es eine oder mehrere Regeln, die angeben, wie von bekannten Werten anderer Zellen auf den Wert für eine bestimmte Zelle geschlossen wird, und welche Berechnungen dabei durchzuführen sind. Eine Regel hat in ATCON die folgende LISP-Syntax:

(⟨Kopf⟩ ⟨Rumpf⟩ ⟨Berechnungsvorschrift⟩).

⟨Kopf⟩ ist die Zelle, für die diese Regel einen Wert ermittelt. ⟨Rumpf⟩ ist eine nichtleere Teilmenge der Zellen des Constraints (dargestellt als Liste), deren Werte alle bekannt sein müssen, damit die Regel angewendet wird. ⟨Berechnungsvorschrift⟩

ist eine LISP-Funktion, die in Abhängigkeit der Zellen in ⟨Rumpf⟩ als Ergebnis einen Wert für die Zelle in ⟨Kopf⟩ liefert. Als Beispiel ist nachfolgend die vorläufige Version einer Regel für die Definition des Constraints „Kräftegleichgewicht“ gegeben. Wie noch gezeigt wird, beschreibt diese Regel die Abhängigkeiten der Zelle p_A von den anderen Zellen noch unvollständig:

```
(pA (pB F AK AR ok) (if (or (not ok)
                            (nearly-zerop AK))
                        :DISMISS
                        (/ (+ F (* pB AR)) AK)))
```

Hierbei ist `:DISMISS` ein spezielles Schlüsselwort in ATCON, welches dem Inferenzmechanismus signalisiert, daß mit diesen Eingaben diese Regel keinen Wert berechnen kann. „`ok`“ bezeichnet die Zelle für die weiter oben erwähnte Constraintbedingung.

Damit ein Constraint korrekt modelliert wird, muß es für jede Teilmenge von Zellen, aus denen ein Wert für die zu bestimmende Zelle bei einer bestimmten Werteausprägung folgt, eine entsprechende Regel geben. Wenn z.B. in dem obigen Constraint die Zelle p_B oder A_R den Wert Null hat, dann reichen zur Berechnung von p_A schon die Zellen F und A_K aus. Dieser Fall muß durch eine entsprechende Regel abgedeckt sein, die in ⟨Rumpf⟩ nur F und A_K enthält. Um die tatsächlichen Abhängigkeiten zwischen den Zellen eines Constraints bei der Herleitung neuer Zellwerte korrekt zu beschreiben, ist zu beachten, daß eine anwendbare Regel nur dann einen Wert abliefern darf, wenn nicht schon eine echte Teilmenge der Zellen in ⟨Rumpf⟩ ausreicht, um diesen Wert zu ermitteln. Das bedeutet: Von mehreren anwendbaren Regeln darf nur diejenige Regel mit der kürzesten ⟨Rumpf⟩-Liste den Wert für die ⟨Kopf⟩-Zelle bestimmen. Wenn in einer Regel eine echte Teilmenge ausreicht, dann muß diese Regel (über das Schlüsselwort `:DISMISS` in der Definition der Berechnungsvorschrift) die Werteberechnung verweigern. Die korrekte Version der obigen Regel lautet dann wie folgt:

```
(pA (pB F AK AR ok) (if (or (not ok)
                            (nearly-zerop AK)
                            (nearly-zerop AR)
                            (nearly-zerop pB))
                        :DISMISS
                        (/ (+ F (* pB AR)) AK)))
```

Für die Werteberechnung der Zelle p_A gibt es in der Constraintdefinition „Kräftegleichgewicht“ insgesamt 21 Regeln.

Die Syntax einer Constraintdefinition lautet in ATCON folgendermaßen:

(constraint ⟨Name⟩ ⟨Zellen⟩ (formulae ⟨Regel$_1$⟩ ... ⟨Regel$_n$⟩)).

Die vollständige Constraintdefinition für das Kräftegleichgewicht ist im Anhang A dargestellt. Für die Beschreibung des Verhaltens eines Zylinders sind drei solcher Constraintbeschreibungen zu erzeugen — eine für jede Verhaltensgleichung.

Der wesentliche Punkt für das Diagnosesystem ist, daß die Constraintdefinitionen automatisch erzeugt werden. Dazu wurde die LISP-Funktion „define-component-constraints“ geschrieben, die für eine Komponente der Anlage aus den einzelnen Verhaltensgleichungen der *Art Deco*-Komponentenbibliothek die entsprechenden Constraintdefinitionen generiert. Die Hauptarbeit bei dieser Funktion leistet ein algebraischer Formelmanipulator, der eine Gleichung nach allen Variablen „auflöst“, und damit die Informationen zum Aufbau der verschiedenen Regeln eines Constraints liefert.

6.2 Symptompropagierung im Constraintnetz

Im letzten Abschnitt wurde gezeigt, wie die Verhaltensgleichungen einzelner Komponenten als Constraints definiert werden. Die Modellierung des gesamten Anlageverhaltens geschieht mit Constraints in einer natürlichen Weise. Die Struktur hydraulischer Anlagen ist durch eine Folge von sich abwechselnden Komponenten und Leitungen gekennzeichnet, die sich jeweils einen Port teilen. Auf der Constraintebene bedeutet dies, daß sich zwei verschiedene Constraints jeweils eine Variable teilen. Dadurch einsteht ein Constraintnetz, in dem die Informationen (Werte von Systemgrößen) zwischen den verschiedenen Komponenten des Systems propagiert werden können.

Wie wird die Propagierung von Werten durch das Constraintnetz realisiert? Wenn ein Constraint genügend Informationen enthält, um einen neuen Wert für eine seiner Zellen zu berechnen, dann äußert sich dies dadurch, daß gewisse Regeln in der Constraintdefinition anwendbar sind. Für jede anwendbare Regel erzeugt ATCON in dem darunterliegenden ATMS eine Justification für die ATMS-Knoten der Zellenwerte, die an dieser Regel beteiligt sind. Die Antezedenten-Knoten der Justification entsprechen den Zellwerten im ⟨Rumpf⟩-Teil der Regeldefinition, und die Konklusion der Justification entspricht dem neu berechneten Wert der Zelle in ⟨Kopf⟩. Weil die Constraintbedingung („`ok`“-Zelle) als Antezedenz-Knoten in jeder Justification enthalten ist, geht die Korrektheitsbedingung in den Label des ATMS-Knotens für den neu hergeleiteten Wert ein. Die Constraintbedingung ist ein ATMS-Knoten, der

die Korrektheitsannahme der Komponente darstellt, deren Verhalten durch dieses Constraint beschrieben ist. Daher enthält der Label jedes hergeleiteten Wertes die Korrektheitsannahmen derjenigen Komponenten, von denen die Berechnung dieses Wertes abhängt.

Nun wird infolge einer Symptombeobachtung der entsprechenden Zelle im Constraintnetz ein neuer Wert zugewiesen. Dieser löst in den Constraints, die diese Zelle benutzen, die Neuberechnung anderer Zellen aus. Dies wird hier als Symptompropagierung bezeichnet. Die Symptompropagierung geschieht nach folgendem Algorithmus:

Algorithmus 6.1: Symptompropagierung

```
1.  Solange es eine Zelle gibt, die einen neuen Wert erhält:
1.1   Wähle eine Zelle, die einen neuen Wert erhält.
1.2   Für jedes Constraint, das diese Zelle benutzt:
1.2.1   Führe jede anwendbare Regel in der Definition des Constraints
        aus, bei der Zelle in <Rumpf> enthalten ist.
```

Bei diesem Verfahren werden nach dem Prinzip der Vorwärtsverkettung der Regeln in den Constraintdefinitionen die globalen Konsequenzen einer lokalen Änderung im Netzwerk bestimmt. Man beachte, daß im obigen Algorithmus jedes Constraint höchstens einmal betrachtet wird. Dazu enthält die Datenstruktur, mit der Constraints dargestellt werden, einen „processed"-Slot zur Markierung des Constraints. Dies ist notwendig, um zu vermeiden, daß Zyklen bei der Propagierung entstehen. Desweiteren wird eine Regel eines Constraints nicht auf das Ergebnis einer anderen Regel desselben Constraints angewandt. Das heißt, eine Regel wird nur ausgeführt, wenn die Werte für die Zellen der Regel von anderem Constraint berechnet wurden als von dem Constraint, zu dem die Regel gehört.

Der Algorithmus verdeutlicht, daß bei der Symptompropagierung der gemessene Wert ausgehend vom Punkt der Beobachtung in alle Richtungen durch das Constraintnetz propagiert wird und dadurch neue Werte für die Parameter vieler anderer Komponenten hervorruft. Dies ist ein wichtiger Punkt, denn er zeigt, daß sich bei der Constraintpropagierung Informationen nicht notwendigerweise in der physikalischen Richtung (von den Eingängen zu den Ausgängen einer Komponente) ausbreiten, sondern in beliebiger Richtung. Die Idee im Rahmen der Diagnose ist, durch die rückwärtige Symptompropagierung im Netz die Abweichung von dem Meßpunkt bis an die Komponenten heranzutragen, die für das Symptom verantwortlich sind.

6.3 Ebenen der Modellierung

Aus der Darstellung des Systems lassen sich drei Ebenen der Repräsentation und der Ausführung eines bestimmten Anlagenmodells identifizieren (Abbildung 6.2). Die erste Ebene enthält die Repräsentation der Anlagentopologie in Form von Komponenten, Ports und Leitungen. Sie wird vom Benutzer bzw. vom Anwendungssystem *Art Deco* durch Instanzierung von Komponentenklassen der Komponentenbibliothek erzeugt, und sie ist die einzige Ebene, mit der der Benutzer im Zuge der Modellbildung zu tun hat.

Wenn die Komponenten einer Anlage auf dieser Ebene instanziert sind, werden die funktionalen Beschreibungen aus den jeweiligen Komponentenmodellen benutzt, um die Constraint-Ebene zu erstellen, einschließlich der zugehörigen ATMS-Knoten und Bedingungen. Wenn Werte eingegeben werden und die Constraints die Berechnung anstoßen, produziert diese zweite Ebene Knoten für die Werte und Justifications auf der ATMS-Ebene oder Justification-Ebene, in der die Constraintbedingungen als unterstützende ATMS-Knoten der Justifications integriert sind.

6.4 Diskussion

Die Constraintpropagierung ist eine Methode, um im Rahmen der modellbasierten Diagnose das Anlageverhalten auf der Basis der Verhaltensbeschreibungen der einzelnen Komponenten der Anlage zu simulieren. Hierbei ist das Verhalten einer Anlage durch die Wertebelegung der physikalischen Größen beschrieben. Eine Verhaltenssimulation bezeichnet den Prozeß der globalen Wertebestimmung in dem Netzwerk anhand lokaler Vorgaben. Die Vorteile einer Verhaltenssimulation durch Constraintpropagierung liegen darin, daß

- das Constraintnetz sich automatisch erzeugen läßt,
- kontinuierliche Variablen benutzt werden können,
- der Rechenaufwand in jedem einzelnen Constraint relativ klein ist, weil die lokalen Berechnungen relativ einfach sind, und
- jede lokal konsistente Lösung ebenfalls global konsistent ist.

Der Nachteil der Constraintpropagierung ist, daß nicht immer alle global konsistenten Werte ermittelt werden, und die Constraintpropagierung terminiert, obwohl einige Constraints noch unbenutzt und einige physikalische Größen im System noch

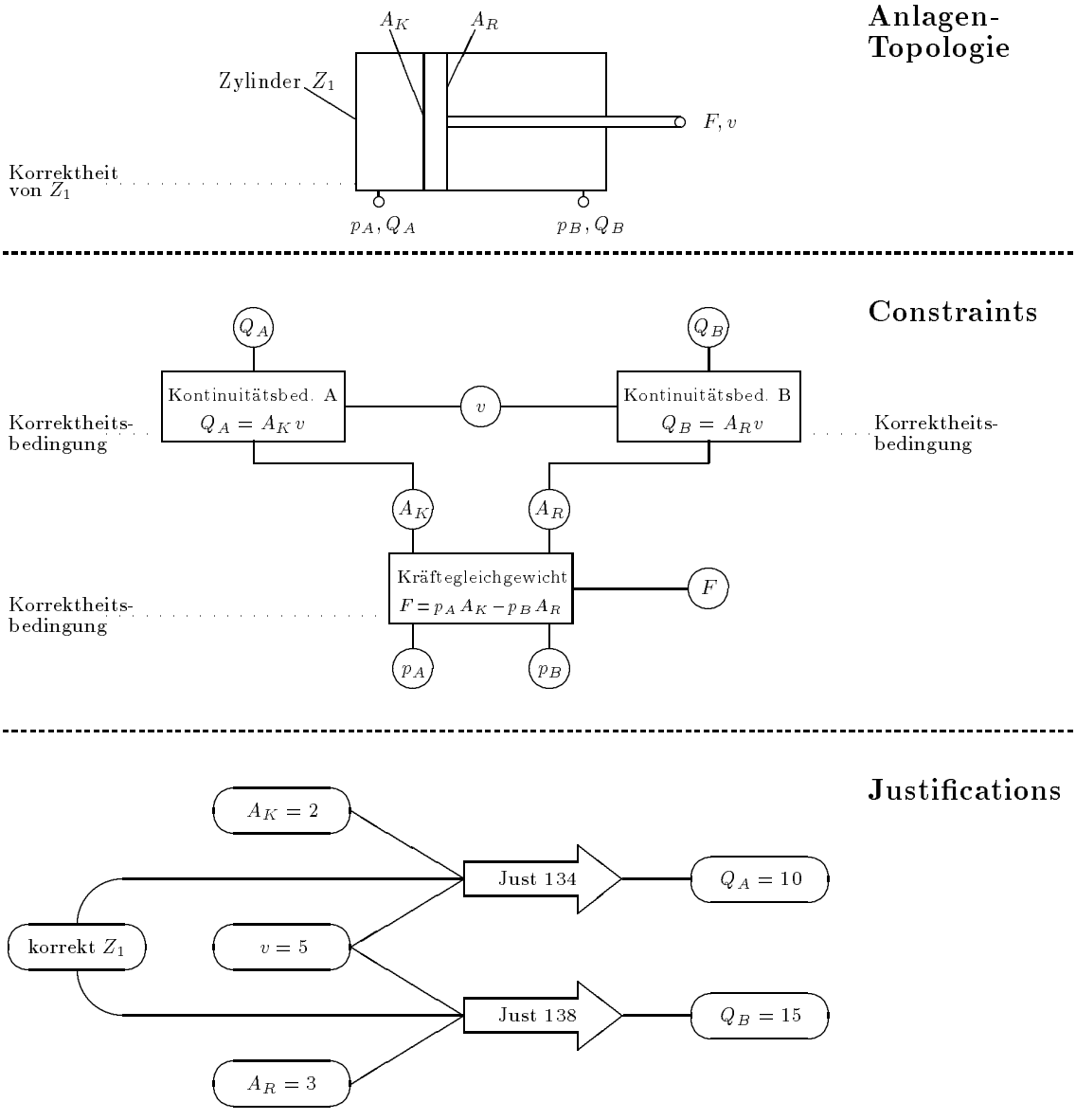


Abbildung 6.2: **Die Modellbildung und -ausführung erfolgt auf drei Ebenen. Die Korrektheitsannahme einer Komponente auf Anlagenebene geht auf der Constraintebene als Korrektheitsbedingung in alle Constraints ein, durch die das Verhalten der Komponente beschrieben ist. Das „Feuern“ eines Constraints (d.h. die Neuberechnung des Wertes einer Zelle) wird realisiert, indem auf ATMS-Ebene eine Justification für die Wert-Herleitung erzeugt wird. Die Korrektheitsbedingung geht dabei in jede Justification als Antezedenz-Knoten ein.**

nicht mit einem Wert belegt sind. Dies tritt einmal ein, wenn nicht genügend Informationen über die Vorgabewerte (Betriebsparameter) der Anlage gegeben sind. Andererseits kann dies bei Komponenten mit nichtlinearem Verhalten auftreten. Die Constraintpropagierung wird häufig an digitalen Schaltkreisen demonstriert [Forbus & de Kleer 1993], bei denen sich das Komponentenverhalten durch lineare Gleichungen beschreiben läßt. Damit die in Abschnitt 6.1 erwähnte rückwärtige Wertepropagierung vom Ausgang zum Eingang einer Komponente funktionieren kann, muß sich das Verhalten dieser Komponente durch eine bijektive Funktion abbilden lassen. In der Hydraulik haben wir es häufig mit nichtlinearem Komponentenverhalten zu tun, so daß eine bijektive Funktion nicht gegeben werden kann. Beispielsweise ist der hydraulische Widerstand proportional zum Quadrat des Flusses.

Bei dem vorliegenden Diagnosesystem geschieht die Konflikterkennung auf der Grundlage der Constraintpropagierung. Die Unvollständigkeit der Constraintpropagierung hat zur Folge, daß an einigen Punkten der Anlage das Modell keine Werte vorhersagt, und deshalb aus der Messung eines Symptoms weniger Konflikte resultieren. Weil die Menge der erzeugten minimalen Konflikte unvollständig ist, erkennt die GDE nicht, daß gewisse Meßpunkte für die Unterscheidung zwischen verschiedenen Kandidaten nützlich sind. Die Folge ist, daß eine größere Anzahl von Messungen notwendig ist.

Der Druck und der Fluß sind die Größen in einem hydraulischen System, welche die Informationen zwischen den Komponenten weiterleiten. Ein grundsätzliches Problem ist, daß sich der Druck in einer hydraulischen Anlage i.a. nicht durch lokale Propagierung bestimmen läßt. Der Druck in einer Anlage stellt in *Art Deco* die Lösung eines Gleichungssystems dar [Stein & Lemmen 1992]. Gleichungssysteme sind i.a. nicht durch lokale Constraintpropagierung lösbar. Denkbar wäre es, für die Berechnung des Druckes bei der Verhaltenssimulation die in *Art Deco* vorhandenen Programmroutinen zur Lösung von Gleichungssystemen zu benutzen. Der wesentliche Grund für die Verwendung von Constraintsystemen bei der Diagnose mit der GDE ist jedoch der, daß in jedem lokalen Propagierungsschritt die Abhängigkeiten von der Korrektheit der beteiligten Komponenten protokolliert werden. Die Information über die Abhängigkeiten kann bei der Werteberechnung mit den *Art Deco*-Routinen nicht ermittelt werden. Bei der GDE können deshalb Fehler nicht diagnostiziert werden, deren Symptome sich in solchen Größen auswirken, die vom Druck abhängen (z.B. die Kraft an einem Zylinderkolben).

Der Fluß in einem System kann durch lokale Propagierung ermittelt werden, sofern durch die Verhaltensgleichungen der einzelnen Komponenten der Fluß an jeder Komponente eindeutig bestimmt ist. Allerdings ist dies bei einer T-Verbindung nicht der Fall. In dem Modell der hydraulischen Anlage wird eine T-Verbindung wie eine Kom-

ponente repräsentiert, die drei Leitungen verbindet. Die einzige Verhaltensgleichung bei einer T-Verbindung ist die, daß die Summe der Flüsse an ihren drei Anschlüssen Null ist. Das Problem der Bestimmung der Flußaufteilung an einer T-Verbindung ist unterbestimmt. Die Constraintpropagierung des Flusses stoppt, sobald der lokal propagierte Fluß die erste T-Verbindung erreicht. Dies schränkt den Nutzen der Verwendung von Constraintsystemen zur Verhaltensmodellierung bei der Diagnose von hydraulischen Anlagen stark ein.

Die unbedingte Forderung bei der Diagnose an den verwendeten Mechanismus zur Verhaltenssimulation ist, daß eine initiale Wertebelegung für alle relevanten Größen an allen Punkten der Anlage berechnet wird. Eine Abweichung (Symptom) kann nur festgestellt werden, wenn es einen Referenzwert gibt, mit dem der beobachtete Wert verglichen werden kann. Damit an jedem Punkt ein Referenzwert für das korrekte Verhalten berechnet werden kann, müssen zu den bereits bestehenden äußeren Vorgaben im Modell weitere Größen an bestimmten Punkten vorgegeben werden. Die Punkte sind so zu wählen, daß sich mittels Constraintpropagierung überall ein Wert einstellt. Das Vorgeben von Größen ist problematisch, denn Vorgaben werden bei der GDE — wie Beobachtungen — als Fakten behandelt, die im Verlauf der Diagnose nicht widerlegbar sind. Wird z.B. die Geschwindigkeit eines Zylinderkolbens mit $v = 8$ m/s im Modell vorgegeben, um einen Referenzwert für den Druck an dieser Komponente zu erhalten, und später an der Anlage festgestellt, daß sich der Kolben tatsächlich mit $v = 6$ m/s bewegt, dann ist es nicht möglich, dieses Symptom im Diagnosesystem zu verarbeiten.

In Ermangelung einer Alternative zur GDE, und damit zur Verhaltenssimulation mittels Constraintpropagierung, wurde das Diagnosesystem auf der Basis der GDE implementiert. Es wird dabei unterstellt, daß bei der Durchführung eines konkreten Diagnoseproblems die fehlenden Werte für Systemgrößen von außen vorgegeben werden können, so daß durch lokale Propagierung an allen Punkten im Modell der Anlage Referenzwerte hergeleitet werden können.

Kapitel 7

Realisierung und Umsetzung

Das Diagnosesystem ist in der Programmiersprache COMMON-LISP implementiert. Den Ausgangspunkt für die Umsetzung des Diagnosesystems liefert eine Implementierung des ATMS aus [Forbus & de Kleer 1993]. Der größte Teil des Zeitaufwandes im ATMS entsteht bei der Labelaktualisierung. Die wesentlichen Operationen dabei sind Mengenoperationen über die Mengen von Basisannahmen (Umgebungen).

In der Implementierung des ATMS von Forbus und de Kleer werden die Standard-LISP-Mengenoperationen verwendet. Dies ist sehr ineffizient. Wenn die Mengen durch Listen repräsentiert werden, in denen die Basisannahmen stets aufsteigend sortiert sind (d.h. die Mengen wohlgeordnet sind), dann können die Mengenoperationen effizienter implementiert werden. Durch eine Neuimplementierung der Mengenoperationen konnte eine wesentliche Effizienzsteigerung des ATMS erreicht werden.

Forbus und de Kleer bieten in ihrem Buch [Forbus & de Kleer 1993] ebenso eine rudimentäre Implementierung der GDE an, die als Leitfaden für die Realisierung des vorliegenden Diagnosesystems gewählt wurde. In dieser Implementierung wird die Kandidatengenerierung über die Bildung von Extensions im ATMS erreicht. Eine Extension ist eine maximale konsistente Umgebung, d.h. eine Umgebung, zu der keine weitere Basisannahme hinzugefügt werden kann, ohne daß ein Nogood entsteht. Ein minimaler Kandidat ist das Komplement einer Extension. Ein Beispiel soll dies verdeutlichen. {A,B,C,D} sei die Menge aller Komponenten in einem System. Die minimalen Konflikte (Nogoods) seien {B,C} und {A,C,D}. Dann gibt es drei Extensions: {A,B,D}, {A,C} und {C,D}. Wie leicht gesehen werden kann, sind die minimalen Kandidaten die Komplemente dieser Mengen, nämlich {C}, {B,D} und {A,B}. Bei der Implementierung von Forbus und de Kleer werden die Extensions im ATMS durch Umgebungen dargestellt. Es gibt eine exponentielle Anzahl von Extensions, weil es exponentiell viele minimale Kandidaten gibt. Durch die enorme

Anzahl von Umgebungen, die das ATMS bei jeder Labelaktualisierung zusätzlich berücksichtigen muß, wird es sehr langsam. Im vorliegenden Diagnosesystem wurde die Kandidatengenierung nach dem im Kapitel 4 beschriebenen Algorithmus von de Kleer und Williams implementiert.

Anhang A

Beispiel für eine Constraintdefinition

Nachfolgend ist die in Abschnitt 6.1 angesprochene vollständige Constraintdefinition für die Modellierung des Kräftegleichgewichts bei einem Zylinder in der ATCON-Notation aufgeführt. Der formulae-Teil der Definition besteht aus 21 Regeln, durch die das Constraint realisiert wird.

```
(constraint cylinder.force-balance ((F cell) (p_A cell) (A_K cell)
                                    (p_B cell) (A_R cell) (ok cell))
 (formulae (F (p_A p_B ok) (if (and ok
                                    (nearly-zerop p_A)
                                    (nearly-zerop p_B))
                               0
                              :DISMISS))
           (F (p_A A_R ok) (if (and ok
                                    (nearly-zerop p_A)
                                    (nearly-zerop A_R))
                               0
                              :DISMISS))
           (F (p_B A_K ok) (if (and ok
                                    (nearly-zerop p_B)
                                    (nearly-zerop A_K))
                               0
                              :DISMISS))
           (F (A_K A_R ok) (if (and ok
                                    (nearly-zerop A_K)
```

```
                        (nearly-zerop A_R))
                   0
                  :DISMISS))
(F (p_A A_K p_B ok) (if (or (not ok)
                            (not (nearly-zerop p_B))
                            (nearly-zerop p_A)
                            (nearly-zerop A_K))
                        :DISMISS
                       (* p_A A_K)))
(F (p_A A_K A_R ok) (if (or (not ok)
                            (not (nearly-zerop A_R))
                            (nearly-zerop p_A)
                            (nearly-zerop A_K))
                        :DISMISS
                       (* p_A A_K)))
(F (p_B A_R p_A ok) (if (or (not ok)
                            (not (nearly-zerop p_A))
                            (nearly-zerop p_B)
                            (nearly-zerop A_R))
                        :DISMISS
                       (- (* p_B A_R))))
(F (p_B A_R A_K ok) (if (or (not ok)
                            (not (nearly-zerop A_K))
                            (nearly-zerop p_B)
                            (nearly-zerop A_R))
                        :DISMISS
                       (- (* p_B A_R))))
(F (p_A A_K p_B A_R ok) (if (or (not ok)
                                (nearly-zerop p_A)
                                (nearly-zerop A_K)
                                (nearly-zerop p_B)
                                (nearly-zerop A_R))
                            :DISMISS
                           (- (* p_A A_K) (* p_B A_R))))
(p_A (F p_B A_K ok) (if (or (not ok)
                            (not (nearly-zerop p_B))
                            (nearly-zerop A_K))
                        :DISMISS
                       (/ F A_K)))
```

```
(pA (F AR AK ok) (if (or (not ok)
                         (not (nearly-zerop AR))
                         (nearly-zerop AK))
                     :DISMISS
                    (/ F AK)))
(pA (F pB AR AK ok) (if (or (not ok)
                            (nearly-zerop AK)
                            (nearly-zerop AR)
                            (nearly-zerop pB))
                        :DISMISS
                       (/ (+ F (* pB AR)) AK)))
(AK (F pB pA ok) (if (or (not ok)
                         (not (nearly-zerop pB))
                         (nearly-zerop pA))
                     :DISMISS
                    (/ F pA)))
(AK (F AR pA ok) (if (or (not ok)
                         (not (nearly-zerop AR))
                         (nearly-zerop pA))
                     :DISMISS
                    (/ F pA)))
(AK (F pB AR pA ok) (if (or (not ok)
                            (nearly-zerop pA)
                            (nearly-zerop AR)
                            (nearly-zerop pB))
                        :DISMISS
                       (/ (+ F (* pB AR)) pA)))
(pB (F pA AR ok) (if (or (not ok)
                         (not (nearly-zerop pA))
                         (nearly-zerop AR))
                     :DISMISS
                    (- (/ F AR))))
(pB (f AK AR ok) (if (or (not ok)
                         (not (nearly-zerop AK))
                         (nearly-zerop AR))
                     :DISMISS
                    (- (/ F AR))))
(pB (F pA AK AR ok) (if (or (not ok)
                            (nearly-zerop AR)
```

```
                                (nearly-zerop A_K)
                                (nearly-zerop p_A))
                            :DISMISS
                           (/ (- (* p_A A_K) F) A_R)))
(A_R (F p_A p_B ok) (if (or (not ok)
                            (not (nearly-zerop p_A))
                            (nearly-zerop p_B))
                        :DISMISS
                       (- (/ F p_B))))
(A_R (F A_K p_B ok) (if (or (not ok)
                            (not (nearly-zerop A_K))
                            (nearly-zerop p_B))
                        :DISMISS
                       (- (/ F p_B))))
(A_R (F p_A A_K p_B ok) (if (or (not ok)
                                (nearly-zerop p_B)
                                (nearly-zerop A_K)
                                (nearly-zerop p_A))
                            :DISMISS
                           (/ (- (* p_A A_K) F) p_B)))))
```

Literaturverzeichnis

[1] Abu-Hanna, A., Gold, Y., An integrated, deep-shallow expert system for multi-level diagnosis of dynamic systems, in *Artificial Intelligence in Engineering: Diagnosis and Learning*, herausg. von J. S. Gero, Elsevier, Amsterdam, 75-94, 1988.

[2] Allen, J. F., Towards a General Theory of Action and Time, *Artificial Intelligence*, **23**, 123-184, 1984.

[3] Bathelt, P., Kerndlmair, M., Zink, T., Expertensysteme für die technische Diagnose, in *Fachberichte Messen-Steuern-Regeln*, Band 14, 505-532, 1986.

[4] Buchanan, B., Shortliffe, E., *Rule-Based Expert Systems. The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, Massachusetts, 1984.

[5] Chen, J., Srihari, S. N., Candidate Ordering and Elimination in Model-based Fault Diagnosis, *Proceedings 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, 1363-1368, 1989.

[6] Console, L., Portinale, L., Theseider Dupré, D., Torasso, P., Combining Heuristic Reasoning with Causal Reasoning in Diagnostic Problem Solving, in *Second Generation Expert Systems*, herausg. von J. M. David, J. P. Krivine und R. Simmons, Springer-Verlag, 47-68, 1993.

[7] Dague, P., Raiman, O., Deves, P., Troubleshooting: When Modeling is the Difficulty, *Proceedings of the 6st National Conference on Artificial Intelligence*, Seattle, WA, 600-605, 1987.

[8] Davis, R., Diagnostic Reasoning Based on Structure and Behavior, *Artificial Intelligence*, **24**, 347-411, 1984.

[9] de Kleer, J., How circuits work, *Artificial Intelligence*, **24**, 205-280, 1984.

[10] de Kleer, J., An assumption-based truth maintenance system, *Artificial Intelligence*, **28**, 127-162, 1986.

[11] DE KLEER, J., Using crude probability estimates to guide diagnosis, *Artificial Intelligence*, **45**, 381-391, 1990.

[12] DE KLEER, J., BROWN, J., A Qualitative Physics Based on Confluences, *Artificial Intelligence*, **24**, 7-83, 1984.

[13] DE KLEER, J., MACKWORTH A., REITER R., Characterizing diagnoses and Systems, *Artificial Intelligence*, **56**, 197-222, 1992.

[14] DE KLEER, J., WILLIAMS, B. C., Diagnosing multiple faults, *Artificial Intelligence*, **32**, 97-130, 1987.

[15] DE KLEER, J., WILLIAMS, B. C., Diagnosis with behavioral modes, *Proceedings 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, 1324-1330, 1989.

[16] DOYLE, J., A truth maintenance system, *Artificial Intelligence*, **12**, 231-272, 1979.

[17] FINK, P., Control and Integration of Diverse Knowledge in a Diagnostic Expert System, *Proceedings 11th International Joint Conference on Artificial Intelligence*, 426-431, 1985.

[18] FINK, P., LUSTH, C., Expert Systems and Diagnostic Expertise in the Mechanical and Electrical Domains, *IEEE Transactions on Systems, Man and Cybernetics (SMC)*, **17**, 340-349, 1987.

[19] FORBUS, K. D., DE KLEER, J., *Building Problem Solvers*, MIT Press, Cambridge, MA, 1993.

[20] GAREY, M. R., JOHNSON, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[21] GENESERETH, M. R., The use of design descriptions in automated diagnosis, *Artificial Intelligence*, **24**, 411-436, 1984.

[22] HAMSCHER, W. C., Model-based troubleshooting of digital systems, Artificial Intelligence Laboratory, *Technical Report TR-1074*, MIT, Cambridge, MA, 1988.

[23] HECKERMAN, D., *Probabilistic Similarity Networks*, MIT Press, Cambridge, MA, 1991.

[24] KOCKSKÄMPER, S., NEUMANN, B., JOSUB, A., MÜLLER, H., Die Anwendung modellbasierten Schließens bei der Diagnose schiffstechnischer Anlagen, *Technischer Bericht LKI-M-93/1*, Labor für Künstliche Intelligenz, Fachbereich Informatik, Universität Hamburg, 1993.

[25] KUIPERS, B., Qualitative Simulation, *Artificial Intelligence*, **29**, 289-338, 1986.

[26] KULIKOWSKI, C. A., WEISS, S. M., Representation of Expert Knowledge for Consultation: The CASNET and EXPERT Projects, in *Artificial Intelligence in Medicine*, herausg. von P. Szolovits, Westview Press, Boulder, CO, 21-56, 1982.

[27] LAMBERT, H., ESHELMAN, L., IWASAKI, Y., Using qualitative physics to guide the acquisition of diagnostic knowledge, in *Artificial Intelligence in Engineering: Diagnosis and Learning*, herausg. von J. S. Gero, Elsevier, Amsterdam, 261-283, 1988.

[28] LENGAUER, T., *Combinatorial Algorithms for Integrated Circuit Layout*, B. G. Teubner, Stuttgart, 1990.

[29] MILLER, J. A., POTTER, W. D., GANDHAM, R. V., LAPENA, C. N., An Evaluation of Local Improvement Operators for Genetic Algorithms, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol 23, No. 5, 1340-1350, 1993.

[30] NG, H. T., Model-Based Fault Diagnosis of Time-Varying Continuous Physical Devices, *Proceedings 6th Conference on Artificial Intelligence Applications*, 9-15, 1990.

[31] PENG, Y., REGGIA, J. A., *Abductive Inference Models for Diagnostic Problem-Solving*, Springer-Verlag, 1990.

[32] POOLE, D., Normality and Faults in Logic-Based Diagnosis, *Proceedings 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, 1304-1310, 1989.

[33] POPLE, H. E., Heuristic Methods for Imposing Structure on Ill-structured Problems: The Structuring of Medical Diagnostics, in *Artificial Intelligence in Medicine*, herausg. von P. Szolovits, Westview Press, Boulder, CO, 119-190, 1982.

[34] PUPPE, F., *Problemlösungsmethoden in Expertensystemen*, Springer-Verlag, 1990.

[35] REGGIA, J. A., NAU, D. S., WANG, P., Diagnostic Expert Systems Based on a Set Covering Model, *Int. Journal of Man-Machine Studies*, **19**, 437-460, 1983.

[36] REITER, R., A theory of diagnosis from first principles, *Artificial Intelligence*, **32**, 57-95, 1987.

[37] ROJAS, R., *Theorie der neuronalen Netze*, Springer-Verlag, 1993.

[38] SHAPIRO, S. C., SRIHARI, S. N., TAIE, M. R., GELLER, J., VMES: A network-based versatile maintenance expert system, *Proceedings of the 1st International Conference on Applications of AI to Engineering Problems*, Springer-Verlag, 925-936, 1986.

[39] STEIN, B., *Functional Models in Configuration Systems*, Dissertation (vorläufige Fassung), Universität-GH Paderborn, Fachbereich Informatik, 1994.

[40] STEIN, B., LEMMEN, R., Art Deco: A System which Assists the Checking of Hydraulic Circuits, in *ECAI '92, Workshop for Model-Based Reasoning*, 1992.

[41] STRUß, P., Model-Based Diagnosis - Progress and Problems, in *Proceedings 3. Internationaler GI-Kongreß Wissensbasierte Systeme*, München, 320-331, 1989.

[42] STRUß, P., Qualitative Modellierung physikalischer Systeme auf dem Weg zu Anwendungen, *Künstliche Intelligenz (KI)*, **4**, 50-53, 1993.

[43] STRUß, P., DRESSLER, O., Physical negation: Integrating fault models into the general diagnostic engine, *Proceedings 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, 1318-1323, 1989.

[44] TANNER, M. C., BYLANDER, T., Application of the CSRL Language to the Design of Expert Diagnostic Systems: The AUTO-MECH Experience, in *Artificial Intelligence in Maintenance*, herausg. von J. J. Richardson, Noyes Publications, 149-162, 1985.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die zum Gelingen der Arbeit beigetragen haben.

Herrn Prof. Dr. Kleine Büning danke ich für die Übertragung der Aufgabe. Besonders bedanke ich mich bei meinem Betreuer Herrn Benno Stein für seine ständige Gesprächsbereitschaft und sein reges Interesse am Fortgang dieser Arbeit.

Den netten Mitarbeitern des Fachbereichs sei gedankt für das angenehme Arbeitsklima und viele hilfreiche Anregungen und Diskussionen.

Ein herzliches Dankeschön gilt meinem Bruder Martin, der das mühevolle Korrekturlesen der Arbeit auf sich genommen und bei der Gestaltung der Arbeit mit LaTeX geholfen hat.

Bei meinen Eltern bedanke ich mich für ihre Unterstützung und den Rückhalt, den sie mir während des gesamten Studiums und insbesondere während der Diplomarbeit gegeben haben.

Hiermit erkläre ich, diese Arbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt zu haben.

Paderborn, den 27. März 1995